BGP HIJACKS TO MAN-IN-THE-MIDDLE DNS SERVERS

DOMAIN REDIRECTION

by

Nils Bijlsma

submitted in partial fulfillment of the requirements
to obtain the degree of Master of Science in Computer Science
at the Delft University of Technology,
to be defended publicly on October 7th, 2024 at 14:00.

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

Thesis committee:

Chair: Georgios Smaragdakis
Committee member: Klaus Hildebrandt
Committee member: Harm Griffioen

ABSTRACT

The border gateway protocol (BGP) is what holds the internet together by making data routing possible between various points on the internet. It is used to exchange routing information between and within networks on the internet with the use of special BGP routers. This routing information consists of a network path and its destination (IP prefix), and is stored in the routers' local routing tables.

Once a pair of BGP routers have been set up, they implicitly trust routes that are shared amongst them. This implicit trust system poses a problem called BGP hijacking: a malicious actor that takes over control of a BGP router, can simply start announcing IP prefixes they do not own. This can lead to different routing paths and thus redirected network traffic, possibly to the infrastructure of the attacker.

In 2017 there were a total of 13,935 routing incidents, consisting of both inadvertent misconfigurations and deliberate IP prefix hijacks. Over 10% (6,000) of all autonomous systems (AS) were affected and about 5% (3,106) of all AS's were a victim at least once. Despite numerous possibilities that do exist in terms of securing BGP, these numbers show that BGP hijacks are still prevalent. As such, there is a lot of room for improvement in terms of preventing hijacks and mitigating their impact.

The aim of this research project was to provide a proof-of-concept framework to aid in performing threat intelligence. A methodology was developed in which the hijacked IP prefixes from BGP hijacks are used as a foundation to gather information about what domains lie in the scope of the impact. Features were determined and extracted from the data, to say something about what domains are more likely to have been targeted. The methodology could also be used as a framework for a real-time detection and response system, by maintaining a live dataset of the different indicators.

Reports of BGP hijacks were analyzed to identify hijacked prefixes. These were referenced against DNS servers to determine what DNS traffic was redirected. The NS records of those servers were then used to identify what domain queries were redirected. Multiple indicators were devised to indicate if domains might have been redirected. This included SSL certificates, DNS records, IP prefix resolutions, as well as the hosting AS's.

We believe this framework serves as a tool to be used for threat intelligence. It uses the basic ideas of BGP hijack detection to look deeper into the impact of such incidents. Instead of only looking at the hijacked prefixes themselves, the DNS servers and domains that are within the scope of the hijack are also considered along with what indicators they might present. If a BGP hijack takes place, it often takes multiple hours to get resolved. Afterwards, the incident is analysed and it is determined what happened. One of

the reasons for this long winded process is that only a relatively small part of the internet is usually affected, so it takes a while before things get noticed. This framework could also be used as a foundation for a detection and response system, such that domains that might be targeted and redirected during such a hijack can act in time.

One important limitation of this research is the incompleteness of the available data. This could be solved by using more data sources and tracking data over time. Another issue in a real-time detection and response system would be the sparse visibility of BGP sensors. Adding other sensor networks could increase the covered surface area of the internet. Each indicator of compromise could be refined further, to better indicate malicious activity.

Contents

1

INTRODUCTION

In this research paper we will be looking at BGP hijacks from a different perspective. We will be detailing current research surrounding hijacks, preventative measures and impact limitation. The problems of BGP and current proposals will be examined. We will go over our methodology and the implementation, lastly we will discuss the results and conclusion.

The border gateway protocol (BGP) [1] is what holds the internet together by making data routing possible between various points on the internet. It is used to exchange routing information within and between networks on the internet by specially set up routers. This routing information consists of the network path to a particular destination (IP prefix) along with some data used for routing policies and is stored in the routers' local routing tables.

Once BGP peering has been established between two BGP routers, both implicitly trust routes that are shared amongst them. There are no security mechanisms in place to ensure correct data is sent, so when the internet became widely available to the public this implicit trust could not be guaranteed anymore. This implicit trust system poses a problem called BGP hijacking: a malicious actor that takes control of a BGP router, can simply start announcing IP prefixes they do not own. This will result in different routing paths and thus redirected network traffic, possibly to the infrastructure of the attacker [2].

The main focus of current BGP research is on monitoring BGP and detecting prefix hijacks, as well as securing BGP to prevent them. Numerous proposals do exist in terms of securing BGP or limiting the possible impact of such routing incidents. One of them is the Resource Public Key Infrastructure (RPKI) that links internet resource information to a trust anchor, which allows an autonomous system (AS) to be authorized to originate certain IP prefixes. To limit the impact when DNS traffic specifically is rerouted, the Domain Name System Security Extensions (DNSSEC) allow for cryptographically signing DNS responses in order for DNS resolvers to verify that the record information is un-

modified.

The main issue with these proposals is that they are time-consuming or difficult to set up correctly and their effectiveness depends on how many entities implement them. In 2017 there were a total of 13,935 routing incidents, which include both inadvertent misconfigurations and deliberate IP prefix hijacks [3]. Over 10% of all AS's were affected (6,000) and about 5% (3,106) were a victim of such an incident at least once.

