

BLOCKCHAINS AND SECURITY:

**Grammar-Based Evolutionary Fuzzing
for JSON-RPC APIs
and
the Division of Responsibilities**

LISETTE VELDKAMP



Blockchains and Security

Grammar-Based Evolutionary Fuzzing for JSON-RPC APIs
and
the Division of Responsibilities

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
in Computer Science

and

MASTER OF SCIENCE
in Communication Design for Innovation

by

L.S. Veldkamp
born in Enschede, the Netherlands



Software Engineering Research Group
Department of Software Technology
Faculty EEMCS, Delft University of
Technology
Delft, the Netherlands

Communication Design for Innovation Group
Department of Science Education
& Communication
Faculty AS, Delft University of Technology
Delft, the Netherlands

Blockchains and Security

Grammar-Based Evolutionary Fuzzing for JSON-RPC APIs
and
the Division of Responsibilities

Abstract

The continual increase in cyber crime revolving blockchain applications calls for secure blockchain systems and clarity on the division of security responsibilities. This research is an integrated project between two master programmes at the Delft University of Technology: Computer Science and Communication Design for Innovation, and focuses on software testing and security responsibilities.

In this study, we investigate if grammar-based fuzzing, a popular approach for identifying bugs in software, is effective on JSON-RPC systems like blockchain applications Ripple and Ethereum. Furthermore, we evaluate whether we can improve upon traditional grammar-based fuzzing by using evolutionary search. We introduce GEFRA, a black-box grammar-based fuzzing tool that generates tests for JSON-RPC APIs. Using a diversity-based fitness function that leverages system feedback, GEFRA is able to effectively guide the search process towards new test cases that obtain additional test coverage.

Additionally, various perspectives on blockchain security responsibilities are investigated. A media content analysis was performed and interviews were conducted with legal and blockchain experts. News media frequently frame end users as responsible for the prevention of blockchain attacks. While attackers are legally responsible, users are left to deal with the consequences if attackers cannot be found. Responsibilities generally end up with users as decentralisation is the core idea of blockchain. Legislation may be the only solution to define a clear division of responsibilities.

Computer Science Thesis Committee:

Chair: Dr.ir. S.E. Verwer, Faculty EEMCS, TU Delft
First supervisor: Dr. A. Panichella, Faculty EEMCS, TU Delft
Second supervisor: Ir. M. Olsthoorn, Faculty EEMCS, TU Delft
External supervisor: Dr. É. Kalmár, Faculty AS, TU Delft
Committee member: Prof.ir. P. Bosman, Faculty EEMCS, TU Delft

Communication Design for Innovation Committee:

First supervisor: Dr. É. Kalmár, Faculty AS, TU Delft
Second supervisor: Drs. C. Wehrmann, Faculty AS, TU Delft
External supervisor: Dr. A. Panichella, Faculty EEMCS, TU Delft

Preface

With these 215 pages, I submit my final report to this university. I can now say that this research project has been the greatest challenge of my studies. With the beginning of my thesis during the COVID-19 lockdowns, and having no idea what research topic I wanted to investigate, I was off to a rocky start. Focusing on two different researches simultaneously proved to be more difficult than expected, and while I did not plan to spend two years on my master's thesis, I am happy with everything I have learned and proud to finally present this work. There are several people that helped me get here that I would like to thank.

First of all, I want to thank all my supervisors. I want to thank Annibale Panichella and Mitchell Olsthoorn, who were always enthusiastic and up for a casual talk. Thank you for giving me the freedom to try out my own ideas, while also making sure I was headed toward promising directions. Next, I want to thank Éva Kalmár, who was always very understanding and eager to help me whenever I was stuck. Thank you for your patience and regularly checking up on me. Your support and creative ideas for my research were very welcome. My gratitude also goes out to Caroline Wehrmann. Although we did not meet often, I knew that I could come to you if I ever needed help. Your critical yet constructive reflections were highly appreciated. I would also like to add a special thank you for the last-minute checking of my colloquium report so I could graduate this study year.

Finally, I want to thank the people close to me. First of all my parents, who have supported me endlessly. Especially my dad, who never turned me away the numerous times I asked for help throughout all of my education, and always encouraged me to take the hard but rewarding path instead of the easy one. Next I want to thank my boyfriend Dimitri, who silently endured my complaining whenever I had a setback. Thank you for making me feel better in those moments and for never doubting I could finish this research project, even when I did. Lastly, I would like to thank my sisters and friends, who provided me with essential distractions throughout this long, yet (mostly) fun process.

L.S. Veldkamp
Delft, the Netherlands
August 16, 2022

Contents

Preface	iii
Contents	v
I General Introduction	3
1 General Introduction	5
1.1 The blockchain technology	5
1.2 Importance of blockchain security	6
1.3 Research aims	8
1.4 Thesis outline	8
II Computer Science	11
2 Introduction	13
2.1 Problem description	13
2.2 Research aim and questions	15
2.3 Contributions	16
2.4 Outline	17
3 Background	19
3.1 APIs	19
3.2 API testing	22
3.3 Search algorithms	24
3.4 Fuzzing	25
4 Related Work	29
4.1 Black-box fuzzing for web APIs	29
4.2 Evolutionary fuzzing	35
4.3 Research gap	36
5 Approach	37
5.1 Building a fuzzing tool	37
5.2 Grammar construction	38

5.3	System server setup	40
5.4	Test case generation and execution	40
5.5	Heuristic	41
5.6	Mutation engine	48
6	Implementation	51
6.1	GEFRA architecture	51
6.2	Grammar-based (mutational) fuzzing	53
6.3	Grammar-based evolutionary fuzzing	55
6.4	Tool usage	64
7	Empirical evaluation	69
7.1	Performance metrics	69
7.2	Benchmark APIs	70
7.3	Experimental protocol	70
7.4	Configuration	71
7.5	Threats to validity and reproducibility	72
8	Results	73
8.1	Evolutionary fuzzing performance	73
8.2	Suitability of fitness functions	81
9	Discussion	85
9.1	Grammar-based <u>E</u> volutionary <u>F</u> uzzer for <u>R</u> PC- <u>A</u> PIs	85
9.2	Limitations	86
10	Conclusions	89
10.1	Research sub questions	89
10.2	Research main question	90
10.3	Recommendations for future work	91
III Communication Design for Innovation		93
11	Introduction	95
11.1	Problem description	95
11.2	Research aim and questions	97
11.3	Contributions	98
11.4	Outline	99
12	Background	101
12.1	Cyber security	101

12.2 Blockchain attacks	102
13 Methodology	107
13.1 Literature study	107
13.2 Content analysis	108
13.3 Interviews with legal professionals	112
13.4 Interviews with blockchain researchers	113
13.5 Methods outline	114
14 Theoretical framework	115
14.1 Frames	115
14.2 Theoretical framework	116
15 Media framing of responsibilities	121
15.1 General observations	121
15.2 Causal responsibility	123
15.3 Treatment responsibility	127
15.4 Discussion	132
16 Legal responsibility	135
16.1 The legal professionals	135
16.2 Interview findings	136
16.3 Discussion	139
17 Developers' influences on user responsibilities	143
17.1 The blockchain professionals	143
17.2 Interview findings	144
17.3 Discussion	147
18 Conclusion	149
18.1 Research sub questions	149
18.2 Research main question	151
19 Discussion	153
19.1 Contributions and relevance	153
19.2 Research limitations	155
19.3 Reliability, validity and ethics	156
19.4 Recommendations for future work	157
Bibliography	159
A Standard configuration of parameters	173

CONTENTS

B Coding protocol	175
C Additional figures content analysis	183
D Legal professionals interview protocol and summaries	187
D.1 Interview protocol	187
D.2 Legal professionals interview summaries	190
E Blockchain professionals interview protocol and summaries	201
E.1 Interview protocol	201
E.2 Blockchain researchers interview summaries	201

Acronyms

API Application Programming Interface.

AUC Area Under the Curve.

CIA Confidentiality, Integrity and Availability.

CRUD Create, Read, Update and Delete.

EA Evolutionary Algorithm.

GA Genetic Algorithm.

GB-EVO Grammar-Based Evolutionary.

GB-MUT Grammar-Based Mutational.

GEFRA Grammar-based Evolutionary Fuzzer for RPC-APIs (the developed tool).

HTTP Hypertext Transfer Protocol.

IQR Interquartile range.

JSON JavaScript Object Notation.

OAS OpenAPI Specification.

OPSEC Operational Security.

P2P Peer-to-Peer.

PoS Proof-of-State.

POW Proof of Work.

REST Representational State Transfer.

RPC Remote Procedure Call.

SUT System Under Test.

URI Uniform Resource Identifier.

PART I

General Introduction

1

General Introduction

Following the success of the cryptocurrency Bitcoin, which was officially introduced in 2009, blockchain technology quickly gained popularity. Bitcoin was the first electronic payment system based on a decentralised peer-to-peer network, without the need for mediation by a third party (such as a bank). Since then, blockchain technology has been applied to many fields. Blockchain applications include the exchange of cryptocurrencies in the financial services sector [64, 80, 102, 120]; secure sharing of medical data in the healthcare sector [64, 102, 120]; improving the efficiency of claim processing in the insurance sector [64, 80, 102, 120]; royalty checking in the music business [64, 80, 102, 120]; monitoring of processes in the logistics and supply chain sector [64, 80, 102, 120], and is believed to be able to reshape the way business is done [12]. Overall, blockchain is seen as a promising technology for information systems and is expected to bring significant benefits to society [5, 14].

1.1 The blockchain technology

The blockchain is a distributed (shared) ledger that can either be private or public (open to anyone). It is a way of recording what happened and in what order. The ledger is essentially a chain of blocks that contain information. Each block contains the hash¹ of the block, the hash of the previous block, and the data to be stored. The kind of data depends on the blockchain application. For a cryptocurrency for example it can be a transaction: containing the sender, receiver and number of coins. The hash is comparable to a fingerprint, being a unique identifier of a block and all of its contents. Upon creation of a new block, the corresponding hash is calculated. Changing something inside the block will change the hash. This way, changes to blocks can be easily detected. Because the hash of the previous block is stored in a new block, a chain of blocks emerges. The first block of the chain is called the genesis block.

It is possible to tamper with a block (resulting in a new corresponding hash value) when all hashes of other blocks are recalculated as well (to make a valid chain again). To counter

¹A hash is the result of passing some data through a formula, resulting in a string of characters. Hash values generated by a formula are always the same length, regardless of the length of the input data.

this, blockchain makes use of a Proof of Work (**POW**) mechanism. **POW** is a cryptographic proof in which one party proves to others that a certain amount of computational effort is spent. **POW** slows down the creation of new blocks, making it difficult to tamper with blocks because **POW** needs to be recalculated for each block as well.

The security of blockchain depends on creative hashing, the **POW** mechanism, and the distributed nature of blockchain. Instead of a central entity, a public blockchain uses a peer-to-peer (**P2P**) network that anyone can join. Each node that joins retrieves a full copy of the blockchain. The node uses the copy to verify that everything is still in order. When a new block is created, it is sent to everyone on the network. Each node verifies the block to make sure it is not tampered with. The process of validating blockchain transactions is called mining. If everything checks out, each node adds the block to their copy of the ledger. All the nodes in the network create consensus: they agree on what blocks are valid and which are not based on a consensus mechanism. Invalid blocks will be rejected by other nodes in the network. To tamper with a block requires tampering with all blocks in the chain, redoing the **POW** for each block, and taking control of more than 50% of the **P2P** network. This is almost impossible to do.

The process of creating a new transaction to add to the blockchain is visualised in Figure 1.1 [111]. Blockchain technology allows mutually distrusting users to complete data exchanges or transactions without the need of any third-party trusted authority [69]. This means it is not required to trust in a third party such as a banking company. In fact, there is no central authority. Transactions can be requested using a private key, which is unique for every user and unknown to others, similar to a password.

A transaction is effectively an exchange of data, which can be conducted using a smart contract. Smart contracts are simple programs stored on the blockchain, used to automatically execute actions (for example exchange coins or medical records) based on certain conditions. A smart contract is similar to a traditional agreement between two parties but it does not require the involvement of a third party [100].

Once a new transaction is created and is written to a block, the nodes in the network decide through a consensus mechanism whether the newly mined block is valid, and if so, they subsequently update their ledger copies to append the block.

The history of any digital asset on the blockchain is transparent and it cannot be modified or deleted. In essence, blockchain technology offers a solution for increased trust, security, transparency, and the traceability of data.

1.2 Importance of blockchain security

While there are many blockchain applications, cryptocurrencies are especially popular. The number of people owning cryptocurrencies worldwide has been estimated to be over 300 million in 2021 [116] and 60% of blockchain uses was found to be in the financial sec-

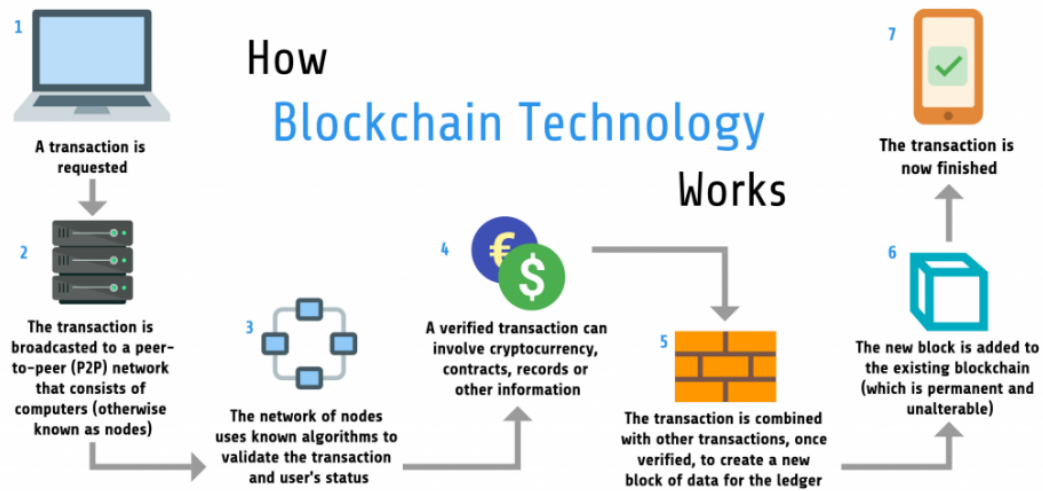


Figure 1.1: Transaction in a blockchain system (IP Specialist, 2022)

tor [14]. The direct link of cryptocurrency applications to money makes them an attractive target for criminals. Additionally, the fact that transactions are pseudonymous² makes it easy for criminals to get away with committing crimes. Numerous attacks have been successfully performed on blockchain applications. In the first nine months of 2019, losses resulting from digital currency crime were estimated to be 4.4 billion³ dollars (around 3.6 billion euros) [26].

Attacks are possible due to vulnerabilities in and related to blockchain applications. With the continual increase in cyber crime revolving blockchain applications, it is crucial that blockchain applications are securely implemented. This means blockchain applications must be thoroughly tested to discover and patch vulnerabilities.

Furthermore, blockchain applications are often open-source and decentralised: instead of a central authority that is responsible for safe-keeping the data, everyone is in charge of the data. This raises the question of who is (deemed) responsible when an attack does happen, and how this is influenced by the blockchain design.

These two aspects of blockchain security, software testing and security responsibilities, are the main subjects of this research.

²Pseudonymity is the near-anonymous state in which someone has a consistent identifier that is not their real name.

³1 billion = 1,000,000,000

1.3 Research aims

The first aspect of this research revolves around the development side of blockchain applications. Users and applications interact with the blockchain using an application programming interface (API), which makes the API a critical part of the blockchain software. Development of automated testing techniques such as fuzzing, contribute to maintaining secure blockchain applications. The following research aim is proposed:

RA1: The aim of this research is to gain insight into the effectiveness of grammar-based fuzzing strategies for JSON-RPC APIs.

The topic of grammar-based JSON-RPC fuzzing and the research approach are more elaborately introduced in Part II of this thesis.

The second aspect is the division of responsibilities when it comes to preventing attacks related to blockchain applications. Without a central authority, who is seen as, and who actually is, responsible for protecting the users of blockchain applications? How does the blockchain design influence this? The aim of this research is as follows:

RA2: The aim of this research is to gain insight into whom is held responsible for cyber security of blockchain applications and to evaluate how developers can influence the division of responsibilities.

The topic and research approach are more elaborately introduced in Part III of this thesis.

Developers are expected to deliver securely designed (and thoroughly tested) blockchain applications, but the responsibility for secure (use of) blockchain applications does not end with deployment and is not only on developers. The combination of the two aspects in this research serves as a step towards more secure blockchain applications and it provides insights into what responsibilities are expected to be carried by users and how developer's decisions regarding the blockchain design can influence this.

1.4 Thesis outline

This master thesis is an integrated project between two master tracks at the Delft University of Technology: Computer Science and Communication Design for Innovation. The thesis is split into three parts. Part I is a general introduction to both research projects.

Next, Part II consists of the Computer Science side of the project. This part focuses on the development of an automated testing tool for JSON-RPC APIs, the standard interface for blockchain applications like Bitcoin and Ethereum [59], to spot vulnerabilities before releasing new features. By implementing and evaluating grammar-based fuzzing

techniques that generate JSON-RPC test cases, the effectiveness of grammar-based fuzzing for blockchain systems is evaluated.

Afterwards, Part III consists of the Communication Design for Innovation side of the project. In this part blockchain user responsibilities are focused on. A content analysis was performed to investigate media framing of blockchain security responsibilities. Furthermore, experts were interviewed to gain insight into how such responsibilities are divided and what impact developers' decisions have on this division of responsibilities.

PART II

Computer Science

2

Introduction

Weaknesses in blockchain applications occur at the point where users interact with the blockchain [3]. Users and applications can interact with a blockchain through an Application Programming Interface (API). APIs enable software systems to make their functionality programmatically available to other programs or end-users [52]. An API takes requests from a user and tells a system what the user wants it to do, after which it returns the system's response to the request.

An API with security vulnerabilities can have a large impact on the applications that are making use of it [74]. Testing is thus crucial to ensure that APIs are behaving as intended. Part II of this thesis focuses on automated testing for APIs in JSON-RPC format.

In this chapter, in Section 2.1 the problem description that was briefly introduced in Chapter I is elaborated on. Afterwards, Section 2.2 presents the main research question and sub-questions. Next, Section 2.3 describes the contributions of this research. Finally, Section 2.4 describes the structure of the Computer Science research part of the thesis.

2.1 Problem description

In Chapter I it was stated that with the continuous increase in cybercrime revolving around blockchain applications, it is crucial that blockchain applications are thoroughly tested to discover and patch vulnerabilities. APIs are however growing in size. Bitcoin for example has 139 unique API operations [17]. To accomplish one task, generally a sequence of different API operations (requests) is required. API back ends usually function like state machines and requests are thus not used in isolation. This makes it necessary to not only test for individual requests, but also for sequences of them. This means that the number of required test cases increases exponentially with the size and complexity of APIs [52]. With APIs growing larger and larger, it has become impossible for developers to test APIs manually.

Fuzzing is a popular technique used to automatically find bugs and vulnerabilities in software systems [50] and is currently the most effective approach to efficiently discover vulnerabilities [68]. Fuzzing involves the generation of test cases that are fed into a target program to induce a crash. The simplest form of fuzzing is random fuzzing, where ran-

2. INTRODUCTION

domly generated inputs are fed to the target program (the program to be tested) in search of faults [90]. The source code of the program is not used to generate input, making it a black-box fuzzing approach. A technique to improve upon random fuzzing is to format input using a template (i.e. grammar-based fuzzing) [90]. In white-box fuzzing, the source code is instrumented to gather information on for example the program paths exercised by the inputs.

Figure 2.1 depicts an overview of the workings of a grammar-based fuzzer. Based on a prespecified grammar, the fuzzer generates well-formed input data. This input data is subsequently fed to the target program, which provides feedback on how the input was handled. The usage of feedback from the target system can guide the fuzzer in its process to generate useful test cases. The newest type of smart fuzzers are evolutionary fuzzers [96]. Evolutionary fuzzers evaluate what each input causes the program to do and change how they proceed based on that evaluation. During the past years, evolutionary testing research has reported encouraging results for automated black-box testing [123].

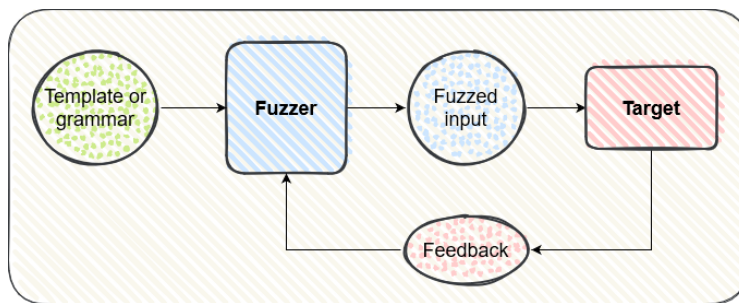


Figure 2.1: Flowchart of a grammar-based fuzzer

In 2010, almost half of web APIs were Remote Procedure Call (RPC) APIs and JSON was one of the main output formats [71]. Today, most distributed applications built use RPC [127]. This includes many blockchain applications like Bitcoin, Ethereum and Ripple, that use RPC encoded in JSON to connect to the blockchain. RPC is a simple and well-understood protocol for APIs that allows data to be exchanged between clients and servers. RPC-based APIs work well for performing actions (commands). In contrast to RESTful APIs, RPC-style APIs do not use HTTP methods (corresponding to Create, Read, Update, and Delete operations) directly to access resources. Instead, RPC defines its own operations that wrap the resource information and invokes these through one of the HTTP methods [71].

RPC APIs are usually components of a micro-services architecture, according to which each component should be small and assigned just one (or very few) responsibilities [52]. This results several small and simple distinct components, which are generally deployed in different containers that can be dynamically allocated and deallocated across different

hosts. This complex architecture makes it difficult to automatically test using white-box approaches. In cases like this, a black-box approach is more suitable since it is able to access the API through an interface, and does not need to cope with complex internal details of how (distributed) components are deployed and run.

Furthermore, white-box approaches require access to source code. The source code of RPC API systems is not always available, for example due to commercial third-party libraries that are part of the implementation.

Another limitation of white-box heuristics is that they need to be re-implemented for every programming language, whereas black-box heuristics are independent from the programming language used to implement the API.

To the best of our knowledge, there are no open-source fuzzing frameworks available that can be used for (black-box) system-level testing for JSON-RPC APIs. Furthermore, the effectiveness of grammar-based fuzzing techniques on JSON-RPC APIs specifically has not been evaluated. This research aims to evaluate grammar-based fuzzing techniques for JSON-RPC APIs, specifically evolutionary fuzzing.

2.2 Research aim and questions

The research aim as was presented in Chapter I is repeated before stating the research question.

RA1: The aim of this research is to gain insight into the effectiveness of grammar-based fuzzing techniques for JSON-RPC APIs.

We define the effectiveness of tests as their ability to discover faults in the tested system. This is statistically correlated to structural coverage [60]. Structural coverage (or code coverage) is the extent to which the source code of a program is executed when a particular collection of test cases is run. A higher structural coverage indicates a larger possibility of discovery of faults [138]. The research aim is therefore reframed into the research question as follows:

How effective is grammar-based JSON-RPC fuzzing to achieve structural coverage?

The main question is divided into two sub-questions. The first step of this research is to develop a black-box tool that automatically generates system-level test cases for JSON-RPC APIs using a grammar (i.e. a template for input). With this tool, a simple grammar-based fuzzing approach and an evolutionary fuzzing approach can be evaluated.

We aim to evaluate whether we can improve upon grammar-based fuzzing by combining it with an evolutionary algorithm that makes use of program feedback in order to construct promising input for test cases. This leads to the following sub-question:

1. How effective is evolutionary fuzzing with regards to structural coverage in comparison to grammar-based fuzzing for JSON-RPC APIs?

The use of program feedback helps guide the search for good test cases. The evolutionary algorithm evaluates how well a test case is based on the received program feedback on that test. This is done using a fitness function. The design of the fitness function is a crucial part of the evolutionary fuzzing approach. We want to evaluate how different fitness functions compare to each other with respect to the performance of the evolutionary fuzzer. This translates into the second research sub question:

2. How do different fitness functions for the evolutionary fuzzing approach compare with regards to structural coverage?

The answers to these questions provide insight into how effective grammar-based fuzzing is for JSON-RPC APIs and whether the addition of an evolutionary algorithm (with a suitable fitness function) can aid grammar-based fuzzing approaches in successfully testing JSON-RPC APIs.

2.3 Contributions

In this study, two approaches are evaluated to find faults and achieve code coverage in software systems that implement a JSON-RPC API: a simple grammar-based fuzzing approach and an evolutionary fuzzing approach. This research project includes the following contributions in the context of automated testing of JSON-RPC APIs:

- **An open-source and grammar-based framework for automated black-box test case generation and execution for JSON-RPC APIs**, which can be easily extended to include additional heuristics and can be used on any target system supporting a JSON-RPC API, given an OpenRPC specification.
- **A comparison of grammar-based fuzzing and evolutionary fuzzing on two commercial and widely used APIs**, showing the potential of both techniques to achieve structural coverage.
- **A replication package of the study**, which is publicly available on GitHub, enabling others to replicate the results from this study.

2.4 Outline

Part II of the thesis consists of Chapters 2 to 10. In the current chapter, Chapter 2, the research topic is introduced and the research questions are presented. In Chapter 3, the reader is provided with background information on JSON-RPC APIs and (evolutionary) fuzzing. Related work on automated API testing is reviewed in Chapter 4. Chapter 5 introduces the approach that was used to answer the research questions.

Chapter 6 goes into detail on the final implementation of the tool. The way in which the tool's performance is empirically evaluated is explained in Chapter 7. The results of the experiments and the corresponding analysis are presented in Chapter 8. Afterwards, the research is discussed in Chapter 9. Finally, in Chapter 10, the research question and accessory sub-questions are answered, and some suggestions for future work are given.

3

Background

The purpose of this research is to evaluate the effectiveness of evolutionary fuzzing for JSON-RPC APIs. In order to perform this evaluation, we design a black-box tool that adopts a grammar-based and evolutionary fuzzing approach to generate test cases for JSON-RPC APIs. In this chapter an introductory background to (RPC) APIs, API testing, search algorithms, and (evolutionary) fuzzing is provided.

3.1 APIs

In Chapter 1 it was explained how blockchain systems work and an overview of a blockchain system was presented in Figure 1.1. This overview did however not specify where the transaction that starts the chain of actions in the blockchain system comes from. To request a transaction, contact should be made with the blockchain system. This is generally done through a web Application Program Interface (API). To complete the overview, an extra part was added to the visualisation of the blockchain system to include the API, resulting in Figure 3.1.

The API acts as an intermediary between two systems. System A communicates with System B through the API. It does so by sending a request to the API, and waiting for a response from the API. In our case, System B is a blockchain system, and System A could for example be a blockchain wallet application that is requesting to transfer cryptocurrency to another account on behalf of a user, or it could be a user directly communicating with the API. Requests are not limited to transactions. Several functionalities of blockchain systems can be accessed through requests. More examples are asking for information about the blockchain ledger, previous transactions, or the current balance of an account.

Without communicating to the blockchain system, applications cannot access information and make transactions with the blockchain ledger. APIs allow other services to integrate the functionalities of the blockchain system from within their applications or websites.

3. BACKGROUND

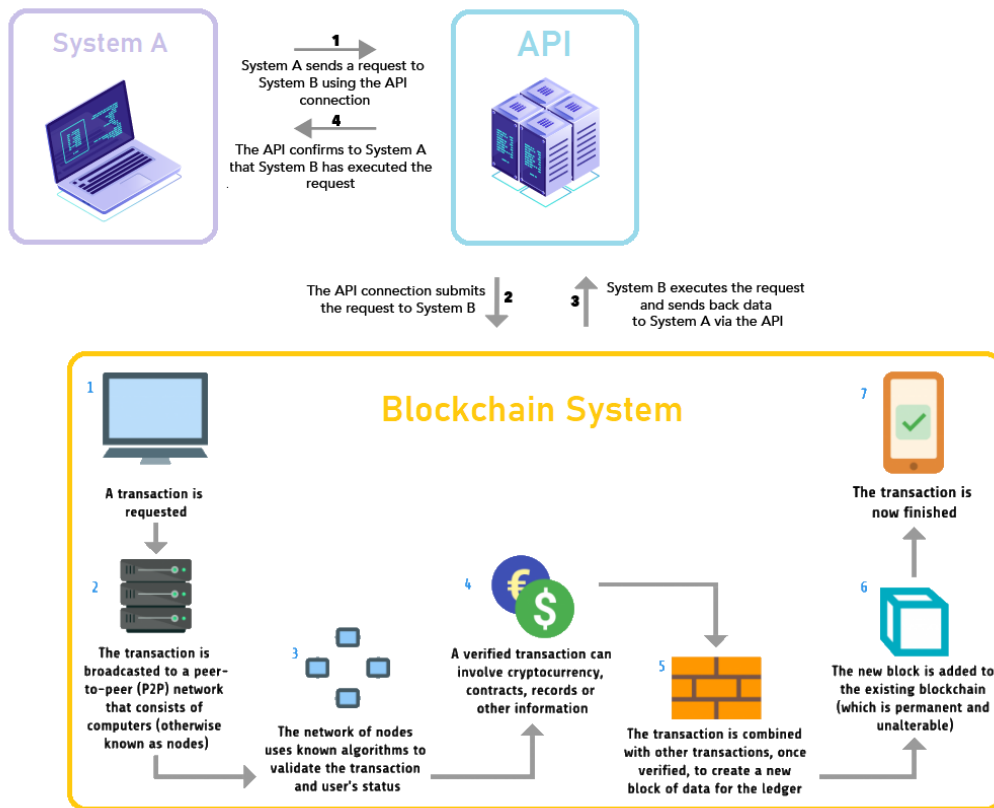


Figure 3.1: The role of APIs in blockchain systems.

3.1.1 HTTP messages

Communication between an API and the server of a system happens through Hyper Text Transfer Protocol ([HTTP](#)) messages. An example of an HTTP request and response message is shown in [Figure 3.2 \[44\]](#). In this example, the request message asks to retrieve a text file called *hi-there* and the server responds with a message containing the contents of the requested file in the body. An HTTP message can either be a request or response message and consists of a start line, zero or more header fields, and an optional body.

The start line indicates the HTTP method for the request message or what happened in the form of a status code for the response message. HTTP defines methods to indicate the action to be performed on a resource. These methods map loosely to the resource related operations of Create (POST), Retrieve (GET), Update (POST/PUT) and Delete (DELETE). The common HTTP methods and their corresponding actions are listed in [Table 3.1 \[76\]](#).

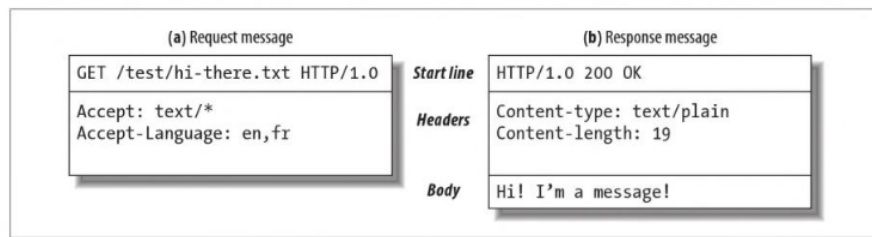


Figure 3.2: An HTTP request and response message.

Type	GET (read)	POST (create)	POST (action)	PUT (create)	PUT (update)	DELETE
Action	Retrieve resource instance	Create new resource instance	Trigger action	Create resource instance at the given path	Update resource instance at a given path	Delete resource instance

Table 3.1: Resource related HTTP methods.

The server always returns a HTTP status code, which provides information about what happened with the request after it reached the server. The status code indicates whether the request was received, successfully handled, redirected or whether there was a client or server error. There are five main categories of status codes. The different classes of status codes are listed in Table 3.2.

Class	Meaning	Description
1XX	Informational	The request was received, continuing process.
2XX	Success	The request was successfully received, understood, and accepted.
3XX	Redirection	Further action needs to be taken in order to complete the request.
4XX	Client error	The request contains bad syntax or cannot be fulfilled.
5XX	Server error	The server failed to fulfill an apparently valid request.

Table 3.2: The classes of HTTP status codes.

The header of an HTTP message allows the client or the server to pass additional information about the request or response and the body of the message. In the example in Figure 3.2, one of the header fields given in the response message is the length (number of characters) of the body of the message. The body of an HTTP message contains (any type of) data. In the example given in Figure 3.2 the body contains the data from the file that was requested.

3.1.2 REST and RPC APIs

There are differences in how various types of web APIs work with HTTP requests. The most used architectural styles for web APIs are Representational State Transfer (**REST**)

3. BACKGROUND

and Remote Procedure Call (RPC) [110]. To illustrate the difference between the two API types, the same request for both REST and RPC APIs is presented in Listing 3.1 and 3.2 respectively. Nowadays the most popular format used for messages from and to web APIs is JSON [78], which is a format readable by both humans and machines.

```
POST /users/100/messages HTTP/1.1
Content-Type: application/json

{"message": "Hello!"}
```

Listing 3.1: REST request in JSON

In a request to a REST API, the start line of the request specifies an HTTP method and a target resource to apply the method on. REST APIs support Create, Read, Update and Delete (CRUD) operations to manipulate data. These translate to the POST, GET, PUT and DELETE HTTP methods. The target resource is identified by a Uniform Resource Identifier (URI), which is essentially a path to a resource. In this example, the request aims to say hello to a user with id 100 by adding a message to the user's messages.

While resource oriented APIs like REST APIs expose internal data through a simple document-processing interface that is always the same (using URIs), RPC APIs exposes internal functionalities through a programming-language-like interface that is different for every service [94]. RPC is the most straightforward form of an API which allows communication with a server in order to remotely execute code. It is essentially a function call, but between two different systems.

In the request to an RPC API, instead of an URI, a function named *SendUserMessage* is specified. The parameters used by this function are defined in the body of the request. RPC APIs define their own functions to access resources and invoke these through POST and GET operations [71], with GET being used to retrieve information and POST being used for everything else.

```
POST /SendUserMessage HTTP/1.1
Content-Type: application/json

{"userId": 100, "message": "Hello!"}
```

Listing 3.2: JSON-RPC request

Because blockchain systems maintain immutable records, they have no need for the CRUD operations that resource oriented APIs provide, making the generic RPC API a more suitable option. Most blockchain systems use RPC APIs to interact with users [137].

3.2 API testing

API testing is a necessity to ensure functionality, reliability and security of the applications that are communicating with the API. Manual API testing is time consuming and error

prone [108]. Especially for large organisations and time-bounded projects this may prove to be infeasible. Automated API testing is fast, minimises errors, and does not need the physical presence of a tester. With automated API testing, important test cases for an API are automatically generated. Test cases for APIs are (sequences of) requests. Tests are generally related to one of the following categories [108]:

1. **Input validation**

The API should be tested using different input parameters and the response should be verified to ensure that the HTTP status code and contents of the response message are fully correct for each set of input parameters. Invalid input parameters should for example result in a 400 HTTP status code.

2. **JSON format validation**

It should be verified that the JSON response message of the API is correctly structured.

3. **Business Logic**

The API should perform the task that it is supposed to do correctly. Data sent by the API as a response to a request should be correct and up-to-date.

4. **Negative test cases**

The API should be able to handle incorrect or invalid parameters, missing or extra values and null values for mandatory fields. Furthermore, large amounts of payload data, special and non-ASCII characters, long strings and integers, and incorrect data types for parameters in requests should not result in unexpected behaviour from the API.

5. **Reliability tests**

It should be tested whether the API can consistently return correct responses.

6. **Call sequencing checks**

When a request has resulted into the modification of data, the firing of an event or a call to another API, requests that follow afterwards should function correctly, taking what previously happened into account.

7. **Security testing**

Unauthorized users should not be able to access and use sensitive code and associated features.

HTTP requests and responses are created by the systems that make use of an API to communicate with each other. The testing categories above are aimed at how the receiving system handles requests. By sending requests and evaluating the response from the receiving system, defects in the code of the receiving system can be found.

3.3 Search algorithms

To find (combinations of) HTTP requests that make good test cases, the solution space should be searched for solutions that trigger new regions of the system’s code. For most systems, it is infeasible to perform an exhaustive search on input domains since they are often large and multi-dimensional. In cases like this, a heuristic can be used to efficiently find solutions. A heuristic function can determine how good a given solution is. This ”goodness” is typically called the *fitness* of an individual, and is estimated by the heuristic that is the *fitness function*. Because heuristics essentially estimate what good solutions are, there is no guarantee that optimal solutions are found.

One of the earliest search techniques and the simplest local search algorithm is **Hill Climbing** [22]. Hill climbing starts with an arbitrary solution to a problem and then climbs the solution space in search for a fitter solution. It does so by repeatedly selecting the local move (one of the neighbour solutions) that leads to the largest improvement of fitness [107]. When no local move can further improve the solution, the search is terminated and a local optimum is reached. While hill climbing is very efficient, it only works well when the neighbourhood is well defined and not too large [134]. Hill climbing is always vulnerable to landing on a local maximum rather than a global maximum. Additionally, it requires fitness functions to be continuous and injective (changing output for any change in the input) [124].

A more advanced version of hill climbing is **Stochastic Hill Climbing**, which allows for global search. Where hill climbing considers only direct neighbours of solutions, stochastic hill climbing mutates the candidate solution and compares the fitness of the mutated solution to the original solution. It then continues the search with the solution that obtained the highest fitness. In theory, every solution in the solution space can be reached through repeated mutation (hence the global search). Mutations can however not be too large. If a solution is mutated in such a way that it is completely replaced, it would result in random search. Instead, most of an individual’s features should be maintained.

Another name for stochastic hill climbing is the (1+1) Evolutionary Algorithm (EA) [20, 33]. EAs are loosely based on biological systems. They maintain a population of individuals that undergo mutation. Individuals with low fitness are gradually eliminated from the population, while high fitness individuals persist. A population emerges of individuals with increasingly higher fitness values. Stochastic hill climbing is a special variant of EAs where the population consists of a single individual that produces one new mutation each generation.

The best known form of EAs is the **Genetic Algorithm (GA)** [134]. A GA encodes problem solutions as a chromosome, which is a sequence of genes. Each gene encodes a trait of the individual. In the context of this research, a trait may be the HTTP method of the request. Through crossover and mutation new individuals emerge in the population, which are evaluated using the fitness function. Similar to stochastic hill climbing, every solution

in the solution space can be reached through repeated mutation [77].

3.4 Fuzzing

Fuzzing is an automated software testing technique that generates test cases by providing invalid, unexpected or random data as input to a computer program [15]. Fuzzing can expose potential attacker entry points such as access violations, unhandled exceptions or buffer/ integer overflows. Essentially, fuzzing is based on the idea of feeding (random) data to a program to induce a crash. Fuzzers can make use of search algorithms to guide the search for test cases. Fuzzers can be classified in various ways. Categories relevant for this research are explained in the following sections.

3.4.1 Smart and dumb fuzzing

Traditional fuzzing relies on randomness. It blindly changes existing input values, without an understanding of the structure of the input data, hence the name "dumb" fuzzing. **Dumb fuzzing** uses either random inputs or random mutations of valid inputs, and due to this may miss security violations that rely on unique corner-case scenarios [15].

Smart fuzzing on the other hand uses "smart" input generation for fuzzing based on domain-specific knowledge of the target program. While it takes effort up front to understand the program design and testing speed is slower, smart fuzzing can provide a greater coverage of security attack entry points compared to random fuzzing as it is likely to fulfil the programs' data structure requirements. State-of-the-art fuzzers generally employ a smart fuzzing strategy [68].

3.4.2 Mutation-based, grammar-based and evolutionary fuzzing

A naive fuzzer generates completely random input and feeds it to a targeted program. There are three types of fuzzing techniques that improve upon a naive fuzzer, namely mutation-based fuzzing, grammar-based (also called generation-based) fuzzing and evolutionary fuzzing [96]:

- **Mutation-based fuzzing** generates new variants of input data based on existing input data samples. It blindly mutates existing data samples to create new input. Mutation-based fuzzers are not aware of the expected input format, and as a result they cannot select mutations in a smart way. A mutation-based fuzzer is an example of a dumb fuzzer. Grammar-based fuzzing and evolutionary fuzzing are examples of smart fuzzing.
- **Grammar-based fuzzing** use a model of the input data to generate input data from scratch. For a grammar-based fuzzer, knowledge of program input is required as

test cases are generated based on the expected input format. While grammar-based fuzzing requires a priori knowledge and is thus difficult to start compared to mutation-based fuzzing, it is able to pass the validation of programs more easily and is more likely to test deeper code of target programs and achieve greater coverage [68].

- **Evolutionary fuzzing** is the newest type of fuzzing. It builds on mutation-based fuzzing by selecting some inputs over others to mutate, using an evolutionary (search) algorithm. Evolutionary fuzzers generally mutate an input sample or select two or more input samples and perform crossover, combining components of the selected inputs to create a new test case. Evolutionary fuzzers evaluate for each input the behaviour of the program and change how they proceed based on that evaluation. In practice, inputs are ranked using a fitness function (often code coverage) and the fittest inputs are selected to be mutated [96]. Evolutionary fuzzers, unlike mutation- and grammar-based fuzzers, require feedback about how well test cases performed.

Grammar-based fuzzing can explore all possible specified inputs, which is a finite number of inputs corresponding to the specification. In contrast, mutation-based and evolutionary fuzzing can in theory generate an infinite number of inputs, as they can deviate from input specifications.

3.4.3 White-, grey-, and black-box fuzzing

Fuzzers can be classified as white-, grey- or black-box with respect the dependence on program source code and the degree of program analysis [34, 68]. Mutation- and grammar-based fuzzers are forms of black-box fuzzing [19].

Black-box fuzzers have no knowledge about the internals of the target program (and do not require access to source code). They apply random input generation.

White-box fuzzers use information obtained from analysis of the source code and program instrumentation to generate inputs for test cases. They require access to the source code of the target program. Program instrumentation enables the receiving of informative messages about the execution of an application at run time. For example, instructions in the program may be for logging information to appear on screen.

Grey-box fuzzers work without source code and collect feedback on the generated inputs solely through program analysis. This information is used to guide the mutation process. Grey-box fuzzing reduces the overhead significantly compared to white-box fuzzing [34]. Black-box and grey-box fuzzers typically have a hard time generating valid inputs due to the lack of program knowledge [68]. Using a grammar solves this issue.

3.4.4 Directed and coverage-based fuzzing

To explore the target program, fuzzers employ the strategy of directed fuzzing or coverage-based fuzzing. **Directed fuzzing** aims to generate test cases that cover specific target code

of the program. **Coverage-based fuzzing** aims to generate test cases that cover as much code of the program as possible. Code coverage indicates the number of code paths covered during testing. Coverage-based fuzzers are expected to test more thoroughly and to detect more bugs [68].

4

Related Work

In this work we focus on grammar-based evolutionary fuzzing for JSON-RPC APIs. Chapter 3 established a background on web APIs, search algorithms, and fuzzing. In this chapter, previous work on black-box fuzzing and evolutionary fuzzing is reviewed to establish what has already been done and to define the research gap that this study should fill.

4.1 Black-box fuzzing for web APIs

As described in Section 3.1.2, and are the most used architectural web API styles. Although to the best of our knowledge, (evolutionary) grammar-based fuzzing for JSON-RPC APIs specifically has been largely unresearched, various fuzzing techniques have been employed and analysed for RESTful APIs. Currently seen as state-of-the-art RESTful API fuzzing tools are RESTler, RestTestGen, RESTest, bBOXRT and EvoMaster [30, 136]. All are grammar-based fuzzers. This section discusses the core elements of grammar-based fuzzing for the various fuzzing tools: the grammar, valid test case generation, mutation for invalid test case creation, and lastly, target system feedback.

4.1.1 Grammar

All of the above mentioned research tools require a specification of the interface to interact with the API. This specification forms the grammar for the fuzzers. The general description format for RESTful APIs is OpenAPI Specification (OAS). The OAS defines a standard interface to RESTful APIs which allows both humans and computers to understand the capabilities of a service without access to source code or documentation, or through network traffic inspection [113]. The OAS includes a list of available operations and the format of the corresponding data of requests and responses. The OAS itself is a JSON object.

Listing 4.1 illustrates an example of the specification of an API operation named *getPetsById* [113]. The method *getPetsById* requires just one parameter: an array of strings, containing the ids of the Pet objects that the user wants to retrieve. The response from the server to a valid HTTP request is an array of Pet objects.

4. RELATED WORK

```
{
  "get": {
    "operationId": "getPetsById",
    "description": "Returns pets based on ID",
    "responses": {
      "200": {
        "description": "pet response",
        "content": {
          "*/*": {
            "schema": {
              "type": "array",
              "items": {
                {
                  "type": "object",
                  "properties": {
                    "name": {
                      "type": "string"
                    },
                    "petType": {
                      "type": "string"
                    }
                  }
                },
                "required": [
                  "name",
                  "petType"
                ]
              }
            }
          }
        }
      }
    },
    "parameters": [ {
      "name": "id",
      "in": "path",
      "description": "ID of pet to use",
      "required": true,
      "schema": {
        "type": "array",
        "items": {
          "type": "string"
        }
      }
    } ]
  }
}
```

Listing 4.1: The OpenAPI Specification for the *getPetsById* operation

Based on the specification that specifies how input should be structured, syntactically valid HTTP requests can be created and sent to the server. Furthermore, it is possible to compare the response from the server with the specified response format to see if they match.

```
POST /users/100/messages HTTP/1.1
Content-Type: application/json

{"id": 2}
```

Listing 4.2: REST request in JSON

4.1.2 Valid request generation

The grammar provides the fuzzer with knowledge on how to construct valid requests but there are different ways to generate requests and corresponding values. RESTler [11] aims to achieve full-grammar coverage, using search techniques like RandomWalk or Breadth-First Search to cover all (dependent) sequences of requests. It makes use of a predefined dictionary to replace parameter values in the grammar (e.g. 0 and 1 for integer values, "sampleString" and an empty string for string values, and "true" and "false" for boolean values).

To create meaningful request sequences, RESTler makes use of *producer-consumer* dependencies among requests. By doing this, the order of execution of requests can be determined. If a variable is produced by request A (as seen in the response values in the OAS), and this variable is required as input for request B, request A should be executed before request B.

Similarly, RESTTESTGEN [119] constructs an *Operation Dependency Graph* to model data dependencies among operations. The graph is updated with values when test cases are generated to dynamically decide when a new operation can be tested (when the required input data is known). Input values are selected by reusing observed data stored in a response dictionary or by generating a new parameter value based on the specification schema (default and example values, specified enumeration values or random input generation).

REStest [74] can only generate test cases containing one operation. It uses custom test data generators to generate input data. These test data generators automatically generate realistic data by (1) extracting values from knowledge bases like DBpedia, (2) reusing values observed in previous API responses; or (3) leveraging manually-defined domain-specific generators (e.g., strings conforming to a regular expression) or data dictionaries.

REStest assigns random values (or non-random values if a test data generator is configured for each parameter) to each parameter for the API operation under test. Furthermore, it evenly distributes test cases within the input space to cover more API functionality. Lastly, REStest addresses the negligence of constraints among input parameters (inter-parameter

dependencies). It does so by relying on constraint programming solvers to support inter-parameter dependencies (i.e. parameters in a request that depend on each other). To enable this, RESTTest integrates IDLReasoner, a library for the automated management of inter-parameter dependencies in RESTful APIs. Inter-parameter dependencies must be defined as a part of the [OAS](#).

bBOXRT [63] is a simple rule-based fuzzer. When generating input, it follows the [OAS](#) to construct valid test cases, while also applying rules to inject faults in such valid requests. Rules that were found to be effective are for example replacing values with an empty value, replacing values with *null*, appending an SQL injection attack to the original input, and inserting random characters at random positions.

Whereas the other tools implement only a black-box approach, EvoMaster [9] has a black-box as well as a white box approach. Both generate random valid inputs according to the grammar defined by the [OAS](#). Using an evolutionary approach, EvoMaster views each test case (which consists of a sequence of one or more HTTP calls) as an individual, which is part of a generation of individuals.

4.1.3 Invalid request creation

To create invalid test cases, valid requests are mutated. Nominal test cases test the API under valid inputs (those conforming to the API specification). Faulty test cases investigate how the API handles invalid inputs, i.e., they expect a client error as a response.

The state-of-the-art research tools all start by generating valid inputs. They then continue by mutating the valid inputs in such a way that faulty test cases emerge. Such error execution scenarios use input data that violate the interface constraints in order to expose defects and unhandled exceptions. RESTler does not generate any intentional invalid test cases. RESTTESTGEN, bBOXRT, RESTTest and EvoMaster do.

RESTTESTGEN [119] uses three simple types of mutation operators to change test cases: removing required parameters, using wrong parameter input types and violating parameter constraints (e.g. exceeding the limit of a parameter value).

bBOXRT [63] follows various predefined rules to inject a single fault in each request. A few rules that were found to be successful in triggering erroneous behaviour are: replacing parameters with empty values, null or random character strings of an equal length; appending an SQL injection attack or random characters to the original input and inserting random characters at random positions.

REStest [74] produces invalid test cases by among other techniques violating inter-parameter dependencies, mutating parameter values, removing parameters, adding parameters, and mutating the [JSON](#) format.

Throughout the search for good test cases, the white-box approach of EvoMaster [9] mutates previously generated test cases either by modifying their structure (i.e., adding or removing HTTP calls) or the data (i.e. parameters) of one specific HTTP request. The black-box approach does not do this and only generates valid requests according to the [OAS](#).

4.1.4 System feedback

All research tools except for bBOXRT analyse to some extent the responses observed during prior test executions in order to select useful tests and generate additional test cases.

REStler [11] uses the responses to learn whether certain request sequences result into a refused by the service. If so, this combination will be avoided in the future (invalid sequences are eliminated). The authors found that this dynamic feedback decreases the number of 4xx response codes and increases coverage.

Likewise, REStTESTGEN [119] analyses the status codes returned by the system to assess if a test case was successfully created. Test cases with status codes 2xx and 5xx are omitted since these represent nominal test cases (2xx) and test cases that reveal implementation faults (5xx). On top of that, REStTESTGEN compares the actual response (observed at execution time) to the intended response syntax (as documented in the [OAS](#)) to expose a potential mismatch between the two.

REStest [74] works in a similar way but focuses only on exposing erroneous behaviour. It evaluates test cases by analysing the status codes. A test case is found to expose erroneous behaviour: if the status code is 5xx (server error), if the status code is 2xx while the request violates the specification of individual parameters, if the status code is 2xx while the request violates one or more inter-parameter dependencies, or if the status code is 4xx while the request conforms to the [OAS](#).

The black-box approach of EvoMaster [9] creates targets that should be covered based on the expected input and output from the [OAS](#). The set targets aims to maximise HTTP status coverage so that an endpoint (API operation) is exercised in different ways: a success call (status code 2xx), a server failure (status code 5xx), or user error (status code 4xx). Like REStTESTGEN and REStest, EvoMaster also checks if the responses of the system conform to the [OAS](#).

4. RELATED WORK

The white-box technique of EvoMaster evolves test cases with an evolutionary algorithm to maximise the effectiveness of the final test suite. The fitness function is based on various objectives: status code coverage, line coverage, branch coverage, and detected faults. The test coverage metrics can be used due to instrumentation of the system under test.

4.1.5 Overview

Testing a RESTful API at the system level involves generating HTTP requests and asserting their responses. A test case comprises one or more requests. In RESTest, a test case represents a single call to an API operation and a set of assertions in the response. The other tools use multiple requests per individual.

All research tools were evaluated on a variety of RESTful services and managed to find software defects, mostly indicated by 5xx HTTP status codes or disconformities with the OAS. Table 4.1 presents an overview of the techniques used for valid request generation, mutation of test cases, and the feedback of the server that is used in order to find useful test cases.

	Valid request generation	Mutation for invalid request creation	System feedback
<i>RESTler</i>	Dictionary Request dependencies	N.A.	Status codes (prune invalid sequences)
<i>RESTTESTGEN</i>	Response dictionary Domain-specific generator Request dependencies	Missing required parameters Wrong parameter input types Parameter constraint violations	Status codes Response validation
<i>RESTest</i>	Knowledge bases Response dictionary Domain-specific generator/dictionary Inter-parameter dependencies	Removing or adding parameters Mutation of parameter values Mutation of JSON format Inter-parameter constraint violations	Status codes Response validation
<i>bBOXRT</i>	Domain-specific generator	Various rules to inject faults (e.g. inserting null values or SQL attacks)	N.A.
<i>EvoMaster (black-box)</i>	Domain-specific generator	N.A.	Status codes Response validation

Table 4.1: Overview of techniques used by state-of-the-art RESTful API fuzzing tools

Other studies adopt similar techniques. Ed-Douibi et al. [35] propose an approach focused on generation of test cases for REST APIs to ensure that APIs meet the requirements as defined in their specifications. Input for method parameters is generated in one of three ways: (1) simple parameter value inference, where examples, default values and enumeration values are used from the specification; (2) dummy parameter value inference, where a dummy value is used that respects the expected type; (3) complex parameter value inference, where the value of the parameter is inferred from the response of an operation. Two

rules are used to test whether APIs match their specification: the nominal test case definition (request complies with schema, should result in 200 status code) and the faulty test case definition (following from missing required key, wrong data types or violated constraints, should result in a 400 status code).

A different tool to validate responses is QuickRest [56]. Like the other tools, QuickRest automatically produces tests for RESTful APIs from the OAS. The tool can be used as a property-based test generator and as a source of validation for results. The main idea of property-based testing is to generate input data and to check if defined properties hold when exercising the system with that input.

As for the best strategy to generate input data, Godefroid et al. [43] conducted a study on how to intelligently generate input data embedded in RESTful API requests in order to find data-processing bugs in cloud services. They observed that static type-value mapping (i.e. mapping each parameter type to a single value) does not address the lack of client-specific, domain-specific or run-time dependent information. Furthermore, example values from the OAS do not help to discover client-specific and run-time dependent information either, although they do help to discover more error types. The authors conclude that in order to cover more error types, besides utilising example values, it is beneficial to re-use values found in responses to previous requests.

4.2 Evolutionary fuzzing

In order to specialise an evolutionary algorithm for a specific problem, one needs to define a problem-specific objective (fitness) function. The objective function allows for comparison between solutions of the search with respect to the search goal (e.g. finding faults in software). Using this information, the search is directed into potentially promising areas of the search space [123].

Evolutionary fuzzers typically make use of code coverage to compute the fitness of a solution (i.e. the quality of a test case). The white-box approach of EvoMaster [9] defines the fitness function in terms of code coverage (statement and branch coverage) and fault detection (5xx status codes).

Another evolutionary fuzzing approach is proposed by Sahin and Akay [97]. The authors created an Artificial Bee Colony (ABC) algorithm for RESTful test suite generation. Like EvoMaster, the ABC algorithm uses code coverage (statement and branch coverage) and the number of server errors (5xx status codes) to calculate the fitness of a test suite. In addition, it tries to minimise the number of test cases (i.e. a smaller test suite should be preferred over a larger test suite that achieves the same code coverage).

Calculating the fitness of a solution using code coverage is a logical choice since high coverage results in a larger probability of discovery of faults [138]. However, in a black-box context code coverage cannot be computed since there is no access to source code. To still be able to guide the search towards promising areas, a different way to compute the

fitness of a solution is required. The input and feedback retrieved from the system contain important information that can be analysed to find a similar measure to code coverage. New inputs and unseen responses from the target system can be a good indicator for the discovery of new code paths (additional code coverage).

4.3 Research gap

The related work illustrates the effectiveness of grammar-based fuzzing for RESTful APIs to generate tests. Even a simple rule-based fuzzer was able to perform well and detect faults. To the best of our knowledge there have not been any studies that measured the effectiveness of grammar-based fuzzing approaches for JSON-RPC APIs, which are action-oriented rather than resource-oriented. In addition, while evolutionary algorithms have been applied for white-box fuzzing approaches, it is not known whether and how evolutionary algorithms can help guide the search for useful test cases in a black-box fuzzing context (without access to code coverage).

In this thesis research, we focus on automatically generating test inputs for components without source code (black-box). We implement and evaluate a novel language-independent evolutionary grammar-based fuzzing tool for JSON-RPC APIs to detect defects and unwanted behaviour.

5

Approach

The aim of this research as described in Chapter 2 is to evaluate the effectiveness of evolutionary fuzzing for JSON-RPC APIs compared to grammar-based fuzzing. To answer the posed research questions, we develop an automated testing tool that generates test cases for JSON-RPC APIs based on a grammar. The search process for test cases is guided by an evolutionary algorithm. The heuristic functions designed in this research aim to favour test cases that are likely to be close to discovering new paths in the target system. This chapter explains the strategy of the tool (and its heuristic functions) to perform grammar-based evolutionary fuzzing.

A high-level overview of the fuzzing strategy of the tool is presented in Section 5.1. The remaining part of this chapter, Section 5.2 up to Section 5.6, elaborates on the various components of the fuzzing tool.

5.1 Building a fuzzing tool

The outline of the fuzzing approach is demonstrated in Figure 5.1. The approach consists of two stages: the preparation stage and the fuzzing stage.

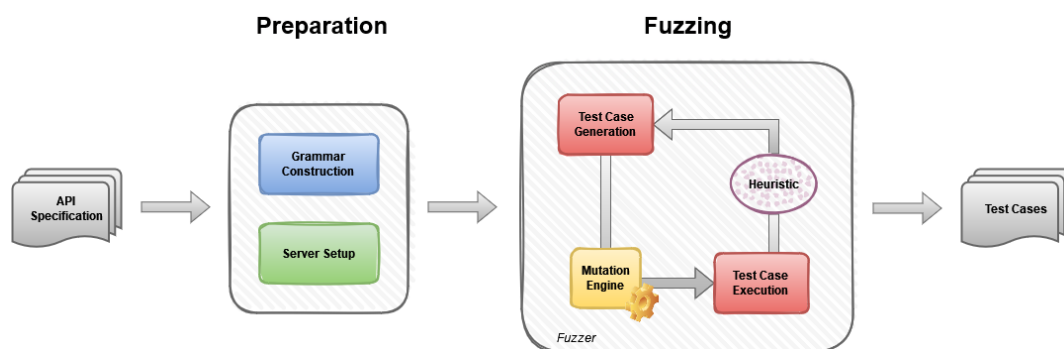


Figure 5.1: Fuzzing process

In the preparation stage the grammar is constructed and the API server of the System Under Test (SUT) is set up and prepared for testing. The fuzzing stage consists of test case gener-

ation and execution, as well as the selected heuristic (fitness function) for the evolutionary fuzzer, which makes decisions on which of the generated test cases will be evolved. The mutation engine holds all operators to mutate test cases.

In the first cycle, the fuzzing process starts with the generation of a (specified) number of test cases based on the grammar. A test case consists out of one or more HTTP requests to the **SUT**. These test cases are the initial population of individuals. In Section 3.4.2, the concepts of grammar-based fuzzing and evolutionary fuzzing were explained. In this research we want to compare evolutionary fuzzing to traditional grammar-based fuzzing. The tool is therefore able to use both approaches. In the grammar-based fuzzing approach, individuals are not mutated or selected to continue to the next cycle. Instead, each cycle (or generation), new individuals are generated based on the grammar.

For the evolutionary fuzzing approach, individuals in the population are mutated, resulting in a group of new individuals (the offspring) of the same size as the initial population. All tests (from the initial and mutated population) are then executed against the **JSON**. Based on the response from the **SUT**, the heuristic makes a selection of test cases to form the next generation, which will be mutated again. The population size remains the same during the cycles, meaning only half of all evaluated tests in each cycle are retained.

Until some predefined termination criteria (e.g. a maximum number of iterations/generations or a time limit) is met, the fuzzing stage is looped through multiple times. This results into several generations of individuals. The different parts of the approach are discussed in more detail in the next sections.

5.2 Grammar construction

Black-box fuzzers generally require a grammar for the **SUT** to generate test cases. This can be constructed from a provided specification of the API operations. As described in Chapter 4, state-of-the-art black-box fuzzers for RESTful APIs make use of a standardised format to describe the APIs: the OpenAPI specification (**OAS**). This format however is specific to RESTful APIs.

For JSON-RPC APIs, a similar format has emerged. The OpenRPC Specification defines a standard, programming language-agnostic interface description [82]. OpenRPC specifications document the operations of JSON-RPC APIs. An example of the OpenRPC specification of an API operation is presented in Listing 5.1 [40]. The API operation named *get_pet_by_id* requires one parameter: a string representing the id of the pet to retrieve. The response from the server contains an object containing two strings: the name and tag of the pet, and an integer value: the id of the requested pet.

```

{
  "name": "get_pet_by_id",
  "description": "Returns a user based on a single ID, if the user
does not have access to the pet",
  "params": [
    {
      "name": "id",
      "description": "ID of pet to fetch",
      "required": true,
      "schema": {
        "type": "integer"
      }
    }
  ],
  "result": {
    "name": "pet",
    "description": "pet response",
    "schema": {
      "allOf": [
        {
          "type": "object",
          "required": [ "name" ],
          "properties": {
            "name": { "type": "string" },
            "tag": { "type": "string" }
          }
        },
        {
          "required": [ "id" ],
          "properties": {
            "id": { "type": "integer" }
          }
        }
      ]
    }
  }
}

```

Listing 5.1: The OpenRPC Specification for the *get_pets_by_id* operation

Essentially, the OpenRPC specification contains the names of all API operations, as well as their the names and schemas of the corresponding parameters. Additionally, it gives information on what the response from the API should look like.

The schema of a parameter specifies the type and constraints (e.g. range or enumeration values) of the parameter. Besides the type of each parameter, the *minimum* and *maximum* value can be specified in the schema for integer types, the *length* or *minlength* and *maxlength* can be defined for array types, and a regular expression *pattern*, as well as a list of predefined *enum*(eration) values, can be specified for string types. Furthermore, the *re-*

quired field allows for specification of which parameters are required. The *allof*, *oneOf* and *anyOf* terms allow for indication of whether multiple parameters or schemas are allowed or even required.

We develop the tool to parse a JSON-RPC specification and construct a grammar from it. Our fuzzing approach generates API operation calls with inputs that match the signature of those API operations. The grammar thus serves as a template for valid HTTP requests (at least from a syntactic point of view). Multiple valid requests per API operation can be generated stochastically based on all the specified properties.

5.3 System server setup

Apart from the OpenRPC specification, the tool requires the url for the API server to be able to connect to it. The url is sufficient to establish a connection to the API and through this connection exchange HTTP requests and responses. This enables the tool to create a test suite for the [SUT](#).

However, since for research purposes we want to evaluate the performance of the fuzzing approaches (on the basis of structural coverage), it is necessary to run the [SUT](#) locally so structural coverage can be computed. Additionally, we want to ensure that the server state (that may have been impacted by a previous test case) is consistent during all tests. To this end, the SUT is shut down, data files are automatically reset and the server is restarted before each test case. To enable these things, a *test driver* needs to be implemented for the [SUT](#). The test driver prepares the [SUT](#) before a test is run. It does so by making use of a script that (re)starts the [SUT](#) and a script that computes the coverage of the [SUT](#). It should be noted that coverage metrics are only used for evaluation of the performance of the approaches, and not in any of the (black-box) fuzzing approaches.

The test driver also allows for dynamic allocation of parameter values (e.g. accounts and corresponding secret keys). Parameter values are retrieved from the server upon start if specified in the test driver. When these parameters occur in requests, they are replaced by the retrieved values. The use of existing parameter values in requests can help to penetrate deeper into the code of the target system (i.e. get past validation barriers).

5.4 Test case generation and execution

It was mentioned previously that test cases are represented by individuals. Following evolutionary terminology, the DNA of an individual consists of chromosomes. In our case, a chromosome is an HTTP request. Chromosomes contain genes. The genes represent the variables of the HTTP request, specifically the HTTP method, the API operation, and the API operation parameters. A gene has a schema and a value. The schema is the part of the grammar (explained in Section 5.2) that corresponds to the parameter. The value is the input

for that parameter. For the HTTP method (which is independent of the grammar), there is no schema. A test driver converts the DNA of an individual to a sequence of HTTP requests that can be sent to the target system.

As seen in Chapter 4, state-of-the-art fuzzing tools for RESTful APIs, mostly use a domain-specific generator to generate test cases. This means parameter constraints (as specified in the OpenRPC specification) besides expected type are taken into account when generating values.

The designed tool initially generates valid test cases using a domain-specific generator. In the following generations of test cases, the fuzzer either generates new valid test cases or mutates previously generated test cases to create invalid ones that cover different code paths of the SUT. Like the state-of-the-art fuzzing tools for RESTful APIs that support mutation, the designed tool mutates test cases by removing, adding or mutating parameters (e.g. changing the type or value).

After the server is started, test cases can be executed. A test case consists of one or multiple HTTP requests that are sent to the server. For each test the server state is reset to ensure reproducibility. The requests included in the test case are sent to the API server and a response (to the last request in the sequence of the test case) is retrieved. Afterwards, a heuristic function evaluates the server response.

During test case execution, additional information (e.g. parameter values for existing accounts) can be used if it was specified before running the tool. This could help to penetrate deeper into the code of the SUT, passing validation barriers.

5.5 Heuristic

The heuristic (or objective) function plays a large role in directing the evolutionary fuzzer towards good test cases. As was mentioned earlier in Section 3.3, a genetic algorithm encodes problem solutions (i.e. test cases) as a chromosome, which is a sequence of genes. In our case, the chromosome is an API operation, containing a sequence of parameters.

The objective function assigns a numerical value (i.e., the fitness) to an individual, which in our context represents the potential of the individual (a test case) to find (or be close to finding) faults in the SUT.

The tool also allows for the use of no heuristic function. In this case, a random selection of individuals will be mutated. The search is in this case not guided by a fitness function.

A **grammar-based fuzzer** only generates valid requests based on an API specification. A **grammar-based mutational fuzzer** repeatedly mutates a random selection of the population of test cases, resulting in potentially faulty test cases. Grammar-based (mutational) fuzzing may be inefficient at exploring the space of API inputs, which is typically very large for complex systems like blockchain applications. We want to evaluate whether a smart se-

lection of test cases to mutate improves the effectiveness and efficiency of the fuzzer. A **(grammar-based) evolutionary fuzzer** evolves the population of test cases over several iterations.

It selects test cases from the previous population to evolve based on an objective function. These test cases are then mutated and evaluated. The hypothesis is that by being selective in which test cases to mutate, coverage can be achieved faster. Intuitively, building upon an input that is known to have progressed towards the objective of the fuzzer (i.e. structural coverage) improves upon starting with a clean sheet every execution. This leaves the question on how to identify test cases that have progressed towards structural coverage.

Section 4.2 described how in white-box approaches, code coverage is generally used as (one of) the objective function(s). By building upon test cases in the population that were able to uncover new code paths in the SUT, there is a high probability that new branches (linked to the previously discovered paths) in the code are reached.

Because our fuzzing approach works in a black-box setting, we do not have direct access to the code coverage obtained by each test case. Instead, we try to derive whether (additional) coverage was obtained by a test case. This is done by analysing the information that we do have access to: the response returned by the API for the test case.

The aim is to maintain a diverse set of test cases (i.e. individuals in the population). The response object that the server returns after a request is sent says much about the system code that was run. Heuristic functions should reward test cases that yield new responses since this is an indicator of the discovery of new code paths. Test cases that result in unseen responses will be given a high fitness value and thus survive onto the next population where the request can be further explored (by mutations applied to the offspring). To keep track of whether responses from the SUT are (relatively) unseen, we implement various experimental fitness functions. They are elaborated on in the following subsections. How they work in practice is explained in Chapter 6.

5.5.1 Response object skeletons

The first designed fitness function is simple. Like an HTTP request, an HTTP response contains parameters and values. The values are often unique. When the input values of a request change, the output values of the response generally change as well. This does not necessarily mean that a new code path has been exercised though (the input is handled the same by the system, but the output is dependent on the input). We do know that when the parameters of the response are different, a new code path has been exercised.

We call this set of parameters (or keys), without values, the skeleton of the response. The *Skeleton Fitness* assigns fitness values to individuals based on how often the fuzzer has come across response skeletons. Individuals that result into response skeletons that have occurred less than the response skeletons of other individuals in the population, are

assigned a higher fitness value. This fitness function aims to maintain a diverse population overall. We assume that the more different response object skeletons are come across, the more code is likely covered by the requests. A diverse population should result into a higher probability of uncovering new code paths.

The *Skeleton Fitness* favours individuals that lead to newly discovered response skeletons. The fitness is calculated as:

$$F_S = \frac{1}{\#ResponseSkeletonOccurrences}. \quad (5.1)$$

Limitations of this fitness function are that values are not taken into account, while they can actually be meaningful and indicate new code paths used. Another issue is that there might be response skeletons that are the same, while the exercised code is different (e.g. different API operations that yield the same response skeleton).

5.5.2 Response object skeletons and request object complexity

The *Skeleton Fitness* is a basic fitness function aimed to maintain a diverse set of individuals over the course of the fuzzing process. It does so by favouring individuals that result into unique response skeletons. The complexity of the request and response object is however not considered.

Instead of just looking at the response object skeleton, the *SkeletonAndComplexity Fitness* also takes the complexity of the request object into account. The fitness value is computed by the multiplication of two factors.

The first factor is the uniqueness of the response object skeleton like in *ResponseSkeleton Fitness*. The second factor is the complexity of the request object. The complexity of a **JSON** object is calculated as the number of keys (on all levels of depth) in a request and corresponding response. The idea behind this is that if a request has few keys, it is unlikely to find a new response structure by mutating this request since there are few fields that can be changed. For a high complexity on the other hand, there may be more variations to be discovered.

The *SkeletonAndComplexity Fitness* favours individuals that lead to newly discovered response skeletons and that have a high number of keys in the request object. The fitness is defined as:

$$F_{SC} = \frac{\#InputParameters}{\#ResponseSkeletonOccurrences} \quad (5.2)$$

5.5.3 Simplified clustering of response objects using categories

The *SimpleResponse Fitness* takes into account both the keys of the **JSON** response object and the values corresponding to these keys. For each API operation in the OpenRPC specification, this approach keeps track of how often parameters with values in certain categories

occur in response objects. During the running of the evolutionary algorithm, the values are grouped in predefined categories. The (few) categories of values depend on the [JSON](#) types (String, Number, Boolean, Array and Null) and are simple (e.g. the category of empty strings). Each parameter value belongs to one of the categories. This fitness function is meant to demonstrate the potential of considering parameter values in a memory-efficient way. The effectiveness of this approach depends heavily on the chosen categories.

A response containing parameters that fall in categories that have a high occurrence count, decreases the fitness value of the individual. An individual gets a maximal fitness score when the entire response has never been seen before. This is the case when a response object contains parameter values that fall within categories that have not occurred before (in a response object to a certain API operation). When many parameter values in a response have occurred before, the individual is assigned a lower fitness score.

The *SimpleResponse Fitness* favours individuals that lead to newly discovered parameters or categories of parameter values for each API operation. The fitness is calculated as:

$$F_{SR} = \frac{\#Params}{paramCategoryOccurrences} \quad (5.3)$$

where *paramCategoryOccurrences* is the sum of the occurrences of the parameters' categories.

The *BasicResponseFitness* is simplistic and requires the predefined categories to be tailored to the [SUT](#). Moreover, the distance between each category is the same, while some categories may be closer together than others (e.g. zero and positive numbers are in reality closer together than negative and positive numbers).

5.5.4 Agglomerative clustering of response objects

As was mentioned in the previous subsection, without in-depth knowledge of the [SUT](#) (which is unavailable for black-box settings), it is difficult to define sensible value categories for response parameter values. Unsupervised learning (clustering) is a solution for this. Clustering is a technique commonly used in data analysis to group a set of similar objects [115]. Objects in the same cluster are more similar to each other than to those in a different cluster. The purpose of clustering is to minimise the intra-cluster distance and maximise the inter-cluster distance.

Agglomerative clustering

Most clustering methods belong to either partitioning and hierarchical methods [126]. Partitioning methods require the number of clusters as input. Since we are working in a black-box setting (and we do not know the optimal number of clusters), partitioning methods cannot be applied. Instead, a hierarchical clustering algorithm needs to be implemented.

Hierarchical methods can be either divisive (top-down) or agglomerative (bottom-up). An agglomerative method is a less complex and particularly effective and popular approach for clustering data [115]. Agglomerative clustering is a bottom-up approach, where each object starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. Similar clusters are sequentially combined until only one cluster is obtained. By observing the similarity between clusters at each step, the best number of clusters can be identified.

Figure 5.2 illustrates the process of agglomerative clustering in a dendrogram. The most similar responses are merged together, resulting in a hierarchy of clusters. The dendrogram is cut off (at the dashed line) to acquire a number of clusters. The number of intersections with the vertical lines yield the numbers of clusters. In the example, three clusters remain.

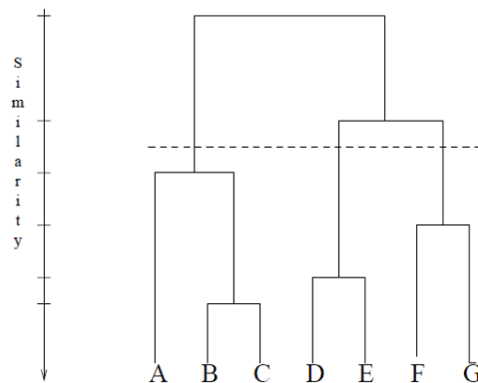


Figure 5.2: Dendrogram

The two key points in hierarchical clustering are the identification of the clusters to merge at each step, and the identification of the optimal terminating step [126]. A meaningful agglomerating and termination criterion, are required.

Agglomerating criterion. The choice of agglomerating criterion defines which clusters are merged into one. This criterion are typically based on distances between clusters, such as the single-linkage approach (which considers the smallest distance between any two points in two clusters) or the complete-linkage approach (which considers the maximum distance between any two points in two clusters) [115]. We chose to use mean linkage as it is less susceptible to noise and outliers compared to single and complete linkage. This is the average distance between each point in one cluster to every point in the other cluster. Mean linkage clustering is most often used in hierarchical clustering algorithms and is the best choice for most applications [53].

Termination criterion. An important question for agglomerative clustering is at what point to cut off the dendrogram (the termination criterion). Typically the cut-off point is chosen that cuts the tallest vertical line [55]. This is the point where clusters are merged that are far apart. The tallest vertical line represents the highest similarity jump for the

merging of two clusters.

Feature vector representation

We represent response parameter values as a feature vector that contains all parameter values (that can be of string, boolean, number, array and JSON object types). Only primitive types (string, boolean and number types) can be stored in the feature vector. This is why for arrays, only the first value of the array (which is a string, boolean, number) is embedded in the feature vector. For JSON objects, all parameter values are extracted from the object and put in the feature vector. To account for the fact that these parameters are part of a JSON object, we also compute a weight vector. The weight of a value is defined by 1 divided by the depth of the parameter. For example, a parameter in a JSON object a has a weight of $\frac{1}{2}$ since this parameter is nested in a JSON object. If this same JSON object a contains another JSON object b , the parameters of b have a weight of $\frac{1}{3}$. It is assumed that parameters with a higher depth are less important to differentiate between responses.

Computation of distance

Without a reliable distance metric between pairs of objects, a meaningful analysis of clusters is not possible [88]. Commonly used distance metrics for hierarchical clustering are the (squared) Euclidean distance, Manhattan distance, Mahalanobis distance, maximum distance, and the cosine similarity [99]. Due to the feature vectors containing different types, it is not straightforward how to apply these distance metrics. The Mahalanobis distance and cosine similarity cannot be applied to feature vectors containing values of different types. The maximum distance is not considered suitable for this problem, since we want to evaluate all parameter values (a vector with one parameter with a very large distance should not be preferred over a vector with multiple parameters that have a small distance). Although the Euclidean and Manhattan distance are also designed for numerical values (coordinates), we can fit these metrics to our problem. The Euclidean distance is defined as:

$$\|\mathbf{a} - \mathbf{b}\|_2 = \sum_i \sqrt{(\mathbf{a}_i - \mathbf{b}_i)^2}.$$

The Manhattan distance (a version of the Euclidean distance) is defined as:

$$\|\mathbf{a} - \mathbf{b}\|_1 = \sum_i |\mathbf{a}_i - \mathbf{b}_i|$$

with feature vectors a and b and i the number of dimensions (the length of the feature vector). The Euclidean distance depends strongly on data normalisation [89]. However, since we do not know the range of numbers, or the diversity of strings, we cannot reliably normalise the vectors. We therefore choose to use the Manhattan distance, which is less dependent on normalisation.

To calculate distance between two vectors, we first compute the distance for each pair of parameters in two feature vectors separately, using a suitable distance metric for that type. For boolean values, the distance is either 0 (values are equal) or 1 (values are not equal). For numbers, the distance is defined as the absolute difference between the numbers. For string values, the Levenshtein distance is often used [99]. This distance metric (also called the edit distance) is the most promising one to compare strings by various edit operations (deletion, insertion, and substitution of characters) [133]. It is defined as the minimum cost (always an absolute value) of transforming one string into another. We then multiply the distance of each parameter by the corresponding weight.

Finally, we use the Manhattan distance to compute the total distance between two vectors. The Manhattan distance assumes that variables are independent of each other. This is likely not the case for response objects. Still, we assume that this distance metric is able to sufficiently differentiate between feature vectors.

Time complexity

Agglomerative clustering has a time complexity of $O(n^3)$ [99], which makes it inefficient and expensive for large data sets, which is why we chose to not perform agglomerative clustering for all individuals in every generation. Instead, we keep track of the "median" and radius of clusters to calculate the distance of individuals in the population to be evaluated.

The median of a cluster is the response object that has the minimum sum of distance to all other responses in the cluster (i.e. it is closest to all other individuals in the cluster). The radius is defined as the distance between the median and the point in the cluster that is furthest removed from the median. Clustering is only performed after a predefined number of generations is processed. When clustering is performed, all responses that did not belong to a cluster, as well as the existing representative (median) responses of the clusters, are clustered. All previous data points (except for the medians) are removed from the clusters. The next generations are evaluated with regard to these updated cluster medians and radiuses (until clustering is performed again).

Diversity-based fitness

We propose to perform clustering so that individuals in the same cluster are similar while individuals in different clusters are more diverse. The fitness value of an individual is defined as the distance of the individual's response object to previously encountered response objects. We call this fitness function the *Diversity-Based Fitness*. Individuals that result in a (relatively) unique response, form a new cluster and are given a high fitness value. Individuals with responses that are very similar (small distance) to those of other individuals are given a low fitness value. The hypothesis is that responses that are unlike what was seen before, are an indication that new code paths are reached. Building from this test case, there is potential to discover other new code paths.

The Diversity-Based Fitness favours individuals that lead to diverse responses for each API operation. The fitness is calculated as:

$$F_{DB} = \frac{1}{1 + \maxSimilarity} \quad (5.4)$$

where *maxSimilarity* is the similarity of the individual's response object to the closest response object in the clustering instance. This is computed based on the distance to the closest object and the parameter weight.

5.6 Mutation engine

The effectiveness of the evolutionary fuzzing approach depends largely on mutation. Good mutation operators are necessary to gather a complete and diverse set of test cases.

An individual can be mutated in various ways. Figure 5.3 illustrates the structure of an individual. An individual (a test case) contains one or multiple HTTP requests. As was explained earlier in Section 3.1.1, an HTTP request requires either a POST or GET operation, an API operation, and corresponding parameters.

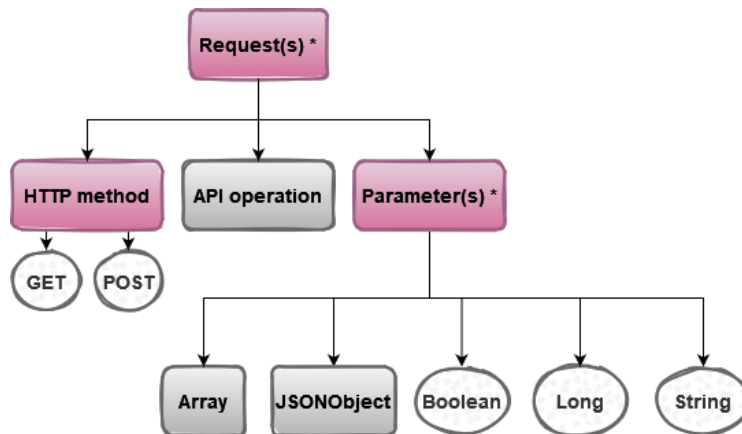


Figure 5.3: The structure of the individual. (* one or more)

When an individual is mutated, a newly generated request is added to the sequence, one of the requests is removed, and/or one of the requests is mutated. The latter can happen in one of three ways:

1. The HTTP method is changed. If it is *POST*, it is changed to *GET*. If it is *GET*, it is changed to *POST*.
2. A new API operation with corresponding parameters is generated (based on the grammar) and replaces the current API operation and parameters. Besides the HTTP

method, this is effectively an entirely new individual. This can be beneficial for exploration of the search space, for example when the population has converged towards specific parts of the search space (e.g. certain API operations).

3. One or more of the API operation’s parameters are mutated.

Chapter 4 described various mutation operators that were shown to be effective in previous work. We implement the most common in our approach: addition of parameters, removal of parameters, and mutation of parameter values (including violation of parameter constraints).

Parameters are either replaced by a newly generated value, or mutated by type-specific mutation operators. Table 5.1 shows the ways in which each type can be changed by mutation operators. For JSON objects, a child (which has its own type as well) is either added, removed, or mutated. For array type values, an element (which has its own type) is either added, removed, or mutated. For numbers, polynomial mutation is applied, or the value is set to the lower or upper bound of the parameter. Polynomial mutation¹ is widely used in evolutionary algorithms as a mutation operator [47], including in related works such as EvoMaster [9]. For string types, (one or more) characters are either added, removed, or mutated (changed into different character(s)). For boolean types, the value is simply flipped (i.e. false becomes true and vice-versa).

Object	Array	Number	String	Boolean
Child mutation	Element mutation	Polynomial mutation	Character(s) mutation	Flipping the value
Child addition	Element addition	Boundary cases	Character(s) addition	
Child deletion	Element deletion		Character(s) deletion	

Table 5.1: Mutation operators of parameter types

When a parameter is replaced by a newly generated value, it either conforms to the OpenRPC specification of that parameter or it does not. This allows for parameter values to occur in requests that have unexpected types (e.g. an API operation expects a parameter value of a boolean type, but this parameter value is mutated into a number).

We do not focus on including a complete set of all mutation operators as the aim of this research is not (strictly) to achieve a maximum amount of coverage, but rather to evaluate whether evolutionary fuzzing improves upon random mutational fuzzing for JSON-RPC APIs.

¹In polynomial mutation, to mutate a value into a value close to the original, a polynomial probability distribution is used. Polynomials are sums of terms of the form $k \cdot x^n$, where k is any number and n is a positive integer.

6

Implementation

In Chapter 5, the strategy was explained for the creation of the fuzzing tool that is central in this research. In this chapter, we introduce **GEFRA**, a black-box Grammar-based Evolutionary Fuzzer for RPC-APIs. **GEFRA** translates OpenRPC specifications to grammars and fuzzes JSON-RPC systems for software defects. **GEFRA** contains a grammar-based (mutational) fuzzing and evolutionary fuzzing approach.

First, the different components of the architecture of **GEFRA** are discussed in Section 6.1. Afterwards, Section 6.2 discusses the grammar-based fuzzing approach. Then, the specifics of the evolutionary fuzzing approach are presented in Section 6.3. Finally, Section 6.4 gives an explanation how to run the tool.

6.1 GEFRA architecture

A high-level overview of the architecture of **GEFRA** can be seen in Figure 6.1.

GEFRA runs on any JSON-RPC API system. All it requires is an OpenRPC specification. With this, the tool is able to generate test cases for any system working with a JSON-RPC API. The OpenRPC specification serves as a grammar for test cases. Listing 6.1 provides an example of the OpenRPC specification for the API operation *account_channels*.

The API operation lists three parameters, out of which one is required. Moreover, the API expects the parameters to be of a certain type and format. The *limit* parameter for example is expected to be an integer value between 10 and 400. **GEFRA** generates test cases based on the specification and respects parameter constraints. Three examples of generated requests for the *account_channels* operation by **GEFRA** are shown in Listing 6.2.

For the initialisation of the population of test cases, **GEFRA** randomly selects API operations from the OpenRPC specification, and constructs syntactically valid requests for them. The HTTP method is set at either *POST* or *GET* with a certain probability. Array, JSON Object, Boolean, Number, and String parameters are generated based on parameter constraints. If these are not specified, the implemented custom generator resorts to default constraints such as a maximum array size and a default regular expression (regex) for strings.

6. IMPLEMENTATION

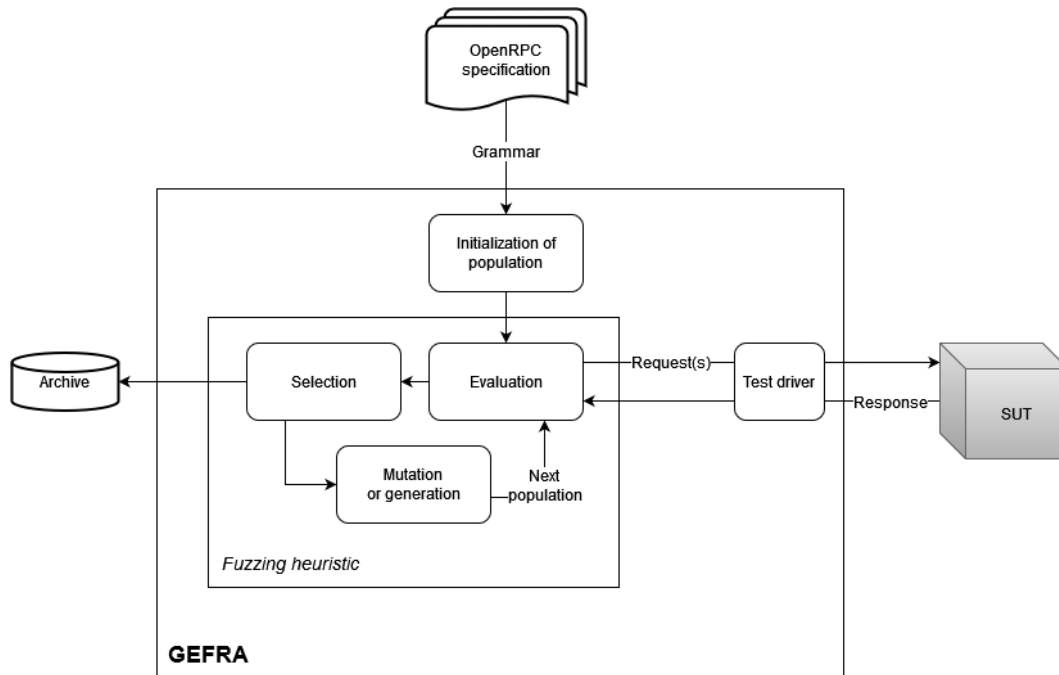


Figure 6.1: A high-level overview of GEFRA

```
{
  "name": "account_channels",
  "params": [
    { "name": "account",
      "required": true,
      "schema": {
        "type": "string",
        "pattern": "^r[1-9a-zA-Z]{25,35}$"
      }
    },
    { "name": "ledger_index",
      "schema": {
        "oneOf": [
          { "type": "integer", "minimum": 0, "maximum": 4294967296 },
          { "type": "string", "enum": ["validated", "current"] }
        ]
      }
    },
    { "name": "limit",
      "schema": {
        "type": "integer",
        "minimum": 10,
        "maximum": 400
      }
    }
  ]
}
```



```

    }
  ]
}

```

Listing 6.1: The OpenRPC Specification for the *account_channels* operation

```

POST {"method":"account_channels","params":[{"limit":317,"account":"
      rUEYbWfD6fqEKpmNHVKgVokEgvZJboc33n"}]}

POST {"method":"account_channels","params":[{"ledger_index":3624039044,"
      account":"rp3cLhgwsFZZs5AoH2VK1YHeYU3FHXVAAf"}]}

GET {"method":"account_channels","params":[{"ledger_index":"current","
      account":"rVhutwmkyva8bt4W9RJVr5bD85ceRJvkN"}]}

```

Listing 6.2: Three generated requests based on the OpenRPC Specification for the *account_channels* operation

After the initial population is generated, the fuzzing process starts. The termination criterion that is set decides when to stop processing generations. The termination criteria that can be set for [GEFRA](#) are a limit of generations, a limit of evaluations (i.e. the number of times the fitness was computed), or a time limit.

The different phases of the fuzzing process depend on the fuzzing approach that is used: grammar-based mutational fuzzing or evolutionary fuzzing. Section 6.2 describes the grammar-based mutational fuzzing approach, whereas Section 6.3 describes the evolutionary fuzzing approach.

Both approaches send requests (test cases) to the [SUT](#) using a test driver. The test driver provides the connection between the fuzzer and the [SUT](#). It resets and starts the server of the [SUT](#) before each test to ensure a clean environment. It converts the individuals from the search process into HTTP requests. Subsequently, the test driver sends the requests to the (API of the) [SUT](#) and monitors the output. The results are passed back to the fuzzer and (in the evolutionary fuzzer) are used by the objective function to calculate the fitness of the test.

[GEFRA](#) also contains a simple archiving mechanism, allowing for the storage of generated system-level test cases for the [SUT](#). Test cases that result in unique response skeletons are stored in the archive. The test suite can be accessed and executed after the fuzzer has finished. The mechanism demonstrates basic performance with regards to structural coverage but it does not reach (near) the coverage obtained by running the fuzzer.

6.2 Grammar-based (mutational) fuzzing

The different phases of the grammar-based mutational ([GB-MUT](#)) fuzzing approach are described in the next subsections.

6.2.1 Evaluation

The grammar-based mutational approach does not guide the search with an objective function. The executed tests are not evaluated.

6.2.2 Selection

Since there is no evaluation, a random selection of the current generation of individuals and their offspring is selected to survive onto the next generation.

6.2.3 Generation and mutation

For a search algorithm to be successful, it needs to establish a good balance between exploration and exploitation [117]. Exploration is the process of visiting entirely new areas of a search space, whereas exploitation is the process of visiting those areas of a search space that are nearby previously visited points.

If new generations are created solely by mutating the current population, the search would be mostly exploitative. To promote exploration, not all individuals in the generation emerge from mutation of existing ones. The offspring is partly created by mutating individuals from the population, and partly by generating new individuals based on the grammar. This is done to ensure a healthy balance between exploration and exploitation. The ratio between generation and mutation is configurable. With a 0% mutation rate, we have a traditional grammar-based fuzzer, to which we want to compare the evolutionary fuzzer. We refer to the grammar-based fuzzer as the baseline.

The HTTP method, API operation(s), and corresponding parameters are embedded in the individual and any part of the individual can be mutated. Figure 6.2 shows all the ways in which an individual can be mutated. These correspond mostly to the related work discussed in Chapter 4. The default probabilities are added to the different paths. Every individual is configured to be mutated twice in order to deviate further from the parent (i.e. the individual that is mutated).

For each mutation, either a request is removed, added, or mutated. When one of the options in the chart is not available (e.g. an individual only has one request), the fuzzer defaults to mutation. If a new request is added, a new request is generated based on the grammar and added to the sequence of the requests of that individual. If a request is mutated, either the HTTP method, the API operation, or one of the (randomly selected) corresponding parameters is mutated. It depends on the parameter type what mutation operators are applicable, but all parameters have a probability to be changed into a newly generated value according to any schema that is documented in the OpenRPC specification. This allows for the occurrence of unexpected types and values (e.g. a value that is expected for a different parameter), which may lead to the discovery of faults in the SUT.

When an array value or **JSON** object is selected to be mutated, one of the elements or keys is randomly chosen. The element or key in question is then mutated based on its type. For example, if a key-value pair (a parameter name and its value) from a **JSON** object is mutated, and the value is a number, the value is mutated using the Long mutation operators. The original value is then replaced by the resulting mutated value.

If string-type parameters specify enumeration values in their OpenRPC specification, there is a probability that the value is changed into one of these. Otherwise, a fraction of characters is mutated. These characters are randomly selected (and thus do not have to be a sequence in the string).

6.3 Grammar-based evolutionary fuzzing

Search algorithms were explained in Section 3.3. The $(\mu + \lambda)$ evolutionary strategy) is a straightforward evolutionary algorithm (EA). Campos et al. [24] found that in most (unit test suite generation) cases, a simple $\mu + \lambda$ algorithm performs better than other, more complex algorithms.

During the evolutionary fuzzing process, for each individual the server is re-started, the request is sent to the API, and a response is retrieved before the fitness of that individual can be computed. This makes the fitness evaluation of an individual quite computationally expensive. $(\mu + \lambda)$ algorithms have been used in literature for computationally expensive fitness functions [1, 29]. Each individual is only evaluated once. With small values for μ and λ , the cost of each iteration is kept to a minimum.

Algorithm 1 presents the $(\mu + \lambda)$ algorithm that is implemented in the tool. The initial population of size μ , after being mutated, produces λ offspring. The algorithm then adds the offspring to the initial population (hence $\mu + \lambda$). The offspring competes with the parents. The solutions are evaluated and the best individuals (a selection of size μ) form the next generation.

During the search, the implemented evolutionary algorithm can make use of crossover operations. A crossover operator is defined to combine an HTTP request of one individual with that of another. For the experiments it was decided to only allow individuals to contain one request so we can draw better conclusions from the comparison (which otherwise could be influenced by the different size of individuals; several requests in an individual would achieve more coverage than one request). Therefore crossover is not enabled. Modification of test cases happens only via the mutation operators.

The different phases of the grammar-based evolutionary (**GB-EVO**) fuzzing approach are described in the next subsections.

Algorithm 1 ($\mu + \lambda$) GA

Input: μ = the population size

```
1 begin
2    $t \leftarrow 0$ 
3    $P(t) \leftarrow \text{initialisePopulation}(\mu)$ 
4   do
5      $t \leftarrow t + 1$ 
6      $P(t) \leftarrow P(t - 1) \cup \text{mutate}(P(t - 1))$ 
7      $\text{evaluate}(P(t))$ 
8      $P(t) \leftarrow \text{select}(P(t), \mu)$ 
9   while termination criterion is not satisfied;
10 end
```

6.3.1 Evaluation

The evaluation in the $(\mu + \lambda)$ EA is performed by the fitness function. In Section 5.5, the ideas behind the designed fitness functions were explained. This part describes how they are implemented and function in practice.

1 - SkeletonFitness

Algorithm 2 shows the *SkeletonFitness* function. It takes the API operation, the request object, and the response object as input.

Algorithm 2 EVALUATE INDIVIDUAL - SkeletonFitness

Input: *met* = API operation used in the last request*req* = last HTTP request object in sequence of individual's requests*res* = last HTTP response object from the server*freqMap* = a dictionary containing skeletons as keys and counts as values**Output:** Computed fitness value *fv*

```
1 begin
2    $\text{stripped} \leftarrow \text{stripValues}(\text{req}, \text{res})$ 
3    $\text{freqMap.put}(\text{stripped}, \text{freqMap.get}(\text{stripped}) + 1)$ 
4    $\text{fv} \leftarrow 1 / \text{freqMap.get}(\text{stripped})$ 
5    $\text{return fv}$ 
6 end
```

The *stripValues* function replaces the parameter values in the response object by predefined standard values. Specifically, strings are replaced by an empty string (""), numbers are replaced by 0, boolean values are replaced by *true*, and arrays are replaced by either

""", 0 or *true*, depending on the type of the contents of the array. Listing 6.3 contains an example of an individual (with one request) that is evaluated.

Parameters from the request are sometimes included in the response by default. Because such values are simply copied from the request, they are deemed irrelevant for the fitness of an individual. The *stripValues* function removes key-value pairs in the response if the pairs occur in the request as well. As can be seen in the stripped response in the example, the parameter values are replaced by the standard values and the *account* parameter is removed completely. We call this stripped version of the response the skeleton.

The skeleton is added to a dictionary that stores the stripped response (in String format) as the key, and the number of total occurrences as the value. The count is updated. Afterwards, the fitness value is computed by the division of 1 by the total occurrences of the response skeleton. In the example, the fitness value will be $\frac{1}{2} = 0.5$. When a response skeleton has never been seen before, the fitness is 1. The fitness value is always between 0 and 1. Individuals that yield relatively unique response skeletons are retained in the population, whereas individuals that retrieve response skeletons that have occurred many times before, do not survive.

```
REQUEST: POST {
  "method": "gateway_balances",
  "params": [{"account": "rPcv9Wm3MhAgyZNnjTaWUKwhHXaTnSEbx9"}]
}

RESPONSE: {
  "result": {
    "ledger_current_index": 30051713,
    "validated": false,
    "account": "rPcv9Wm3MhAgyZNnjTaWUKwhHXaTnSEbx9",
    "status": "success" }
}

STRIPPED response: {
  "result": {
    "ledger_current_index": 0,
    "validated": true,
    "status": "" }
}

FREQUENCY map (after update):
{"result":{"ledger_current_index":0,"validated":true,"status":""}}=2,
{"result":{"validated":true,"ledger_hash":"","status":""}}=14,
{"result":{"random":"","status":""}}=5
```

Listing 6.3: Demonstration of the SkeletonFitness evaluation steps

2 - SkeletonAndComplexityFitness

Algorithm 3 shows the *SkeletonAndComplexityFitness* function. This objective function is very similar to *SkeletonFitness*, except it takes an additional property of the request into account: the complexity of the request. If we stick to the same example as in Listing 6.3, the complexity is calculated by counting the parameters in the request. In this case, there is one parameter, namely *account*. The complexity is thus equal to 1. The frequency is calculated in the same way as for *SkeletonFitness*, and is 0.5. The fitness is computed by multiplication of the complexity of the request by the frequency of the response skeleton: $1 * 0.5 = 0.5$. The fitness has a minimum value of 0 (since complexity and frequency are always positive). The maximum fitness value is equal to the maximum number of keys in requests.

Algorithm 3 EVALUATE INDIVIDUAL - SkeletonAndComplexityFitness

Input: *met* = API operation used in the last request

req = last HTTP request object in sequence of individual's requests

res = last HTTP response object from the server

freqMap = a dictionary containing skeletons as keys and counts as values

Output: Computed fitness value *fv*

```
1 begin
2   stripped  $\leftarrow$  stripValues(req, res)
3   frequencyMap.put(stripped, skeletonFrequencyMap.get(stripped) + 1)
4   complexity  $\leftarrow$  countParameters(req)
5   frequency  $\leftarrow$   $1 / \textit{frequencyMap.get}(\textit{stripped})$ 
6   fv  $\leftarrow$  complexity * frequency
7   return fv
8 end
```

The higher the complexity of inputs, the higher the probability of finding faults [118]. Essentially, this representation of the complexity of the request, functions as a weight for the skeleton frequency. If a skeleton has occurred often, but there are many parameters in the request (i.e. many parameters that can be mutated that may result into different responses), the fitness is boosted by the complexity. If a skeleton frequency is the minimum value possible (1), but the request contains 0 parameters, the fitness value assigned to the individual will be 0, as there is seemingly nothing in the request to mutate. The complexity of the request is an indicator for the potential of the individual, but it is not foolproof. It is very well possible for a request to contain only a few parameters, while the used API operation could define many. By mutating the individual, new (previously unused) parameters may be added, which could lead to new responses.

3 - SimpleResponseFitness

The above fitness functions disregard the fact that besides the used parameters in the response, the parameter values differentiate between responses as well. *SimpleResponseFitness* is a basic attempt to consider not only the parameter names in the response, but also the parameter values. It is a slightly more refined version of *SkeletonFitness*. Instead of just storing the number of occurrences of the complete skeletons, this fitness function stores the number of occurrences for each parameter value category. Algorithm 4 shows the pseudocode of the *SimpleResponseFitness* function.

Algorithm 4 EVALUATE INDIVIDUAL function - SimpleResponseFitness

Input: *met* = API operation used in the last request

req = last HTTP request object in sequence of individual's requests

res = last HTTP response object from the server

freqMap = a dictionary keeping counts of all value categories per parameter per API operation

Output: Computed fitness value *fv*

```

1 begin
2   score ← 0
3   numberOfKeys ← 0
4   foreach key  $k_i \in res$  do
5     cat ← identifyCategory(res.get( $k_i$ ))
6     occurrences = getOccurrences(met,  $k_i$ , cat)
7     freqMap(met,  $k_i$ , cat) ← (occurrences + 1)
8     numberOfKeys ← numberOfKeys + 1
9   end
10  cost ← occurrences/numberOfKeys
11  fv ← 1/cost
12  return fv
13 end

```

The response **JSON** object contains keys (i.e. the parameters) and values. For each key in the response, it is evaluated by *identifyCategory* to which predefined category the value belongs. The value categories depend on the value type: String, Number, Boolean, Array and Null (corresponding to the **JSON** types). The categories can be found Table 6.1. They are a simple representation of what kind of values exist. String categories only differentiate between zero, short and long (> 10 characters) lengths. The use of Regex expressions could help to create more specific categories. However, for all value categories it is a problem that it is unknown what categories are suitable for the **SUT**. The defined categories may be meaningless for the systems we test, which is a significant weakness of this approach.

6. IMPLEMENTATION

Number	String	Boolean	Array	Other
INTEGER_POSITIVE	EMPTY_STRING	TRUE	EMPTY_ARRAY	NULL
INTEGER_NEGATIVE	SHORT_STRING	FALSE	STRING_ARRAY	
INTEGER_ZERO	LONG_STRING		NUMBER_ARRAY	
LONG_POSITIVE			BOOLEAN_ARRAY	
LONG_NEGATIVE				
LONG_ZERO				

Table 6.1: Categories for parameter values in SimpleResponseFitness

After the value is categorised, we want to compare it to the value categories of that parameter that have occurred previously. The total count of categories for all parameters of all API operations is stored in a dictionary (*freqMap*). The dictionary is never reset. Upon evaluation, the count of the parameter value's category path is increased by 1. The cost is defined as the sum of the counts of the present value categories per parameter in the response, normalised by the total number of parameters. The fitness is defined by 1 divided by the cost. The workings of *SimpleResponseFitness* are demonstrated in Listing 6.4. The sum of counts of the categories present in the response is 12 and the response contains 6 parameters. The fitness value in this example therefore is: $1 / (12/6) = 0.5$.

```

REQUEST: POST {
  "method":"account_channels",
  "params":[{"account":"rQK8G7FT5BHhG9RBVWNmJeKfQCb3NvFaQn"}]
}

RESPONSE object: {
  "result": {
    "error_message":"Account not found.",
    "request":{
      "account":"rQK8G7FT5BHhG9RBVWNmJeKfQCb3NvFaQn",
      "command":"account_channels"},
    "error_code":19,
    "error":"actNotFound",
    "status":"error"
  }
}

VALUE CATEGORIES in this object:
/result/error_message: LONG_STRING
/result/request/account: LONG_STRING
/result/request/command: LONG_STRING
/result/error_code: INTEGER_POSITIVE
/result/error: LONG_STRING
/result/status: SHORT_STRING

```



```

FREQUENCY map for the account_channels API operation (after update):
/result/error_message={LONG_STRING=4},
/result/request/account={LONG_STRING=3},
/result/request/command={LONG_STRING=6},
/result/error_code={INTEGER_POSITIVE=2},
/result/error={LONG_STRING=2},
/result/status={SHORT_STRING=3},
/result/request/ledger_index={LONG_POSITIVE=1, SHORT_STRING=1},

```

Listing 6.4: Demonstration of the SimpleResponseFitness evaluation steps

4 - Diversity-Based Fitness

The diversity-based fitness approach was explained in 5.5.4. It makes use of agglomerative clustering. Clustering instances are updated after a predefined number of generations of individuals. During the evaluation process, responses that do not fall within (the radius of) any cluster (i.e. do not belong to a cluster), are stored to be clustered next time together with all median responses.

Evaluation of individuals is done by comparing their response objects to the clustering instances. Pseudocode of the *DiversityBasedFitness* evaluation algorithm is shown in Algorithm 5.

Algorithm 5 EVALUATE INDIVIDUAL function

Input: *met* = API method used in the last request

req = last HTTP request in sequence of individual's requests

res = last HTTP response from the server

Output: Computed fitness value *fv*

```

1 begin
2   stripped ← stripValues(req, res)
3   features, weights ← convertToVector(res, stripped)
4   clusteringInstance ← getClusteringInstance(met, stripped)
5   maxSimilarity ← clusteringInstance.calculateMaxSimilarity(features, weights)
6   fv ← 1/(1 + maxSimilarity)
7   return fv
8 end

```

We keep track of various clustering instances, identified by the response skeleton. This is necessary since distance can only be calculated between vectors of the same length containing the same types. A clustering instance contains one or more clusters of (the parameters of) response objects. If two response objects are in a different clustering instance, the distance between them is maximal. We assume that if a response object contains different keys, different code paths were exercised.

6. IMPLEMENTATION

The *stripValues* function is the same as explained in *SkeletonFitness*. Besides stripping the response from parameter values, it removes parameter-value pairs that were present in the request. Based on the parameters that are left, a feature vector is created by *convertToVector*. It takes the values of these parameters from the response. The sequence of values (which corresponds to the sequence of keys in the response skeleton) forms the feature vector. A corresponding weight vector is created as well, storing the depths of the parameters in the response object. The depth specifies how many levels of contained objects are included in the **JSON** representation.

The clustering instance for the response skeleton at hand is retrieved and the maximum similarity of the feature vector to any of the clusters is computed (i.e. how similar is this response to the closest other response). *calculateMaxSimilarity* computes the similarity of the individual's response to the closest cluster's median. The similarity is calculated as: $\frac{1}{1+distance*weight}$. A large distance results in low similarity. If a response is completely new (e.g. a new clustering instance is initiated), the distance is maximum, and the *maxSimilarity* is 0. The fitness value is computed as: $\frac{1}{1+maxSimilarity}$.

To illustrate the calculation of distance between two individual's responses, two requests, their responses, and the response skeleton are shown in Listing 6.5.

```
REQUEST 1: POST {
  "method": "tx_history",
  "params": [{
    "start": "957269095"
  }]
}

REQUEST 2: POST {
  "method": "tx_history",
  "params": [{
    "start": "708045535919000813722
I9j18Z2V0t1KP630J5n4y5544769592053jvk11TmD8a3Ir1219118314143"
  }]
}

RESPONSE 1: {
  "result": {
    "error_message": "Internal error.",
    "request": {
      "start": "708045535919000813722
I9j18Z2V0t1KP630J5n4y5544769592053jvk11TmD8a3Ir1219118314143",
      "command": "tx_history"
    },
    "error_code": 73,
    "error": "internal",
    "status": "error"
  }
}
```

```

}

RESPONSE 2: {
  "result": {
    "error_message": "You don't have permission for this command.",
    "request": {
      "start": "957269095",
      "command": "tx_history"
    },
    "error_code": 6,
    "error": "noPermission",
    "status": "error"
  }
}

STRIPPED RESPONSE 1 and 2: {
  "result": {
    "error_message": "",
    "request": {
      "start": "",
      "command": ""
    },
    "error_code": 0,
    "error": "",
    "status": ""
  }
}

```

Listing 6.5: clustering

The responses have the same response skeleton, which means they are part of the same clustering instance, and can be compared to each other. The distances between all feature vectors parameter pairs are:

$$\begin{bmatrix} \text{"Internalerror."} \\ tx_history \\ 73 \\ \text{"internal"} \\ \text{"error"} \end{bmatrix} - \begin{bmatrix} \text{"You don't have permission for this command."} \\ tx_history \\ 6 \\ \text{"noPermission"} \\ \text{"error"} \end{bmatrix} = \begin{bmatrix} 20 \\ 0 \\ 67 \\ 10 \\ 0 \end{bmatrix} * \begin{bmatrix} 1 \\ 0.5 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

The distance between string values is calculated by the Levenshtein distance. Because the Levenshtein distance can quickly become very large, it is capped at 20. We believe that there is no significant difference between strings with a 20-character edit distance and strings with a higher edit distance than that in this context. To compute the similarity between the two vectors, we multiply each distance by the parameter weight.

Finally, we compute the Manhattan distance as $(20*1+0*0.5+67*1+10*1+0*1) = 97$. This implies a similarity of $\frac{1}{1+97} \approx 0.01$, indicating that the responses are not similar. Seeing

as three of the five parameter values are significantly different, this seems reasonable. We do see that numbers greatly influence the similarity as the distance for numbers has no maximum. Furthermore, in this case, while the `error_code` parameter values are numbers, it seems that these values are used as nominal categories rather than numerical values. Our experimental distance metric has no way to differentiate between the two.

6.3.2 Selection

The selection mechanism of a genetic algorithm is the process that favours the selection of better individuals in the population for the mating pool. The main selection mechanisms are elitist selection and tournament selection.

The elitist selection scheme ranks all individuals (current population + offspring) based on their fitness. Consequently, it selects the n individuals with the highest fitness values to form the next generation of individuals, with n being the population size.

Tournament selection, the most popular selection method [38], holds a tournament among s randomly selected competitors (from the current population + offspring), with s being the tournament size. The winner of the tournament is the individual with the highest fitness among the tournament competitors. The winner is then passed on to the next generation. The process is repeated until n individuals are selected. The next generation has a higher average fitness than the average population fitness. This tournament selection technique allows individuals of lower quality to participate in the improvement of the population [62]. This promotes diversity of the population.

The **GB-EVO** fuzzer implements both tournament and elitist selection. In all experiments, we use tournament selection without replacement (individuals participate in exactly one tournament each). Setting the tournament size to the population size would result in elitist selection, whereas a tournament size of 1 is essentially random selection.

Specifically, we use a tournament size $s = 4$. Each individual participates in only one tournament. Each tournament yields $\frac{s}{2}$ winners. This way we always end up with exactly n individuals in the next generation (since the selection pool is of size $2n$: the current population and its offspring).

6.3.3 Generation and mutation

The generation and mutation of individuals is the same as for the grammar-based mutational fuzzer.

6.4 Tool usage

GEFRA can be run inside a docker container using various arguments such as the termination criteria, the population size, the heuristic function used and the mutation-generation

ratio. Figure 6.4 shows how an experiment using the tool (and a docker image) is started from the command line.

A guide to build the docker images containing the tool and the SUTs (Ripple and Ganache) is available on GitHub. A release of GEFRA that was used during the experiments can be downloaded from GitHub on the releases page.

It is possible to use GEFRA to generate tests for other JSON-RPC systems as well. This requires the change of a few settings in the implementation of the tool. Specifically, the url to the server and the path to the OpenRPC specification file must be set. The test drivers for the SUTs in this research cannot be used for other SUTs. However, it is possible to set the test driver to the implemented *SimpleTestDriver*, a basic test driver that works with a server url. Afterwards, the software project can be build, and used on the desired JSON-RPC system to generate (and store) tests.

6. IMPLEMENTATION

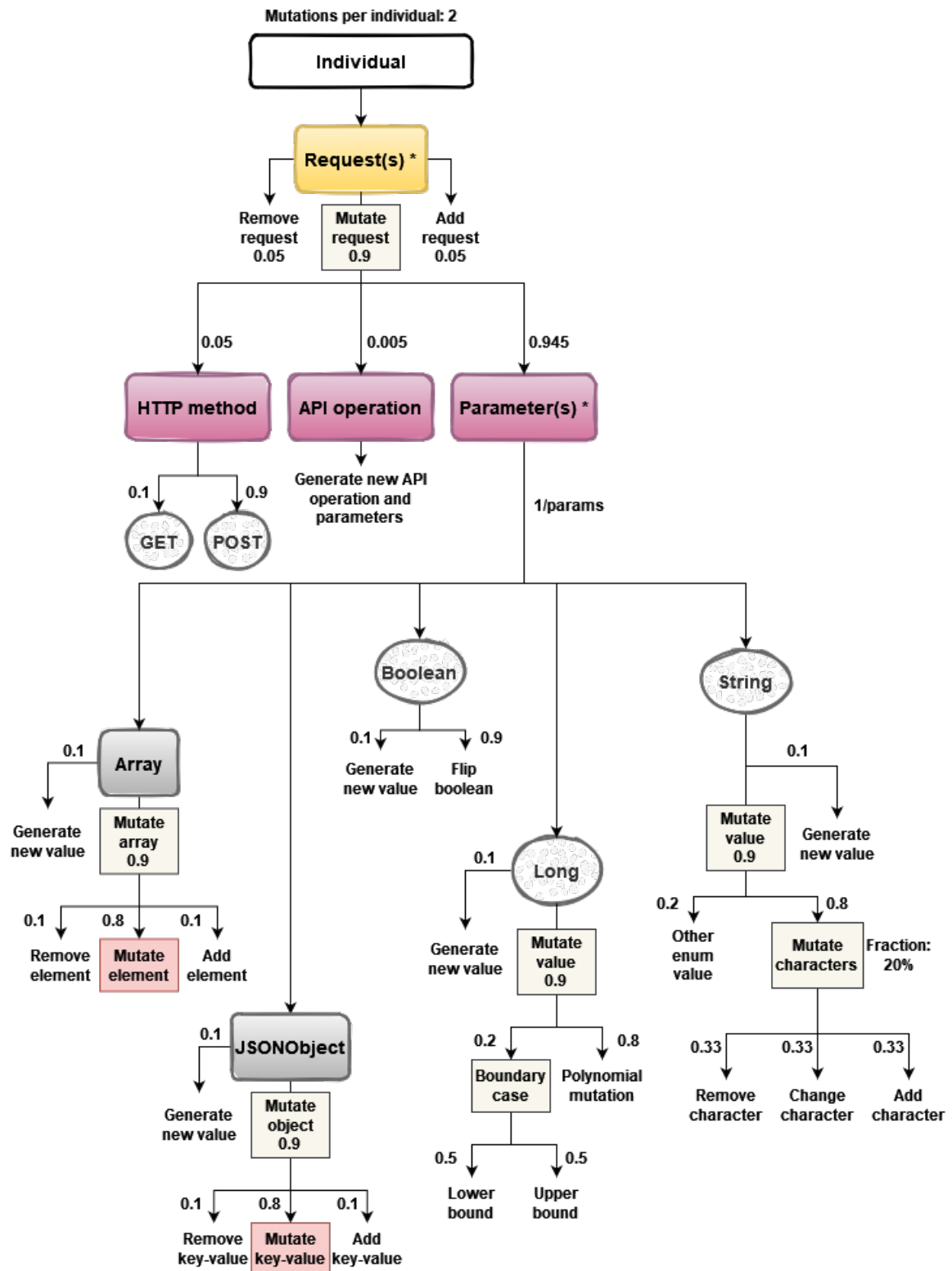
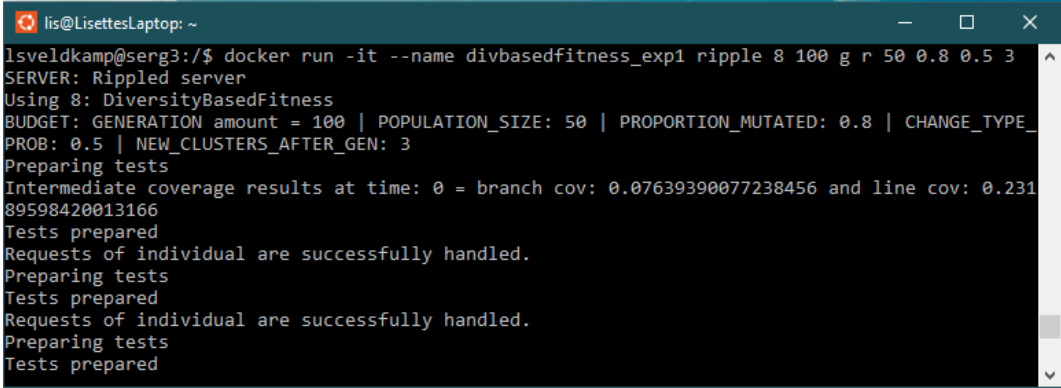


Figure 6.2: Flow of the custom mutation of an individual

A terminal window titled 'lis@LisettesLaptop: ~' showing the execution of a Docker command. The command is 'docker run -it --name divbasedfitness_exp1 ripple 8 100 g r 50 0.8 0.5 3'. The output shows the tool's initialization, including server status, configuration parameters (8: DiversityBasedFitness, BUDGET: GENERATION amount = 100, POPULATION_SIZE: 50, PROPORTION_MUTATED: 0.8, CHANGE_TYPE_PROB: 0.5, NEW_CLUSTERS_AFTER_GEN: 3), and a series of test preparation steps. The output includes 'Preparing tests', 'Intermediate coverage results at time: 0 = branch cov: 0.07639390077238456 and line cov: 0.23189598420013166', and a repeating cycle of 'Tests prepared' and 'Requests of individual are successfully handled.'

```
lis@LisettesLaptop: ~  
lsveldkamp@serg3:/$ docker run -it --name divbasedfitness_exp1 ripple 8 100 g r 50 0.8 0.5 3  
SERVER: Rippled server  
Using 8: DiversityBasedFitness  
BUDGET: GENERATION amount = 100 | POPULATION_SIZE: 50 | PROPORTION_MUTATED: 0.8 | CHANGE_TYPE_  
PROB: 0.5 | NEW_CLUSTERS_AFTER_GEN: 3  
Preparing tests  
Intermediate coverage results at time: 0 = branch cov: 0.07639390077238456 and line cov: 0.231  
89598420013166  
Tests prepared  
Requests of individual are successfully handled.  
Preparing tests  
Tests prepared  
Requests of individual are successfully handled.  
Preparing tests  
Tests prepared
```

Figure 6.3: Running the tool from the command line

7

Empirical evaluation

We have implemented a prototype tool, named GEFRA, for the proposed grammar-based evolutionary fuzzing approach. GEFRA contains a traditional grammar-based fuzzing approach and a grammar-based evolutionary fuzzing approach with several objective functions. In this chapter, the empirical study to evaluate the grammar-based fuzzing approaches is described.

First, in Section 7.1, the performance metrics are deliberated on. Then, the benchmark APIs that the fuzzing approaches were tested on are introduced in Section 7.2. Section 7.3 goes into the experimental protocol. The configuration of parameters for the fuzzing approaches is talked about in Section 7.4. Finally, Section 7.5 discusses the threats to validity and reproducibility of this research.

7.1 Performance metrics

Since the effectiveness of tests is statistically correlated to code coverage [60], the measurable goal of evolutionary testing is to reach the highest coverage possible. Moreover, a larger code coverage indicates a higher possibility of discovery of faults [138]. We look at branch coverage to evaluate the effectiveness of the approach. Branch coverage is equivalent to code path coverage. Branch coverage increases when the target program executes a new branch in the code. We investigate the final branch coverage that can be achieved by the implemented fuzzing approaches, as well as the coverage over time to learn about the efficiency of the approaches.

A different metric that provides insight into the effectiveness of a fuzzing approach is the discovery of 5xx HTTP status codes. Fuzzing tools, as described in Chapter 4, often make use of HTTP codes as an indicator of faults. We keep track of all HTTP status codes that occurred. Furthermore, while fuzzing, test cases that result in 5xx status codes (server errors) are automatically stored in the archive.

7.2 Benchmark APIs

To measure the performance of the evolutionary fuzzing tool, it is used on two large and popular blockchain systems using a JSON-RPC API, namely Ganache¹ (an Ethereum simulator) and Ripple². Both servers are run locally.

Ethereum provides a complete OpenRPC specification in their repository [41]. Ripple does not have an OpenRPC specification. We created a basic specification based on the information as documented on their website [67] using the OpenRPC playground [83] to help with construction of a valid specification. All non-admin API operations and corresponding parameters (and parameter constraints) are included. This resulted into 34 unique PI operations. The Ethereum specification contains 47 API operations.

The objective of the fuzzing approach is to achieve structural coverage. The Ganache blockchain system is written in (mostly) JavaScript. The Ripple blockchain system is written in C++. To compute the coverage of the **SUT**, we rely on the *nyc* tool (for Ganache) and on the *gcov* tool (for Ripple).

7.3 Experimental protocol

All experiments are conducted on a cluster of computers with an Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz (20 cores, 40 threads) and Ubuntu 20.04. Unless mentioned otherwise, experiments are repeated 10 times. Repeating the experiments is important to account for noise from the operation system (OS) and randomness in the fuzzer. To reduce the influence of OS noise even more, only 10 threads are used simultaneously to minimise any interference.

GEFRA contains two types of fuzzers: a grammar-based mutational (**GB-MUT**) fuzzer and a grammar-based evolutionary (**GB-EVO**) fuzzer. We assess the performance of the **GB-EVO** fuzzing approach by comparing it to the **GB-MUT** fuzzing approach. The **GB-MUT** fuzzer with a mutation rate of 0% (a traditional grammar-based fuzzer) serves as a baseline. In addition, we compare the **GB-EVO** fuzzing approach to the **GB-MUT** fuzzing approach with the same mutation-generation ratio. The **GB-MUT** fuzzer is a logical baseline since our evolutionary approach extends it with an objective function to mutate individuals selectively rather than random.

The raw coverage values are noisy. Some branches are always covered regardless of the fuzzing approach used, while many others cannot be reached through the API. We do not know what amount of **SUT** coverage can be achieved through the AI. The Ripple and Ganache systems are too complex to analyse with respect to which code paths can be exercised. Therefore, we look at the absolute number of branches covered. Before conducting

¹Available at <https://github.com/trufflesuite/ganache-cli-archive.git>

²Available at <https://github.com/XRPLF/rippled/releases/tag/1.6.0>

any experiments, we evaluate what branch coverage is obtained by running the benchmark API servers for 6 hours without interference of the tool. While most experiments ran for longer than 6 hours, no additional coverage is obtained after half an hour of running the server by itself. Moreover, the standard deviation is negligible, indicating that the number of branches covered does not vary much. The obtained branch coverage for the benchmark APIs is visible in Table 7.1. The servers are run in stand-alone mode, which means they cannot contact other (blockchain) nodes. The coverage is deducted from the final results to obtain a realistic idea of the number of branches covered by the implemented fuzzing approach.

In total, the Ripple system contains 130,246 branches. The Ganache system contains 3177 branches. It is unknown how many of the branches can be covered through the API, which is why we evaluate the absolute number of covered branches.

	Ripple		Ganache	
	<i>Mean</i>	<i>STD</i>	<i>Mean</i>	<i>STD</i>
Branch coverage	7293	3.77	120	0

Table 7.1: The mean and standard deviation of branches covered after running the benchmark API servers for 6 hours (3 run average).

We use the Wilcoxon rank-sum test with the threshold $\alpha=0.05$ to determine the significance of differences among a simple grammar-based fuzzer that only generates new requests (the baseline), grammar-based mutational fuzzer (**GB-MUT**), and the grammar-based evolutionary fuzzer (**GB-EVO**). The statistical analysis is complemented with the Vargha-Delaney statistic (\hat{A}_{12}) to measure the effect size of the results. If $\hat{A}_{12} > 0.5$, the fuzzing approach yields better coverage than the approach it is compared to. If there is no difference between the performances of two fuzzing approaches, $\hat{A}_{12} = 0.5$.

7.4 Configuration

There are many parameter settings that can be configured for the implemented fuzzing approaches. Appendix A lists the available parameters and their default values. Most are related to the mutation process of individuals. These parameter values are displayed in Figure 6.2 as well. Other parameters are related to the evolutionary algorithm.

A full grid search of optimal parameters is not feasible due to time constraints, and also not the purpose of this research. We aim to find a configuration that is merely suitable to demonstrate the effectiveness of evolutionary fuzzing. Through experimentation and inspiration from related work (the EvoMaster [9] implementation), we define an acceptable baseline for the default configuration.

Important parameters are the population size, the proportion of individuals that is mutated each generation (instead of replaced by a newly generated individual) and the number

of requests that an individual can have at maximum. The maximum number of requests is set at 1 so the comparisons we make are based on an equal amount of requests that are sent to the server. The choice of a large population size puts more emphasis on the exploration (rather than exploitation) of the search landscape [8]. Since the SUTs contain 34 and 47 distinct API operations with many different sets of parameters, exploration is very important to not get stuck in local optima.

Furthermore, for the diversity-based fitness function, the number of generations that is processed before re-clustering is set at 3.

7.5 Threats to validity and reproducibility

Threats to validity are mostly related to the random nature of fuzzers. To address this threat, we run each fuzzing heuristic 10 times on each benchmark API independently using docker containers. Furthermore, we analyse the median results and draw conclusions based on non-parametric statistical tests (Wilcoxon and Vargha-Delaney).

Another threat relates to the generalisability of our results. The fuzzing approaches are tested on two complex and popular blockchain systems written in C++ (Ripple) and JavaScript (Ethereum). Additional experiments on other JSON-RPC systems would be beneficial to increase confidence in generalisability to other projects. There are unfortunately few open-source software systems that use JSON-RPC API and describe how the server can be set up and run locally. Added to this, most JSON-RPC systems do not have an OpenRPC specification (or any other description of the API operations).

All experiments were conducted independently in docker containers. The results can be reproduced by running docker containers (one for each experiment) and specifying the arguments corresponding to the experiment to conduct. Upon finishing the experiment, a CSV file is produced containing the results. The docker images (for the two SUTs Ripple and Ganache) can be build following the guide that is available on GitHub.

8

Results

In Chapter 6, the workings of the grammar-based evolutionary fuzzing solution for JSON-RPC API systems have been laid out. Chapter 7 explained how the solution was empirically evaluated. Specifically, the effectiveness (branch coverage) and efficiency (coverage over time) of the tool is evaluated on two blockchain systems: Ripple and Ethereum.

This chapter presents the found results per research sub question. First, it is evaluated whether an evolutionary fuzzing approach can outperform traditional grammar-based fuzzing in terms of branch coverage (over time) in Section 8.1. Then, Section 8.2 evaluates the efficiency of the implemented fitness functions used in the evolutionary fuzzing approach with regards to branch coverage.

8.1 Evolutionary fuzzing performance

The first research sub question is the following:

1. How effective is evolutionary fuzzing with regards to structural coverage in comparison to grammar-based fuzzing for JSON-RPC APIs?

To answer this question, we first need to evaluate the performance of the grammar-based fuzzing approach and whether mutational fuzzing (the basis for the evolutionary fuzzer) is able to outperform it. Subsection 8.1.1 presents the results of the experiments with the grammar-based fuzzing and the grammar-based mutational fuzzing approach on the two benchmark APIs: Ripple and Ganache. Subsection 8.1.2 continues with the evaluation of the evolutionary fuzzing approach.

8.1.1 Grammar-based fuzzing

Before experimenting with the evolutionary fuzzing approach, we want to determine whether GB-MUT fuzzing is actually able to achieve more coverage for both SUTs, compared to the grammar-based fuzzing approach (i.e., generating input values without ever mutating existing input). We will refer to the latter as the baseline.

8. RESULTS

Initially, we use a 50% mutation ratio for the **GB-MUT** fuzzer. This means that for each generation, half of the individuals are created by mutating a random selection of individuals from the population, and half of the individuals are newly generated based on the grammar. Performing mutations on test cases can result in invalid requests (i.e. faulty test cases). It is thus expected that more code paths are discovered (resulting in more covered branches). The baseline solution (0% mutation ratio) generates new individuals each generation based on the grammar.

For the first **SUT**, Ganache, the obtained branch coverage quickly converges. This is illustrated in Figure 8.1. After 200 evaluations (approximately 100 minutes), all runs of the experiments have reached a limit. This holds for both the baseline and the **GB-MUT** fuzzer, although the **GB-MUT** fuzzer is able to reach the limit quicker, after only 2 generations (approximately 50 minutes).

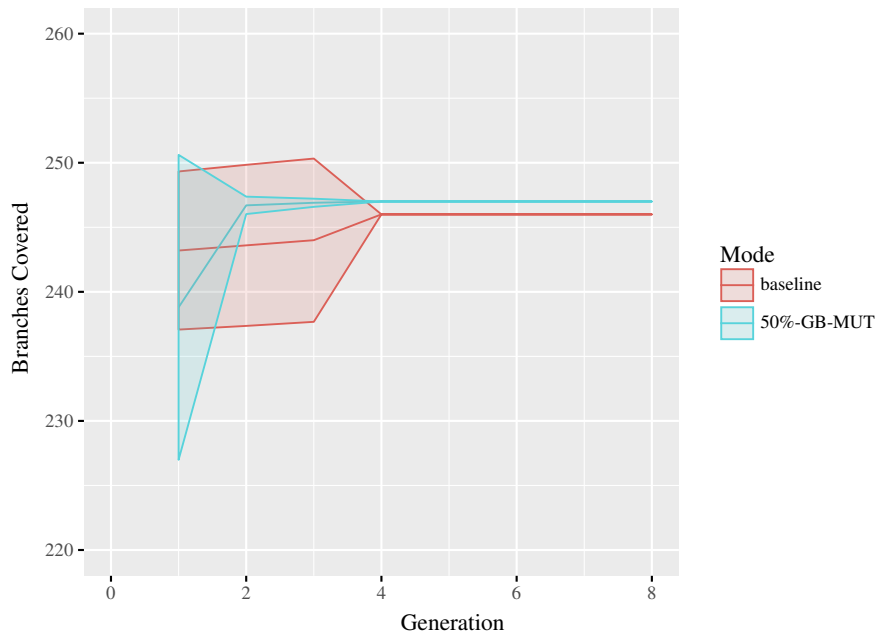


Figure 8.1: Ganache branches covered over generations (10 runs per experiment). One generation contains 50 individuals and equals 50 fitness evaluations. On average, evaluation of one generation takes approximately 25 minutes.

The baseline covers 246 branches of the **SUT**. The grammar-based mutational fuzzer covers 247 branches. The interquartile range¹ (IQR) is 0 for both approaches, indicating that each run of the experiments resulted in the same amount of branches covered. The

¹The interquartile range of a dataset is the difference between the first quartile and third. It measures the spread of the middle 50% of values.

consistent difference of one branch can be explained by the fact that there is one branch that can only be achieved by a test case that is mutated. However, due to the fact that there seems to be a limit to the coverage that can be obtained, and it is so quickly reached (after only 200 evaluations), we cannot evaluate whether **GB-MUT** fuzzing is more effective than the baseline. The total amount of branches that can be covered through the **API** is unknown, but it is unlikely that every code path that can be reached is covered.

Upon further investigation we find that the number of different responses that are retrieved from the **API** is minimal. Common responses indicate an "Incorrect number of arguments" and "Method not supported" message. This indicates that the OpenRPC specification (provided by Ethereum) may not be valid. The fuzzer is unable to reach past the validation barriers of the **SUT**, and more than 247 branches cannot be covered.

The second **SUT**, Ripple, shows different results. We compared several configurations for the **GB-MUT** fuzzer by considering different mutation-generation ratios. Figure 8.2 illustrates the branches covered after running the baseline and **GB-MUT** fuzzer for 100 generations, a population size of 50, and different mutation percentages.

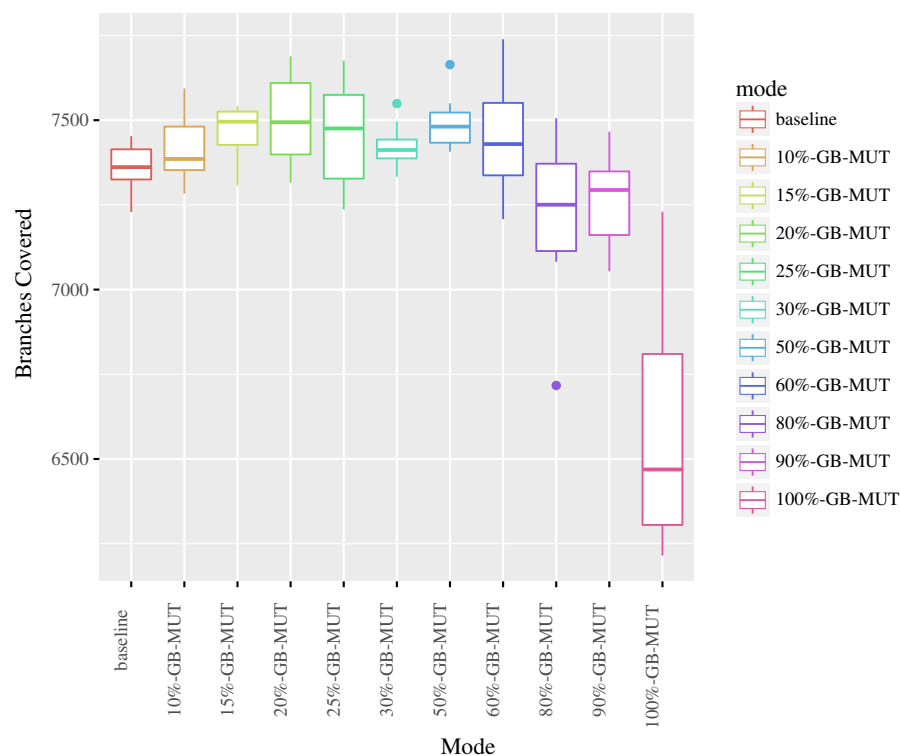


Figure 8.2: Branches covered on Ripple (10 runs per experiment) after 100 generations of 50 individuals each. On average, evaluation of one generation takes approximately 12 minutes.

8. RESULTS

As mentioned previously, the baseline fuzzer is equal to the **GB-MUT** fuzzer with a mutation rate of 0%. The baseline is able to achieve a median of 7362 newly covered branches. The **GB-MUT** fuzzer performs better than the baseline for almost all mutation percentages, obtaining up to 7494 discovered branches, indicating that test cases deviating from the grammar are effective in discovering new branches. 100% mutation performs worst, which is expected, since mutations alter individuals in small ways, and the Ripple blockchain system has a large search space (34 API operations). In this case, a combination of exploration and exploitation works better than only exploration (the baseline) or only exploitation (the 100% **GB-MUT** fuzzer).

Table 8.1 shows the median and IQR of the branches covered by the baseline and **GB-MUT** fuzzer. While all **GB-MUT** fuzzers below a mutation rate of 80% perform better than the baseline, the 20%-**GB-MUT**, 30%-**GB-MUT** and 50%-**GB-MUT** fuzzer perform significantly better than the baseline (p -value < 0.05), but not significantly better than each other. The 90%-**GB-MUT** and 100%-**GB-MUT** fuzzers perform significantly worse than the baseline. The significance of the differences between the **GB-MUT** fuzzing approaches and the baseline can be found in Table 8.2.

Branches covered			GB-MUT fuzzer vs baseline		
	Median	IQR		p -value	\hat{A}_{12}
baseline	7361.5	89	10% mutation	0.375	0.67 (medium)
10% mutation	7385.5	128.25	20% mutation	0.027	0.81 (large)
20% mutation	7494	211	30% mutation	0.037	0.72 (medium)
30% mutation	7412	55.50	50% mutation	0.004	0.90 (large)
50% mutation	7481	89	60% mutation	0.193	0.65 (small)
60% mutation	7429.5	213.75	80% mutation	0.131	0.35 (small)
80% mutation	7250.5	257.75	90% mutation	0.049	0.27 (medium)
90% mutation	7294	187.75	100% mutation	0.002	0.01 (large)
100% mutation	6469	504.75			

Table 8.1: Median and IQR of covered branches for Ripple (100 generations, population size of 50).

Table 8.2: Wilcoxon rank-sum test and Vargha-Delaney statistic for Ripple branch coverage (100 generations, population size of 50).

Figure 8.3 depicts the obtained branch coverage over time by the baseline, the 20%-**GB-MUT** fuzzer (significantly better than the baseline) and the 100%-**GB-MUT** fuzzer (significantly worse than the baseline). There are no significant differences between the Area Under the Curve (AUC) of the better performing **GB-MUT** fuzzers. None of the 20%-**GB-MUT**, 30%-**GB-MUT** or 50%-**GB-MUT** fuzzers reaches coverage significantly faster than each other.

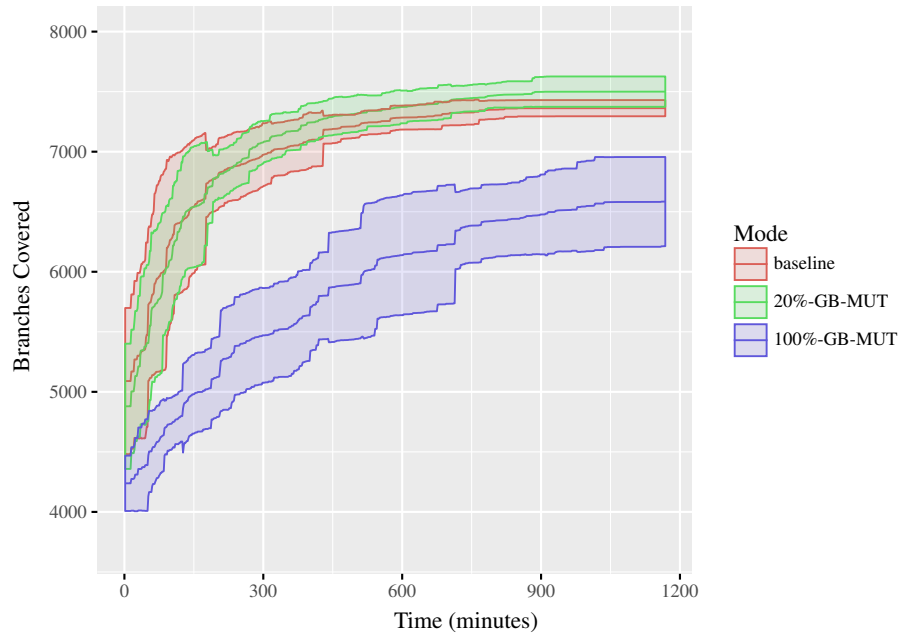


Figure 8.3: Ripple branches covered over time (10 runs per experiment). 100 generations of 50 individuals each.

In conclusion, there is no significant difference between using a lower or higher mutation rate for the **GB-MUT** fuzzer, as long as the rate is not too low (0%) or high (> 80%) so the balance between exploration and exploitation is preserved. We do observe that the grammar-based mutational approach performs significantly better than the baseline in achieving branch coverage. This means that there is potential for the evolutionary fuzzing approach to improve upon the **GB-MUT** approach. For both of the **SUTs**, no server errors (5xx HTTP status codes) were found.

8.1.2 Evolutionary fuzzing

The results showed that grammar-based mutational fuzzing is able to outperform traditional grammar-based fuzzing. We now investigate whether the **GB-EVO** fuzzer is more effective at obtaining branch coverage than the baseline and the **GB-MUT** fuzzer.

In the previous experiments we saw that the baseline performs quite well, likely due to the large search space. For the next experiments we increase the population size to 100 to take into account the large search space. Furthermore, we use a mutation-generation ratio of 80%. This allows us to better observe the effects of the evolutionary selection, which differentiates the **GB-EVO** fuzzer from the **GB-MUT** fuzzer. We use the diversity-based

8. RESULTS

fitness function, as defined in Section 6.3.1.

Figure 8.4 displays the distribution of the branches covered by the three fuzzing approaches: the baseline, GB-MUT fuzzing with an 80% mutation-generation ratio, and GB-EVO fuzzing with an 80% mutation-generation ratio. The medians and IQRs are presented in Table 8.3. We observe that the GB-EVO approach achieves a greater number of branches covered compared to the baseline and GB-MUT approach. The median number of branches covered over the 10 experiment runs is 7742: 240 branches more than the baseline. The GB-EVO fuzzer performs significantly better than the baseline (p -value=0.006), but is not able to significantly outperform the GB-MUT approach with regards to the final obtained coverage. Furthermore, the 80%-GB-MUT fuzzer is not significantly better than the baseline. This was also the case for the experiments with a population size of 50, described in Section 8.1.1. The significance statistics are visible in Table 8.4.

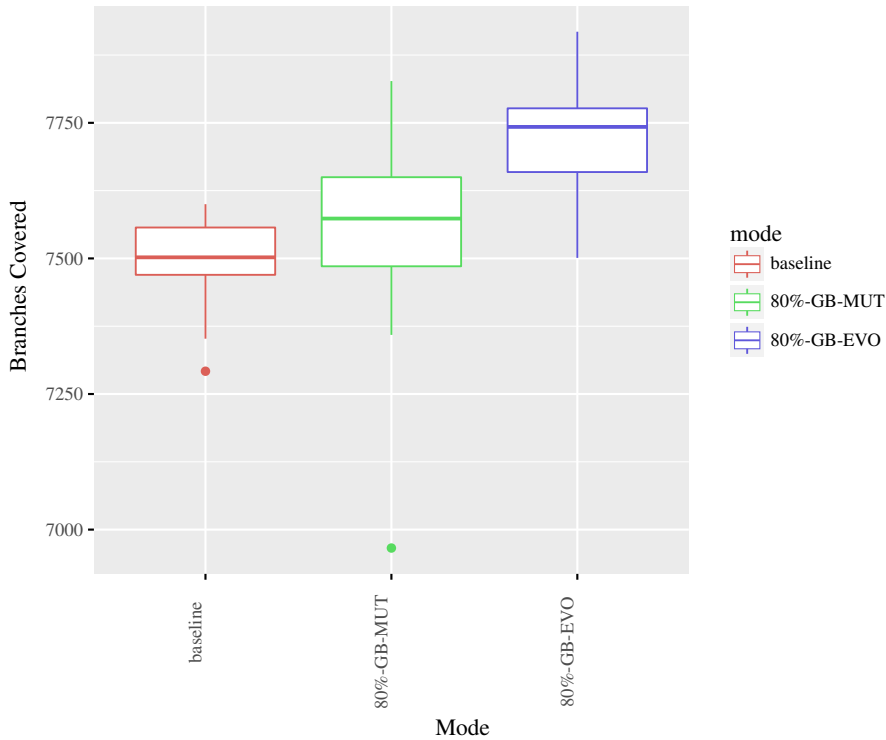


Figure 8.4: Branches covered on Ripple (10 runs per experiment) after 100 generations of 100 individuals each. One generation equals 100 fitness evaluations and on average takes approximately 22 minutes to be evaluated.

Branches covered		
	Median	IQR
baseline	7502	87.25
80%-GB-MUT	7573.5	164.25
80%-GB-EVO	7742.5	117.50

Table 8.3: Median and IQR of covered branches for Ripple (after 10000 evaluations).

GB-MUT fuzzers vs baseline		
	p -value	\hat{A}_{12}
80%-GB-EVO	0.006	0.09 (large)
80%-GB-MUT	0.557	0.33 (medium)

Table 8.4: Wilcoxon rank-sum test and Vargha-Delaney statistic for Ripple branch coverage (after 10000 evaluations).

The **GB-EVO** fuzzer does not obtain significantly higher coverage than the **GB-MUT** fuzzer. However, we consider the efficiency of the fuzzing approaches as well, by looking at the obtained coverage over time. The branch coverage obtained over time for the **GB-EVO**, **GB-MUT** and the baseline is plotted in Figure 8.5.

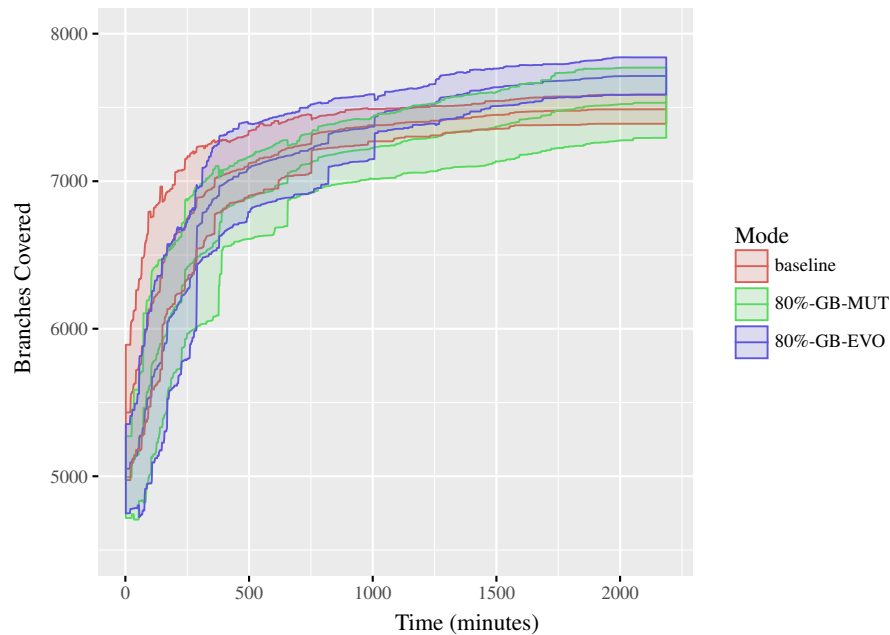


Figure 8.5: Rippled branches covered over time (10 runs per experiment).

It can be seen that baseline starts off best. This makes sense as in the beginning of the fuzzing process there are lots of easily reached branches to be discovered. A high level of exploration at this point yields better results. After several generations, the baseline is surpassed by the evolutionary fuzzer. In Figure 8.6 we can see that this is around the 35th generation (i.e. after 3500 individuals have been processed). The **GB-MUT** is much slower,

8. RESULTS

overtaking the baseline only after the 80th generation.

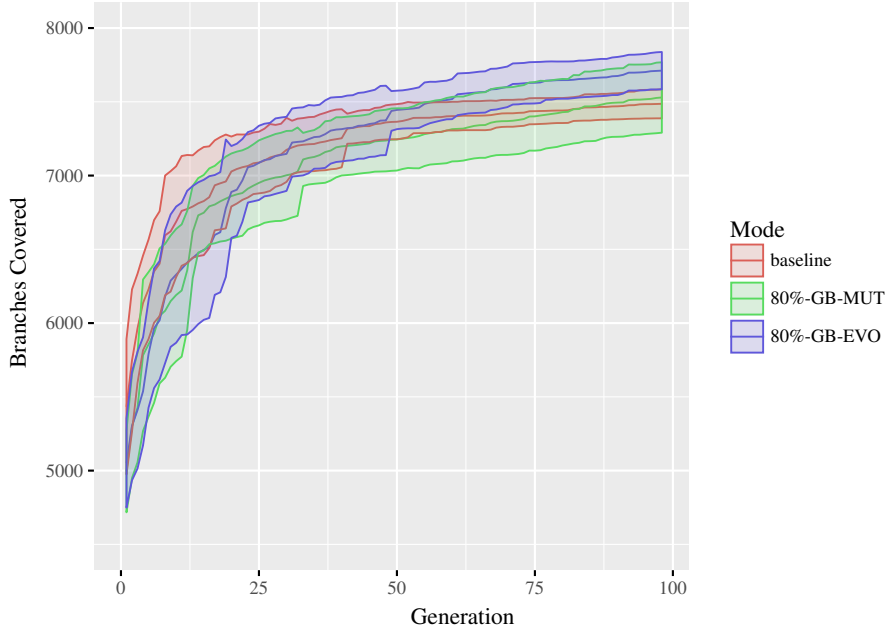


Figure 8.6: Ripple branches covered over generations (10 runs per experiment). One generation contains 100 individuals and equals 100 fitness evaluations. On average, evaluation of one generation takes approximately 22 minutes.

To see whether the **GB-EVO** fuzzer is significantly faster in obtaining coverage than the **GB-MUT** fuzzer, we compute the **AUC**. Table 8.5 contains the median values and **IQR** for the **AUC** of coverage over time. The corresponding significance statistics can be found in Table 8.6. Furthermore, the distribution of the **AUC** is depicted in Figure 8.7. The **GB-EVO** fuzzer has a significantly greater **AUC**, which means it achieves structural coverage significantly faster than the **GB-MUT** fuzzer.

AUC of covered branches		
	Median	IQR
baseline	9,493,502.5	245,052.25
80%-GB-MUT	9,336,025	167,497
80%-GB-EVO	9,523,086	130,390.50

Table 8.5: AUC for covered branches over time for Ripple (10000 evaluations).

GB-MUT fuzzers vs GB-EVO		
	p -value	\hat{A}_{12}
baseline	1.00	0.52 (negligible)
80%-GB-MUT	0.01	0.78 (large)

Table 8.6: Wilcoxon rank-sum test and Vargha-Delaney statistic for Ripple AUC for covered branches over time (10000 evaluations).

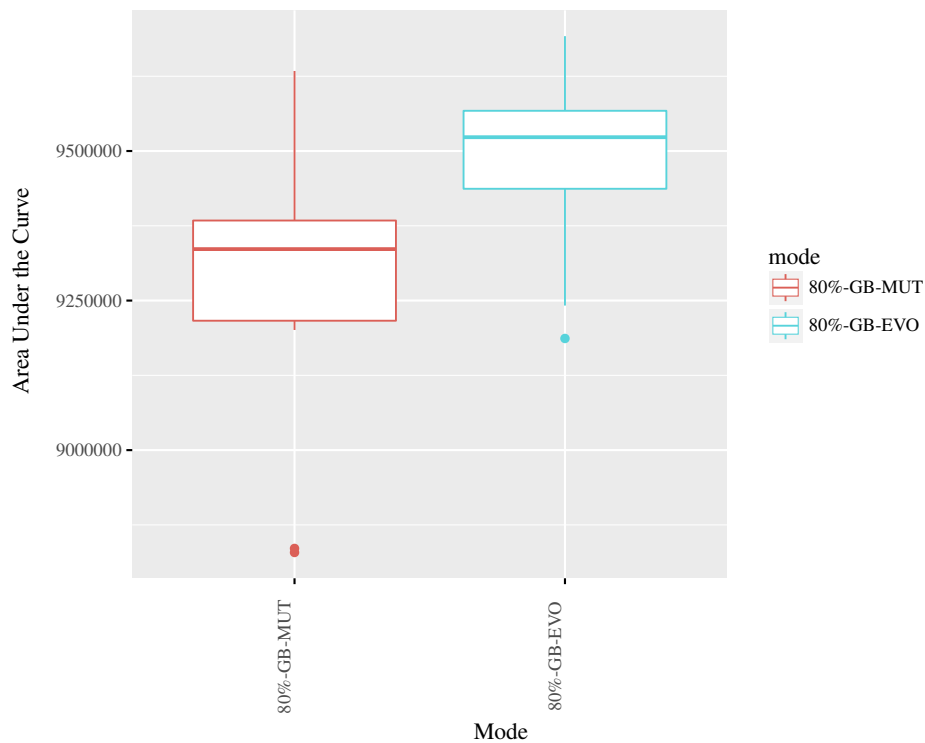


Figure 8.7: AUC for branches covered over time (10 runs per experiment).

In summary, evolutionary fuzzing was able to achieve near 7750 of branches in the Ripple blockchain system. This is a significant improvement from traditional grammar-based fuzzing (which covered 7500 branches). Furthermore, evolutionary fuzzing achieves structural coverage significantly faster than grammar-based mutational fuzzing.

8.2 Suitability of fitness functions

From the previous sub question we found that evolutionary fuzzing (with a suitable fitness function) obtains a significantly higher amount of covered branches than grammar-based fuzzing. Furthermore, we saw that evolutionary fuzzing is more efficient than grammar-based mutational fuzzing, achieving coverage quicker.

The second research sub question focuses on the suitability of various fitness functions for evolutionary fuzzing. The sub question is:

2. How do different fitness functions for the evolutionary fuzzing approach compare to each other with regards to structural coverage?

In Chapter 5, four fitness functions aimed at promoting diversity were defined. The above

8. RESULTS

experiments were conducted using the *DiversityBasedFitness* fitness function. We want to investigate whether the same results in performance can be achieved by using simpler fitness functions, specifically the *SkeletonFitness*, the *SkeletonAndComplexityFitness*, and the *SimpleResponseFitness* functions.

The *DiversityBasedFitness* clusters responses based on the keys (i.e. parameter names) in the response object and the values of these keys. *SkeletonFitness* only considers the keys of a response object and disregards the values. *SkeletonAndComplexityFitness* is essentially a weighted version of *SkeletonFitness*, with the weight being the number of parameters in the request object. Lastly, *SimpleResponseFitness* considers the keys of a response object, and groups the values into a limited number of predefined categories.

The **GB-EVO** approach with the above fitness functions was evaluated on the Ripple system. Again, a mutation-generation ratio of 80% and a population size of 100 was chosen. The experiments were run for 100 generations. The median and **IQR** values are shown in Table 8.7. Figure 8.8 illustrates the distribution of obtained branch coverage by using the four fitness functions in the **GB-EVO** fuzzer.

The significance of the differences can be found in Table 8.8. All fitness functions perform significantly worse than the *DiversityBased* fitness. Moreover, they cannot (significantly) outperform the baseline. This means that the *SkeletonFitness*, *SkeletonAndComplexityFitness* and *SimpleResponseFitness* are inadequate objective functions to promote diversity in the population.

Branches covered		
	Median	IQR
baseline	7502	87.25
skeleton	7571.5	169.75
skeletonAndComplexity	7563.5	62.50
simpleResponse	7505	128.75
diversityBased	7742.5	117.50

Table 8.7: Median and IQR of covered branches for Ripple (after 10000 evaluations).

Fitness functions vs diversityBased		
	<i>p</i> -value	\hat{A}_{12}
simpleResponse	0.002	0.09 (large)
skeletonAndComplexity	0.049	0.19 (large)
skeleton	0.010	0.16 (large)

Table 8.8: Wilcoxon rank-sum test and Vargha-Delaney statistic for Ripple branch coverage (after 10000 evaluations).

Since all fitness functions, including the *DiversityBasedFitness*, make use of the response skeleton to differentiate between individuals, the significant difference in performance must be related to the consideration of values in the fitness functions.

Both response skeleton fitness functions completely disregard the parameter values in object responses. These results show that parameter names alone are insufficient to (correctly) differentiate between response objects. During experimentation, we found that the

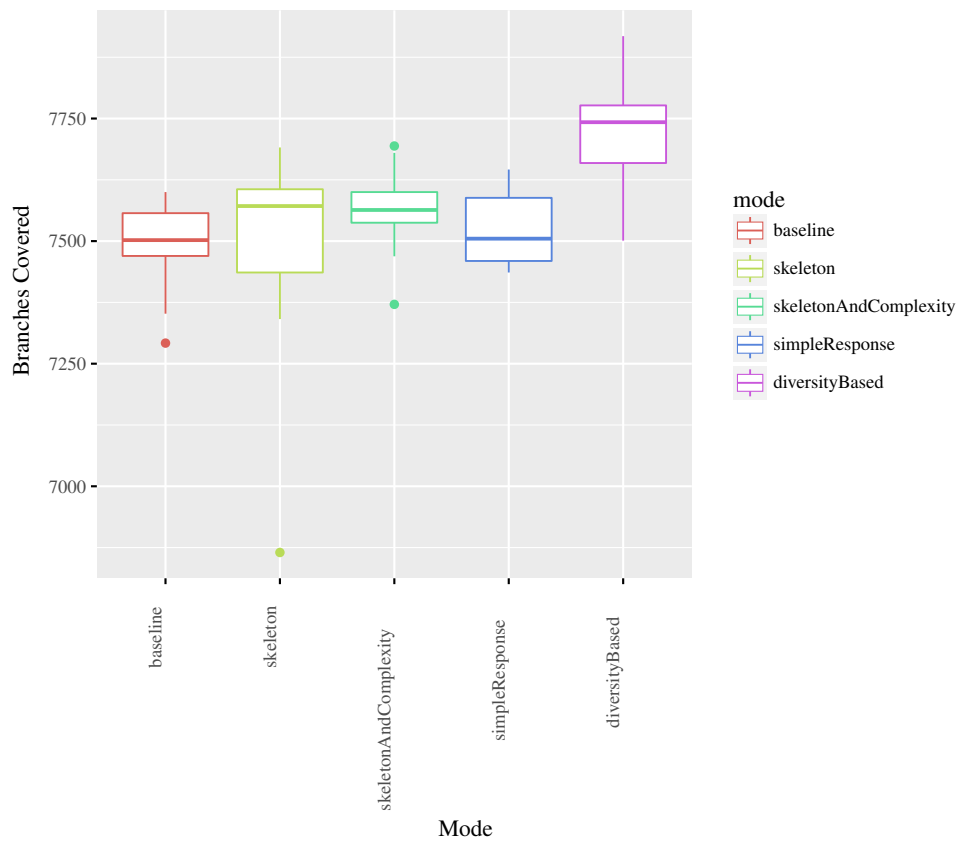


Figure 8.8: Ripple branches covered over generations (10 runs per experiment). One generation contains 100 individuals and equals 100 fitness evaluations, and on average takes approximately 22 minutes to be evaluated.

Ripple system can return various error responses with exactly the same parameter fields (e.g. *error_message* and *error_code*), while the values of parameters are different, indicating that different code paths were exercised. The skeleton fitness functions are unable to differentiate between such responses. In case of the *SimpleResponseFitness*, the predefined categories used are simply too basic to differentiate between different parameter values. For example, many distinct *error_message* values would fall under the category of a short string. With more refined categories tailored to the SUT, this approach might perform better (but would no longer be black-box).

8. RESULTS

Ultimately, the *DiversityBasedFitness* is the only fitness function of the four that, used in the **GB-EVO** fuzzer, outperforms not only the traditional grammar-based fuzzer (with regards to effectiveness), but also the **GB-MUT** fuzzer (with regards to efficiency). This illustrates that the parameter values of a response object play a crucial role in the identification of unique response objects.

9

Discussion

In Chapter 8 the results of this study were presented. The discussion with regard to the conducted research is outlined in this chapter. Section 9.1 discusses the implemented tool that was developed in this study and its findings. Section 9.2 elaborates on the limitations of this research.

9.1 Grammar-based Evolutionary Fuzzer for RPC-APIs

This study introduces a black-box grammar-based evolutionary fuzzing approach, specifically tailored to JSON-RPC systems. To the best of our knowledge, [GEFRA](#) is the first open-source fuzzing tool to find faults in JSON-RPC APIs. Since no access to the source code is required, [GEFRA](#) can generate tests for JSON-RPC systems written in any programming language, running on local or remote servers.

The main finding of this research is that evolutionary fuzzing, using an objective fitness that favours diversity in individuals, can improve upon traditionally grammar-based fuzzing for JSON-RPC APIs. The designed objective fitness makes use of agglomerative clustering in a way that it is scalable for large populations of individuals.

Fuzzers optimize code coverage, and there is a strong relation between increasing code coverage and finding code defects. Evolutionary fuzzing obtains a greater structural coverage in comparison to grammar-based fuzzing, provided it is given enough time. Moreover, this research has shown that offspring selection with a suitable objective function improves efficiency of fuzzing in comparison to random selection. When the fuzzer runs for a short period of time, exploration is preferred over exploitation, in which case a traditional grammar-based fuzzer is more effective. The minimum run time required for evolutionary fuzzing to be effective depends on the complexity of the [SUT](#). For a complex system (with many API operations and corresponding parameters), evolutionary fuzzing requires more time to be able to obtain greater structural coverage than grammar-based fuzzing.

Ideally, we would like to evaluate what percentage of all code branches can be covered with evolutionary fuzzing. Unfortunately, we do not know how many branches can be reached through the API. The complexity of the benchmark systems make it impossible (for a non-

expert) to determine the number of reachable branches (out of the total 3,177 branches in Ganache and 130,246 branches in Ripple). Nevertheless, the total obtained coverage by the evolutionary fuzzer (including the branches run by starting the server) is however thought to be relatively low: 11.6% for Ganache (after four generations \approx 1.5 hours) and 11.5% for Ripple (after 100 generations \approx 36 hours). It should be noted that there was large computational overhead in the experiments due to the computation of coverage in between generations and re-starting the server for each test. Compared to this, the internal operations of the fuzzer (e.g. mutations, clustering) are negligible with regards to time. This is supported by the observation that different fuzzing approaches do not differ with regards to the time it costs (on average) to evaluate one population.

Furthermore, [GEFRA](#) was not able to find faults (i.e. system server errors) in the benchmark APIs. This was not surprising considering the fact that both systems are widely used and well-tested.

9.2 Limitations

While producing reasonably good test cases (that discover new code paths), [GEFRA](#) has a number of limitations. Successful fuzzing depends on the quality of random mutations. By repeating experiments multiple times we tried to limit the influence of this randomness, but it is possible that a result is either much better or worse than it would be on average. From the experiments we found that after 10 iterations (each experiment was run 10 times) the standard deviation of obtained branch coverage can reach a value up to 5.6% of the mean for Ripple; 0 for Ganache.

Besides the stochastic nature of fuzzing, there are other aspects of the fuzzing approach that influence the findings of this research. Firstly the quality of the OpenRPC specification determines the coverage that can be achieved. Without a correct specification of API operations and parameters, it is impossible to pass validation barriers in the code, resulting in many branches remaining unvisited. For the Ripple system, we constructed an OpenRPC specification ourselves based on the available information. While handled diligently, parameter constraints could have been misinterpreted, or incorrectly converted to regular expressions¹ (in the case of string values), leading to the generation of syntactically invalid requests. Added to this, the tested JSON-RPC APIs are both related to blockchain applications, and for Ganache we were unable to evaluate the effectiveness of evolutionary fuzzing due to the limited number of branches that could be covered. The results may not be generalisable to other JSON-RPC systems.

Secondly, the performance of [GEFRA](#) depends on hyper-parameters, which can be cumbersome to tune for each [SUT](#). Specifically, the population size and mutation-generation ratio have a great effect on the obtained code coverage. Due to the time it takes to run ex-

¹A regular expression (or regex) is a sequence of characters that specifies a search pattern in text.

periments, it is difficult to find an optimal combination of parameters. One experiment (10 iterations) can take up to 36 hours. We define an acceptable baseline for the parameters that is able to illustrate the effectiveness of evolutionary fuzzing but performance may be better with hyper-parameters that are tweaked to the [SUT](#).

10

Conclusions

In this chapter, the research questions as first presented Chapter 2 are answered. First, the research sub-questions are answered in Section 10.1. The answers form the basis for the conclusions of the main research question in Section 10.2. Finally, several suggestions for future work are discussed in Section 10.3.

10.1 Research sub questions

- 1. How effective is evolutionary fuzzing with regards to structural coverage in comparison to grammar-based fuzzing for JSON-RPC APIs?*

In order to evaluate the effectiveness of the two grammar-based fuzzing approaches, a tool (named **GEFRA**) was developed that implements both strategies. Using the tool, experiments were run on two JSON-RPC systems: Ganache and Ripple. The implemented fuzzing approaches sent several generations of individuals (JSON-RPC requests) to the System Under Test (**SUT**). The initial population is generated based on a grammar, which is constructed from the OpenRPC specification of the **SUT**. While grammar-based fuzzing only generates new individuals based on a grammar, evolutionary fuzzing selects a group of individuals each generation and mutates them to build the next generation.

We found that traditional grammar-based fuzzing is able to obtain a maximum coverage of 246 branches on Ganache, and 7502 on Ripple, depending on how long the fuzzer runs. In the conducted experiments (of 10,000 evaluations), no hard limit was reached (yet).

Evolutionary fuzzing with random selection (**GB-MUT** fuzzing) can improve upon traditional grammar-based fuzzing for Ganache by only 1 branch (an increase of $< 1\%$), which leaves no room for a fitness function to optimise the selection fuzzing process. For Ripple, **GB-MUT** fuzzing consistently increased the number of covered branches by approximately 2% for 5,000 evaluations (a difference of 133 branches).

Evolutionary fuzzing with the use of an objective function was able to improve upon the performance of **GB-MUT** fuzzing. Over time, evolutionary fuzzing was quicker to obtain coverage. In comparison to traditional grammar-based fuzzing, evolutionary fuzzing

10. CONCLUSIONS

was able to significantly (p -value =0.006) obtain a larger branch coverage (an increase of approximately 3.2% for 10,000 evaluations, a difference of 241 branches).

In conclusion, there is a significant difference in performance between evolutionary fuzzing and traditional grammar-based fuzzing. Evolutionary fuzzing consistently achieves greater structural coverage. Furthermore, a suitable fitness function can significantly increase the efficiency of the fuzzer compared to random selection.

2. How do different fitness functions for the evolutionary fuzzing approach compare with regards to structural coverage?

To evaluate which aspects are important in a fitness function, four fitness functions were implemented:

1. *SkeletonFitness*
2. *SkeletonAndComplexityFitness*
3. *SimpleResponseFitness*
4. *DiversityBasedFitness*

Each has the same purpose: to favour individuals that yield response objects from the SUT that are relatively unseen. We found that the fitness function that considers both the keys *and* the corresponding values in the response object performs best. This *DiversityBasedFitness* represents response objects as feature vectors and makes use of agglomerative clustering to cluster the responses. The other fitness functions, which base the computation of the fitness value on the response object keys only, or on basic value categories, do not perform better than the grammar-based fuzzer.

Parameter values of a response object play a crucial role in the identification of unique response objects. They should be taken into account when designing a fitness function for a diversity-based evolutionary algorithm.

10.2 Research main question

The main research question Was:

How effective is grammar-based JSON-RPC fuzzing to achieve structural coverage?

We find that traditional grammar-based fuzzing is able to cover between a few hundred (on Ganache) and thousands of branches (on Ripple). Grammar-based mutational fuzzing, which enables the generation of faulty test cases, is able to increase the structural coverage for both SUTs. Evolutionary fuzzing, provided with a suitable fitness function, is able

to converge towards maximal structural coverage significantly faster than grammar-based mutational fuzzing.

Overall, grammar-based JSON-RPC fuzzing approaches work well to obtain structural coverage. However, not surprisingly, the effectiveness of grammar-based JSON-RPC fuzzing on a system depends strongly on the quality of the grammar. Furthermore, for evolutionary fuzzing, a meaningful fitness function is required. Finally, from this research follows that with regards to effectiveness and efficiency, an evolutionary fuzzing strategy is a good choice for a JSON-RPC API fuzzer.

10.3 Recommendations for future work

While the evolutionary fuzzing approach is shown to be effective in obtaining structural coverage, there is no critical selection for the tests that should be kept. Because the tool operates in a black-box setting (and code coverage can thus not be used as a metric), is not straightforward which of the generated test cases contribute to additional structural coverage. One could use a fitness threshold, but deciding upon this value is not simple either (when is a test diverse enough?). Future work could focus on the design of a suitable selection procedure to retain a minimal number of test cases while maximising the structural coverage that they achieve.

Additionally, there are several techniques that can be applied to further optimise the fuzzing approach. First of all, while the baseline is always eventually overtaken by the evolutionary fuzzer, it is generally able to achieve branch coverage faster in the beginning. A dynamic mutation rate might improve the performance of the evolutionary fuzzer. In the beginning one might want to generate many test cases to cover all parts of the grammar, and after some time the focus can shift towards mutation instead of generation (exploration vs exploitation).

Furthermore, the current implementation was evaluated with one request to the API per individual. Key-value pairs of response objects are therefore not stored and re-used in follow-up requests. Related work showed that this is however an effective way to cover deeper code paths. The fuzzing approach could be extended with this feature.

Finally, instead of single-objective function optimisation, future work could look into multi-objective function optimisation. The current (best) fitness function only takes into account the response object that was retrieved for a response. A multi-objective function can consider other relevant information as well, such as the HTTP status code or the time it takes for the server to respond to a request.

PART III

Communication Design for Innovation

11

Introduction

In Part II of this report, we evaluated a grammar-based fuzzing approach that automatically generates test cases, which can expose faults in the code base of blockchain applications. The results of Part II show that this approach is effective in obtaining test coverage. Once defects are known, the code may be patched to make the system more robust and to prevent attacks that exploit these defects.

It is difficult, if not impossible, to find and patch all code defects, and even with fool-proof code attacks may happen due to other vulnerabilities like human or physical vulnerabilities (e.g. a flood or power outage). Part III of this report addresses media framing of responsibilities concerning common blockchain attacks and discusses what developers can do to influence how responsibilities are currently assigned.

First, in Section 11.1 the problem description that was introduced in Chapter 1 is discussed in more detail. Next, in Section 11.2 the main research question and sub-questions are formulated. Section 11.3 describes the contributions of this research. Finally, in Section 11.4 the structure of the Science Communication research part of the thesis is presented.

11.1 Problem description

Today computer and smartphone users, especially employees, are expected to be aware of the risks associated with using the internet. To increase risk awareness, training, education programs and information campaigns are organized. Unfortunately these turn out to be largely ineffective [4]. It is often stated that the weakest link in the cyber security chain is human. This may be true, but with numerous elements in the cyber security chain, it is not clear this means that users should be responsible for their cyber security. Cybersecurity is in its infancy when we compare it to more established fields such as traditional crime prevention. If a bike owner forgets to lock his bike and it gets stolen, insurance will not reimburse the stolen bike, as this precaution is widely regarded as a reasonable expectation, and it is clear that the bike owner is responsible for taking this precaution.

When it comes to blockchain technology, it is not evident who is expected to take responsibility for the prevention of cyber attacks on blockchain applications and for handling the consequences. Victims of cyber attacks are often held (partly) responsible for the crime,

owing to not having taken deliberate action to make themselves less vulnerable [93]. This is an example of victim blaming. The phenomenon of victim-blaming occurs when the victim of a crime or abuse is held partly or entirely responsible for the actions committed against them [105]. In other words, the victims are held accountable for the maltreatment they have been subjected to. If they fail to take all the right precautions and end up a victim, a certain degree of responsibility for the consequences rests with them [93].

The media play a role in this phenomenon. The way in which news media cover an issue influences how people, including victims and perpetrators, think [51, 86]. Attitudes can be changed through media consumption [51]. The nature of media coverage of blockchain attacks is still unknown in the Netherlands. The fact that victims of cyber attacks are often victim-blamed [81] highlights the practical relevance of a study that investigates the framing of responsibilities in the media.

Blockchain technology eliminates the need for trusted third parties and with this disrupts how things are normally done by centralised institutions. In centralised institutions generally a group, or sometimes an individual, can control the other involved parties and in essence force them to collaborate [106]. Because it is clear who is in charge of the decision-making, centralised institutions can be held responsible for their actions. In blockchain networks on the other hand, decisions are made by a large group of distributed participants that are not individually significant for the result. This means that participants automatically take on responsibilities that are otherwise absorbed by a trusted third party [84, 135]. In fact, Østbye [84] found that despite the decentralized nature of blockchain technology, participants cannot legally use a lack of causal links as a justification to evade responsibility.

Many of the harms caused by cyber attacks are addressed by general and specialized laws and regulations [84]. However, in blockchain attacks legal liability can only reach so far to hold attackers liable due to problems with enforcement and the fact that attacks are not always covered by legal norms. In some cases, there is a basis for liability for developers or trading platforms, while there is none for attackers [84].

Although the legal responsibility of participants in blockchain applications has been studied in the literature [84, 85, 135], it is unknown what happens when blockchain attacks happen in reality. Attackers cannot always be held responsible, either because of limits in regulation, or because attackers cannot be identified. This leaves the question on what this means for who is responsible to deal with the consequences of blockchain attacks in practice.

Cybersecurity threats to blockchain applications are accelerating and becoming more impactful over time [45]. Strawser and Joy Jr [112] evaluated whether the average person is able to implement adequate cyber security measures, and whether there exists a prevailing belief that failing to meet a security norm is a matter of negligence. The authors arrive at the conclusion that vigilance is a duty of all parties involved but there should be reasonable

security solutions provided that users can effectively use without any special expertise.

However which security solutions are reasonable and practical is undefined. Moreover, it is unclear to what extent developers of applications can (and perhaps should) influence how security responsibilities are divided and communicated. Applications could be designed in a more user-friendly and restricted way, taking away some of the responsibilities from users, and thus creating a more secure environment. Such design rules could even be imposed by legislation. There are some studies about design of frameworks that take away part of the freedom of users to ensure better security [61, 98], as well as studies that try to find ways to encourage users to assume more responsibility for cyber security, such as the study by LaRose et al. [65]. They found that the average user can be induced to take a more active role in online safety.

Lack of clarity when it comes to user responsibilities is not helpful to improve security. Guidelines and rules for developers and users to arrange cyber security responsibilities for blockchain applications in a way that is suitable for users and effective at preventing blockchain attacks are necessary.

11.2 Research aim and questions

Before moving on to the research question, the research aim as proposed in Chapter 1 is repeated.

RA2: The aim of this research is to gain insights into whom is held responsible for cyber security of blockchain applications and to evaluate how blockchain developers can influence the division of responsibilities.

This research aim is reframed into the research question as follows:

How are cyber security responsibilities assigned for blockchain attacks in the Netherlands and what can blockchain developers do to influence this?

The main question is divided into three sub-questions, of which the first is the main part of the research, while the second and third sub-question serve to offer a different perspective on blockchain responsibilities. The first step of this research is to investigate who is depicted by news media as responsible for the prevention and consequences of blockchain attacks, and leads to the following sub-question:

1. How is cyber security responsibility for blockchain applications framed in news media channels in the Netherlands?

The media framing of responsibilities may differ from who is effectively responsible. Since culprits are generally difficult to find after the attack, they are usually not the ones paying for the damage. We obtain a general idea of who is responsible legally and in practice for dealing with the consequences of common blockchain attacks through the second sub-question:

2. How do legal experts view responsibilities for the consequences of blockchain attacks in the Netherlands?

It is not reasonable to assign responsibility for cyber security to the average user if they cannot be expected to take the precautions they are supposed to take. In addition to the framing of responsibilities and legal responsibilities, we aim to gather some insights into what role developers can play to assign responsibilities in a more effective way that improves cyber security when designing a blockchain application. The third sub-question therefore is:

3. What can developers do to establish an effective division of responsibilities?

Each sub-question aims to view blockchain responsibilities from a different yet relevant perspective. Sub-question 1, which forms the main part of the research, reflects on how news media portray responsibilities concerning blockchain security. Sub-question 2 and 3 consider blockchain responsibilities from different perspectives to present the reader with a broader outlook on blockchain security responsibilities. Sub-question 2 touches upon the legal implications of the use of blockchain applications, while sub-question 3 seeks to highlight various aspects of how the current implementation of blockchain technology impacts user responsibilities.

11.3 Contributions

In this work, we try to create a deeper understanding of the way responsibilities are assigned for cyber security in blockchain applications, and use this to formulate suggestions for developers to assign responsibilities in a way that improves cyber security. The main contributions are:

- **A theoretical framework and coding schedule:** We develop a theoretical framework to form the basis for a textual content analysis to gather insights into causal and treatment responsibilities. Based on this theoretical framework a coding schedule is constructed that can be used to perform a content analysis focused on framing of responsibilities.
- **Insights into responsibilities:** This research provides new insights into how news media depict responsibilities concerning blockchain attacks and how this differs from practice.

- **Various considerations regarding blockchain technology:** This research touches upon several aspects of the current implementation of blockchain and the (potentially unwanted) effects of this on user responsibilities.

11.4 Outline

Part III of the thesis consists of Chapters 11 to 19. In the current chapter, Chapter 11, the research topic is introduced and the research questions are presented. In Chapter 12, the reader is provided with a background information on cyber security and blockchain attacks to get acquainted with blockchain and its security challenges. Subsequently, Chapter 13 introduces the methods that are used to answer the research questions.

In Chapter 14, we construct a theoretical framework for blockchain security responsibilities based on framing theories. The created theoretical framework is linked to content in the media to form a coding schedule. Next, in Chapter 15 the results of the corresponding content analysis are presented. Afterwards, in Chapter 16, a discussion on who is responsible in practice for the consequences of blockchain attacks according to legal experts can be found. Then, in Chapter 17 we evaluate how the implementation of blockchain technology affects user responsibilities. Chapter 15, Chapter 16 and Chapter 17 present the results corresponding to sub-questions 1, 2 and 3 respectively.

The research question and its sub-questions are answered in Chapter 18. Finally, this thesis will be finished with a discussion in Chapter 19.

Figure 11.1 illustrates the relation of the upcoming chapters to the research sub questions and used methods.

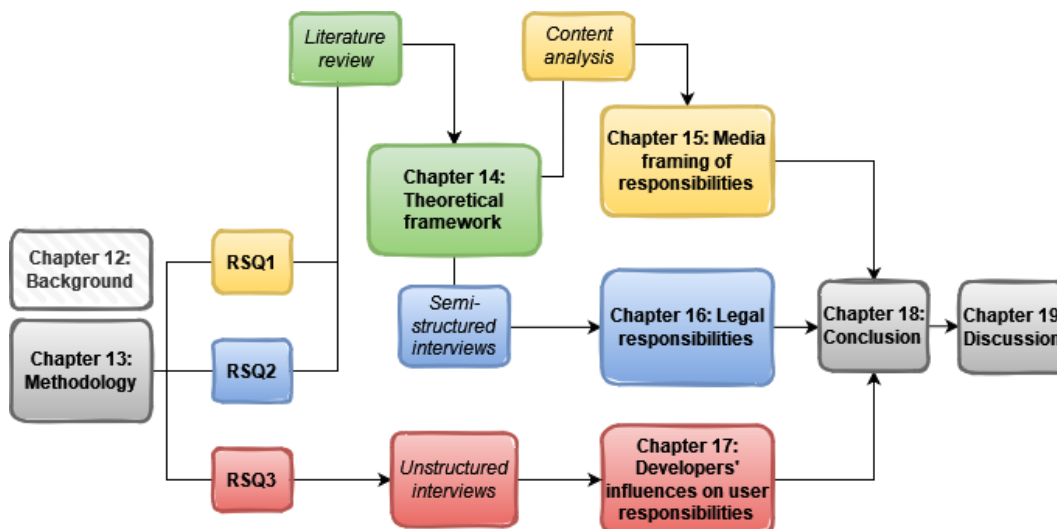


Figure 11.1: Overview of the upcoming chapters.

12

Background

This research aims to investigate who is held responsible for cyber security against attacks performed on blockchain applications and to evaluate how blockchain developers can influence the division of responsibilities. In this chapter, the context of the research is established by providing the reader with a background on cyber security in Section 12.1. Additionally, different types of blockchain attacks and corresponding vulnerabilities are discussed in Section 12.2.

12.1 Cyber security

Cyber security is a broad concept and there exist many definitions for it, some of which have been investigated by [Craig et al.](#) and [Schatz et al.](#) [31, 101]. The authors state that the definitions of cyber security are highly variable (inconsistent), context-bound, and often subjective. In order to reason about the responsibility for cyber security, there should not be any confusion regarding the term. [Craig et al.](#) argue that a concise, broadly acceptable definition is needed to facilitate technological and scientific advances [31].

Such a definition is provided in the ISO/IEC 27000-series: a series of best practice recommendations, published by the ISO (International Organization for Standardization) and the IEC (International Electrotechnical Commission). The ISO/IEC 27032:2012 document defines cyber security as the “preservation of the confidentiality, integrity and availability of information in cyberspace” [121]. This definition will be used throughout this study. The aspects of cyber security in the definition are based on the CIA (Confidentiality, Integrity and Availability) triad [27], visualised in Figure 12.1 [125]. The triad contains three categories of threats to information: unauthorised information release (threat to confidentiality), unauthorised information modification (threat to integrity) and unauthorised denial of use (threat to availability).

In line with the definition for cyber security presented above, a cyber security attack can be described as any attack performed to compromise the confidentiality, integrity or availability of information in cyberspace.



Figure 12.1: The CIA triad

12.2 Blockchain attacks

The blockchain technology implements all three elements of the **CIA** triad by default [37, 72]. Blockchain is public, but it's tamper-proof and node failure tolerant. It provides security without the need or help of a trusted third party. Still, although blockchain provides improved security compared to traditional database systems, like all technology, it is not flawless. Its preventative mechanisms may impair its resistance to other types of frauds and maliciousness [130].

Blockchain applications have proven to be a popular target for criminals. In the first nine months of 2019, losses resulting from digital currency crime added up to 4.4 billion dollars (around 3.6 billion euros) [26]. Based on a dataset containing 86 incidents during the period of 2011-2018, Chia et al. [28] propose three main classes of blockchain attacks:

1. **Operational Security (OPSEC)**: Incidents compromising an organisation or individual's control of information and access to (business-critical) assets.
2. **Smart Contracts**: Incidents resulting from improperly written smart contracts deployed and executed on a blockchain.
3. **Consensus Protocol Incentives**: Incidents arising from malicious exploitation of consensus protocols that create opportunities and benefits for blockchain participants.

Traditional **OPSEC**-related issues remain the largest source of incidents in the dataset (66%), while blockchain-specific incidents related to Smart Contracts and Consensus Protocol Incentives make up for respectively 22% and 12% of the incidents [28]. All three classes are described in more detail below.

OPSEC

OPSEC is a container class of blockchain incidents based on traditional cyber attacks. Attacks in this class are not specific to the blockchain technology itself. Attacks are generally possible due to a lack of sufficient (available and standard) cyber security solutions [28]. Blockchain applications are an attractive target for criminals because of the significant amounts of cryptocurrencies that go around.

Most blockchain **OPSEC** incidents are related to private key theft [66]. Blockchain uses public-key cryptography to give users ownership over their blockchain data (e.g. cryptocurrency units). The person controlling the private key is the owner of the blockchain assets. Blockchain applications use the private key to confirm a user's identity and complete a transaction on the blockchain. The process of a transaction does not require the disclosure of the identities of participants. Blockchain addresses are not tied to a person and the security of assets solely depends on the safekeeping of the private key. The security of the private key is a precondition for information being unfalsifiable [87]. Each blockchain address has a corresponding private key. While the address can be shared, the private key should be kept secret as it provides access to the user's funds. Unlike traditional public key cryptography, blockchain users are responsible for their own private keys. A private key is generated and taken care of by the user instead of a third-party. Users therefore act as their own bank. If a user loses their private key, it will be impossible to get access to his digital assets on the blockchain, and there is nothing anyone can do about it [128].

The private key of a user is generally stored in a user wallet. User wallet credentials are therefore a target for criminals. To obtain wallet credentials, attackers use both traditional methods like social engineering techniques and dictionary attacks (brute force attacks based on a list of likely passwords) and new sophisticated methods like finding weaknesses in cryptographic algorithms. Such weaknesses are for example vulnerable signatures, flawed key generation, or hot/cold wallets (wallets that are (or are not) connected to the internet have certain vulnerabilities).

For the most part the security issues with the blockchain technology are not connected to the algorithms used in the blockchain itself, but to the human factor. Humans are often called the weakest link in information security systems [21, 92, 122]. This is no exception for blockchain technology, where generally the consumers of the technology are the easiest targets [75]. People tend to overestimate the security of the blockchain and overlook its weaknesses.

Social engineering is increasingly being applied to cryptocurrency users [128]. Criminals try to obtain a user's name, password, or private keys [10]. One of the most common serious threats is phishing, in which criminals attempt to steal user credentials using fake emails or websites or both [7]. The goal of phishing can be anything from trying to get people to send money, hand over sensitive information, or even just download malware (for

example a keylogger) unwittingly. Blockchain-related social engineering schemes include for example clone phishing (creating fake copies of trustworthy websites, applications or emails), social networking (creating fake accounts to spread phishing links posing as a well known person) and fake cryptocurrency wallets (placing fake wallets in popular app stores to retrieve a user's private key) [7]. Andryukhin [7] distinguishes two types of phishing attacks: social engineering schemes and technical schemes. Social engineering schemes are based on deception and subsequent independent wrong actions of the victim, while technical schemes use vulnerabilities and imperfections of software and infrastructure.

Chainalysis reported that more than 50% of all cybercrime revenue in 2017 was generated from phishing scams without hacking the blockchain infrastructure itself (i.e. social engineering schemes) [114, 129]. So far, social engineers targeted mainly individual users and profited financially from their attacks. However, with the increasing adoption of blockchain technology in the business environment, criminals will draw their attention to companies and their employees [128]. They will use similar tactics to gain access to companies' information and information systems.

While fraudulent investing scams (tricking users into investing by promising a high return) are a blockchain-related social engineering scheme [7], it is not part of the OPSEC class since such incidents do not compromise the control of information and access to blockchain-related assets. The blockchain technology is essentially used as a masquerade to lure victims in, but blockchain applications are not a target.

Smart Contracts

Smart contracts are applications running on top of the blockchain. They are intelligent, self-checking contract applications that provide a foundation for digital asset ownership and a range of decentralized applications in the blockchain domain [49]. The health, insurance and business management sectors are examples where smart contract applications are used [100]. Smart contracts eliminate the need for an intermediary when two parties want to exchange valuable digital or physical assets. They are used as follows. First, two parties create a smart contract between them. Both parties remain anonymous. The contract is stored on the public blockchain ledger. Once the contract is placed within the blockchain, it is nearly impossible to have it removed [100]. The contract terms and conditions are written as code and include triggering events (like deadlines). When the conditions are fulfilled by the desired time, the contract gets triggered to execute the digital transaction. A visualization of the lifecycle of a smart contract is presented in Figure 12.2 [46].

Smart contracts incidents occur when a smart contract does not work the way it was intended, introducing faults that adversaries may exploit [28]. Smart contracts are an application built on top of a blockchain network and therefore have security vulnerabilities due to program faults (deficiencies in programming language, execution environment, and

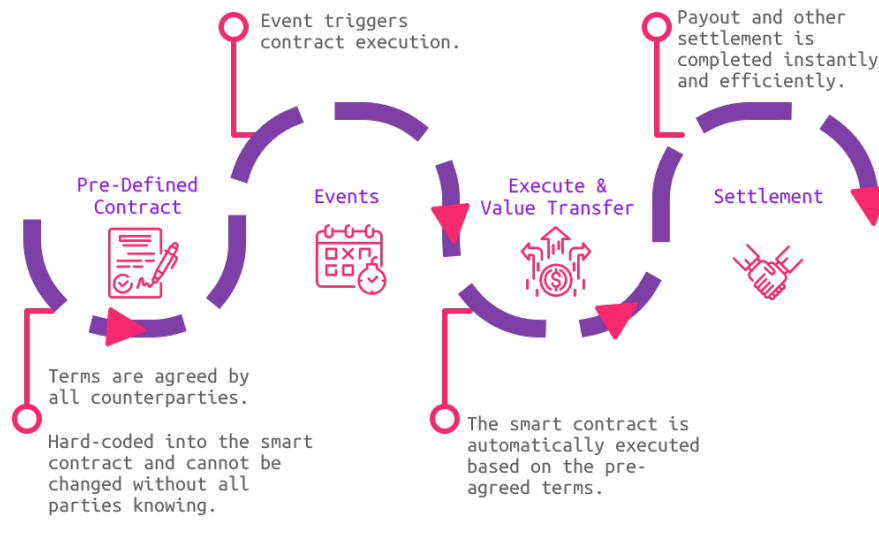


Figure 12.2: The lifecycle of a smart contract

coding style) [69, 95]. These coding vulnerabilities can lead to a large variety of attacks that circumvent the rules intended by a smart contract. Smart contract attacks typically exploit code bugs to drain funds from the contract wallet [100].

Consensus Protocol Incentives

In contrast to **OPSEC**-related and Smart Contract incidents, Consensus Protocol Incentives incidents relate to the core technology of blockchain. In Chapter 1 it is described that to update the public blockchain ledger, nodes (i.e. servers) in the blockchain network create consensus. Consensus is created based on a consensus protocol. A consensus protocol ensures a common, unambiguous ordering of blocks, and guarantees the integrity and consistency of the blockchain across all nodes [39]. Commonly used consensus protocols are Proof-of-Work (**POW**) and Proof-of-State (**PoS**) [13].

To reach consensus with **POW**, all nodes in the network are constantly performing computations (mining) to solve complex mathematical problems. This is required to validate new blocks of transactions and costs computational power. To give incentive to miners to perform these computations, rewards (blockchain assets, i.e. cryptocurrency) are handed out to the miner that first solves the mathematical problem. Mining pools (a collection of miners) use shares to track activities of each miner. Other consensus protocols such as **PoS** employ different tactics to assign the work and corresponding rewards.

Consensus mechanism vulnerabilities can allow a criminal to break the rules intended by

the blockchain. Attacks on the consensus protocol include the 51% attack (where someone that holds the majority of the hashrate in the network can perform double spending or halt payments between some or all users), the alternative history attack, the Finney attack and similar attacks that can result in **double spending** for the attacker [6]. Furthermore, **eclipse** attacks, which occlude a merchant's view of the blockchain ledger by alienating him from the rest of the network, can be used as a tool to increase the effectiveness of a double spending attack [16].

Moreover, mining pool vulnerabilities allow attackers to apply different tactics to gain more shares so they will receive a greater portion of the reward. Examples of attacks are the **block withholding** attack (intentionally not submitting any found blocks to the pool, making the mining pool lose all rewards contained within the block) and the **bribery** attack (rational miners accept bribes from attackers to maximize their profit, enabling the attacker to get a high percentage of the hash power). Such attacks sabotage the effort of other miners, resulting in the attacker finding the cryptographic solution following from the mining first and claiming more shares of the reward.

The majority of incidents arising from incentives are due to second-order, unintended effects of blockchain use [28]. The improper mining of a block or the censorship of nodes are examples of such unintended effects.

13

Methodology

In Chapter 11 the research question and sub-questions that are central in this research were presented. In this chapter, the methods used to answer these questions are discussed.

The first sub-question was addressed with a literature study and a content analysis. The literature study helped shape a theoretical framework. How this literature study was performed is elaborated on in Section 13.1. The theoretical framework formed the basis for the content analysis, which is further explained in Section 13.2. To answer the second sub-question we conducted four exploratory interviews with legal (research) experts. This is talked about in Section 13.3. The third sub-question was answered by talking to two blockchain development experts. This is explained in in Section 13.4. A summary of the methods used for the different sub-questions and where their results can be found in the thesis is given in Section 13.5.

13.1 Literature study

The starting point of the first research sub-question (*How is cyber security responsibility for blockchain applications framed in news media channels in the Netherlands?*) was to discover commonly used theories and frames regarding victim blaming and responsibility framing. A systematic literature review was conducted to find studies that provide insights into how responsibilities are assigned in news media. The primary goal of this literature study was to shape a framework of theories that could form the basis of a coding schedule for a content analysis. Literature was searched using Scopus and the following query:

```
( framing OR frames OR framework )
AND ( victim )
AND ( blame OR blaming OR culpability
      OR responsibility OR responsible )
AND ( blamed OR assigned OR assigning
      OR attribution OR attributed )
AND ( media OR news OR text)
```

The aim of the literature search is to find frames that can be applied to cyber crimes where there is at least one party that fell victim. The focus of the search query therefore is on media, news or textual framing in scenarios where there is a victim and culpability is attributed in some way. From the search resulted 47 scientific articles. After reviewing all titles and abstracts, 15 relevant studies were selected for full text reading. Articles that did not perform or discuss a study about media framing in the context of crimes were excluded from the literature review.

The theoretical framework that was created based on the frames as encountered in the literature study is presented in Chapter 14. The constructed theoretical framework is used to perform a media content analysis focused on blockchain security responsibilities.

13.2 Content analysis

The aim of the first research question is to investigate how news media attribute responsibilities for blockchain security. For this purpose a quantitative content analysis was conducted. The content analysis was performed following the strategy as described by [Hansen and Machin \[48\]](#). The authors define the following eight key steps in the process of media content analysis.

1. Define the research problem
2. Review relevant literature and research
3. Select media and sample
4. Define analytical categories
5. Construct a coding schedule and protocol
6. Pilot the coding schedule and check reliability
7. Data preparation and analysis
8. Report findings and conclusions

The research problem was described in Section 11.1. By analysing a body of communication, we want to gather insights into how media sources frame responsibilities for the prevention and consequences of blockchain attacks. In other words, when blockchain security is compromised, who is deemed responsible according to news media?

Relevant literature was explored in the systematic literature review as described in Section 13.1. Based on the studies that were found and the theories that were deemed relevant, the theoretical framework that is presented in Chapter 14 was constructed.

The body of media content that is analysed consists of online news sites in the Netherlands. By performing the content analysis on digital media, a large body of text can be analysed due to the high accessibility of online news. Furthermore, offline news media has been gradually displaced by online news media [54]. There is a decay of print newspapers in many countries around the world [18]. Online news media provides us with a reasonable reflection of the news that society interacts with.

Initially it was opted to retrieve media articles using LexisNexis. LexisNexis provides access to known and most read news papers. However, access to LexisNexis is only possible through universities and Delft University of Technology did not have the required access. Therefore news articles were selected using the media content tool *Coosto*. This tool offers the functionality to search news websites and forums using a search query. It yields a varied selection of media that is online accessible. Because the Coosto search is thorough, the results also include several duplicate articles, social media posts, and comments placed under articles. Furthermore, many articles found in the search are not from renown newspapers but rather from blockchain enthusiast magazines or general news sites that report articles written by other news sources.

To sample relevant content an elaborate query is constructed. The unit of analysis is a written news article that (1) discusses a blockchain application, usually related to decentralized finance or cryptocurrencies, (2) discusses an attack related to a blockchain application and (3) mentions the presence of a victim or perpetrator. Briefly put, the query indicates the context (that should concern blockchain), that an attack took place, and that there is someone mentioned in the text that is either a victim or an attacker (some sort of responsibility needs to be discussed).

Specific cryptocurrencies not included in query since this would lead to articles about attacks unrelated to blockchain applications, that have some sort of a payment in an cryptocurrencies. Additionally, because many articles discussed an attack that made use of cryptocurrency payments to blackmail victims, while the attack itself had nothing to do with blockchain applications, the term ransomware was excluded in the query. The used query is the following:

```
( blockchain OR blockchainapplicatie OR blockchainapplicaties
  OR defi OR "decentralized finance"
  OR "digital currency" OR "digital currencies"
  OR crypto OR cryptocurrency OR cryptocurrencies OR cryptovaluta
  OR cryptocoin OR cryptocoins OR cryptomunt OR cryptomunten
  OR "digitale munt" OR "digitale munten" )
AND ( hack OR hacks OR attacks OR attack OR aanval OR aanvallen
  OR oplichting OR opgelicht OR fraude OR scam OR scams
  OR gestolen OR verlies OR diefstal OR gejat
```

13. METHODOLOGY

```
OR misleidende OR misleidend OR misleiding
OR kwijtraken OR kwijtgeraakt )
AND ( slachtoffer OR slachtoffers OR gedupeerde OR gedupeerd OR gedupeerden
OR schuldig OR schuldige OR schuldigen OR aanvaller OR aanvallers
OR dader OR daders OR crimineel OR criminelen OR hacker OR hackers
OR verantwoordelijk OR verantwoordelijke OR verantwoordelijken )
- ransomware
```

Articles were selected over the timespan of a full year. This way a variety attacks that happened throughout the year are covered in the media. The selected time period is from September 2020 up to and including August 2021. In total, 1103 articles over the course of 1 year were retrieved from the *Coosto* search.

All articles were considered for the analysis but only relevant ones were further analyzed. For articles to be included in the analysis, the following selection criteria needed to be met. First, the article must without a doubt discuss an attack related to a blockchain application (e.g. not an attack that uses cryptocurrencies as payment). Second, the article must be a news article. Finally, the article must be accessible. Articles behind a paywall or with an invalid link are not further investigated. One exception are articles from *NRC Handelsblad*, since a subscription was used to access these articles.

The coding scheme began with an initial set of categories based on existing research on frames in responsibility attribution and was expanded during the coding of the sampled material. We use a top-down approach: the dimensions that are analysed to classify content are directly related to the theoretical framework that was constructed. The focus of the content analysis is to investigate what parties are assigned causal and treatment responsibility in news media. By considering both types of responsibilities, we acknowledge that there is a difference between being responsible for causing an attack, and being responsible for preventing or resolving the situation.

The type of blockchain attack that is discussed in an article may influence other categories in articles that are analysed. The analytical categories should thus include the type of blockchain attack, the party that is assigned causal responsibility, the reason why, the party that is assigned treatment responsibility, and the treatment that is mentioned. The party mentioned as the victim and perpetrator are relevant dimensions as well since they indicate who news media portray to be harmed by a blockchain attack and who inflicted this harm. This may differ from who is assigned causal or treatment responsibility.

Another analytical category that is analysed is the type of framing that is adopted by the articles. The type of framing that is used tells us whether blockchain attacks are seen as a societal issue or simply as individual issues dissociated from larger social contexts. Episodic framing tends to cause audiences to attribute guilt to individuals, while thematic framing puts blame on societal factors beyond an individual's control [131]. Other relevant

dimensions are the name of the newspaper and the date of the article.

Based on the analytical categories a formal coding schedule was constructed. The coding schedule is presented in Table 13.1. Each column in the coding schedule corresponds to the individual variables analysed (such as the date an article was written, the newspaper, or type of blockchain attack discussed) and each row corresponds to each news article.

Article Number	Date	Newspaper	Type of Attack	Framing	Involved Parties		Causal Responsibility		Treatment Responsibility	
					Victim	Perpetrator	Party	Category	Party	Category

Table 13.1: The coding schedule

To guide the coding process, a coding protocol was written. The coding protocol serves as a manual for coders, giving clear instructions about how the coding is to be done. It helps ensure that the content analysis is replicable and that the results are reliable. The complete coding protocol is included in Appendix B.

The coding protocol was piloted on a small sample of the selected media content to gain insight into the content which was to be analysed and to fine-tune the coding schedule and protocol. This was done to avoid loss of ability to relate different values and variables to each other. Additional (missing) categories of classification were added to the causal and treatment responsibility variables based on the articles encountered in the pilot.

Moreover, a partially inductive approach was taken during the content analysis by adding new values to variables in the coding protocol (e.g actors that were depicted as having causal or treatment responsibility) as the coding progresses and as new values (e.g. new actors having causal or treatment responsibility) appeared.

The reliability of the coding process was tested by having three independent coders (fellow students) perform coding on a subset of the samples based on the coding protocol. The subset consisted of 5% (i.e. 15) of the relevant articles. There should be insignificant divergence in how the same material is being categorised by different coders. The Krippendorff's alpha statistic was applied to measure intercoder reliability. Krippendorff's alpha can be used with more than two coders and minimises the effect of chance in agreements on the codes [79]. In this case it was chosen to compare the external coders separately to the main coder. Coefficients of 0.80 or greater are generally considered fairly reliable [79], although 0.667 is the lowest conceivable limit. Table 13.2 shows the computed Krippendorff's alpha for three external coders. Coder 1 and Coder 2 coded the articles without additional information; Coder 3 was given an explanation of the coding protocol. Krippendorff's alpha was computed for the different categories in the coding schedule. The average Krippendorff's alpha was 0.55, which is low and indicates unreliability. This likely is related to a lack of understanding of blockchain attacks and framing concepts. Coder 3, to

whom an explanation about the coding protocol (and the concepts in it) was provided, was able to achieve an average Krippendorff's alpha above the conceivable limit (0.74).

Krippendorff's alpha	Coder 1	Coder 2	Coder 3	Average
Framing	0.46	0.67	0.34	0.49
Type of attack	0.64	0.41	0.94	0.66
Victim	0.41	0.53	0.65	0.53
Perpetrator	0.004	0.33	0.64	0.32
Causal responsibility	0.82	0.26	0.91	0.66
Causal category	0.83	0.16	0.82	0.60
Treatment responsibility	0.58	0.45	0.91	0.65
Treatment category	0.67	0.009	0.72	0.47
Average	0.55	0.35	0.74	0.55

Table 13.2: Krippendorff's alpha calculated for three other coders.

For managing and analysing the content analysis data Microsoft Excel was used. The relevant codes, as described in the content analysis protocol, are entered into the coding schedule. Based on the complete content analysis data, comparisons across the different news articles are made.

The key findings of the content analysis that provide insights into how blockchain security responsibilities are assigned by news media are reported in Chapter 15.

13.3 Interviews with legal professionals

In order to answer the second sub-question (*How do legal experts view responsibilities for the consequences of blockchain attacks in the Netherlands?*), online semi-structured interviews were conducted with professionals specialised in the legal field. The interviews were conducted in order to explore legal considerations and relevant legislation when it comes to blockchain-related crimes. Specifically, the legal experts were asked to give their thoughts and opinions in regard to who is legally responsible for the prevention of the attack, who is likely to be found responsible for the consequences and what aspects are important to consider when determining who is responsible. Insights from these interviews enable us to see how responsibility for blockchain security in theory (liability) and practice (whether enforcement is possible) are related.

Since it makes sense for responsibilities of blockchain participants to be dependent on the root cause of the considered attack, it was decided to discuss a variety of cases with the legal professionals to cover different aspects of blockchain security. The selected cases are real life examples of the different types of blockchain attacks as presented in Section 12.2. Although attacks that do not include a blockchain application do not fall under any of the three blockchain attack types, fraudulent schemes that use blockchain as a cover are very prevalent in today's society (as seen in the performed content analysis) and

therefore it may be worth to investigate as well. An additional category was therefore added to include blockchain investment schemes. This leaves us with four blockchain attack types: OPSEC incidents, Smart Contract incidents, Consensus Protocol Incentives, and Fraudulent Schemes.

The legal experts were selected from different universities in the Netherlands (University of Leiden, University of Groningen, VU Amsterdam) that research legal implications of technology, or organisations affiliated with such universities (the Dutch Blockchain Coalition). It was important that the experts had experience with (research in) the legal field, and that they were knowledgeable about blockchain technology. Several (eight) experts were contacted to set up an interview.

In total, four interviews were conducted. Each interview lasted about 60 minutes and was conducted through an online video connection using Microsoft Teams. The interview protocol containing a description of the selected cases and questions is enclosed in Appendix D.1. The summaries of the interviews can be found in Appendix D.2. The questioned experts were asked to validate our interpretation of the answers given during the interviews. The findings resulting from the interviews are discussed in Chapter 16.

13.4 Interviews with blockchain researchers

The third sub research question (*What can developers do to establish an effective division of responsibilities?*) focuses on the influences that blockchain developers have on user responsibilities. Exploratory non-structured interviews were conducted with blockchain researchers to identify relevant aspects of blockchain designs that have an impact on the responsibilities that users are (automatically) assigned. The question that was focused on is how developers play a role in defining (perhaps unwittingly) responsibilities for users or other parties. It was decided to have an open discussion with the experts due to the large scope of this question. This way, there was enough time during the interview to further explore any relevant points that came up.

The blockchain experts were selected from universities in the Netherlands (University of Amsterdam, Delft University of Technology) that conduct research on blockchain technology and/or its societal implications. The contacted experts perform research in the field of blockchain security, either from a technical or societal point of view. Four experts were contacted to set up an interview.

In total, two interviews were conducted. Each interview lasted about 60 minutes and was conducted through an online video connection using Microsoft Teams. The interview protocol containing an introduction to the topic is enclosed in Appendix E.1. Our interpretation of some interesting concerns that were raised per interview can be found in Appendix E.2. The blockchain experts were asked to validate our interpretation of the insights gathered from the interviews. The results of the interviews are discussed in Chapter 17.

13.5 Methods outline

The upcoming chapters answer the three research sub-questions. The corresponding methods for each research sub-question can be found in Table 13.3.

RSQ	Method	Chapter
1	Literature study (theoretical framework)	14
	Media content analysis	15
2	Interviews legal experts	16
3	Interviews blockchain experts	17

Table 13.3: Methods used per research sub-question and the chapter in which they are answered.

Theoretical framework

This chapter presents the literature study of this research project, as well as the theoretical framework that was constructed based on it. First, relevant frames that were encountered in the literature study are described in Section 14.1. Then, in Section 14.2, these theories are linked to the context of this research, namely blockchain security, to shape a theoretical framework that is the basis for the content analysis.

14.1 Frames

In crisis communication the most common frame is attribution of responsibility, which offers causes of or potential solutions to a crisis [132]. Other frames that are currently used in news media are the human interest frame, the conflict frame, the economic consequences frame, and morality frames [58, 131]. For the context of this research, the attribution of responsibility frame is most relevant as the aim of the content analysis is to investigate which parties are attributed responsibilities.

The attribution of responsibility frame is found in news stories where individuals, groups, organisations or governments are assigned blame or credit for an action [58]. Siefkes-Andrew and Alexopoulos [109] explain that the attribution theory addresses how people interpret the causes of their own and others' behaviors. Following from the theory, people attribute other's actions to either internal or external causes. Internal attribution means that the cause of behaviour is attributed to an internal quality, while external attribution assigns the cause of behaviour to an external event or situation that is outside the person's control.

Variations in framing can shift thinking, attitudes, and decisions [23]. Furthermore, how news media frame an issue, and its causes and solutions, influences not only how the public thinks, but how victims and perpetrators think [86]. Semantics matter as well [42, 109]. For example, it was found that the use of active verbs influenced participants to attribute more responsibility to the story characters compared with the use of passive verbs [109].

Several of the studies that were reviewed differentiated between episodic and thematic framing [42, 86, 131]. Media frames issues primarily as either episodic or thematic [131]. Episodic frames dominate media coverage [86]. They discuss isolated incidents. Thematic

framing refers to the broader portrayal and presentation of issues through information about their systemic causes, trends, and consequences.

Studies have shown that attributions of cause and responsibility are sometimes correlated with episodic or thematic frames [86, 131]. The way an issue is framed influences who the public believes is responsible for a problem and who is responsible to remedy a situation. Episodic frames tend to cause audiences to attribute guilt to individuals, while thematic frames puts blame on societal factors beyond an individual's control.

In general, readers who encounter episodic framing tend to attribute blame to individuals and tend to support individual-level solutions. By contrast, readers who encounter thematic framing are more likely to call for broader, systematic changes [42, 86]. By focusing on anecdotal events and personal stories, the media emphasises deficiencies in and modifications of the individuals' choice of behaviours that are dissociated from larger social contexts, thereby potentially diverting public attention away from systemic flaws in the political, social, and economic environment [131].

In many of the reviewed literature, the research involved victim blaming [2, 25, 32, 36, 42, 91]. Individuals often tend to irrationally blame victims for the maltreatment they have been subjected to. This behaviour is supported by just-world beliefs [25]. Individuals typified by strong just-world beliefs believe the world is a fair and predictable place: bad things only happen to people who deserve it.

Various articles that were reviewed in the literature study used the attribution of responsibility frame in the media content analysis they performed [57, 86, 132]. The attribution of responsibility frame includes both causal responsibility (blaming someone) and treatment responsibility (demanding a solution from someone) [58]. Causal and treatment responsibility can be attributed either to the individual or to institutions [132]. Newspaper stories most often focused on a single case (episodic framing), and not on the issue as a whole (thematic framing), which would lead to more framing of individual behavior [86].

14.2 Theoretical framework

The attribution of responsibility frame as described above is placed within the context of blockchain security. In Section 12 three types of blockchain attacks are discussed. These types of blockchain attacks correspond to different layers of blockchain applications. OPSEC incidents relate to the application layer of the blockchain technology. OPSEC incidents attack not the blockchain technology itself but applications or users that make use of it. Smart Contract incidents are in the Smart Contract layer.

Although attacks that do not include a blockchain application do not fall under any of the three blockchain attack types, fraudulent schemes that use blockchain as a cover are very prevalent in today's society. It was therefore decided to include this category in the framework as well. These kind of attacks can be seen as an external layer, not part of the blockchain.

The theoretical framework, connecting the types of blockchain attacks to involved parties, is presented in Figure 14.1. The involved parties are selected based on what parties were identified in the framing studies and missing parties that were encountered in news articles.

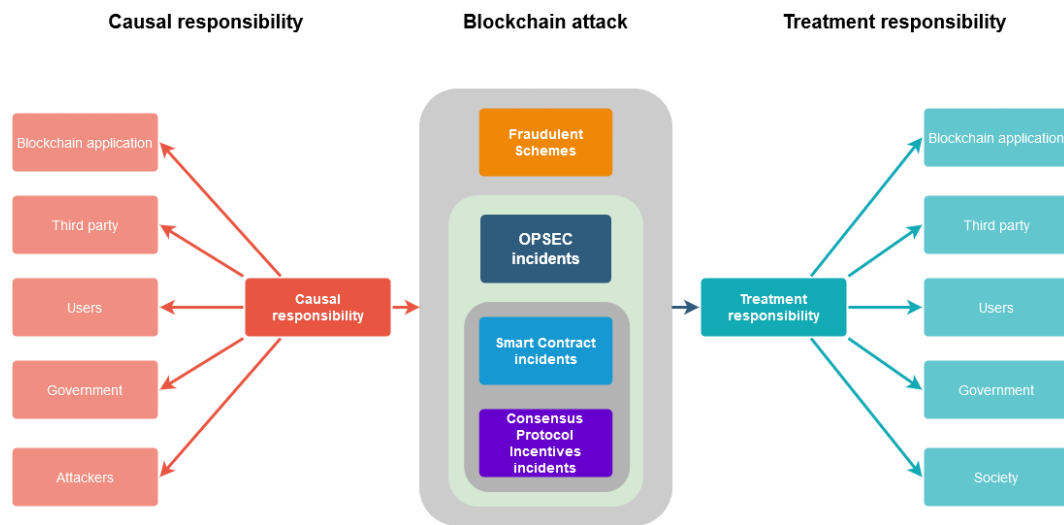


Figure 14.1: Causal and Treatment responsibility

For a party to be attributed causal responsibility the cause must be identified. Several categories were set up, again based on what was encountered in the framing studies and on what was encountered in news articles about blockchain attacks. The categories are presented in Figure 14.2. For blockchain applications, possible causes are the characteristics of the blockchain, the attractiveness of blockchain technology to perform an attack, the newness of the blockchain technology, the lack of security of blockchain applications, faulty behaviour by blockchain applications, and finally a lack of communication by blockchain applications. For third parties (like crypto exchanges) possible causes for attacks are negligence and sloppiness. For users it could be due to bad user behaviour and/or lack of awareness. Insufficient regulation, poor law enforcement and lack of supervision on the use of blockchain applications are reasons for governments to be causally responsible. For attackers reasons are ethical hacking, for purposes of fun, social status, or political reasons. These categories are explained in detail in the coding protocol in Appendix B.

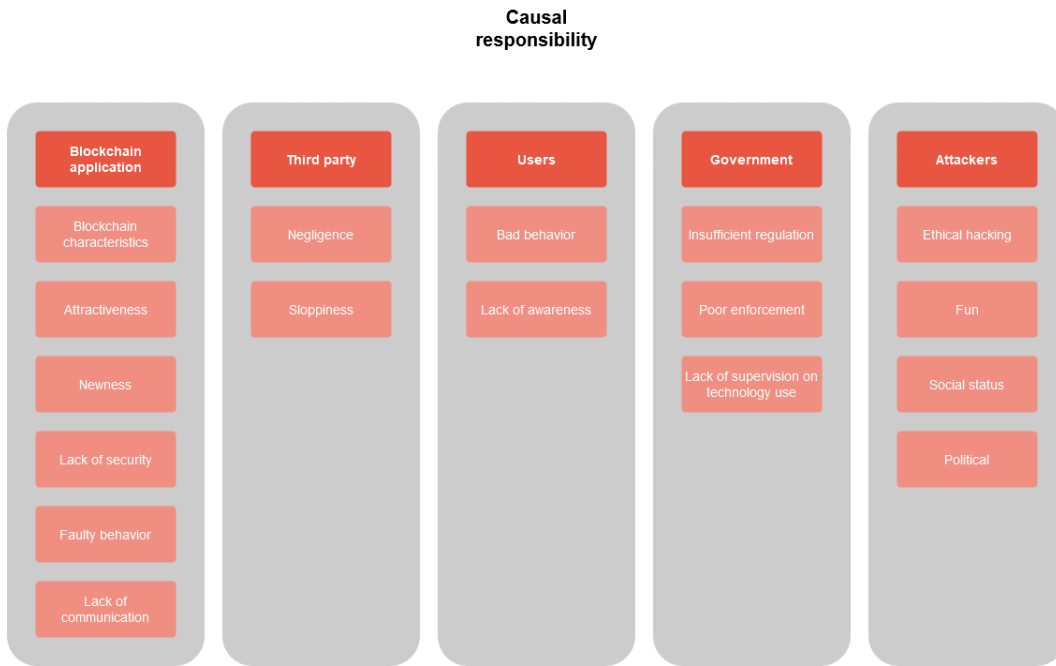


Figure 14.2: Causal responsibility categories

Similar to causal responsibility, corresponding categories for treatment responsibility of involved parties are shown in Figure 14.3. Again, the categories are explained in detail in the coding protocol in Appendix B. For treatment responsibility the attackers are not a responsible party. Instead, society can play a role as seen in the reviewed literature studies, in the form of providing education, financial support or guidance in the aftermath of the attack, and covering blockchain attacks more in the media to give users a realistic image of blockchain risks.

The constructed framework forms the basis for the content analysis. In the content analysis, news articles are analysed with respect to causal and treatment responsibility. The categories (the causal responsibility parties and categories, as well as the treatment responsibility parties and categories) that are present in the framework, form the categories in the coding schedule.

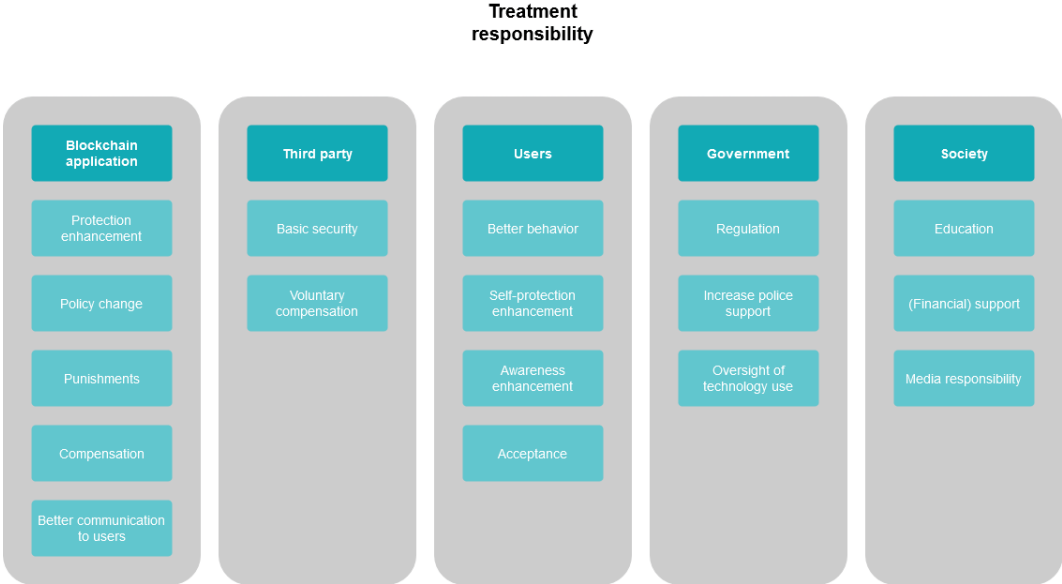


Figure 14.3: Treatment responsibility categories

15

Media framing of responsibilities

In Section 13.2, it was explained how the media content analysis was performed. This was done according to the coding protocol that is included in Appendix B. In this chapter, the results of the content analysis are presented and analysed.

First, in Section 15.1, some general observations that came from the analysis are discussed. Afterwards, the concept of causal responsibility and the way it presented itself in the news articles is elaborated on in Section 15.2. In Section 15.3 the same is done for the concept of treatment responsibility. The results are evaluated in Section 15.4.

15.1 General observations

The content analysis was conducted on a total of 1103 articles. The articles were retrieved using *Coosto* and a search query (described in Section 13.2) focused on finding articles dated between September 2020 and August 2021 on blockchain attacks. 828 (75.1%) of the articles were deemed irrelevant. Articles were classified as irrelevant if they were not a news article or did not discuss a blockchain attack (51.2%), if they were a duplicate (29.2%), or if they were inaccessible (either due to a paywall or a broken link) (19.6%). The analysis was conducted on the other 275 (24.9%) relevant articles.

In far out the most of the relevant articles (88.4%) episodic framing was used. Only for 11.6% of the articles thematic framing was the chosen approach. This aligns with the fact that episodic framing is prominent in news media [86].

Figure 15.1 and Figure 15.2 illustrate the distribution of the blockchain attacks that are discussed in the analysed news articles. Articles can discuss one or more type of blockchain attack. 32% of the articles did not specify what kind of attack happened, or the description was too vague to distinguish between the different types of attacks. Most articles discussed a fraudulent investment scheme (33.5%) or an OPSEC incident (28.4%). This is expected since such attacks employ regular social engineering techniques, which occur frequently as they require low effort. As described in Chapter 12, more than half of cybercrime revenue was generated from social engineering attacks without hacking the blockchain infrastructure itself. Smart contract incidents and Consensus Protocol Incentives (CPI) incidents have very low media coverage (4.7% and 2.9% respectively). This could be due to the fact that these

15. MEDIA FRAMING OF RESPONSIBILITIES

attacks are more complex to understand and thus explain to readers. It is a possibility that articles that did not specify the type of attack actually discussed a smart contract or CPI incident.

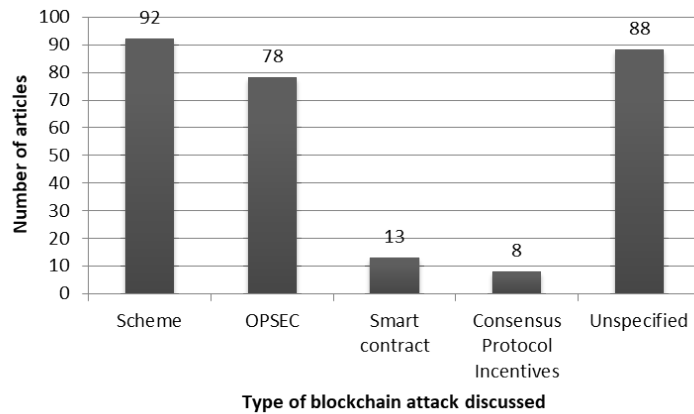


Figure 15.1: The types of blockchain attacks that were reported on in the articles.

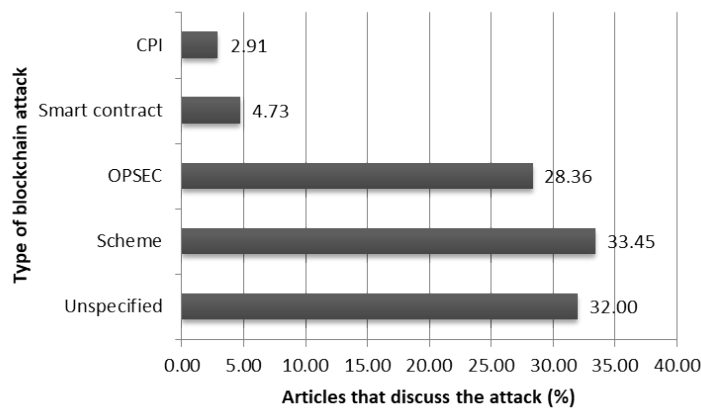


Figure 15.2: The percentage of articles that discuss a type of blockchain attack. $n = 275$.

The parties that were identified as the victim of blockchain attacks are illustrated in Figure 15.3a. An article can mention multiple parties as a victim or perpetrator. Users and investors (individuals wanting to invest in blockchain assets) are mentioned as being a victim of blockchain attacks in farout most of the articles (61.1%). Interestingly, the blockchain application (which comes down to the people behind the application) is also often portrayed as a victim (36% of the cases). Celebrities whose images were used for schemes and external organization(s) are least often mentioned as being a victim.

Almost all analysed articles (90.5%) mentioned the actual attacker as the perpetrator

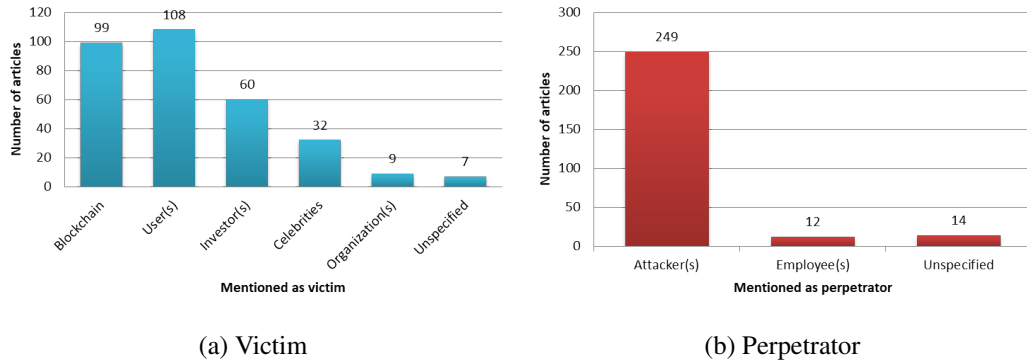


Figure 15.3: Party mentioned as the victim and perpetrator of the attack

of the attack. This is visible in Figure 15.3. In 5.1% of the articles the perpetrator was not mentioned at all. In 4.4% of the articles employee(s) of a third party (in this case a telephone network company which was mentioned in various articles on the same case) were identified as being the attacker(s). In Section 14.1 it was described that the use of active verbs can influence participants to attribute more responsibility to those characters compared with the use of passive verbs. The fact that attacker(s) are almost always mentioned in the articles can be an indicator for attribution of responsibility to the attacker(s).

15.2 Causal responsibility

Attribution of causal responsibility indicates that a party was blamed for causing the incident. In 197 of the relevant articles (71.6%) causal responsibility was not attributed to any party. In 78 articles (28.4%) it was. Figure 15.4 shows the distribution of who was assigned causal responsibility in the remaining articles.

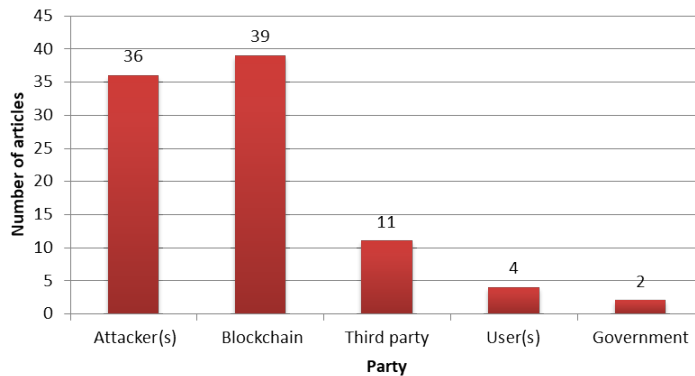


Figure 15.4: Assigned causal responsibility

15. MEDIA FRAMING OF RESPONSIBILITIES

Blockchain was mentioned most often (in 39 articles) as being causally responsible, closely followed by the attacker(s) (in 36 articles). In several articles (11) a third party was attributed causal responsibility. In few articles, the end user(s), and the government were attributed causal responsibility (4 and 2 articles respectively).

In total there were 78 articles (28.4% of all significant articles) that attributed causal responsibility to one or more parties. Table 15.1 presents the occurrences of all categories of causal responsibility in total and per type of attack. Figure C.1 in Appendix C presents the same data in a visual way. We will first consider all type of attacks. Afterwards each attack type will be analysed separately.

Party	Causal category	All attacks		OPSEC		Smart Contract		CPI		Scheme		Unspecified	
		Article count	%	Article count	%	Article count	%	Article count	%	Article count	%	Article count	%
Blockchain application	Lack of security	26	66.7	1	25	5	100	2	50	-	-	19	76
	Attractiveness	6	15.4	2	50	-	-	1	25	2	100	2	8
	Blockchain characteristics	3	7.7	-	-	-	-	2	50	-	-	1	4
	Newness	3	7.7	-	-	1	20	-	-	-	-	2	8
	Lack of communication	2	5.1	1	25	-	-	-	-	-	-	1	4
	Faulty behaviour	1	2.6	-	-	-	-	-	-	-	-	1	4
	Blockchain responsible	39	50	4	26.7	5	45.5	4	100	2	25	25	61.0
Third party	Negligence	7	63.6	3	50	-	-	-	-	4	80	-	-
	Sloppiness	4	36.4	3	50	-	-	1	100	1	20	-	-
	Third party responsible	11	14.1	6	40	-	-	1	25	5	62.5	-	-
User(s)	Lack of awareness	4	100	-	-	1	100	-	-	3	100	-	-
	User(s) responsible	4	5.1	-	-	1	9.1	-	-	3	37.5	-	-
Government	Insufficient regulation	2	100	-	-	-	-	-	-	1	100	1	100
	Government responsible	2	2.6	-	-	-	-	-	-	1	12.5	1	2.4
Attacker(s)	Ethical hacking	26	72.2	-	-	8	100	-	-	-	-	18	81.8
	Fun	9	25	-	-	1	12.5	-	-	-	-	8	36.4
	Political	7	19.4	6	100	-	-	-	-	-	-	1	4.5
	Attacker(s) responsible	36	46.2	6	40	8	72.7	-	-	-	-	22	53.7
Total causal responsibility		78	28.4	15	19.2	11	84.6	4	50	8	8.7	41	46.6
<i>Total significant articles</i>		275	100	78	100	13	100	8	100	92	100	88	100

Table 15.1: Causal responsibility

It is important to note that the percentages of the different causal categories do not add up to 100% due to the fact that articles can attribute causal responsibility based on multiple categories of reasons. The shown percentage is computed by dividing the articles that attributed causal responsibility to the corresponding party by all articles that attributed causal responsibility to one or more parties.

Moreover, some articles discuss multiple types of blockchain attacks, which is the reason that the sum of the article counts for the different types of blockchain attacks exceeds the total attributed causal responsibility article count of 78.

15.2.1 Overall analysis

While it was already visible in Figure 15.4 that blockchain is most often attributed causal responsibility, it now becomes clear that in half of all articles that attribute causal responsibility, it is to blockchain applications. The same holds for attacker(s), albeit 46.2% rather

than 50%.

For blockchain applications, the category that was most often used to assign causal responsibility was **Lack of security**. 66.7% of all articles that attributed causal responsibility to blockchain applications did so because of a lack of security. Examples of quotes in articles that led to this categorisation indicate that an attack was possible due to a vulnerability in the system code or that insufficient precautions were taken to handle an attack. Specific examples are *”Door een fout in de code van het netwerk te misbruiken, kon de hacker een nieuw paar creëren met een lage liquiditeitspool.”* and *”De beveiligingsmaatregelen [...] kwamen onvermijdelijk onder vuur te liggen tijdens de hack en er werden zorgen geuit over de reden waarom het bedrijf geen gebruik had gemaakt van cold storage (waarin privé-sleutels offline worden opgeslagen).”*.

The **Attractiveness** of blockchain technology led in 15.4% of articles (that attributed causal responsibility to one or more parties) to attribution the blockchain application. Examples of article quotes indicate that blockchain offers lucrative opportunities to criminals due to its immutability or the anonymity that it offers. Specifically, the quotes are: *”Volgens [...], blijven blockchainprojecten lucratieve mogelijkheden aan cybercriminelen bieden. Anders dan in het traditionele financiële systeem kunnen frauduleuze transacties niet worden teruggedraaid.”* and *”Een gerichte aanval loont, want dankzij de bitcoinhousse is identiteitsdiefstal lucratiever dan ooit. Via de achterliggende blockchaintechnologie kun je bitcointransacties volgen, maar wie de transactie doet, blijft anoniem. De pakkans is vrijwel nul.”*.

The other categories occurred significantly less often in the analysed articles. **Blockchain characteristics** was responsible for attribution of causal responsibility in 7.7% of the cases. A quote corresponding to this category is *”Doordat de altcoin gebruik maakt van het Proof-of-Work-mechanisme, is het netwerk kwetsbaar voor dergelijke aanvallen. Het is hiermee een groot nadeel die bij het mechanisme komt kijken”*, which says that the **POW** mechanism makes the network vulnerable for attacks.

The same percentage of 7.7% holds for the **Newness** of blockchain applications (*”Er gaan grote bedragen rond in deze markt, maar veel projecten staat nog in de kinderschoenen waardoor af en toe een nieuwe fout wordt ontdekt en gelijk wordt misbruikt door hackers”* - indicating that many blockchain projects are still in their infancy).

Furthermore, **Lack of communication** was in 5.1% of the cases a reason for attribution of causal responsibility to blockchain applications. *”[...] cryptocurrency exchange Coinbase ligt momenteel onder vuur vanwege een naar verluidt ‘vreselijke klantenservice’, nadat het crypto-bedrijf meldingen kreeg van gebruikersaccounts die zijn gehackt en hierdoor geld hebben verloren., Het probleem zou vervolgens verergerd zijn, doordat de crypto-exchange niet reageerde op ondersteuningsverzoeken.”*, indicating that the crypto-exchange provided terrible customer service, is an example of a quote that led to this categorisation.

Finally, **Faulty behaviour** of blockchain applications (*”De exchange heeft inmiddels*

laten weten dat zij niet op tijd zijn geweest met het offline halen van het platform. Dit zou mogelijk een oplossing kunnen zijn geweest gedurende de cyberaanval." - indicating that the exchange did not take their platform offline in time, which could be a solution to mitigate consequences of the attack) was just in one article mentioned.

For **Attacker(s)** three categories of causal responsibility were come across in the analysed articles. Most often used as a reason for causal responsibility was *Ethical hacking*, which was in 72.2% of the articles where the attacker(s) were found responsible the cause. Quotes in articles that illustrate this category are "*Het bedrijf liet vervolgens in een verklaring weten dat het om een ethische hacker ging (Mr. White Hat), die zocht naar zwakke plekken in de ICT-systemen, om mensen met slechte bedoelingen voor te zijn.*" (indicating that an ethical hacker was looking for weak spots in the system to find them before criminals would) and "*Etherhood legde uit dat de cryptocurrency gestolen was om het veilig te houden, voordat insiders de kwetsbaarheid konden misbruiken.*" (indicating that cryptocurrency was stolen with the sole purpose of keeping it safe).

The other categories were **Fun** ("*Voor de fun*", *antwoordde de meest succesvolle cryptohacker ooit op de vraag waarom hij of zij Poly Network aanviel.*" - indicating that the hacker attacked the network for fun), responsible for 25% and **Political** reasons ("*Ze zijn volgens justitie in dienst van het Noord-Koreaanse leger; het geldt dat zij stalen, zou naar het regime van de Noord-Koreaanse leider Kim Jong-un zijn gegaan.*" - indicating that attackers were in service of a national army), 19.4%.

The third most mentioned party as being causally responsible are third parties. **Negligence** was most often the reason (63.6%) for attribution of causal responsibility. An example quote from the analysis that led to this category is the following: "*[...] voerde aan dat YouTube niet alleen de frauduleuze advertenties niet heeft verwijderd, maar ook aanzienlijk heeft bijgedragen aan de zwendel door gerichte advertenties te verkopen die verkeer naar de video's leidden en onterecht de YouTube-kanalen te verifiëren waarop de video's stonden.*", which indicates that a third party (YouTube) has not deleted fraudulent advertisements that lured victims in.

Sloppiness ("*In praktijk bleek het verificatieproces eenvoudig te omzeilen. Het was namelijk mogelijk om beide verificatiecodes op de nieuwe simkaart uit te voeren.*" - indicating that the verification process was easy to bypass) is the other category and was mentioned in 36.4% of the cases where a third party was found causally responsible.

User(s) and governments were only attributed causal responsibility in 4 and 2 articles respectively due to **Lack of awareness** ("*De hoeveelheid stomiteiten waar ze vervolgens ingetuind zijn zou hilarisch zijn als het niet zo verdrietig was geweest.*" - indicating that victims were incredibly stupid for believing the scammers) and **Insufficient regulation**

“Volgens de ministers zijn er zonder wettelijke maatregelen risico’s voor consumenten en beleggers. Ook maken ze zich zorgen over misbruik van het virtuele systeem, zoals fraude, witwassen en financiering van terrorisme.” - indicating that without legislation, consumers and investors are at risk).

15.2.2 Analysis per attack types

It is likely that the type of blockchain attack influences what parties are attributed causal responsibility. As explained in Section 12.2, the origins of the attack differ per blockchain attack type. Consensus Protocol Incentives attacks are targeted at the core technology and principles of blockchain. Smart contract attacks aim at exploiting vulnerabilities in the code of smart contracts, which lie on top of the blockchain. OPSEC attacks and investment schemes do not attack the blockchain infrastructure at all, but rather use other (third party) platforms to target victims.

Figure 15.5 illustrates the attributed causal responsibility per type of blockchain attack. Although the number of articles that classify a type of attack as well as attribute causal responsibility are fairly low, it can be seen that blockchain applications are assigned causal responsibility for all type of attacks.

Attacker(s) are attributed causal responsibility quite often for OPSEC (6 articles) and smart contract (8 articles) incidents. This can be ascribed to the fact that there were several news articles (from different sources) that covered the same case where adversaries performed an OPSEC attack for political reasons, as well as numerous articles that covered the same case where an ethical hacker performed an attack by exploiting a bug in a smart contract.

Third parties are assigned causal responsibility a high number of times as well, specifically for OPSEC attacks (6 articles) and schemes (5 articles). This could be related to the fact that third parties are often to some extent involved in OPSEC attacks and investment schemes.

End users are attributed causal responsibility only in a few articles that discussed a scheme or smart contract attack.

15.3 Treatment responsibility

Attribution of treatment responsibility indicates that a party was expected to offer a solution for the incident or prevent similar incidents in the future. In 206 of the relevant articles (74.9%) treatment responsibility was not attributed to any party. In 69 articles (25.1%) treatment responsibility was assigned. Figure 15.6 shows the distribution of who was assigned treatment responsibility in the remaining articles.

It can be viewed that user(s) are attributed treatment responsibility most often (in 45 articles), even more than twice as often as blockchain applications (which are attributed

15. MEDIA FRAMING OF RESPONSIBILITIES

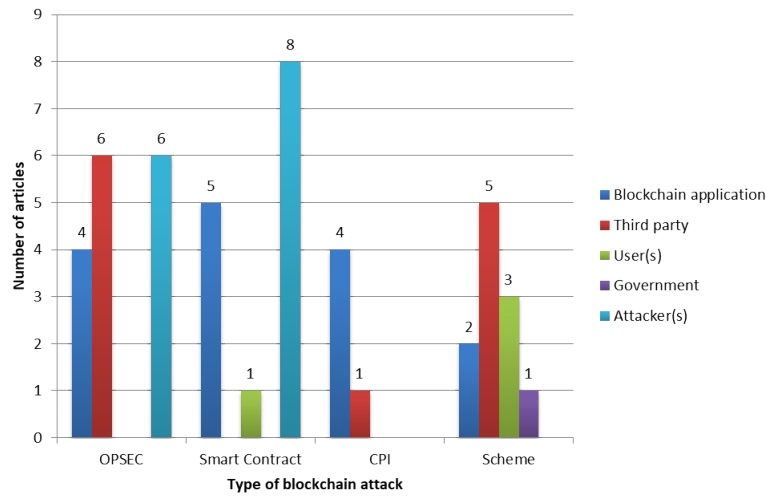


Figure 15.5: Causal responsibility per type of blockchain attack

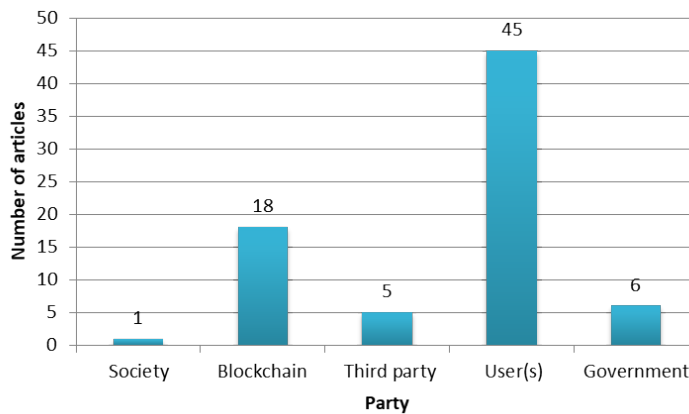


Figure 15.6: Assigned treatment responsibility

treatment responsibility in 18 articles). Society, third parties, and governments are attributed treatment responsibility in very few articles (in 1, 5 and 6 article(s) respectively).

Table 15.2 presents the occurrences of all categories corresponding to the attributed treatment responsibility. Figure C.2 in Appendix C shows the same data in a visual way. Again, we will first analyse the results for articles on all type of attacks together before considering the attributed treatment responsibility per type of blockchain attack.

Similar to the table presenting the results of attributed causal responsibility, the percentages of the different treatment categories do not add up to 100% due to the fact that articles can attribute treatment responsibility based on multiple categories of reasons. The shown percentage is computed by dividing the articles that attributed treatment responsibil-

15.3. Treatment responsibility

Party	Treatment category	All attacks		OPSEC		Smart Contract		CPI		Scheme		Unspecified	
		Article count	%	Article count	%	Article count	%	Article count	%	Article count	%	Article count	%
Blockchain application	Compensation	11	61.1	1	100	-	-	-	-	1	100	9	52.9
	Protection enhancement	7	38.9	-	-	1	100	-	-	-	-	6	35.3
	Better communication	1	5.6	-	-	-	-	-	-	-	-	1	5.9
	Punishments	1	5.6	-	-	-	-	-	-	-	-	1	5.9
	Blockchain responsible	18	26.1	1	4.0	1	7.7	-	-	1	5.6	15	60
Third party	Voluntary compensation	3	60	2	66.7	-	-	-	-	1	50	-	-
	Basic security	2	40	1	33.3	-	-	-	-	1	50	-	-
	Third party responsible	5	7.2	3	12	-	-	-	-	2	11.1	-	-
Users	Better behaviour	23	51.1	15	60	-	-	2	100	2	14	5	45.5
	Self-protection enhancement	23	51.1	10	40	-	-	-	-	12	86	2	18.2
	Acceptance	4	8.9	-	-	-	-	-	-	-	-	4	36.4
	Users responsible	45	65.2	22	88	-	-	2	25	13	72.2	10	40
Government	Regulation	6	100	-	-	-	-	-	-	3	100	3	100
	Government responsible	6	8.7	-	-	-	-	-	-	3	16.7	3	12
Society	Education	1	100	1	100	-	-	-	-	-	-	-	-
	Society responsible	1	1.4	1	4	-	-	-	-	-	-	-	-
Total treatment responsibility		69	27.3	25	32.1	1	7.7	2	25	18	19.6	25	28.4
<i>Total significant articles</i>		275	100	78	100	13	100	8	100	92	100	88	100

Table 15.2: Treatment responsibility

ity to the corresponding party by all articles that attributed treatment responsibility to one or more parties.

Furthermore, due to the fact that some articles discuss multiple types of blockchain attacks, the sum of the article counts for the different types of blockchain attacks exceeds the total attributed treatment responsibility article count of 69.

15.3.1 Overall analysis

In 65.5% of all articles that attribute treatment responsibility, it is attributed to the end user(s). This is a much higher number than the attributed treatment responsibility for blockchain applications (26.1%), and for the other parties (8.7% for governments, 7.2% for third parties and 1.4% for society).

Often mentioned solutions for end users to prevent blockchain attacks are **Better behaviour** and **Self-protection enhancement**. Both treatment responsibility categories were present in 51.1% of all articles that attribute treatment responsibility to users.

Two examples of quotes from articles that led to the better behaviour category are "[...] *het belang van het beheren én beveiligen van je cryptocurrency. De meest veilige en dus beste oplossing is een hardware wallet: een fysiek apparaat dat je private keys opslaat in een beveiligde chip.*" (the article provides tips for secure storage of keys) and "*Gebruikers kunnen waardevolle accounts beter beveiligen met een app op de eigen telefoon, zoals de gratis authenticator-apps van Google of Microsoft.*" (the article advises users to use an authenticator application to keep their accounts safe).

Likewise, "*Het advies is altijd alert te blijven met linkjes, codes of andere manieren om*

u ergens naar toe te leiden. Geloof niet te gemakkelijk grote winsten die beloofd worden.” (the article tells readers to be careful with urls and to not believe things that are too good to be true) and *”Je moet desondanks goed opletten welk platform je gebruikt en wat de beveiliging is.*” (the article indicates that people should make sure the platform they are using is safe) are quotes indicating self-protection enhancement.

Acceptance of blockchain security risks was mentioned as a solution in much less of these articles (8.9%). An article quote illustrating this category is the following: *”Het maakt niet uit [...], er zit hier altijd een risico aan verbonden wanneer het gaat om slimme contracten en de steeds complexere implementaties hiervan”, aldus het Value DeFi Protocol.*”, which indicates that there is always a risk when it comes to smart contracts.

As for blockchain applications, **Compensation** is described most often as a treatment (61.1% of articles that attributed treatment responsibility to blockchain applications). An example of an article that was classified in this category contained the quote *”Om de gebruikers te helpen, heeft Roll een fonds van \$500.000 gelanceerd. Dit zal worden verdeeld onder alle getroffen gebruikers.*”, which indicates that a fund was created by the blockchain application to reimburse affected users.

The category of compensation is followed by **Protection enhancement** (38.9%), indicated by quotes such as *”Betere beveiligingsprocedures zullen nodig zijn om DeFi-gerelateerde criminaliteit terug te dringen.*”, which describes that better security procedures for blockchain applications are required to reduce crime related to decentralised finance.

Better communication (*”De gebruikers zijn het hiermee duidelijk eens dat hier [bij de klantenservice] een enorme verbetering gemaakt moet worden.*”, indicating that customer service should be improved) and **Punishments** (*”Het hooggerechtshof van Londen beveelt Binance de accounts van hackers [...] te volgen en te bevriezen. Als deze zaak slaagt, behaalt Binance mogelijk een van de grootste prestaties bij het veiligstellen van het geld van zijn gebruikers [...].”*), indicating that a blockchain application should freeze the assets of hackers) are only described in one article each as a solution against blockchain attacks.

Governments, third parties and society are attributed treatment responsibility in very few articles, especially compared to users and blockchain applications. When the government was mentioned as being responsible for a treatment, the treatment entailed **Regulation**. A quote that illustrates this is: *”De problemen [...] hebben duidelijk gemaakt dat er dringend regelgeving nodig is voor de nieuwe en snel groeiende markt voor cryptovaluta’s.*”, saying that regulation is required urgently).

As for third parties, **Voluntary compensation** (*”T-Mobile heeft de meeste slachtoffers deels gecompenseerd, maar de provider acht zich niet verantwoordelijk voor online accounts die met een sms-code beveiligd zijn.*”, indicating that a third party has voluntarily compensated part of the victims but does not feel responsible) and **Basic security** (*”[...] is geen uit-*

zonderlijk geavanceerde malware en kan door organisaties vrij eenvoudig worden geweerd door basale onderdelen van de cybersecurity zoals spamfiltering, up-to-date Windows hosts en systeem administratie, op orde te hebben”, indicating organisations can easily defend against specific malware by implementing basic security measures) were classified as treatments. Voluntary compensation was in 60% of the cases the reason for attribution of treatment responsibility to third party, basic security in 40% of the cases.

Finally, one article mentioned that society could offer **Education** regarding blockchain risks, illustrated by the quote *”De hogescholen laten weten geen aparte voorlichting te geven over de risico’s van cryptovaluta. „Mogelijk volgt dat nog”, [...] „Als dit een probleem blijkt, gaan wij ons hierover buigen.”*”, which states that universities of practice are open to teaching about the risks of cryptocurrencies.

15.3.2 Analysis per attack type

Figure 15.7 presents the parties that were attributed treatment responsibility per type of blockchain attack. While for smart contract and consensus protocol incentives incidents almost no articles assigned treatment responsibility to a party, this is different for OPSEC incidents and fraudulent investment schemes. Specifically for these type of attacks users are attributed treatment responsibility in a relatively large number of articles.

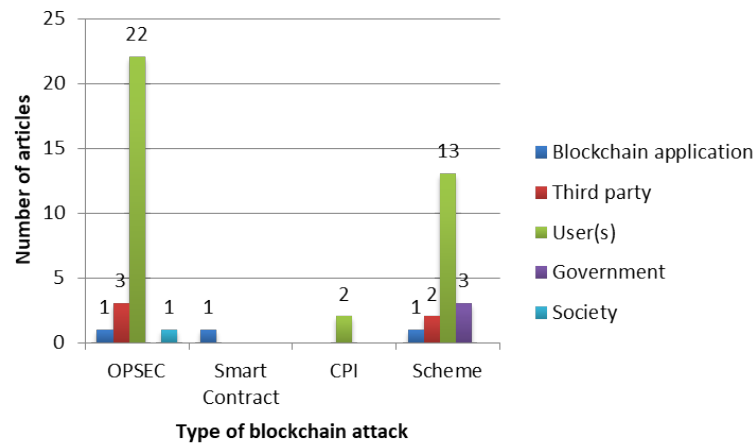


Figure 15.7: Treatment responsibility per type of blockchain attack

For OPSEC incidents, users were classified as being treatment responsible in 22 articles, whilst blockchain applications, third parties, and society were assigned treatment responsibility in just a few articles (1, 3 and 1 article respectively). As can be seen in Table 15.2, 88% of all articles about OPSEC attacks that attributed responsibility, attributed it to users. Overall, 28% of all OPSEC articles attributed treatment responsibility to users.

Similarly, in articles discussing a fraudulent investment scheme, users were classified as being treatment responsible in 13 articles, which makes up for 72.2% (Table 15.2) of all articles about schemes that attributed responsibility to one or more parties. Overall, 14% of all articles about fraudulent investment schemes attributed treatment responsibility to users. Blockchain applications, third parties, and the government were only found responsible in a minimum amount of articles (the blockchain application was mentioned as being treatment responsible in 1 article, third parties in 2 articles, and the government in 3 articles).

15.4 Discussion

In Chapter 11 it was explained that the way in which news media cover an issue influences how people think, and attitudes can be changed through media consumption. The news media content analysis illustrates that causal responsibility (blaming someone) was only attributed in 28.4% of the articles, and treatment responsibility (demanding a solution from someone) even less: in 25.1% of the articles. This indicates that the majority of news articles do not pay attention to why a blockchain attack happened or what solutions could prevent attacks in the future. This may contribute to confusion about blockchain security responsibilities.

Furthermore, attacks related to smart contracts and consensus protocol incentives are not reported on often. Of all analysed articles (275), 4.7% covered a smart contract attack and 2.9% discussed an incident related to consensus protocol incentives. This is in contrast to the coverage on OPSEC incidents and fraudulent investment schemes. OPSEC incidents were talked about in 28.4% of the articles, while fraudulent investment schemes were the topic of main interest in 33.5% of the analysed articles. It makes sense that fraudulent investment schemes have high coverage in news media, as social engineering schemes are prevalent in today's society.

These results are similar to the analysed incidents in the study by Chia et al. [28], which was discussed in Section 12.2. If we dismiss articles that featured fraudulent investment schemes (since these were not part of the constructed database of incidents by Chia et al.), OPSEC incidents were covered in 79% of the media (against 66% in the database), smart contract incidents 13% (22% in the database) and CPI incidents 8% (12% in the database). This suggests that media coverage of different type of attacks conform mostly to the type of attacks that happen in reality.

More than half (61.1%) of all analysed articles name individuals as the victim of the discussed blockchain attack, suggesting that in most cases of described blockchain attacks, individuals suffer losses. Blockchain applications are also often portrayed as the victim of the crime (in 36% of the articles). This is not surprising since blockchain applications are often the direct target of perpetrators.

Causal responsibility was most often attributed to blockchain applications (in 50% of the articles that attributed causal responsibility), to criminals (46.2%) or to third parties (14.1%). End users and governments were almost never attributed causal responsibility. This shows that news media does not view blockchain attacks as the fault of end users, but rather that of (besides the perpetrators) the blockchain or involved third party applications.

Common reasons for the attribution were lack of security for blockchain applications (enabling criminals to perform the attack), ethical hacking for attackers, and negligence of third parties. Ethical hacking has a high number of occurrences due to the fact that a very large blockchain attack where an ethical hacker was involved was covered numerous times in several news media. Lack of awareness from users and insufficient regulation by the government were mentioned just a few times. This leads to the belief that the current (lack of) governmental regulation is not seen by news media as something that enables blockchain attacks.

Users are attributed treatment responsibility most often out of all involved parties (in 65.2% of the articles that attributed treatment responsibility), while society and governments (1.4% and 8.7%) are almost never mentioned. This suggests that there is no the belief among society that governmental regulation or better education can be a solution for the prevention of blockchain attacks. Instead, users are expected to behave better (take more practical security measures) and to enhance their self protection (be more careful and skeptical when it comes to blockchain applications). Blockchain applications are not off the hook either. A little over a quarter (26.1%) of the articles that attributed treatment responsibility, attribute it to blockchain applications. Solutions frequently mentioned are compensation to victims or improving the security of the platform.

It was expected that the type of blockchain attack influences what parties are attributed causal and treatment responsibility. A distinction can be made between Consensus Protocol Incentives (CPI) incidents and smart contract incidents on the one hand, and OPSEC incidents and fraudulent investment schemes on the other hand. CPI and smart contract attacks are specifically aimed at the blockchain technology, while OPSEC incidents and fraudulent investment schemes target individuals (often through third party platforms).

With this reasoning, one would expect that for CPI and smart contract incidents causal responsibility would be most often assigned to blockchain applications (as they are presumed to run a secure application), whereas for OPSEC incidents and fraudulent investment schemes third parties would be found causally responsible. This was however not the case. Blockchain applications were frequently found causally responsible in all types of blockchain attacks, not significantly more times for smart contract or CPI incidents.

Looking at treatment responsibility, it became clear that for CPI and smart contract incidents users were almost never attributed treatment responsibility, while they were attributed treatment responsibility for OPSEC incidents and fraudulent investment schemes

many times, significantly more than any other party. This can be explained by the fact that such attacks generally employ social engineering techniques, which means that victims have control over the course of the attack (and could in principle prevent it).

In summary, we find that while news media frame blockchain applications as often being causally responsible for attacks, treatment responsibilities are still mostly assigned to end users. This suggests that, according to news media, users are expected to take proper security measures to prevent blockchain attacks, while they are not experts and it is unclear whether they are actually equipped to do so. The next chapter, Chapter 16, discusses the legal perspective on who is assigned responsibility with regards to handling the consequences of blockchain attacks.

16

Legal responsibility

Chapter 15 described how responsibilities regarding blockchain security are framed in news media. The framing of responsibilities in the media does not necessarily correspond with who is found to be responsible in practice. To gather insights into who is liable, four professionals in the legal field were interviewed on various types of blockchain attacks. How the interviews were conducted is described in Section 13.3.

This chapter describes the expert opinions on the responsibilities concerning cyber security for blockchain applications. The interviewed professionals are briefly introduced in Section 16.1. Then, in Section 16.2, the findings of the interviews are discussed. The chapter ends with a discussion of the obtained insights from the interviews in Section 16.3.

16.1 The legal professionals

The legal experts that were interviewed are Sandra van Heukelom-Verhage, Dr. Oskar Josef Gstrein, Dr. Thibault Schrepel and Prof. Bart Schermer. All are to some extent involved in blockchain research and have a background in law or governance. The professionals are briefly introduced below.

Sandra van Heukelom-Verhage (further referred to as expert 1) is a partner at the law firm Pels Rijcken and works as a constitutional and administrative lawyer with a focus on innovation and technology. She is a member of the Blockchain Coalition and is the former president of the Legal Expert Group within the Blockchain Coalition (January 2018 - February 2020).

Dr. Oskar Josef Gstrein (expert 2) is an Assistant Professor at the University of Groningen. He has a background in (European Human Rights) law and philosophy and currently conducts research for the Governance and Innovation department at the University of Groningen. He is active in the research fields of International European Law and Multidisciplinary Approaches for Law. Among other things, Gstrein studies emerging technologies and governance (including blockchain), cybersecurity, and human rights in the digital age.

Dr. Thibault Schrepel (expert 3) is an Associate Professor of Law at VU Amsterdam. His latest research is focused on blockchain antitrust and how law and technology could

cooperate. Recently he published a freely available book on this topic called “Blockchain + Antitrust”. Part of the research Schrepel has conducted is about how blockchain can work for modern societies, and how public permissionless blockchains (when anyone can write on the blockchain ledger) make it difficult to enforce the law.

Prof. Bart Schermer (expert 4) is a Professor of Law and Digital Technology at the University of Leiden. He has a background in Law and Information Technology. His research focuses on privacy and cybercrime, specifically the relation between enforcement and human rights in the online world. He is also a fellow at the E.M. Meijers Institute for Legal Studies and active as a partner at Considerati, a legal consultancy firm specialized in legal and policy advice for IT and new media. Furthermore, he is a member of the Cybercrime expert group for the Court of Appeal in The Hague and a member of the Human Rights Committee of the Advisory Council on International Affairs.

16.2 Interview findings

Based on the different types of blockchain attacks that were discussed in Section 12.2, the legal experts were asked to give their thoughts on examples of each type of blockchain attack. The general opinions per attack type are discussed below. Our interpretation of what was said during the interviews was validated by all experts.

16.2.1 OPSEC incidents

The information described in this section is based on statements from expert 2, expert 3 and expert 4, and aligns with all three expert opinions.

OPSEC incidents often happen through the use of a third party platform, for example an application store or a social media platform. Attacker(s) use such platforms to get in (online) contact with their victims to steal private information. In a general sense everyone bares their own damage until it is proven that someone else is to blame (civil and liability law). Whether a third party platform is responsible for harm done to users depends on the criteria they have set for themselves on what it means for content to be safe (enough) to be published.

This might change as there is currently a draft¹ of new European legislation named the Digital Services Act² (DSA). Currently platforms active in the EU have the duty to do everything they can to protect customers (and thus detect evil contents). Legally speaking

¹The European Parliament approved the Digital Services Act (DSA) on the 5th of July, 2022.

²The Digital Services Act (DSA) clarifies the responsibilities of digital services like online platforms regarding activities and information for consumers. The DSA helps combat illegal contents online by clarifying the role of providers of digital services and creating procedures for carefully handling this kind of contents.

this means that platforms must employ all the means they can to do this, but there is no duty to obtain a certain result.

Legislation and liability should be aligned with the power of prevention. If a party is not in control, they should not be liable. Technically it is possible for online platforms to check all published contents. However, because there is too much information, it is infeasible to do so. With more powerful algorithms in the future this might be subject to change. At the moment, if it cannot be proven that a platform was aware of malicious contents, they are not liable.

In any case, the attackers are liable, but if they remain unidentified, victims cannot retrieve their lost funds. For some incidents governments have designated funds available to help victims³. A similar fund could be set up for blockchain incidents so victims can be compensated when perpetrators cannot be tracked down. Such a fund currently does not exist.

16.2.2 Smart Contract incidents

The information described in this section is based on statements from expert 3 and expert 4, and aligns with both expert opinions.

In principle, the person who exercises power, is liable. When a mistake in smart contract code results in damages, whoever has power over the blockchain platform is liable. Looking at a software development context however, users are usually out of luck. Companies are not necessarily liable for offering a defective product on the market. For large organisations, generally customers have to agree to predefined terms and conditions. With this, companies can contractually exclude liability.

To determine liability, first one needs to verify whether a party is guilty of culpable behaviour. Next, the point of contact and the culpable behaviour need to be identified and clear. Usually blockchain applications are open-source. There is no clear point of contact. Developers of the blockchain application design it together or separately from each other, without the presence of a legal entity or foundation. It depends on the expectations a cryptoplatform has set, and to what extent they are involved with the code implementation or have contractually excluded liability, whether they are liable.

A large problem concerning enforcement is the international context of blockchain. What jurisdiction is applicable and what legal forum can be used? When using an exchange that is based in a different country, it may not be possible to get justice in your home country. Platforms have to conform to financial legislation in the countries where they are active. Crypto exchanges in the Netherlands need to register at DNB and AFM (and thus conform to certain rules), but they are not under financial supervision.

³In the Netherlands there is a fund called the Waarborgfonds Motorverkeer for victims involved in car accidents where the perpetrator is unknown or uninsured.

It is unlikely that compensation from a crypto exchange that contained a vulnerability causing losses for users is enforceable. Cryptoplatforms may however choose to compensate losses to win back trust from users of their service.

16.2.3 Consensus Protocol Incentives (CPI) incidents

The information described in this section is based on statements from expert 3 and expert 4, and aligns with both expert opinions.

The user should be aware of the inherent risks of blockchain (attacks on blockchain's core elements). When using a blockchain application you are essentially responsible (barring your own damage until it is proven that someone else is to blame). There will always be a risk that the blockchain system falls apart, and that when that happens there is no one (no central authority) to recover damages from. The attacker(s) are liable, but it is challenging to identify them.

Overall CPI incidents are complex as there is not one single party to blame. Schrepel explains in his book the concept of a blockchain nucleus [106]. This is a group of participants that come together to achieve a form of control over the blockchain by collaborating, by circumventing (some of) the (economic, political, logical, sociological, architectural and legal) constraints imposed on them, and by changing them in the long run. The nucleus can become a legal entity that can be assigned liability. There are three types of blockchain participants: founders/core developers (who implement the original rules of the blockchain: design the code and determine the consensus protocol to be used), users (who propose new transactions but cannot easily exercise coordinated control), and miners (who validate transactions assembled into blocks). Generally in decentralized permissionless blockchain applications (like Bitcoin or Ethereum) no one can unilaterally impose changes.

Participants are to be held liable for illegal conduct committed within the nucleus or perimeter that they can control or influence to a great degree. The width of that perimeter depends on the case. Lack of control should result in a lack of liability. Still, concrete and quantifiable frameworks are necessary to define the nucleus in each case to ensure legal certainty, limit legal errors and reduce regulatory costs.

16.2.4 Investment schemes

The information described in this section is based on statements from all experts, and aligns with all four expert opinions.

Blockchain is used as another layer of complexity to set up schemes. This extra layer makes the scam more plausible. Since everyone knows there is a lot of money in blockchain, cryptocurrencies are attractive to use as bait. In practice blockchain investment schemes are no different from other investment schemes that do not involve blockchain technology.

Blockchain should not be viewed as something entirely different from other technologies. There is no separate law for blockchain but existing legislation (for financial supervision, liability law, criminal law) can be applied to technological applications.

The people that set up the scheme are likely liable (due to intent of deceit and forgery) but it can be difficult to find them to recover any damages. The blockchain infrastructure makes it easier to conceal traces of the money going into the scheme, making it a complex task to trace it back to the perpetrators. It is more a question of operability than of liability (the creators of schemes are liable). The liability is in this case not enforceable, but it is in fact there. In practice this means that victims cannot be compensated.

In some cases victims are in a (non-legal) sense responsible for losing their money since they consented to invest money themselves. People always have some responsibility themselves. If perpetrators are not found, users have to deal with the damages themselves. Other possible liable parties are intermediaries.

Intermediaries (e.g. banks or social media platforms) have a *duty of care*⁴ for consumers and a *duty to inform*⁵ (informatieplicht) while consumers have a *duty to investigate*⁶. There is a balance between the duty of care and the duty to investigate. In case a breach of duty of care can be indicated, an intermediary can be held liable, but it may be problematic to determine such a breach. Generally social media platforms are not liable (according to the DSA⁷) for rogue content on their platform unless they are informed about it and did not intervene but platforms may be required by national courts to implement measures to prevent future violations.

Influencers that promote investment schemes can also be found liable, that is if they were aware of the fraudulent construction that they were promoting to others, in which case they were doing it for the purpose of self-enrichment. If they were not aware of it, they cannot be blamed for the damages they cost others.

16.3 Discussion

For all of the different cases, it is clear that a crime has been committed, and the question is not that of responsibility but of enforcement. Overall, whoever is in control (generally the perpetrators), is liable. The crime territory goes across borders and the blockchain technology allows criminals to carry out crimes with ease. While laws are in place and deemed

⁴A duty of care is a legal obligation which is imposed on an individual or organisation, requiring adherence to a standard of reasonable care while performing any acts that could foreseeably harm others.

⁵A seller has a duty to inform (an obligation to provide information). He should tell the customer what he knows about the product or service that he is offering.

⁶A consumer has a duty to investigate. He should investigate whether the product or service that is offered meets their demands.

⁷The Digital Services Act (DSA) clarifies the responsibilities of digital services like online platforms regarding activities and information for consumers. The DSA helps combat illegal contents online by clarifying the role of providers of digital services and creating procedures for carefully handling this kind of contents.

sufficient, it is difficult to enforce them. The enforcement mechanisms have limitations. The way that rules are set up (nationally) is a restriction to enforce them, especially for blockchain applications, where there are no territorial borders. The game of the crime is being moved away from the space of the territory. It requires a large amount of public resources to follow blockchain data traces (and money) to the end point.

Intermediary parties such as online platforms could be held responsible if a breach of duty of care can be proven, but this is often difficult. In the future online platforms may be held responsible more often, when technology is advanced enough to detect all fraudulent uses but currently this is understood as infeasible. In some cases an intermediary party can be held liable, if it can be proven that they were aware of the illegal contents on their platform and they had the power to do something about it.

In general, it seems that people underestimate risk and law enforcement is complicated. Cryptocurrencies live in a wild west territory - it is relatively new and there exist many optimistic stories about people becoming wealthy quickly. The lack of a central authority and point of contact, which is simultaneously promoted as one of the big advantages of blockchain technology, also means that there is no government protection for users. A government fund may be constructed in Europe that compensates users if perpetrators remain unidentified.

To identify the wrongdoer is not always possible, especially when the blockchain is impacted from within (due to the anonymity blockchain offers). If it however is impacted from the outside (by making use of a centralized system - where identities are known - like a cryptoplatfrom or social media platform) the perpetrator can usually be identified. If the perpetrator can be identified, the victim(s) can be reimbursed. If not, and if platforms are not liable, victims have to suffer the losses. Many consumers trade blockchain assets through cryptoplatfroms, so it is likely these platforms will be regulated similarly to banks in the future.

In the previous chapter, we found that users are rarely attributed causal responsibility for blockchain attacks. The findings of the interviews support this. Users are not liable - attackers are. We also found that while causal responsibility lies mostly with blockchain applications according to news media, currently blockchain applications are difficult to hold (legally) accountable.

Furthermore, users are often attributed treatment responsibility (particularly for OPSEC attacks and fraudulent investment schemes). The interview findings highlight that when there is no other party to hold liable (e.g. if the criminals cannot be found, which is not uncommon), users have to suffer the consequences. It is therefore not strange that users are attributed treatment responsibility, since there is no other party that can be relied on to help deal with the consequences when an attack happens.

The way that blockchain technology works (i.e. being an international network, offering

anonymity to a degree, allowing for use of smart contracts by anyone) is not unrelated to this. The next chapter, [Chapter 17](#), goes into the influences of developers and the blockchain design on the division of security responsibilities.

17

Developers' influences on user responsibilities

In Chapter 15 it is presented how responsibilities regarding blockchain security are framed in news media. Chapter 16 explains how responsibilities are assigned in the legal sense, and highlights some problems with enforcement on blockchain applications. This chapter aims to view the division of responsibilities from a blockchain developer's perspective. Through interviews with blockchain researchers, some interesting thoughts were gathered on this topic. Section 13.4 discusses how the interviews were conducted.

This chapter shares some views on how the implementation of blockchain technology influences the division of responsibilities. The interviewed professionals are first introduced in Section 17.1. Afterwards, my interpretation of the discussions with the experts on different aspects of blockchain is described in Section 17.2.

17.1 The blockchain professionals

The blockchain experts that were interviewed are Dr. Oğuzhan Ersoy and Dr. Balázs Bodó. Both are researchers of blockchain based technologies. Dr. Ersoy works mainly on the design of secure blockchain protocols, while Dr. Bodó is interested (among other things) in how features and faults of blockchain design create significant deviations from the societal expectations embodied in institutions, laws, and ethical frameworks. The professionals are briefly introduced below.

Dr. Oğuzhan Ersoy is a Postdoc researcher at the Radboud University. He works in the area of decentralised systems and blockchain technology, specifically on provable secure, privacy-preserving, and scalable and incentive-compatible protocols. Previously during his PhD he also studied the security and economic aspects of decentralised systems (including blockchain).

Dr. Balázs Bodó is an Associate Professor working at the University of Amsterdam at the Institute for Information Law. He researches various aspects of the societal effects of blockchain technologies. In 2018 he started the Blockchain & Society Policy Research Lab

to study the non-technical, societal, economic, policy and legal implications of blockchain based technologies.

17.2 Interview findings

In the open discussions with Dr. Oğuzhan Ersoy (further referred to as expert 1) and Dr. Balázs Bodó (expert 2) various aspects of blockchain responsibilities were touched upon. In this section the relevant findings from the interviews are described.

When it comes to blockchain technology, there are various assumptions that users have, that may not be valid. One of those assumptions is the decentralised nature of blockchain. Whether the current implementation of blockchain really is fully decentralised is questionable.

Is blockchain actually decentralised?

The information described in this section is inspired by statements from expert 1 and expert 2.

In the DAO hack (July 2016) on the Ethereum blockchain network, due to a fault in a smart contract an attacker was able to spend the large sums of money stored in the contract. The general idea of blockchain is that it is not invertible (immutable) and transactions are final. In this case however, Ethereum developers proposed to fork the blockchain (to roll back Ethereum network's history to before the attack). This was very controversial since blockchains are supposed to be immutable and censorship-resistant. While most stakeholders adopted the change, allowing for the fork to be implemented, not all did, resulting in two separate Ethereum blockchains. The fact that this could happen highlights that there is no full decentralisation and censorship is possible.

Another interesting aspect of the decentralisation of blockchain is that all hashing power is concentrated in just a few large mining pools. This indicates that in reality the network is not actually decentralised, as just a few decision-makers hold the majority of the hashing power and can exert power over the network. For BTC for example, the four largest mining pools together hold over 50% of the hashing rate.

Apart from the core blockchain technology, many elements of blockchain applications are actually not decentralised. Crypto exchanges, wallet management applications, and other blockchain-related applications are all a step away from true decentralisation as such applications are centralised. Very few people actually run their own server node to perform blockchain activities.

Besides the fact that blockchain may not be as decentralised as one might think, we may wonder if decentralisation really is something that is worth pursuing.

Is decentralisation desirable?

The information described in this section is inspired by statements from expert 2.

The blockchain system reflects a(n) (American libertarian) zero-trust society, where no one can be trusted. It goes against the beliefs that the state is not the enemy and that intermediaries have an important role to play in helping citizens. Full decentralisation requires lay people to do things that we are not equipped to do. Lay people do not have the knowledge to for example store private keys in a secure manner. Mistakes will be made.

Centralisation solves this by leaving such tasks up to professionals (e.g. they can make sure that people are not locked out of their bank account, they can restore forgotten passwords for them). Full decentralisation removes this solution and ships all responsibility to the end user. There is no censorship, which also means there are no rules.

If a user makes a mistake, there is no one to help. Few users completely understand what they are expected to do (or are doing), since they lack expert knowledge. If you lose your private key, if a smart contract contains a bug that is exploited, if some other bad thing happens, the investors lose their money, and there is no one to turn to. Unlike traditional financial systems, any of these decentralised intermediaries offer less consumer protection than traditional solutions.

For developers it is completely rational to push all responsibility to users, to avoid any liability and the possibility to go bankrupt when having to pay damages in compensation. In that sense, decentralisation perfectly achieves that. Still, although there is no legal clarity, there likely are liable parties. There are centralised legal entities (core developers, decision-makers, owners). For example the Ethereum foundation (that is behind the core development and able to influence in which direction the system goes) will likely have a difficult time denying that they have something to do with the system. It is a relative new field for judges but there are good arguments to be made that blockchain systems, like any other system, have liable parties that can be made legally liable for their actions.

Moxy Marlinspike (founder of the Signal app) makes the case that blockchain technologies immediately tended towards centralisation through platforms in order for them to be realised. Most participants do not even know or care decentralisation is happening, suggesting that decentralisation itself is not actually of importance to the majority of users. Distributed systems may make things more complicated and more difficult rather than less complicated and less difficult. Perhaps the only amount of decentralisation people want is the minimum amount required for something to exist.

Not anonymous

The information described in this section is inspired by statements from expert 1.

Another assumption users have is that blockchain ensures anonymity. It is however proven that pseudonymous identities could be linked to real identities through network analysis and investigation.

Furthermore, crypto exchanges function as gates between blockchain systems and the real world. This means that crypto exchanges do know real identities of blockchain users.

Blockchain implementation: consensus protocol and smart contracts

The information described in this section is inspired by statements from expert 1 and expert 2.

An important element of blockchain technology is the consensus protocol. Various consensus protocols exist (Proof of Work, Proof of Stake, Byzantine fault tolerance, hybrid versions). Trade-offs in usability, security, eco-friendliness, developer-friendliness are made when selecting one protocol. No one protocol is the best.

Factors that need to be considered are whether the blockchain needs to be accessible to everyone, and whether anyone can create transactions (private or public, permissioned or permissionless blockchains). Energy consumption is also a factor, PoW uses a very large amount of energy, while PoS has low energy consumption, but other (security) challenges. Scaling is another important point. Some protocols do not scale well, making them inefficient for wide usage.

One of the most crucial security aspects of blockchain applications are smart contracts. All other elements (e.g. consensus protocols) of the blockchain are extensively researched. A fault in a consensus protocol can affect the entire blockchain (e.g. with a 51% attack or selfish mining attack) but these attacks have been greatly studied. Smart contracts are not studied that well. Every smart contract is different, has its own functionality and implementation. Errors are made at the smart contract level. Normally users are protected if they buy a defective product but if users use a defective smart contract, it is questionable who is liable.

Users should check contracts before using them but without being an expert this is difficult, even if most of them are publicly available. There are some companies that do formal verification (e.g. penetration testing of smart contracts). Most contracts are also reviewed online.

Improving security through market competition or legislation

The information described in this section is inspired by statements from expert 2.

Quality (in terms of for example security, honesty, well maintained) of blockchain applications can be impacted in two ways:

1. Legislation: through legislation certain requirements for blockchain applications can be enforced.
2. Market competition: market competition ideally produces quality when firms compete and consumers go for the provider with the best quality.

Currently there is market competition in the blockchain field. For blockchain applications it is however difficult for end users to determine the level of quality. There exist third party auditors that can verify some aspects of blockchain projects (e.g. smart code contract auditors) which can grow trust with end users and can help projects in getting an advantage against competitors. However, it does not necessarily prevent mistakes.

Market competition alone will likely not work for blockchain to produce honest players. It is better to enforce rules instead of using the market place as an incentive. The blockchain space is a wild west. The market is not about quality but just on how to make more money in less time. Market competition will not guarantee better security. Applications compete on money instead of quality, and law enforcement is required, and is coming. There is no way for blockchain networks to avoid financial regulation eventually since all financial institutions need to comply with certain laws.

What are developers' responsibilities?

The information described in this section is inspired by statements from expert 2.

It is a complex question what responsibilities developers have towards users. First we need to make a distinction between good faith and bad faith developers. For good faith developers product liability rules should apply, in the same way that other services have to guarantee consumer protection. Relying on good faith of developers or counterparties without regulation does not work. Regulation is required to define responsibilities for blockchain applications and to hold them legally accountable.

17.3 Discussion

The interviews raised some valid concerns regarding blockchain technology. Blockchain is not as decentralised as it is made out to be. Most people that are involved with blockchain technology, use centralised access points. Moreover, the hashing power is concentrated in

only a few large mining pools, and censorship of the blockchain is in principle possible. [Marlinspike](#) [73], a computer security researcher and creator of the *Signal*¹ app has stated that with public blockchain, we may end up with the worst of both worlds: centralised control, but still distributed enough to become mired in time. Besides this, blockchain technology is not actually anonymous, as users can be linked to real identities in practice. Blockchain may represent advances in encryption and security, but it is vulnerable in some of the same ways that other technology is, and has new vulnerabilities of its own [70].

With these issues, we may wonder if use of blockchain technology is even desirable. [Schneier](#) [104], an internationally renowned security technologist, argues that blockchain does nothing to solve any existing problem with financial (or other) systems. Adding blockchain causes new problems and can make systems worse. Blockchain is designed to shift trust in people and institutions to trust in technology: the cryptography, the protocols, the software, the computers and the network [103]. Full decentralisation assigns all security responsibilities to users. The majority of blockchain users do not have the expertise to audit the computer code used by the blockchain, yet are expected to trust it. When that trust turns out to be misplaced, there is nothing left to do (unlike with a human legal system, where matters can be reverted).

While some researchers believe firmly that blockchain is expected to bring significant benefits to society [5, 14], the current market competition for blockchain applications does not guarantee better quality and security. Only regulation can force blockchain applications to satisfy a security standard, and to provide users with some security guarantees. The fact that blockchain applications need to be regulated by governments in order to provide security for users is ironic, as the purpose of blockchain is to shift trust in governments to technology [103]. Added to this, since current applications of blockchain applications tend towards centralisation anyways, perhaps we are using blockchain for the wrong purposes.

¹Signal is a free, privacy-focused messaging app you can use on smartphones and via desktop.

18

Conclusion

In this chapter, the research questions as first presented Chapter 11 are answered. First, the research sub-questions are answered in Section 18.1. The answers form the basis for the conclusion of the main research question in Section 18.2.

18.1 Research sub questions

1. How is cyber security responsibility for blockchain applications framed in news media channels in the Netherlands?

To answer the first research sub question, we analysed 275 relevant articles discussing blockchain attacks, during the period of September 2020 and August 2021, with respect to causal and treatment responsibility.

From the content analysis we found that end users are named as victims of blockchain attacks in the majority of news articles, indicating that overall news media acknowledges that users are harmed in blockchain attacks. Nevertheless, responsibility is not often attributed in the analysed articles. A little over a quarter of the articles attributes causal responsibility to blockchain applications (50%, mostly due to a lack of security), attackers (36%), third parties (14%), end users (5%), and/or governments (3%). Furthermore, in a quarter of the articles, treatment responsibility is attributed to users (65%), blockchain applications (26%), governments (9%), third parties (7%) and/or society (1%).

The results show that causal responsibility, when attributed, is focused mostly on blockchain applications and criminals. Treatment responsibility however, is most often attributed to users, with blockchain applications being second in line with a much lower number. Society and governments are almost never attributed treatment responsibility, leading to the belief that attacks on blockchain applications are not viewed by news media as a societal issue. Instead, users are attributed the responsibility to prevent attacks targeted at them.

Farout most of the articles talk about OPSEC attacks (28% of the articles) and fraudulent investment schemes (33% of the articles). CPI incidents and smart contracts are generally not covered or specifically mentioned in the analysed articles (discussed in only 3% and 5% of the articles respectively). The majority of news articles about OPSEC incidents and

fraudulent investment schemes make recommendations to users on better behaviour and self-protection enhancement.

2. How do legal experts view responsibilities for the consequences of blockchain attacks?

There was a consensus among the different legal professionals that perpetrators of (all categories of) blockchain attacks are always liable. The existing laws are applicable and sufficient for blockchain attacks. It is however difficult for police and law enforcement bodies to identify the perpetrator and to press charges. The international and pseudonymous nature of blockchain technology plays a big part in this. The legal experts agreed that to overcome the difficulty of law enforcement, international cooperation is necessary and governments must regulate blockchain applications better. This way, law can be effectively enforced.

In practice, users generally are unable to claim compensation for any damages from blockchain applications if criminals remain unidentified. Involved intermediaries like social media platforms can in some cases (specifically for fraudulent investment schemes and OPSEC incidents) be liable for enabling attackers to perform attacks, but these attacks are not new or specific to blockchain. Blockchain applications are relatively new (and so are the attacks specifically targeted at them) and it is unclear what legal obligations they have and how these can be enforced.

3. What can developers do to establish an effective division of responsibilities?

Essentially, the influence of developers on the division of responsibilities is limited. Full decentralisation (the idea of blockchain) by definition assigns all responsibilities to end users. There is no censorship, and no rules, in a fully decentralised system.

Interestingly, current blockchain implementations are not actually fully decentralised. The past years it has become clear that hard forks (example of censorship) are possible, the majority of the hashing power used in blockchain networks is concentrated in just a few mining pools, and access points to blockchain are centralised (e.g. crypto exchanges, wallet managers).

We cannot rely on the good faith of developers (or any other involved party for that matter) to implement secure and foolproof blockchain applications. Currently blockchain applications are regulated through market competition and this does not work. Users are unable to determine good or bad quality. Core elements of blockchain technology (like consensus protocols) are heavily researched, but smart contracts are not, while they are very error-prone. Most blockchain attacks are possible due to bad designs. Normally users are protected when buying a defective product (through consumer protection legislation), but this is not necessarily the case when using defective smart contracts. Regulation (e.g. in the form of product liability rules) is required to ensure security.

One might wonder whether distributed systems actually make things less complicated and less difficult at all. From the start blockchain technologies tended towards centralisation

through platforms in order for them to be realised. Decentralisation itself is likely not actually of importance to the majority of users.

18.2 Research main question

The main research question is as follows:

How are cyber security responsibilities assigned for blockchain attacks and what can blockchain developers do to influence this?

The answer to this question is a summation of the answers to the three sub-questions. The general conclusion of the first sub-question is that while blockchain attacks are most often described without attribution of causal or treatment responsibility, when they are, blockchain applications are most often portrayed to be causally responsible. Treatment responsibility is however frequently assigned to end users, specifically for OPSEC incidents and fraudulent investment schemes. This is characterised by the high presence of recommendations for better behaviour and self-protection enhancement in news articles about OPSEC incidents and fraudulent investment schemes. For smart contract incidents and CPI incidents the number of articles that described these incidents, whilst also attributing treatment responsibility to one or more parties, is too low to draw a meaningful conclusion.

From the second sub-question it followed that attackers are legally responsible according to current legislation, but if they remain unidentified (which is easier with blockchain technology compared to traditional crime) and cannot be prosecuted, users are unable to retrieve compensation. Blockchain applications are not liable either. Thus, while users are not framed as having causal responsibility and also do not have legal responsibility, they often have treatment responsibility. In essence, they are still expected to prevent attacks happening to them. This makes sense since in practice users are not entitled to any compensation from blockchain applications and attackers are typically never identified.

The results from the third sub-question highlight that the influence of developers on the division of responsibilities is limited as there is not much they can do: full decentralisation (the core idea of blockchain) by definition hands over all responsibilities to end users. This leaves legislation as the only solution to define a clear division of responsibilities, and to provide a legal basis for end users to claim compensation from blockchain applications when a blockchain attack happens.

All in all, this research provides valuable insights into three different perspectives on blockchain security responsibilities. First, we found that news media mostly attribute responsibility for the prevention of blockchain attacks to end users¹. Second, we learned that generally

¹It should be noted that attacks specifically aimed at the blockchain infrastructure are not covered in news media as often as traditional cyber security attacks involving blockchain applications.

18. CONCLUSION

end users have to deal with the consequences of blockchain attacks. It is unknown what specific legal obligations blockchain applications have. Due to this, it is difficult to hold blockchain applications accountable when an attack happens. Last but not least, this study highlights that the blockchain design, full decentralisation, essentially shifts all responsibilities to users.

19

Discussion

This chapter outlines the discussion with regard to the conducted research. First, the relevance of the findings of the study is discussed in a societal and scientific context in Section 19.1. Next, Section 19.2 elaborates on the limitations of this research. Afterwards, in Section 19.3 the reliability and validity of the research are discussed, as well as the ethics concerned with the research. Finally, recommendations for future research are given in Section 19.4.

19.1 Contributions and relevance

The first part of this thesis focused on the software testing of blockchain applications. Developers are to some extent responsible for the security of the application they create; they are expected to thoroughly test it. However, applications are rarely foolproof, and attacks happen. The present study examined whom is responsible for the cyber security of blockchain applications according to news media and in a legal sense. In addition, the study identified aspects of blockchain implementations that influence responsibilities of users.

To this day there have not been studies that investigate news media framing in the context of blockchain application attacks. In Chapter 11 it is stated that the phenomenon of victim-blaming occurs often for cyber attacks. Victims of cyber attacks are held responsible, owing to not having taken deliberate action to make themselves less vulnerable [93]. The content analysis of news media showed that interestingly enough this is not the case for blockchain attacks. Causal responsibility is generally not assigned to users, but to the attackers or the blockchain application itself. On the other hand, treatment responsibility is in most of the articles assigned to users. It seems that while users are not deemed causally responsible, they are supposed to provide a solution for incidents that happen.

This study leads to believe that this could be related to the decentralised nature of blockchain. Blockchain activities are delegated away from a central organisation and distributed among users (pushing all responsibilities on them). This study found that practical problems with the enforcement of attackers are likely the cause of why users normally do not receive any compensation. Without a clear legal entity to hold liable, it makes sense that users themselves are deemed responsible for treatment.

In Chapter 11 the contributions of this research were described, specifically a generic theoretical framework and coding schedule, insights into responsibilities, and various considerations regarding blockchain technology. Firstly, the constructed theoretical framework and corresponding schedule can be used to analyse any sort of news media (e.g. news papers, social media, television) that discusses blockchain attacks. The framework and coding schedule can be easily adapted to focus on other cyber security contexts, or they can be expanded to focus on the design (attack) layers of other technological applications. Secondly, this research illustrated that users are not causally responsible for blockchain attacks according to news media. This may reflect the way society thinks. Still, users are the party that is mostly assigned treatment responsibility, especially for schemes and OPSEC attacks. Strangely enough, for consensus protocol incentives incidents and smart contract incidents there are rarely any treatment solutions offered. This perhaps indicates that solutions for such incidents do not exist or are (relatively) unknown. From a legal perspective, there is a lack of (international) regulation in the blockchain field, making users responsible for the consequences of attacks due to attackers not being able to be identified and blockchain applications not being liable. Thirdly, this study highlights various considerations regarding blockchain technology that offer a different outlook on blockchain, such as issues with market competition, current regulation and the untrue assumptions that users have regarding blockchain technology, specifically the extent to which current implementations of blockchain are decentralised.

The ambiguity regarding the extent to which blockchain is truly decentralised is an interesting subject to focus on for science communication professionals. It is unclear whether users are aware of the fact that how they are using blockchain (usually through a centralised application like a crypto exchange) is only possible because of a centralised third party. A science communication professional could fulfill a role in helping end users understand what exactly users are involved in and what risks they face when making use (or not) of such a centralised third party to access the blockchain network.

While this research is focused on the Netherlands, blockchain applications are used worldwide. Security and regulatory issues that are present here in the Netherlands, are likely also present in other countries. It is plausible that the results of this study are relevant in other countries in the EU as well.

Blockchain security is a small part of cyber security in general. This study focused on blockchain security specifically, but some of the results may be generalised to hold for cyber security as well. It is expected that cyber security responsibility framing follows a similar trend, making end users responsible for the prevention of cyber attacks. While the consensus protocol incentives incidents and smart contract incidents may be specific to blockchain, OPSEC attacks and fraudulent investment schemes have always been a signif-

ificant part of cyber crime. In any case, for blockchain security or cyber security in general, security measures may be too complex for end users to adhere to, so attribution of treatment responsibility may be unreasonable.

19.2 Research limitations

There are many ways to perform this type of research. The choice of theories in the framework, the coding protocol, the interview methods, the analysis of the news media: decisions were made for each step of the research. Below the limitations of the made decisions are discussed per method.

For the construction of the theoretical framework, the used query to find framing theories was the starting point of the literature research. The research idea in the beginning was however slightly different and focused on victim blaming. This is reflected by the query. The focus on victim blaming may have limited the amount of theories that were encountered in the literature study. Nevertheless, the framework that was eventually constructed based on the literature was suitable to be used for the content analysis.

There were several limitations to the content analysis. First the sampled articles are discussed. The used *Coosto* tool only provides access to free articles. This limited the articles that could be sampled as it was for example not possible to analyse renown or popular news papers in the Netherlands. Furthermore, the search yielded many duplicate articles that required manual filtering. It is possible that some mistakes were made, like accidentally identifying an article as duplicate or analysing duplicate articles. Additionally, the search yielded numerous inaccessible articles (i.e. articles behind a paywall or articles with a broken link). Another limitation is that it was not possible to download all textual articles, meaning it was not possible (without much manual effort) to perform a textual analysis (e.g. counting frequent terms, or identifying exact duplicates).

Second, the great majority of articles were deemed irrelevant. This could be due to the query not being specific enough. However, in various articles keywords from the query were not present, suggesting that the *Coosto* tool might not work entirely as one would expect.

Third, some articles were (near) duplicates but did not reference another news paper as a source. It was infeasible to remember for each article whether it was already come across (and whether the differences were considerable enough to count it as a separate sample), which led to such articles being included in the analysis. This may impact the results of the analysis since (near) duplicates yield (almost) the same coding.

Regarding the interviews with legal professionals, not all experts had working experience in the legal field. Nevertheless, all have worked in either the legal field or as a researcher in the legal field, which is why this does not necessarily have a negative effect on the research.

Still, experts working as lawyers in the specific field of cyber security might have been able to offer more detailed views on liability or would know more about specific legal cases that happened.

On top of that, due to the limited time that was planned for the interviews it was not possible to question the experts on all cases. Based on the second interview additional cases were added to question experts on, which is why not all interviewees commented on the same cases. The first two professionals were later asked to comment on the three additional cases as well, but they did not have the time to do so. Despite this, enough information was gathered from the other two legal experts to properly analyse the additional cases.

The third research question on what developers can do to establish an effective division of responsibilities is not a very straightforward question. It proved difficult to find professionals that felt this was within their area of expertise. Four professionals in the blockchain research field were approached, and two agreed to an interview.

It is possible that other experts would have different opinions. This is however not necessarily a bad thing as the research sub-question aimed not to gather a complete and correct overview of everything developers' can do to influence users' responsibilities, but to highlight various relevant considerations regarding blockchain implementations.

Due to the openness of this question, the exploratory interviews had a very large scope and it was difficult to identify a clear answer to the question. The professionals that were interviewed had many insights but not necessarily related to the research question. The open conversations allowed for many insights to be gained, but some time was spent going too much into detail on one topic, for example on some technical aspects of blockchain technology. This was in hindsight not that relevant and the time could possibly be better spent.

19.3 Reliability, validity and ethics

This study is conducted in a systematic way to safeguard reliability. Each step was documented throughout the research process and used protocols for the methods are included in the appendices of this report. Mostly the content analysis provided a risk for the reliability of the research as the coding process can be subjective. For this reason, the coding protocol was constructed in such a way that benefits consistency among multiple codes to a high degree. Furthermore, the coding schedule was piloted and a check for intercoder reliability was performed.

To guarantee validity of the research, the content analysis of the news media was based on verified framing theories that have been often used in prior studies. For the conducted interviews, professionals were contacted that had spent much time in the legal or blockchain research field. The interviewed experts often highlighted different aspects in their answers but overall the different answers were consistent.

Conducting interviews with professionals required some ethical considerations. Because much was said in a short period of time (one hour), the interviewees were asked whether the interview could be recorded. All interviewees agreed to this. After the research was finished, all recordings were deleted. Furthermore, after the interviews were processed, the interviewees were asked to approve what was written about what they had said during the interview. Not all interviewees felt comfortable with the public publishing of the interview summaries. It was therefore opted to write about the general interpretations following from the interview findings, rather than the use of specific quotes or statements.

19.4 Recommendations for future work

While this study provides insights into whom is held responsible for cybersecurity of blockchain applications, the body of (relevant) content that was analysed was rather small (275 articles) and consisted of mostly fairly unknown news sources. A follow-up study on blockchain security responsibilities should be done on a larger sample, preferably consisting of renown news sources to provide a better reflection of the news society reads. Furthermore, online news sites may perhaps attract different readers than social media platforms. It could be interesting to investigate how responsibilities are assigned in a less official context, as texts are likely more opinionated.

Moreover, this study was focused on news media in the Netherlands. A second recommendation is to compare assigned causal and treatment responsibilities among different countries. Each country (especially outside of Europe) has their own legislation that may impact how blockchain security is looked upon. In China for example trading in cryptocurrencies is illegal. It begs the question whether news media also discuss blockchain attacks differently due to this. Besides causal and treatment responsibility, the tone (positive/negative towards blockchain technology) of articles may also be an interesting additional element to analyse, as it could be helpful to decide upon whether blockchain technology is actually perceived as something good or bad (dangerous) according to news media.

Finally, it might be interesting to move away from blockchain applications, but focus on third party (centralised) applications that embed blockchain systems in their application. From the results of the third research sub-question it was found that most individuals make use of these centralised applications (e.g. crypto exchanges) rather than perform activities on the blockchain directly. Such applications are currently monitored by De Nederlandsche Bank (DNB) to regulate only two laws related to money laundering and financing terrorism. They are however not under supervision by DNB or AFM. This means that there is no supervision on financial business risks and there is no financial consumer protection of any kind. Such applications are centralised so there is a legal entity that can be held liable, but in practice it is not obvious whether end users are entitled to compensation when something goes wrong. A study looking into consumer protection in the context of such applications could provide insights into this.

Bibliography

- [1] Raja Ben Abdessalem, Annibale Panichella, Shiva Nejati, Lionel C Briand, and Thomas Stifter. Automated repair of feature interaction failures in automated driving systems. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 88–100, 2020.
- [2] Victoria Aboungo, Elizabeth Kaselitz, Raymond Aborigo, John Williams, Kat James, and Cheryl Moyer. Why do community members believe mothers and babies are dying? behavioral versus situational attribution in rural northern ghana. *Midwifery*, 83:102657, 2020.
- [3] Shaik V Akram, Praveen K Malik, Rajesh Singh, Gehlot Anita, and Sudeep Tanwar. Adoption of blockchain technology in various realms: Opportunities and challenges. *Security and Privacy*, 3(5):e109, 2020.
- [4] Hussain Aldawood and Geoffrey Skinner. Educating and raising awareness on cyber security social engineering: A literature review. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, pages 62–68. IEEE, 2018.
- [5] Omar Ali, Ashraf Jaradat, Atik Kulakli, and Ahmed Abuhalmeh. A comparative study: blockchain technology utilization benefits, challenges and functionalities. *IEEE Access*, 9:12730–12749, 2021.
- [6] Ayman Alkhalifah, Alex Ng, ASM Kayes, Jabed Chowdhury, Mamoun Alazab, and Paul Watters. A taxonomy of blockchain threats and vulnerabilities. *Preprints*, 2019.
- [7] AA Andryukhin. Phishing attacks and preventions in blockchain based projects. In *2019 International Conference on Engineering Technologies and Computer Science (EnT)*, pages 15–19. IEEE, 2019.
- [8] Andrea Arcuri and Gordon Fraser. Parameter tuning or default values? an empirical investigation in search-based software engineering. *Empirical Software Engineering*, 18(3):594–623, 2013.
- [9] Andrea Arcuri, Juan Pablo Galeotti, Bogdan Marculescu, and Man Zhang. Evomas-ter: A search-based system test generation tool. *Journal of Open Source Software*, 6(57):2153, 2021. doi: 10.21105/joss.02153. URL <https://doi.org/10.21105/joss.02153>.

- [10] Irina Astrakhantseva, Roman Astrakhantsev, and Alexey Los. Cryptocurrency fraud schemes analysis. In *SHS Web of Conferences*, volume 106. EDP Sciences, 2021.
- [11] Vaggelis Atlidakis, Patrice Godefroid, and Marina Polishchuk. Restler: Stateful rest api fuzzing. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 748–758. IEEE, 2019.
- [12] Marcella Atzori. Blockchain technology and decentralized governance: Is the state still necessary? Available at SSRN 2709713, 2015.
- [13] Parma Bains. Blockchain consensus mechanisms: A primer for supervisors. *FinTech Notes*, 2022(003), 2022.
- [14] David Berdik, Safa Otoum, Nikolas Schmidt, Dylan Porter, and Yaser Jararweh. A survey on blockchain for information systems management and security. *Information Processing & Management*, 58(1):102397, 2021.
- [15] Swarup Bhunia and Mark Tehranipoor. Chapter 13 - security and trust assessment, and design for security. In Swarup Bhunia and Mark Tehranipoor, editors, *Hardware Security*, pages 347–372. Morgan Kaufmann, 2019. ISBN 978-0-12-812477-2. doi: <https://doi.org/10.1016/B978-0-12-812477-2.00018-6>. URL <https://www.sciencedirect.com/science/article/pii/B9780128124772000186>.
- [16] George Bissas, Brian Levine, A. Ozisik, Gavin Andresen, and Amir Houmansadr. An analysis of attacks on blockchain consensus. 10 2016.
- [17] Bitcoindeveloper. Rpc api reference. URL <https://developer.bitcoin.org/reference/rpc/>. Accessed: 2022-04-07.
- [18] Pablo J Boczkowski, Eugenia Mitchelstein, and Facundo Suenzo. The smells, sights, and pleasures of ink on paper: the consumption of print newspapers during a period marked by their crisis. *Journalism Studies*, 21(5):565–581, 2020.
- [19] Marcel Boehme, Cristian Cadar, and Abhik Roychoudhury. Fuzzing: Challenges and reflections. *IEEE Softw.*, 38(3):79–86, 2021.
- [20] Pavel A Borisovsky and Anton V Eremeev. Comparing evolutionary algorithms to the (1+ 1)-ea. *Theoretical Computer Science*, 403(1):33–41, 2008.
- [21] Jan-Willem Hendrik Bullée, Lorena Montoya, Wolter Pieters, Marianne Junger, and Pieter Hartel. On the anatomy of social engineering attacks—a literature-based dissection of successful attacks. *Journal of investigative psychology and offender profiling*, 15(1):20–45, 2018.

-
- [22] Edmund K Burke and Yuri Bykov. The late acceptance hill-climbing heuristic. *European Journal of Operational Research*, 258(1):70–78, 2017.
- [23] Suzette Caleo. Research framing, victim blaming: Toward an empirical examination of victim precipitation and perpetrator predation paradigms. *Industrial and Organizational Psychology*, 11(1):134–137, 2018.
- [24] José Campos, Yan Ge, Gordon Fraser, Marcelo Eler, and Andrea Arcuri. An empirical evaluation of evolutionary algorithms for test suite generation. In Tim Menzies and Justyna Petke, editors, *Search Based Software Engineering*, pages 33–48, Cham, 2017. Springer International Publishing. ISBN 978-3-319-66299-2.
- [25] Mary Catlin, Kyle C Scherr, Christopher P Barlett, Erin Jacobs, and Christopher J Normile. Bounded blame: The effects of victim–perpetrator relationship and victimization history on judgments of sexual violence. *Journal of interpersonal violence*, 36(15-16):NP8800–NP8823, 2021.
- [26] Gertrude Chavez-Dreyfuss. Cryptocurrency crime surges, losses hit \$4.4 billion by end-september: Ciphertrace report. *Reuters*. URL <https://www.reuters.com/article/us-crypto-currencies-crime/cryptocurrency-crime-surges-losses-hit-4-4-billion-by-end-september-ciphertrace-report-idUSKBN1Y11WH>.
- [27] Yulia Cherdantseva and Jeremy Hilton. A reference model of information assurance & security. In *2013 International Conference on Availability, Reliability and Security*, pages 546–555. IEEE, 2013.
- [28] Vincent Chia, Pieter Hartel, Qingze Hum, Sebastian Ma, Georgios Piliouras, Daniël Reijnders, Mark Van Staalduinen, and Pawel Szalachowski. Rethinking blockchain security: Position paper. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1273–1280. IEEE, 2018.
- [29] Tinkle Chugh, Karthik Sindhya, Jussi Hakanen, and Kaisa Miettinen. A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. *Soft Computing*, 23(9):3137–3166, 2019.
- [30] Davide Corradini, Amedeo Zampieri, Michele Pasqua, and Mariano Ceccato. Empirical comparison of black-box test case generation tools for restful apis. In *2021 IEEE 21st International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pages 226–236, 2021. doi: 10.1109/SCAM52516.2021.00035.

- [31] Dan Craigen, Nadia Diakun-Thibault, and Randy Purse. Defining cybersecurity. *Technology Innovation Management Review*, 4(10), 2014.
- [32] Alexa Dodge. “try not to be embarrassed”: A sex positive analysis of nonconsensual pornography case law. *Feminist Legal Studies*, 29(1):23–41, 2021.
- [33] Stefan Droste, Thomas Jansen, and Ingo Wegener. On the analysis of the (1+ 1) evolutionary algorithm. *Theoretical Computer Science*, 276(1-2):51–81, 2002.
- [34] Martin Eberlein, Yannic Noller, Thomas Vogel, and Lars Grunske. Evolutionary grammar-based fuzzing. In Aldeida Aleti and Annibale Panichella, editors, *Search-Based Software Engineering*, pages 105–120, Cham, 2020. Springer International Publishing. ISBN 978-3-030-59762-7.
- [35] Hamza Ed-Douibi, Javier Luis Cánovas Izquierdo, and Jordi Cabot. Automatic generation of test cases for rest apis: A specification-based approach. In *2018 IEEE 22nd international enterprise distributed object computing conference (EDOC)*, pages 181–190. IEEE, 2018.
- [36] Osamuyimen Egbon and Chijoke Oscar Mgbame. Examining the accounts of oil spills crises in nigeria through sensegiving and defensive behaviours. *Accounting, Auditing & Accountability Journal*, 2020.
- [37] Sidi Boubacar ElMamy, Hichem Mrabet, Hassen Gharbi, Abderrazak Jemai, and Damien Trentesaux. A survey on the usage of blockchain technology for cyber-threats in the context of industry 4.0. *Sustainability*, 12(21):9179, 2020.
- [38] Yongsheng Fang and Jun Li. A review of tournament selection in genetic programming. In Zhihua Cai, Chengyu Hu, Zhuo Kang, and Yong Liu, editors, *Advances in Computation and Intelligence*, pages 181–192, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-16493-4.
- [39] Emmanuelle Ganne. *Can Blockchain revolutionize international trade?* World Trade Organization Geneva, 2018.
- [40] GitHub. open-rpc/examples, . URL <https://github.com/open-rpc/examples/blob/master/service-descriptions/petstore-expanded-openrpc.json>. Accessed: 2022-06-19.
- [41] GitHub. ethereum-json-rpc-specification, . URL <https://github.com/etclabs-core/ethereum-json-rpc-specification/blob/master/openrpc.json>. Accessed: 2022-08-01.

-
- [42] Tara Goddard, Kelcie Ralph, Calvin G Thigpen, and Evan Iacobucci. Does news coverage of traffic crashes affect perceived blame and preferred solutions? evidence from an experiment. *Transportation research interdisciplinary perspectives*, 3:100073, 2019.
- [43] Patrice Godefroid, Bo-Yuan Huang, and Marina Polishchuk. Intelligent rest api data fuzzing. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2020*, page 725–736, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370431. doi: 10.1145/3368089.3409719. URL <https://doi-org.tudelft.idm.oclc.org/10.1145/3368089.3409719>.
- [44] David Gourley, Brian Totty, Marjorie Sayer, Anshu Aggarwal, and Sailu Reddy. *HTTP: the definitive guide*. ” O’Reilly Media, Inc.”, 2002. URL https://books.google.nl/books?hl=nl&lr=&id=3EybAgAAQBAJ&oi=fnd&pg=PT4&dq=http+status+codes&ots=X61T_afW_o&sig=ca19b3ENLi50-tBka6-ICgYVktA&redir_esc=y#v=onepage&q=httpstatuscodes&f=false.
- [45] Constantin Gurdgiev and Adam Fleming. Informational efficiency and cybersecurity: Systemic threats to blockchain applications. In *Innovations in Social Finance*, pages 347–372. Springer, 2021.
- [46] Hack. Smart contract development. URL <https://hack.bg/dlt-blockchain-development-services/smart-contracts-development/>. Accessed: 2022-03-29.
- [47] Mohammad Hamdan. A dynamic polynomial mutation for evolutionary multi-objective optimization algorithms. *International Journal on Artificial Intelligence Tools*, 20(01):209–219, 2011.
- [48] Anders Hansen and David Machin. *Media and communication research methods*. Macmillan International Higher Education, 2018.
- [49] Huru Hasanova, Ui-jun Baek, Mu-gon Shin, Kyunghee Cho, and Myung-Sup Kim. A survey on blockchain cybersecurity vulnerabilities and possible countermeasures. *International Journal of Network Management*, 29(2):e2060, 2019.
- [50] Adrian Herrera, Hendra Gunadi, Liam Hayes, Shane Magrath, Felix Friedlander, Maggi Sebastian, Michael Norrish, and Antony L. Hosking. Corpus distillation for effective fuzzing: A comparative evaluation, 2019. URL <https://arxiv.org/abs/1905.13055>.
- [51] Jennifer Hoewe and Cynthia Peacock. The power of media in shaping political attitudes. *Current Opinion in Behavioral Sciences*, 34:19–24, 2020.

- [52] Steyn Huurman, Xiaoying Bai, and Thomas Hirtz. Generating api test data using deep reinforcement learning. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, pages 541–544, 2020.
- [53] Angur Mahmud Jarman. Hierarchical cluster analysis: Comparison of single linkage, complete linkage, average linkage and centroid linkage method. *Georgia Southern University*, 2020.
- [54] Sung Wook Ji and Junwon Lee. Cultural dimensions of online vs. offline media competition: an application of niche theory. *Journal of Media Economics*, pages 1–18, 2021.
- [55] Tushar Kansal, Suraj Bahuguna, Vishal Singh, and Tanupriya Choudhury. Customer segmentation using k-means clustering. In *2018 international conference on computational techniques, electronics and mechanical systems (CTEMS)*, pages 135–139. IEEE, 2018.
- [56] Stefan Karlsson, Adnan Čaušević, and Daniel Sundmark. Quickrest: Property-based test generation of openapi-described restful apis. In *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, pages 131–141. IEEE, 2020.
- [57] Sei-Hill Kim and Matthew W Telleen. Talking about school bullying: News framing of who is responsible for causing and fixing the problem. *Journalism & Mass Communication Quarterly*, 94(3):725–746, 2017.
- [58] Jelena Kleut and Norbert Šinković. “is it possible that people are so irresponsible?”: Tabloid news framing of the covid-19 pandemic in serbia. *Sociologija*, 62(4):503–523, 2020.
- [59] Kyungchan Ko, Chaehyeon Lee, Taeyeol Jeong, and James Won-Ki Hong. Design of rpc-based blockchain monitoring agent. In *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1090–1095. IEEE, 2018.
- [60] Pavneet Singh Kochhar, Ferdian Thung, and David Lo. Code coverage and test suite effectiveness: Empirical study with real bugs in large systems. 03 2015. doi: 10.1109/SANER.2015.7081877.
- [61] Elmarie Kritzinger and Sebastiaan H von Solms. Cyber security for home users: A new way of protection through awareness enforcement. *Computers & Security*, 29(8):840–847, 2010.

- [62] Douiri Lamiae, Abdelouahhab Jabri, Abdellah El Barkany, and A.-Moumen Darcherif. *Optimization of Fresh Food Distribution Route Using Genetic Algorithm with the Best Selection Technique*, pages 175–199. Springer Singapore, Singapore, 2021. ISBN 978-981-33-6710-4. doi: 10.1007/978-981-33-6710-4_8. URL https://doi.org/10.1007/978-981-33-6710-4_8.
- [63] Nuno Laranjeiro, João Agnelo, and Jorge Bernardino. A black box tool for robustness testing of rest services. *IEEE Access*, 9:24738–24754, 2021. doi: 10.1109/ACCESS.2021.3056505.
- [64] Chetna Laroiya, Deepika Saxena, and C Komalavalli. Applications of blockchain technology. In *Handbook of Research on Blockchain Technology*, pages 213–243. Elsevier, 2020.
- [65] Robert LaRose, Nora J Rifon, and Richard Enbody. Promoting personal responsibility for internet safety. *Communications of the ACM*, 51(3):71–76, 2008.
- [66] Aleksandr Lazarenko and Sergey Avdoshin. Financial risks of the blockchain industry: A survey of cyberattacks. In *Proceedings of the Future Technologies Conference*, pages 368–384. Springer, 2018.
- [67] XRP Ledger. Documentation. URL <https://xrpl.org/public-api-methods.html>. Accessed: 2022-06-19.
- [68] Jun Li, Bodong Zhao, and Chao Zhang. Fuzzing: a survey. *Cybersecurity*, 1(1):6, 2018.
- [69] Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen. A survey on the security of blockchain systems. *Future Generation Computer Systems*, 107:841–853, 2020.
- [70] Stuart Madnick. Blockchain isn’t as unbreakable as you think. *Available at SSRN 3542542*, 2019.
- [71] Maria Maleshkova, Carlos Pedrinaci, and John Domingue. Investigating web apis on the world wide web. In *2010 eighth ieee european conference on web services*, pages 107–114. IEEE, 2010.
- [72] Jims Marchang, Gregg Ibbotson, and Paul Wheway. Will blockchain technology become a reality in sensor networks? In *2019 Wireless Days (WD)*, pages 1–4. IEEE, 2019.
- [73] Moxie Marlinspike. My first impressions of web3, Jan 2022. URL <https://moxie.org/2022/01/07/web3-first-impressions.html>.

- [74] Alberto Martin-Lopez, Sergio Segura, and Antonio Ruiz-Cortés. Restest: Black-box constraint-based testing of restful web apis. In *International Conference on Service-Oriented Computing*, pages 459–475. Springer, 2020.
- [75] Charles McFarland, Tim Hux, Eric Wuehler, and Sean Campbel. Blockchain threat report. *McAfee*, june 2018. URL <https://www.mcafee.com/enterprise/en-us/assets/reports/rp-blockchain-security-risks.pdf>.
- [76] Thijs Metsch, Andy Edmonds, et al. Open cloud computing interface-restful http rendering. In *Open Grid Forum-OCCI Working group technical report*, 2011.
- [77] Seyedali Mirjalili. Genetic algorithm. In *Evolutionary algorithms and neural networks*, pages 43–55. Springer, 2019.
- [78] Andy Neumann, Nuno Laranjeiro, and Jorge Bernardino. An analysis of public rest web service apis. *IEEE Transactions on Services Computing*, 2018.
- [79] Alireza Nili, Mary Tate, and Alistair Barros. A critical analysis of inter-coder reliability methods in information systems research. 2017.
- [80] Michael Nofer, Peter Gomber, Oliver Hinz, and Dirk Schiereck. Blockchain. *Business & Information Systems Engineering*, 59(3):183–187, 2017.
- [81] Raoul Notté, ER Leukfeldt, and Marijke Malsch. Double, triple or quadruple hits? exploring the impact of cybercrime on victims in the netherlands. *International Review of Victimology*, page 02697580211010692, 2021.
- [82] OpenRPC. What is openrpc?, . URL <https://open-rpc.org/>. Accessed: 2022-06-19.
- [83] OpenRPC. Playground., . URL <https://playground.open-rpc.org/>. Accessed: 2022-06-19.
- [84] Peder Østbye. Who is causally responsible for a cryptocurrency? *Available at SSRN 3339537*, 2019.
- [85] Peder Østbye. Who is liable if a cryptocurrency protocol fails? *Available at SSRN 3423681*, 2019.
- [86] Jane O’Boyle and Queenie Jo-Yun Li. #metoo is different for college students: Media framing of campus sexual assault, its causes, and proposed solutions. *Newspaper Research Journal*, 40(4):431–450, 2019.
- [87] Om Pal, Bashir Alam, Vinay Thakur, and Surendra Singh. Key management for blockchain technology. *ICT Express*, 2019.

-
- [88] Sakshi Patel, Shivani Sihmar, and Aman Jatain. A study of hierarchical clustering algorithms. In *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 537–541, 2015.
- [89] Barbara Piasecka, Marc Robinson-Rechavi, and Sven Bergmann. Correcting for the bias due to expression specificity improves the estimation of constrained evolution of expression between mouse and human. *Bioinformatics*, 28(14):1865–1872, 2012.
- [90] Corina S. PÄfsÄfreatu, Rody Kersten, Kasper Luckow, and Quoc-Sang Phan. Chapter six - symbolic execution and recent applications to worst-case execution, load testing, and security analysis. volume 113 of *Advances in Computers*, pages 289–314. Elsevier, 2019. doi: <https://doi.org/10.1016/bs.adcom.2018.10.004>. URL <https://www.sciencedirect.com/science/article/pii/S0065245818300640>.
- [91] Amanda My Linh Quan, Lindsay A Wilson, Salima S Mithani, David T Zhu, A Bota, and Kumanan Wilson. Reporting on the opioid crisis (2000–2018): role of the globe and mail, a canadian english-language newspaper in influencing public opinion. *Harm Reduction Journal*, 17(1):1–11, 2020.
- [92] Rayne Reid and Johan Van Niekerk. From information security to cyber security cultures. In *2014 Information Security for South Africa*, pages 1–7. IEEE, 2014.
- [93] Karen Renaud, Stephen Flowerday, Merrill Warkentin, Paul Cockshott, and Craig Orgeron. Is the responsabilization of the cyber security risk reasonable and judicious? *computers & security*, 78:198–211, 2018.
- [94] Leonard Richardson and Sam Ruby. *RESTful web services*. ” O’Reilly Media, Inc.”, 2008.
- [95] Muhammad Saad, Jeffrey Spaulding, Laurent Njilla, Charles Kamhoua, Sachin Shetty, Dae Hun Nyang, and David Mohaisen. Exploring the attack surface of blockchain: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 2020.
- [96] Gary J. Saavedra, Kathryn N. Rodhouse, Daniel M. Dunlavy, and W. Philip Kegelmeyer. A review of machine learning applications in fuzzing. *CoRR*, abs/1906.11133, 2019. URL <http://arxiv.org/abs/1906.11133>.
- [97] Omur Sahin and Bahriye Akay. A discrete dynamic artificial bee colony with hyper-scout for restful web service api test suite generation. *Applied Soft Computing*, 104:107246, 2021. ISSN 1568-4946. doi: <https://doi.org/10.1016/j.asoc.2021.107246>. URL <https://www.sciencedirect.com/science/article/pii/S1568494621001691>.

- [98] Abubakar Sadiq Sani, Dong Yuan, Jiong Jin, Longxiang Gao, Shui Yu, and Zhao Yang Dong. Cyber security framework for internet of things-based energy internet. *Future Generation Computer Systems*, 93:849–859, 2019.
- [99] K Sasirekha and P Baby. Agglomerative hierarchical clustering algorithm-a. *International Journal of Scientific and Research Publications*, 83(3):83, 2013.
- [100] Sarwar Sayeed, Hector Marco-Gisbert, and Tom Caira. Smart contract: Attacks and protections. *IEEE Access*, 8:24416–24427, 2020.
- [101] Daniel Schatz, Rabih Bashroush, and Julie Wall. Towards a more representative definition of cyber security. *Journal of Digital Forensics, Security and Law*, 12(2): 53–74, 2017.
- [102] Manuel Schlegel, Liudmila Zavolokina, and Gerhard Schwabe. Blockchain technologies from the consumers’ perspective: what is there and why should who care? In *Proceedings of the 51st Hawaii international conference on system sciences*, 2018.
- [103] Bruce Schneier. Blockchain and trust, Feb 2019. URL https://www.schneier.com/blog/archives/2019/02/blockchain_and_.html.
- [104] Bruce Schneier. On the dangers of cryptocurrencies and the uselessness of blockchain, Jun 2022. URL https://www.schneier.com/blog/archives/2019/02/blockchain_and_.html.
- [105] Julia Churchill Schoellkopf. Victim-blaming: A new term for an old trend. 2012.
- [106] Thibault Schrepel. *Blockchain+ Antitrust: The Decentralization Formula*. Edward Elgar Publishing, 2021.
- [107] Bart Selman and Carla P Gomes. Hill-climbing search. *Encyclopedia of cognitive science*, 81:82, 2006.
- [108] Abhinav Sharma, M Revathi, et al. Automated api testing. In *2018 3rd International Conference on Inventive Computation Technologies (ICICT)*, pages 788–791. IEEE, 2018.
- [109] Ashlie J Siefkes-Andrew and Cassandra Alexopoulos. Framing blame in sexual assault: An analysis of attribution in news stories about sexual assault on college campuses. *Violence against women*, 25(6):743–762, 2019.
- [110] Hari Krishna SM and Rinki Sharma. Survey on application programming interfaces in software defined networks and network function virtualization. *Global Transitions Proceedings*, 2021.

-
- [111] IP Specialist. How blockchain technology works. URL <https://medium.com/@ip-specialist/how-blockchain-technology-works-e6109c033034>. Accessed: 2022-03-29.
- [112] Bradley J Strawser and Donald J Joy Jr. Cyber security and user responsibility: surprising normative differences. *Procedia Manufacturing*, 3:1101–1108, 2015.
- [113] Swagger. Openapi specification. URL <https://swagger.io/specification/v2/>. Accessed: 2022-06-05.
- [114] Chainalysis Team. The rise of cybercrime on ethereum. URL <https://blog.chainalysis.com/reports/the-rise-of-cybercrime-on-ethereum>. Accessed: 2021-01-23.
- [115] Eric K. Tokuda, Cesar H. Comin, and Luciano da F. Costa. Revisiting agglomerative clustering. *Physica A: Statistical Mechanics and its Applications*, 585:126433, 2022. ISSN 0378-4371. doi: <https://doi.org/10.1016/j.physa.2021.126433>. URL <https://www.sciencedirect.com/science/article/pii/S0378437121007068>.
- [116] TripleA. Cryptocurrency across the world. URL <https://triple-a.io/cryptocurrency-ownership/>. Accessed: 2022-02-26.
- [117] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.*, 45(3), jul 2013. ISSN 0360-0300. doi: 10.1145/2480741.2480752. URL <https://doi-org.tudelft.idm.oclc.org/10.1145/2480741.2480752>.
- [118] Spandan Veggalam, Sanjay Rawat, Istvan Haller, and Herbert Bos. Ifuzzer: An evolutionary interpreter fuzzer using genetic programming. In *European Symposium on Research in Computer Security*, pages 581–601. Springer, 2016.
- [119] Emanuele Viglianisi, Michael Dallago, and Mariano Ceccato. Resttestgen: automated black-box testing of restful apis. In *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, pages 142–152. IEEE, 2020.
- [120] Rahul Rao Vokerla, Bharanidharan Shanmugam, Sami Azam, Asif Karim, Friso De Boer, Mirjam Jonkman, and Fahad Faisal. An overview of blockchain applications and attacks. In *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, pages 1–6. IEEE, 2019.
- [121] Basie von Solms and Rossouw von Solms. Cybersecurity and information security—what goes where? *Information & Computer Security*, 2018.
- [122] Rossouw Von Solms and Basie Von Solms. From policies to culture. *Computers & security*, 23(4):275–279, 2004.

- [123] Tanja EJ Vos, Felix F Lindlar, Benjamin Wilmes, Andreas Windisch, Arthur I Baars, Peter M Kruse, Hamilton Gross, and Joachim Wegener. Evolutionary functional black-box testing in an industrial setting. *Software Quality Journal*, 21(2):259–288, 2013.
- [124] Michael D Vose. *The simple genetic algorithm: foundations and theory*. MIT press, 1999.
- [125] Debbie Walkowski. What is the cia triad? URL <https://www.f5.com/labs/articles/education/what-is-the-cia-triad>. Accessed: 2022-03-29.
- [126] M. Wallace, G. Akrivas, and G. Stamou. Automatic thematic categorization of documents using a fuzzy taxonomy and fuzzy hierarchical clustering. In *The 12th IEEE International Conference on Fuzzy Systems, 2003. FUZZ '03.*, volume 2, pages 1446–1451 vol.2, 2003. doi: 10.1109/FUZZ.2003.1206644.
- [127] Stephanie Wang, Benjamin Hindman, and Ion Stoica. In reference to rpc: It’s time to add distributed memory. *Workshop on Hot Topics in Operating Systems (HotOS '21)*, 2021.
- [128] Kristin Weber, Andreas E Schütz, Tobias Fertig, and Nicholas H Müller. Exploiting the human factor: Social engineering attacks on cryptocurrency users. In *International Conference on Human-Computer Interaction*, pages 650–668. Springer, 2020.
- [129] Jiajing Wu, Qi Yuan, Dan Lin, Wei You, Weili Chen, Chuan Chen, and Zibin Zheng. Who are the phishers? phishing scam detection on ethereum via network embedding. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.
- [130] Jennifer J Xu. Are blockchains immune to all malicious attacks? *Financial Innovation*, 2(1):1–9, 2016.
- [131] Xiuyun Yang. An assessment of the media’s portrayal of murders at chinese mines. *The Extractive Industries and Society*, 7(3):1066–1076, 2020.
- [132] Xiuyun Yang and Bo Wang. Framing and blaming: Media coverage of coal mining accident coverups in china. *The Extractive Industries and Society*, 8(2):100895, 2021.
- [133] Li Yujian and Liu Bo. A normalized levenshtein distance metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1091–1095, 2007. doi: 10.1109/TPAMI.2007.1078.
- [134] Andreas Zeller, Rahul Gopinath, Marcel Böhme, Gordon Fraser, and Christian Holler. *The fuzzing book*, 2019.

- [135] Dirk A Zetsche, Ross P Buckley, and Douglas W Arner. The distributed liability of distributed ledgers: Legal risks of blockchain. *U. Ill. L. Rev.*, page 1361, 2018.
- [136] Man Zhang and Andrea Arcuri. Open problems in fuzzing restful apis: A comparison of tools, 2022.
- [137] Peilin Zheng, Zibin Zheng, Xiapu Luo, Xiangping Chen, and Xuanzhe Liu. A detailed and real-time performance monitoring framework for blockchain systems. In *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, pages 134–143. IEEE, 2018.
- [138] Xiaogang Zhu, Sheng Wen, Seyit Camtepe, and Yang Xiang. Fuzzing: A survey for roadmap. *ACM Comput. Surv.*, jan 2022. ISSN 0360-0300. doi: 10.1145/3512345. URL <https://doi-org.tudelft.idm.oclc.org/10.1145/3512345>. Just Accepted.



Standard configuration of parameters

<i>General</i>		<i>Fuzzing</i>	
Population size	100	Clustering after X generations (div-based)	3
Requests limit	1	Proportion mutated	0.8
<i>Request generation</i>		Add new individual to population	0.0
Probability to generate HTTP method GET (otherwise it is POST)	0.1	Sample from archive	0.0
Probability to include optional parameter during generation	0.25		
<i>Mutation</i>		ArrayGene: remove element	0.1
Mutations per individual	2	ArrayGene: add element	0.1
Probability to mutate HTTP method	0.05	ObjectGene: remove child	0.1
Probability to mutate API operation	0.005	ObjectGene: add non-required child	0.1
Probability to add request	0.05	LongGene: boundary case (upper or lower)	0.1
Probability to delete request	0.05	LongGene: no outside boundary cases	False
Schema-based generation instead of mutation	0.1	StringGene: fraction string to mutate	0.2
Probability to change parameter type	0.5	StringGene: pick another enum value	0.2
Use other schemas for new types	True		

Table A.1: Default configuration for GEFRA



Coding protocol

The coding schedule is filled in by answering for each article the following questions:

1. **Is the article relevant for this analysis?**

The article should not be a(n) (exact) duplicate. If the article refers to another article, this other article is the source. If this already exists in the list, the article at hand is deemed a duplicate.

An article is only relevant if it discusses (without a doubt) a blockchain attack.

If the article is irrelevant, the reason why is written down in the *Notes* column and is one of the following:

- **Inaccessible:** The article is not or no longer available (e.g. the article is placed behind a paywall or the link is invalid).
- **Duplicate1:** A direct link to the (same) article in another newspaper is mentioned in the article.
- **Duplicate2:** The same article is published in different newspapers but there are no references. However, the newspapers do have a connection or share the same platform/website setup (e.g. Tubantia and Stentor have the same parent company).
- **No blockchain attack:** The article describes some sort of attack or crime but this is not actually an attack that involves blockchain technology to some extent or the target of the attack is unrelated to the blockchain technology.
- **Unclear:** The article does not make clear (enough) whether it discusses an attack on a blockchain application or on something different in order to steal a blockchain asset (e.g. cryptocurrencies).
- **Unrelated:** The article is either not a news article (e.g. a court case document or comment on a forum), or it is not about blockchain (attacks), or it falls outside of the scope of this analysis (e.g. the article discusses cryptocurrency payments made as a consequence of a ransomware attack, how to invest in cryptocurrencies, or other topics that are not about blockchain-related attacks).

Only for relevant articles the rest of the questions are answered.

2. **What is (mostly) the narrative of the article?**

An article is either mostly episodic or thematic. In this column, what is applicable is written down:

- a) **Episodic:** Does the article focus its narrative on one person's experience? (i.e. an isolated incident/individual, anecdotal events, personal stories, dissociated from larger social contexts)
Briefly put: the article discusses one particular incident or issue presented in a personal case/specific event.
- b) or **Thematic:** Does the article focus on a more general and abstract level about the issue as a whole? (i.e. a broad portrayal and presentation of issues through information about their systemic causes, trends and consequences)
Briefly put: the article discusses a broader issue of which the attack was a part, and possibly society's role, this requires research and data collection.
A few sentences stating that hacks happen often (with very few examples illustrating such hacks) do not count.

3. **What blockchain attack(s) is/are discussed in the article?** A relevant article can discuss one or more of the following attacks, which are written down in the *Type of attack* column:

- a) **OPSEC-related incidents:** The article discusses an incident that compromises an organisation or individual's control of information and access to (business-critical) assets (i.e. trust, key, and information management issues that are no different for a cryptocurrency exchange than for a bank) (e.g. identity theft, sim-swapping).
- b) **Smart contract incidents:** The article discusses an incident resulting from improperly written smart contracts deployed and executed on a blockchain (e.g. exploits in smart contracts).
- c) **Consensus Protocol Incentives incidents:** The article discusses an incident arising from malicious exploitation of consensus protocols (e.g. 51% attacks, or block withholding attacks against the entire network or the incentive structure within pools).
- d) **Scams:** The article discusses an incident where the blockchain hype is used as a means to fool people and lure them in (e.g. pyramid/Ponzi schemes or investment fraud).

-
- e) **Unspecified:** The article is not specific enough to categorise the attack (e.g. the article discusses a 'hack' or server break-in and does not go into detail).

4. **Who is described as the victim(s) of the attack** A relevant article might describe a party or parties that fell victim to the attack or has lost something. If an item is described as a victim (e.g. a cryptocurrency that plummeted), this does not count since it is not a person or an entity represented by people.

This party can be one of the following and is written in the *Victim* column of the coding schedule:

- a) **Blockchain** application
This includes a cryptoplatforms, cryptoexchanges, DeFi (Decentralised Finance) applications, or any application that integrates the blockchain technology (and their developers).
- b) (End)users of blockchain applications (e.g. wallet hacks)
- c) **Investor(s):** The article discusses people that were looking to invest as the victims (that are not necessarily using an application).
This category will likely be merged with Users since it comes down to the same thing.
- d) **Celebrities** (including politicians)
- e) External **organizations** (e.g. banks, companies, startups, municipalities)
- f) **Unspecified:** The article does not speak of a victim.

5. **Who is described as the perpetrator(s) of the attack?** A relevant article might describe the party or parties that performed or planned the attack. This party can be one of the following and is written in the *Perpetrator* column of the coding schedule:

- a) **Attacker(s)/Hacker(s)/Criminal(s):** The article talks about attackers, hackers or criminals that performed the attack.
- b) **Employee(s):** The article talks about employee(s) (possibly) being involved in the attack.
- c) **Country:** The article mentions that a country is behind the attack (e.g. military hackers from North-Korea).
- d) **Unspecified:** The article does not speak of an adversary.

6. **Who has causal responsibility (blaming someone)?**

A relevant article might describe a party or parties that have causal responsibility. This can be one or more of the following parties and corresponding categories of causal responsibility. If the answer to one or more of the next questions is yes, than

that party has causal responsibility.

The party is written down in the *Causal responsibility* column of the coding schedule, the corresponding category in the *Category* column and the quote from the article that lead to this decision in the *Quote* column.

a) **Blockchain** application

This includes cryptoplatforms, cryptoexchanges, DeFi (Decentralised Finance) applications, or any application that integrates the blockchain technology (and their developers).

- **Lack of security:** Does the article indicate in some way that the attacked application is failing to offer secure storage of blockchain assets (e.g. information or cryptocurrencies)?
Or does the article indicate in some way that the attack was possible due to a bug/vulnerability/problem/something wrong in the application?
Or is it mentioned that security depends on the application used/that there is a lack of security overall?
- **Faulty behavior:** Does the article indicate in some way that the attack happened or escalated due to decisions (not) made by the application (that could have stopped or mitigated the consequences of the attack)?
- **Blockchain characteristics:** Does the article indicate in some way that the attack was possible because of vulnerabilities inherent to the way blockchain works (e.g. certain consensus algorithms making the network vulnerable for certain attacks)?
- **Lack of communication:** Does the article indicate in some way that the attacked application is providing insufficient communication to users to help them (e.g. poor customer service, not responding to users).
- **Newness:** Does the article indicate in some way that due to the newness of blockchain applications, at times faults happen that are exploited by adversaries?
- **Attractiveness:** Does the article indicate in some way (quotes from sources used in the article are allowed) that it (could be or) is lucrative for criminals to attack blockchain projects (for example due to the fact that transactions cannot be rolled back or the anonymity that blockchain offers)?

b) **Third party** / other platform than the blockchain application that was involved (in some way)

- **Negligence:** Does the article mention in some way that a third party was or might be negligent in trying to prevent the attack or was expected to prevent it, or that it even helped enable the attackers? For example by not removing fraudulent ads or even making a profit off of them.

Or does the article mention that a third party did something (for any reason) (e.g. releasing a security measure enabling the attack) that enabled the attack?

- **Sloppiness:** Does the article indicate in some way that attacks are possible due to weaknesses in (the security of) other (third party) platforms that are to some extent related to or integrated in the blockchain project?

c) **(End)users**

- **Bad behavior:** Does the article indicate in some way that the user(s) did not sufficiently keep their private information safe/that they should have taken more security precautions?
- **Lack of awareness:** Does the article indicate in some way that the user(s) did not do (enough) background research/blindly trusted the perpetrator(s)/put their trust in something? Or does the article indicate in some way that the user(s) were stupid to trust the scheme or consider the application/environment to be safe?

d) **Government** (institutional)

- **Insufficient regulation:** Does the article indicate in some way that there is a lack of regulation/legislation, education or security regarding the attack?
- **Poor regulatory enforcement/lack of police support:** Does the article indicate in some way that the government is unable to enforce rules on blockchain attacks?
- **Lack of supervision on use of technology:** Does the article indicate in some way that there is a lack of supervision on the use of technology (e.g. the attack was possible because users (when using the application) are not sufficiently supervised)?

e) **Society** as a whole (insitutional)

- **Good reputation:** Does the article indicate in some way that blockchain applications are considered to always be secure?
- **Lack of awareness:** Does the article indicate that society does not consider blockchain attacks a priority issue or serious crime (or indicate that blockchain attacks are not a problem)?
- **Society ignorance:** Does the article indicate that society does not demand accountability from anyone?
- **Media responsibility:** Does the article indicate that there is a victim blaming and inaccurate (media) coverage concerning blockchain attacks?

f) **Perpetrator**

- **Ethical hacking:** Does the article indicate in some way that the perpetrator(s) possibly wanted to help the application by exposing vulnerabilities or teach them a lesson?
- **Fun:** Does the article indicate in some way that the perpetrator(s) attacked the application for fun or as a challenge?
- **Political:** Does the article indicate in some way that the attacker(s) performed the attack for political reasons?
- **Social status/show-off:** Does the article indicate in some way that the perpetrator(s) attacked the application to show off their skills?

g) **Undefined** (when there is no causal responsibility assigned)

7. Who has treatment responsibility (demanding a solution)?

a) **Blockchain** application

This includes a cryptoplatforms, cryptoexchanges, DeFi (Decentralised Finance) applications, or any application that integrates the blockchain technology (and their developers).

- **Protection enhancement:** Does the article indicate in some way that the application should/must or has to implement measures to prevent vulnerabilities in the application/improve the security (for example in order to be able to continue business as usual) (e.g. introduce a bug bounty program)? Or is it mentioned that in-place security measures were able to (at least partly) effectively block the attack?
- **Policy change:** Does the article indicate in some way that the application needs to change their user policies in order to be more secure/prevent future incidents?
- **Punishments:** Does the article indicate in some way that applications need to punish offenders (e.g find them and freeze their assets) in order to continue to exist/continue business as usual?
- **Better communication to users:** Does the article indicate in some way that communication/customer support should be improved (in order to continue business as usual)?
- **Compensation:** Does the article indicate that the application or service has compensated or will compensate victims to some extent?

b) **Third party**

- **Basic security:** Does the article indicate that an external party can prevent the attacks by having security measures in place? For example, is it stated that companies should be able to counter malware by having spam filters and up to date systems, or by training customer service?

-
- **Voluntary compensation:** Does the article indicate that an external party (e.g. a bank or other company) has compensated or will (or should) compensate victims to some extent?
- c) (End)users (if answer is yes to one of the following questions)
- **Better behavior:** Does the article indicate in some way that the user(s) should better take care of/keep safe their private information when using the application? Or that user(s) could do something to be more safe (e.g. store money in a cold wallet, use safer alternatives, make back up of wallet and keys).
 - **Self-protection enhancement:** Does the article indicate in some way that user(s) should watch out/be cautious of scams when using the application/do some investigation before using an app (e.g. only use applications that have official permits, investigate applications)?
Or that user(s) should not share personal information (e.g. on social media platforms)?
Quotes from another source count as well (if there are no counterarguments).
 - **Awareness enhancement:** Does the article indicate in some way that victims (and witnesses) should report (similar) crimes?
 - **Acceptance:** Does the article indicate in some way that users have to be aware of and accept the risk that their account could be hacked and their funds could be lost? Or less explicit: that there will always be risk in the blockchain community (e.g. for cryptocurrencies)?
Or is it somehow indicated that attacks are normal and part of the DeFi (Decentralized Finance) business?
Or that it is inherent to the blockchain technology that there is no authority that can fix things?
- d) **Government** (institutional)
- **Regulation:** Does the article indicate in some way that the government or EU should create (or improve) legislation/regulation for blockchain applications or related activities (like cryptocurrencies trading)?
 - **Increase police support:** Does the article indicate in some way that more police force should be dedicated towards blockchain attacks and tracking down attackers?
 - **Oversight of technology use:** Does the article indicate in some way that users should be monitored and have restrictions (to some extent) when it comes to the usage of blockchain applications?
- e) **Society** (institutional)

B. CODING PROTOCOL

- **Education:** Does the article indicate in some way that education from schools/(college)universities on blockchain risks is an option to prevent attacks?
 - Adequate education and **support:** Does the article indicate in some way that society needs to provide (financial) support for victims?
 - **Media responsibility:** Does the article state that there is a need for better reporting, accurate coverage and neutral coverage on blockchain applications or attacks?
- f) **Undefined** (when there is no treatment responsibility assigned)

C

Additional figures content analysis

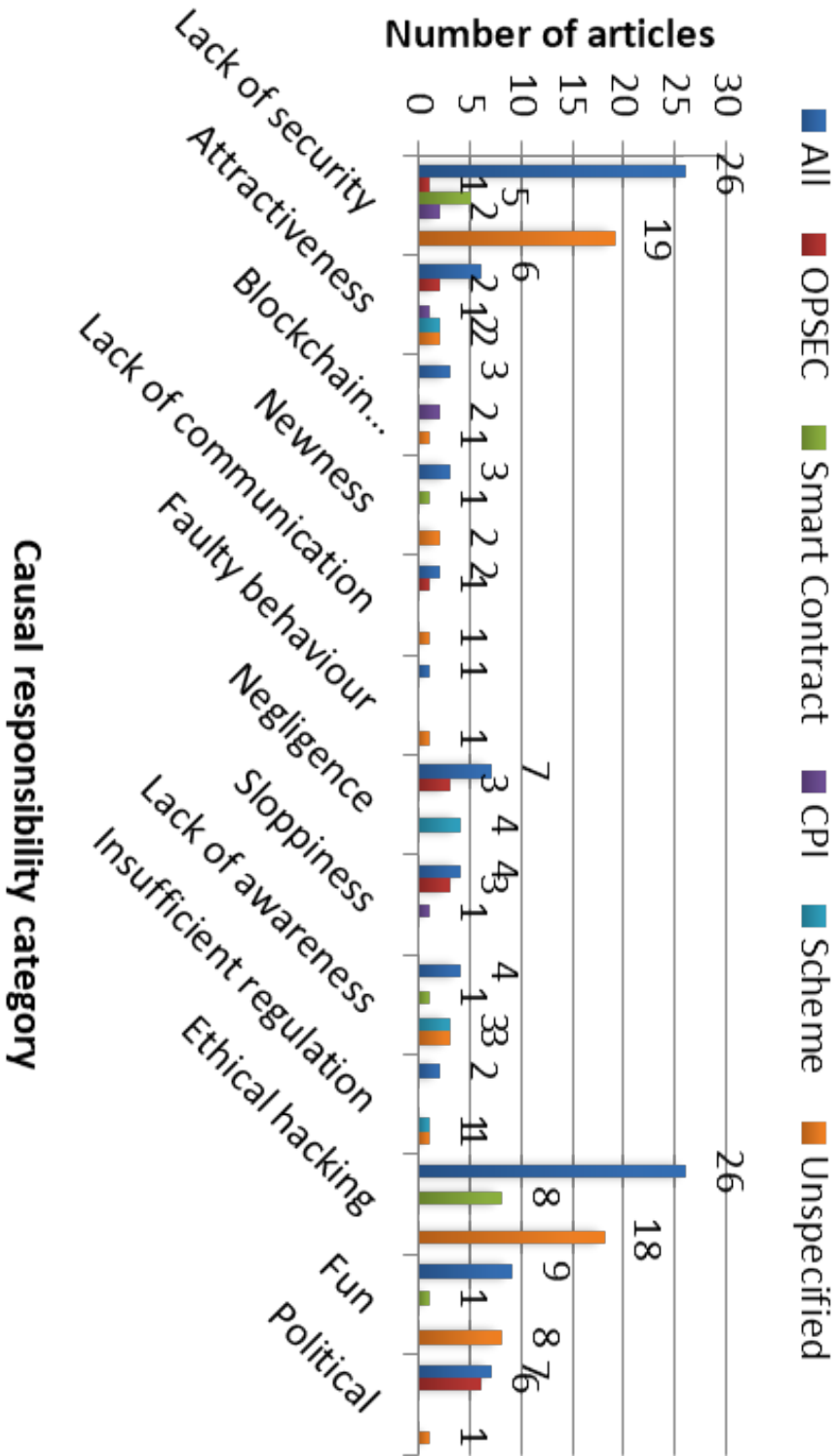


Figure C.1: Categories of causal responsibility per blockchain attack type

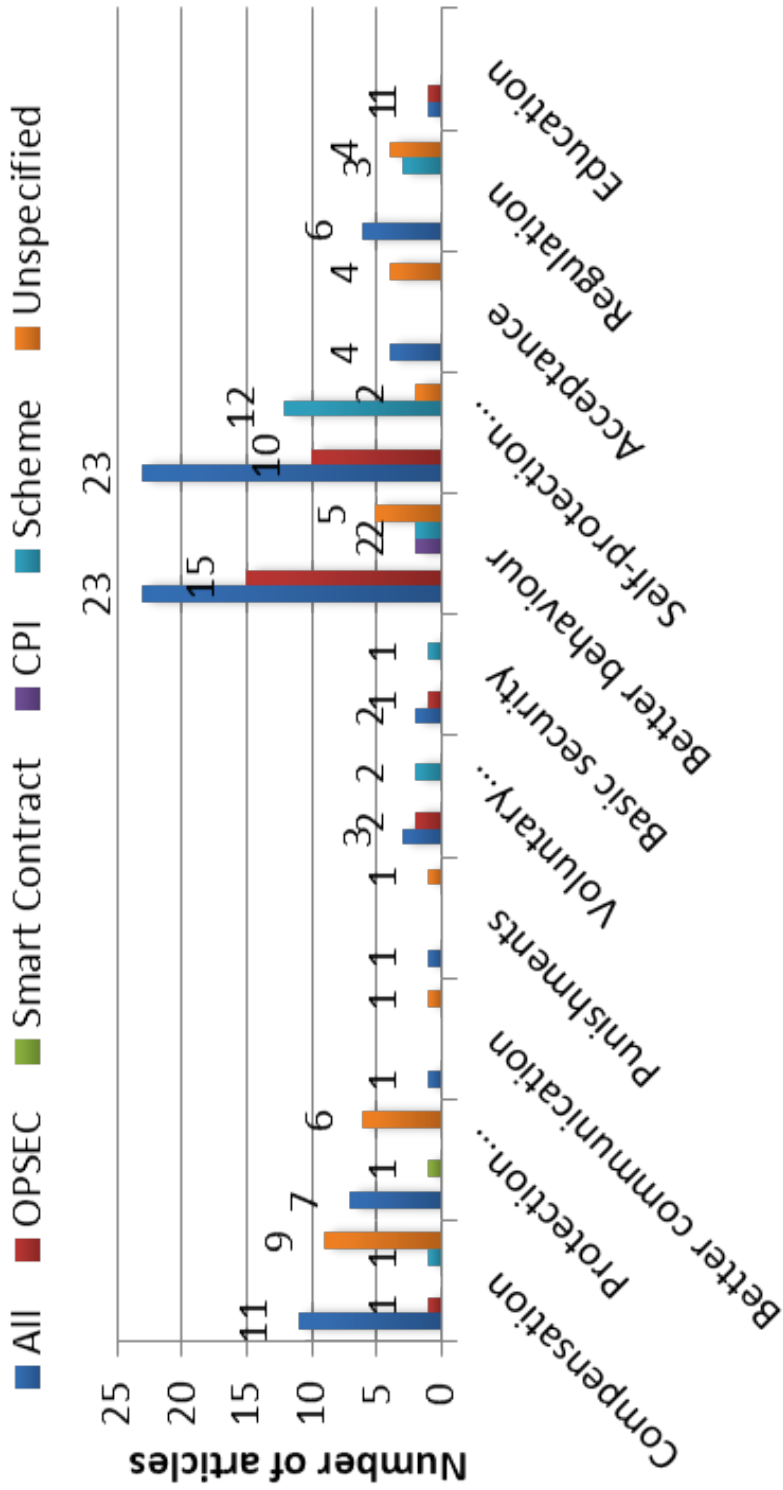


Figure C.2: Categories of treatment responsibility per blockchain attack type



Legal professionals interview protocol and summaries

D.1 Interview protocol

The interviews were structured as an open discussion. Various cases were described to the legal professionals and they were asked to give their opinion regarding who was legally responsible for the prevention of the attack and who was likely to suffer the consequences for each case. Not all legal professionals were questioned on all five cases.

Case 1: Poly Network (August 2021) (Smart contract incident)

What happened? A hacker stole more than \$600 million through the cryptocurrency exchange Poly Network. Poly Network operates on various blockchain networks. Cryptocurrency tokens are swapped between the blockchains using smart contracts that contain instructions on when to release the assets to the counterparties. The hacker exploited a vulnerability in one of these smart contracts and overrode the contract instructions to steal cryptocurrency funds from Poly Network users.

In this case, the hacker returned all of the assets soon after, stating to be an ethical hacker wanting to expose vulnerabilities so the developers could fix them.

What would however happened if the hacker had instead kept the assets? **Who was legally responsible for the prevention of the attack, and who is likely to have to suffer the consequences and why?**

Case 2: Bitcoin SV (August 2021) (Consensus protocol incentives incident)

What happened? Bitcoin SV became the victim of a 51% attack. In a 51% attack, one party holds the majority of the hashrate in the blockchain network, and the network is no longer truly independent. During the attack, the attackers were able to double spend (repeat transactions) and in this way steal money from users.

This attack exploits an inherent vulnerability of blockchain but is mostly a risk for small

blockchain networks. For large blockchain networks, it becomes infeasible to obtain the majority of mining nodes. In the case of Bitcoin SV, **who was legally responsible for the prevention of the attack, and who is likely to have to suffer the consequences and why?**

Case 3: Fake chrome extensions for cryptowallets (2020) (OPSEC-related incident)

What happened? Fake chrome extensions that masqueraded as legit crypto-wallet extensions were available in the Google Play Store. The extensions worked the same as the legit wallet applications. However, while the extensions enabled users to access their wallet, login details were siphoned off to the attackers, allowing them to steal the users' cryptocurrencies.

Google allowed the fake extensions to be in the Google Play Store. People that downloaded the extensions from the Google Play Store were robbed of their credentials and money. **Who was legally responsible for the prevention of the attack, and who is likely to suffer the consequences and why?**

Case 4: The BitConnect Ponzi scheme (January 2018) (Scheme)

What happened? BitConnect was a cryptocurrency that was connected with an investment program of the same name. BitConnect collected money from people by selling itself as a way for people to easily invest in cryptocurrency and to make a lot of money doing so. BitConnect then used money from new investors to pay old ones (otherwise known as a Ponzi scheme), and its advocates built pyramid schemes where they would get a cut for convincing others to invest by promising them the same deal. Some advocates used their YouTube platform to promote BitConnect. Eventually BitConnect insiders closed the exchange platform and soon after the BitConnect tokens lost their value, leaving the investors with nothing. **Who was legally responsible, and who is likely to suffer the consequences and why?**

Case 5: Fake investment advertisements featuring celebrities (June 2019) (Scheme)

What happened? Advertisements were placed on Facebook featuring famous Dutch people's names and pictures calling on users to invest in cryptocurrencies. The advertisements stated for example that the celebrities became rich due to online investment programs. The celebrities were however not at all involved and unaware of their image being used in this way.

The investment programs promoted by the advertisements turned out to be fraudulent. With the money the victims supposedly invested in cryptocurrencies, cryptocurrencies were bought but sent to addresses that the victims did not have access to. Because the compa-

nies behind the advertisements and investment programs are located abroad, and victims voluntarily transferred money, the police are unable to act.

Facebook (and other social media companies) try to remove fraudulent advertisements as they are a breach of their regulations but adversaries seem to be able to keep bypassing the control mechanisms. **Who was legally responsible for the prevention of this attack, and who is likely to suffer the consequences and why?**

D.2 Legal professionals interview summaries

This section contains the summaries of interviews of the experts that gave (explicit) permission for publication.

D.2.1 Summary interview Dr. Oskar Josef Gstrein

Dr. Oskar Josef Gstrein

Dr. Oskar Josef Gstrein is an Assistant Professor at the University of Groningen. He has a background in (European Human Rights) law and philosophy and currently conducts research for the Governance and Innovation department at the University of Groningen. He is active in the research fields of International European Law and Multidisciplinary Approaches for Law. Among other things, Gstrein studies emerging technologies and governance (including blockchain), cybersecurity, and human rights in the digital age.

Case 1: Fake chrome extensions for cryptowallets (2020) (OPSEC)

Gstrein stated that Google has the obligation to check whether extensions that are available in their store are safe. However, safety and security have a context-heavy dimension. It depends on the criteria Google sets for themselves what it means for an application to be safe enough to be placed in the Google Play Store. Google would need to look at how applications or extensions share information and verify that it does not share information that it should not. Some responsibility may reside with Google in that sense.

If users look specifically for the application or extension in order to download it, they may have some share in responsibility. In any case, it is not the legal responsibility of users to report fraudulent extensions to Google or any other services.

Case 2: The BitConnect Ponzi scheme (January 2018) (Scheme)

Gstrein pointed out that this case is no different from other Ponzi schemes that did not involve blockchain technology. He mentioned the significance of a contract that may have been in place. If there was a contract, then the construction is a fraud when the promise (of high returns of investment) is not kept. The difficulty then lies in who is responsible for the creation of these contracts.

The blockchain structure of cryptocurrencies makes it easier to conceal traces of the money going into the scheme, making it a complex task to trace creators of the Ponzi scheme. It is more a question of operability than of responsibility. The responsibility (or liability) is in this case not enforceable, but it is in fact there. In practice this means that while the creators of the Ponzi scheme are liable, due to the responsibility not being enforceable (the fraudsters are long gone when the crime is noticed), victims cannot be compensated.

Case 3: Fake investment advertisements featuring celebrities (June 2019) (Scheme)

Gstrein said that the cryptocurrencies here are used as another layer of complexity to set up the scam. The extra layer makes the scam more plausible. Since everyone knows there is a lot of money in blockchain, cryptocurrencies are attractive to use as bait. The use of cryptocurrencies in addition to the image of celebrities in the scam here tricks people to give money to something they would normally never give money to.

In this case, the question is how well Facebook actually has to check the contents (of advertisements) with which they make money, and how trustworthy Facebook information needs to be.

The celebrities have the strongest legal position due to infringement of personality rights and economic and reputational damage to their image, but this is unrelated to blockchain. While blockchain is used as another layer of complexity, the scam itself is not different from other investment scams.

As for banks, when they are aware that there is a fraudulent transaction (money transfer to a scamming party), they should act. It is however difficult for them to detect whether a transaction is fraudulent. This falls within in the area of consumer protection or insurance against fraud.

Case 4: Phone call investment scam (2020) (Scheme)

Gstrein commented that this scheme is not new: only the blockchain context is, since the blockchain hype make attacks like these more successful. Furthermore, cryptocurrencies make it easier for attackers to access the money without being caught.

In this case, banks were not too much involved but they are a protective intermediary. It depends however on how much the bank knows, whether they should have intervened. This falls within the consumer protection territory. It is not desirable for a bank to intervene at the slightest possibility of fraud. This would mean that clients would often be asked to explain why they make a transaction when they attempt to do so. There is a balance between the impact of such scams on society overall and the possibility to transfer money quickly. Moreover, there are already some protective measures in place: it is generally not possible to transfer too much money at once. What 'too much' money is depends on the client but needs to be generalised to some extent to avoid discrimination.

It is difficult to retract general responsibility for banks unless there is a very clear indication that something fishy is going on. Because the scam hinges on cryptocurrency, it is even more difficult to detect fraud. A transaction could also be an investment in cryptocurrencies rather than a scam.

Final conclusions from the interview (my interpretation)

Gstrein states that social engineering schemes are not new when looked at in an abstract manner. They are simply updated versions with the addition of a layer of complexity: blockchain. Usually it is clear that a crime has been committed, and the question is not that of responsibility but of enforcement. The crime territory goes across borders and the blockchain technology allows criminals to carry out crimes with ease. While laws are in place and deemed sufficient, it is difficult to enforce them. The enforcement mechanisms have limitations. The way that rules are set up (nationally) is a restriction to enforce them, especially for blockchain applications, where there are no territorial borders. The game of the crime is being moved away from the space of the territory. It requires a large amount of public resources to follow blockchain data traces (and money) to the end point.

In essence, it is clear by law who is responsible (the attackers) but it is difficult to enforce this responsibility. Cryptocurrencies live in a wild west territory - it is relatively new and there exist many optimistic stories about people getting rich. People are quick to underestimate risk and law enforcement is complicated.

D.2.2 Summary interview Dr. Thibault Schrepel

Dr. Thibault Schrepel is an Associate Professor of Law at VU Amsterdam. His latest research is focused on blockchain antitrust and how law and technology could cooperate. Recently he published a freely available book on this topic called “Blockchain + Antitrust”. Part of the research Schrepel has conducted is about how blockchain can work for modern societies, and how public permissionless blockchains (when anyone can write on the blockchain ledger) make it difficult to enforce the law.

Case 1: Fake chrome extensions for cryptowallets (2020) (OPSEC)

There is currently a draft of new European legislation named the Digital Services Act¹ (DSA). Currently platforms active in the EU have the duty to do everything they can to protect customers (and thus detect evil contents). Legally speaking this means that platforms must employ all the means they can to do this, but there is no duty to obtain a certain result. This is comparable to the duty of doctors for the care and safety of patients. Doctors have the duty to do everything they can to care for patients but there is no duty to cure them. In the US there is absolutely no obligation for online platforms to check contents.

Legislation and liability should be aligned with the power of prevention. If a party is not in control, they should not be liable. Technically it is possible for online platforms to check all published contents. However, because there is too much information, it is infeasible to do so. With more powerful algorithms in the future this might be subject to change. At the moment, if it cannot be proven that a platform was aware of malicious contents, they are not liable.

In this case, Google is not liable if they did not know about the fraudulent application extensions; the creators are. If the creators are not found, the victims cannot retrieve their lost funds. For some incidents governments have designated funds available to help victims². A similar fund could be set up for blockchain incidents so victims can be compensated when perpetrators cannot be tracked down. Such a fund currently does not exist.

Case 2: Poly Network (August 2021) (Smart Contract)

Here the rule of power of command and control applies. If you exercise power, then you are liable. In this case PolyNetwork would likely be legally responsible. They have the power to design the platform and make decisions regarding the implementation. When a mistake in the code results in cryptocurrency being stolen, whoever has power over the platform is liable.

¹The Digital Services Act (DSA) clarifies the responsibilities of digital services like online platforms regarding activities and information for consumers. The DSA helps combat illegal contents online by clarifying the role of providers of digital services and creating procedures for carefully handling this kind of contents.

²In the Netherlands there is a fund called the Waarborgfonds Motorverkeer for victims involved in car accidents where the perpetrator is unknown or uninsured.

Case 3: Bitcoin SV (August 2021) (Consensus Protocol Incentives)

This is a complex case because there is not one single person to blame. Schrepel explains in his book the concept of a blockchain nucleus. This is a group of participants that come together to achieve a form of control over the blockchain by collaborating, by circumventing (some of) the (economic, political, logical, sociological, architectural and legal) constraints imposed on them, and by changing them in the long run. Schrepel argues that the nucleus should become a legal entity that can be assigned liability. There are three types of blockchain participants: founders/core developers (who implement the original rules of the blockchain: design the code and determine the consensus protocol to be used), users (who propose new transactions but cannot easily exercise coordinated control), and miners (who validate transactions assembled into blocks). Generally in decentralized permissionless blockchain applications (like Bitcoin or Ethereum) no one can unilaterally impose changes.

In his book (chapter 1) Schrepel illustrates that when a problem comes to the surface, one can see where the power lies. In just a few minutes, code changes were implemented to fix a vulnerability in Bitcoin ABC after core developers made a phone call to the main mining farm to get the patch approved. Schrepel argues in his book that a key weakness (but also a key strength) of decentralized systems is that tasks are not always clearly defined. It pushes members of the blockchain community to verify each other's work (e.g. a participant finding the vulnerability, core developers implementing a patch and getting it approved by the (largest) mining pool of workers, and participants pushing other miners to implement the patch).

The design of the blockchain application influences the possibility to double spend. In this case it is likely that a group of blockchain core developers would be liable since they implemented the code and are able to propose code changes. Big miner parties are able to force updates to be implemented (since they maintain a large portion of miners' computing power, they influence the voting process), and social influencers can push followers for updates to be adopted; thus more parties were able to influence the implementation. Participants are to be held liable for illegal conduct committed within the nucleus or perimeter that they can control or influence to a great degree. The width of that perimeter depends on the case. Lack of control should result in a lack of liability. Concrete and quantifiable frameworks are necessary to define the nucleus in each case to ensure legal certainty, limit legal errors and reduce regulatory costs.

Case 4: The BitConnect Ponzi scheme (January 2018) (Scheme)

This is criminal law. What matters most is the intention. If the people behind the Ponzi scheme intended to scheme the victims, they are liable. The same goes for the influencers. If they were aware of the scheme and still convinced people to join in, they are liable. Proof is however required (for example email contact that makes it clear that they were

in fact aware of the scheme). It depends on the country what the legal standard to prove this is. The victims are in a (non-legal) sense responsible for losing their money since they consented to invest money themselves.

Case 5: Fake investment advertisements featuring celebrities (June 2019) (Scheme)

There is a difference between hacking accounts or creating new accounts. If accounts are hacked, the social platform would be liable since they are responsible for network security. Technically it should be possible to prevent hacks if necessary. For new accounts however, the platform cannot be liable. At this time platforms might not be able to detect new accounts that imitate real people. This is however transforming and in the future it might be possible. According to the DSA social media platforms are not liable in such cases. Schreipel thinks that this might change since it should be possible to reliably detect the fraudulent use of personal images.

The party behind the account(s) is liable. If perpetrators are not found, users have to deal with possible damages. This issue (fraudulent investment schemes) is however not specific for blockchain applications.

Final conclusions from the interview (my interpretation)

Overall, whoever is in control, is liable. Third parties such as online platforms can be held liable. In the future this might happen more often, when technology is advanced enough to detect all fraudulent uses but currently this is understood as infeasible.

In the future a government fund may be constructed in Europe that compensates users if the perpetrator remains unidentified. To identify the wrongdoer is not always possible, especially when the blockchain is impacted from within (due to the anonymity blockchain offers). If it however is impacted from the outside (by making use of a centralized system - where identities are known - like a social media platform) the perpetrator can usually be identified. If the perpetrator can be identified, the victim(s) can be reimbursed. If not, and if platforms are not liable, victims have to suffer the losses.

D.2.3 Summary interview Prof. Bart Schermer

Prof. Bart Schermer is a Professor of Law and Digital Technology at the University of Leiden. He has a background in Law and Information Technology. His research focuses on privacy and cybercrime, specifically the relation between enforcement and human rights in the online world.

He is also a fellow at the E.M. Meijers Institute for Legal Studies and active as a partner at Considerati, a legal consultancy firm specialized in legal and policy advice for IT and new media. Furthermore, he is a member of the Cybercrime expert group for the Court of Appeal in The Hague and a member of the Human Rights Committee of the Advisory Council on International Affairs.

Case 1: Fake chrome extensions for cryptowallets (2020) (OPSEC)

In a general sense everyone bares their own damage until it is proven that someone else is to blame (civil and liability law). Whether Google is (partly) liable depends on the expectations Google has set for users. It would matter for example whether Google is presenting the Google Play Store as a secure (or even the only) environment where applications can be downloaded from. Schermer states that one would assume that Google excluded liability in their terms and conditions. This is however not watertight. It comes down to whether Google can be blamed for allowing an application in their store that was not well (enough) checked, resulting into harm done to users.

Liability might be a stretch as Google does not know what people do with an application extension or what exactly the extension is meant to do. It is difficult to determine whether Google would have a responsibility there. For Apple it might be different, since they have (and are known for) a clear vetting process for the App Store, which is the only place where applications can be downloaded. Eventually it comes down to the question to what extent Google can be blamed for not checking the application thoroughly.

Case 2: Poly Network (August 2021) (Smart Contract)

Looking at different contexts, such as software development, users are usually out of luck. For example, in the past there was a zero day vulnerability present in Microsoft software that caused many companies damage. None of those companies however held Microsoft accountable for the damage. Perhaps Microsoft handled the implementation of code carelessly, but this does not necessarily mean that Microsoft is liable for offering the product on the market. Up until now consequential damage is not recovered from the developing party. For a large organisation like Microsoft, generally customers have to agree to predefined terms and conditions. With this, companies can contractually exclude liability.

Usually blockchain applications are open-source. There is no clear point of contact. Developers of the blockchain application design it together or separately from each other,

without the presence of a legal entity or foundation. The individual developers (that together create and build the standards of the application) have nothing to do with any legal entity. If users makes use of the application, they themselves are responsible for the consequences.

In the Poly Network case there is the question of causality. Is PolyNetwork responsible for the damages that users suffer when using their product? It may be difficult to reach Poly Network because it is not clear who the point of contact is. If there is not a legal entity behind the platform, who can you talk to? In the world of cryptocurrency often there are open platforms, add-ons and smart contracts. In many settings these are build by volunteers or start-ups. Even if they are liable, there is not much to gather from them as they do not have the funds.

So are users out of luck in the blockchain context? We have to look at liability. First, one needs to verify whether a party is guilty of culpable behaviour. Next, the point of contact and the culpable behaviour need to be identified and clear. It depends on the expectations a cryptoplatfrom has set, and to what extent they are involved with the code implementation or have contractually excluded liability, whether they are liable.

One problem concerning enforcement is the international context of blockchain. What jurisdiction is applicable and what legal forum can be used? If you use an exchange that is based in a different country, it may not be possible to get your justice in your home country. Platforms have to conform to financial legislation in the countries where they are active. Crypto exchanges need to register at DNB and AFM (and thus conform to certain rules), but they are not under financial supervision.

It is likely that compensation from a crypto exchange that contained a vulnerability causing losses for users is not enforceable (as the platform is not liable). Cryptoplatfroms may however compensate losses to win back trust from users of their service.

Case 3: Bitcoin SV (August 2021) (Consensus Protocol Incentives)

Majority attacks are an inherent risk of blockchain applications. As a user you should be aware that such things can happen. When using a blockchain application you are essentially responsible (baring your own damage until it is proven that someone else is to blame). However, considering the case from a criminal law perspective, when someone performs an attack like this with the intent to scam or deceive others, then it may qualify as deceit or forgery, and they are liable. It is difficult to hold a blockchain network or platform liable due to the decentralization of blockchain applications. There will always be a risk that the blockchain system falls apart, and when that happens there is no one (no central authority) to recover damages from. The attacker(s) are liable, but it is challenging to identify them.

Dutch institutes AFM and DNB can impose legal obligations to financial institutions like crypto exchanges. They also inform the public in a general sense. Investing in cryptocurrencies is outside of the common sector of regulation. There is no regulated supervision for blockchain applications. Cryptoplatfroms active in the Netherlands have to register

at DNB, for which they need to comply with certain rules regarding money laundering and financing of terrorism but other than that there is little legislation (there is no supervision on consumer protection).

Schermer expects that there will be more legislation at some point. Blockchain is difficult to regulate due to the decentralisation (no central point of contact). Many consumers trade blockchain assets through cryptoplatforms however (which do have a central authority), so likely these platforms will be regulated similarly to banks. In some countries it is even illegal to trade cryptocurrencies (for example in China). While a total ban may not happen here, stricter regulations for cryptoplatforms are not unlikely to be implemented. This does not prevent individuals from setting up their own blockchain node, mining cryptocurrencies or setting up a wallet themselves, but most people trade cryptocurrencies using a cryptoplatform (which is easier and more secure for the average user).

Case 4: The BitConnect Ponzi scheme (January 2018) (Scheme)

Blockchain should not be viewed as something entirely different from other technologies. There is no separate law for blockchain but existing legislation (for financial supervision, liability law, criminal law) can be applied to technological applications. The people that set up the scheme are likely liable (due to intent of deceit and forgery) but it is likely impossible to find them to recover any damages.

For the influencers that promoted the scheme, it depends on the extent to which (it can be proven that) they were (or should have been) aware of the scam. Another thing is that it is against the law (in the Netherlands specifically the Wft: Wet financieel toezicht) to provide financial or investment advice without a permit for this. If the influencers were guilty of this, it could provide a basis for a stronger case against them.

Case 5: Fake investment advertisements featuring celebrities (June 2019) (Scheme)

In this case the reputation of Dutch celebrities are violated because of associations with the fake advertisements. Their portrait right and right to privacy are violated. In the article that Schermer wrote on horizontal privacy (titled *Het recht op privacy in horizontale verhoudingen*) he discusses the (European) Digital Services Act³, which entails a sort of duty of care⁴ for internet platforms. The details of the implementation are not clear yet and it may be problematic to determine a breach of this duty of care.

Specifically in the case of the Dutch celebrity John de Mol, who sued Facebook, Facebook was informed by de Mol and they did not delete the advertisements fast enough in

³The Digital Services Act (DSA) clarifies the responsibilities of digital services like online platforms regarding activities and information for consumers. The DSA helps combat illegal contents online by clarifying the role of providers of digital services and creating procedures for carefully handling this kind of contents.

⁴A duty of care is a legal obligation which is imposed on an individual or organisation, requiring adherence to a standard of reasonable care while performing any acts that could foreseeably harm others.

this opinion. According to the Electronic Commerce Directive⁵ Facebook is liable because they were made aware of the issue and had the ability to interfere. Generally social media platforms are not liable for content on their platform unless they are informed about it but platforms may be required by national courts to implement measures to prevent future violations. In this case, the judge has ruled that Facebook must proactively ensure that no new advertisements featuring de Mol will be placed.

This case is more about the Dutch celebrities and social media platforms than blockchain and the victims of the scam. Facebook can be held (partly) liable for the content (if they did or do not remove fraudulent ads), but one might wonder to which extent it is Facebook's fault when a user falls for the scam. People always have some responsibility themselves. The probability that a user can recover the damages from Facebook is likely very low.

Final conclusions from the interview (my interpretation)

In some cases a third party (like a social media platform) can be held liable, if it can be proven that they were aware of the issue and they had the power to do something about it.

Usually however when the attacker is gone, the user is out of luck since there is no one to sue and recover damages from. This is due to the lack of a central authority and point of contact, which is simultaneously promoted as one of the big advantages of blockchain technology. The government cannot interfere (e.g. no wiretapping, no inflation). This also means however that there is no government protection for users, which people sometimes tend to forget. The average user assumes that they are protected in a similar way as they would be when depositing money in a bank (where the government guarantees the bank), but those rules do not apply to blockchain assets.

⁵The e-Commerce Directive aims to remove obstacles to cross-border online services in the EU internal market and provide legal certainty for businesses and consumers. It establishes harmonized rules on issues such as the transparency and information requirements for online service providers; commercial communications; and electronic contracts and limitations of liability of intermediary service providers.



Blockchain professionals interview protocol and summaries

E.1 Interview protocol

The interviews were structured as an open conversation. Both experts were told about the purpose of the interview, which was to identify various aspects where developers play a role in defining (perhaps unwittingly) responsibilities for users or other parties. What followed was a divergent discussion on various blockchain security aspects.

E.2 Blockchain researchers interview summaries

This section contains the summaries of interviews of the experts that gave (explicit) permission for publication.

E.2.1 Summary interview Dr. Oğuzhan Ersoy

Dr. Oğuzhan Ersoy is a Postdoc researcher at the Radboud University. He works in the area of decentralised systems and blockchain technology, specifically on provable secure, privacy-preserving, and scalable and incentive-compatible protocols. Previously during his PhD he also studied the security and economic aspects of decentralised systems (including blockchain).

Blockchain security

When it comes to blockchain technology, there are various assumptions that users have, that may not be valid. One of those assumptions is the decentralised nature of blockchain. Whether the current implementation of blockchain really is fully decentralised is questionable.

Is blockchain actually decentralised?

In the DAO hack (July 2016) on the Ethereum blockchain network, due to a fault in a smart contract an attacker was able to spend the large sums of money stored in the contract. The general idea of blockchain is that it is not invertible (immutable) and transactions are final. In this case however, Ethereum developers proposed to fork the blockchain (to roll back Ethereum network's history to before the attack). This was very controversial since blockchains are supposed to be immutable and censorship-resistant. While most stakeholders adopted the change, allowing for the fork to be implemented, not all did, resulting in two separate Ethereum blockchains. The fact that this could happen highlights that there is no full decentralisation and censorship is possible.

Another interesting aspect of the decentralisation of blockchain is that all hashing power is concentrated in just a few large mining pools. This indicates that in reality the network is not actually decentralised, as just a few decision-makers hold the majority of the hashing power and can exert power over the network. For BTC for example, the four largest mining pools together hold over 50% of the hashing rate.

Is blockchain actually anonymous?

Another assumption users have is that blockchain ensures anonymity. It is however proven that pseudonymous identities could be linked to real identities through network analysis and investigation.

Furthermore, crypto exchanges function as gates between blockchain systems and the real world. This means that crypto exchanges do know real identities of blockchain users.

Implementation of consensus protocols and smart contracts

An important element of blockchain technology is the consensus protocol. Various consensus protocols exist (Proof of Work, Proof of Stake, Byzantine fault tolerance, hybrid versions). Trade-offs in usability, security, eco-friendliness, developer-friendliness are made when selecting one protocol. No one protocol is the best.

Factors that need to be considered are whether the blockchain needs to be accessible to everyone, and whether anyone can create transactions (private or public, permissioned or permissionless blockchains). Energy consumption is also a factor, PoW uses a very large amount of energy, while PoS has low energy consumption, but other (security) challenges. Scaling is another important point. Some protocols do not scale well, making them inefficient for wide usage.

One of the most crucial security aspects of blockchain applications are smart contracts. All other elements (e.g. consensus protocols) of the blockchain are extensively researched. A fault in a consensus protocol can affect the entire blockchain (e.g. with a 51% attack

or selfish mining attack) but these attacks have been greatly studied. Smart contracts are not studied that well. Every smart contract is different, has its own functionality and implementation. Errors are made at the smart contract level. Normally users are protected if they buy a defective product but if users use a defective smart contract, it is questionable who is liable. Users should check contracts before using them but without being an expert this is difficult, even if most of them are publicly available. There are some companies that do formal verification (e.g. penetration testing of smart contracts). Most contracts are also reviewed online.