We intend to look at the scope of the impact of BGP hijacks to look at this problem from a different perspective. We will specifically be looking at hijacks that rerouted DNS traffic and what domains could have been affected. Therefore our main question can be formulated as: *can we determine what domains could have been targeted in a BGP hijack, based on the DNS traffic that was rerouted during said hijack?*

The goal is to provide a proof-of-concept framework to perform threat intelligence about the potential impact of routing incidents and the associated indicators of compromise. We developed a methodology that uses hijacked IP prefixes from BGP hijacks to provide information about what domains lie within the scope of the impact. We then go further to determine and extract features to say something about what domains were more likely to have been targeted.

2

Related Work

This chapter goes over background information regarding BGP and BGP hijacks and looks at current research. In section 2.1, an overview will be presented about how BGP works and what it does. Section 2.2 will describe the possible abuse of the implicit trust system that BGP has and how IP prefix hijacks are performed. The subsections will go over two different types of BGP hijacks. Section 2.3 will go over some of the proposed solutions that aim to prevent hijacks by securing BGP or limit the impact of hijacks.

2.1. How does BGP work?

An autonomous system is a set of routers that belong to a single administrative entity and normally uses one interior gateway protocol to propagate routing information within their own set of routers. The AS presents a clearly defined routing policy to the internet [4]. BGP is a a routing protocol that is used to exchange routing information between BGP routers. BGP supports two types of routing: exchanges among different AS's and exchanges within a single AS. When used between different AS's, external BGP (eBGP) is used and BGP sessions perform inter-AS routing. When used within a single AS, interior BGP is used (iBGP) and BGP sessions perform intra-AS routing [1]. BGP peers advertise routes to each other in update messages and the routes are stored in their local routing tables. The routing information that BGP systems exchange includes the destination, the complete route to said destination, as well as additional information about the route. The destination is represented as an IP address prefix, the route is called an AS path and is a list of AS numbers that the route passes through and the additional route information is included as optional attributes, which are used for routing policies. These routing policies can for example be used to choose from multiple paths to a particular destination or to control the further distribution of routing information. BGP routers use this routing information to maintain a table of network layer reachability information (NLRI). This table is used to construct a graph of AS connectivity, this network topology enables routers to eliminate loops and enforce routing policies.

2.2. How do BGP hijacks work?

BGP hijacking is the process of taking control over an IP prefix assigned to a victim AS [2]. This is achieved by exploiting the implicit trust system that BGP has to change the routing paths used for forwarding network traffic. BGP peers are established by manual configuration and by default accept IP prefixes announced by the other router. So if a router announces a prefix, the peers install the prefix into their routing table and they advertise it to their own BGP peers. This way the prefix is propagated to other AS's, without verification if the originating AS owns the prefix it announces. The attacker AS essentially impersonates the victim AS and pretends to be the owner of the prefixes by announcing them. If these routing updates propagate and get accepted, network paths change which leads to subverted network traffic.

2.2.1. Partial BGP Hijacking

Partial BGP hijacking is a type of hijack where the traffic is not completely redirected, but only a certain percentage of it. It occurs when two autonomous systems announce an identical IP prefix with the same prefix length [2]. BGP normally prefers more specific routes and chooses the update with the longest prefix. However, when both announced prefixes have the same length, BGP determines which path is the best option based on the protocol rules as well as local routing policies. Since the path decision is not unanimous for all routers receiving the route announcement, the traffic is only partially rerouted.

2.2.2. Complete BGP Hijacking

Complete BGP hijacking occurs when an attacker announces a longer and thus more specific prefix than the legitimate owner of the prefix [2]. As stated before, BGP prefers longer prefixes, thus every router that receives the announced prefix will accept the route. This leads to all the traffic that passes these routers to be redirected.

2.3. Proposed Solutions

There have been proposals that aim to solve the implicit trust problem BGP has and secure BGP itself, other measures try to limit or completely negate the impact of BGP hijacks. RPKI is intended to prevent hijacks from taking place, since announced IP prefixes can be verified to be from the legitimate AS. DNSSEC is more concerned with impact mitigation because DNS resolvers will simply refuse forged DNS responses, but the BGP hijack itself can still be performed and traffic can still be redirected. It is important to note that DNSSEC only limits the impact in the case of redirected DNS traffic.

2.3.1. RPKI

The resource public key infrastructure is a public key infrastructure framework designed to help secure the routing infrastructure of the internet [5]. It aims to connect internet resource information such as AS numbers and IP addresses to a trust anchor. It enables an entity to assert that it is the legitimate holder of IP addresses or AS numbers. The legitimate holder of IP address space can use this to authorize one or more AS's to originate routes to their address space. This authorization of origin AS's can prevent BGP

route hijacking because these authorizations can be verified.

Resource certificates describe the allocation of IP addresses or AS numbers to the subject, by the issuer. Any resource holder who is authorized to sub-allocate resources is able to issue new resource certificates that correspond to these sub-allocations, which is done with their own resource certificate. This creates a chain of resource allocations, which is verifiable because the chain of resource certificates leads back towards a trust anchor.

End-entity (EE) certificates are also issued by resource holders with their resource certificates. The primary use for EE certificates is to create Route Origin Authorizations (ROA) which are digitally signed. These objects assert that an AS is permitted to originate routes to the given prefixes which can then be verified by other parties.

2.3.2. DNSSEC

DNSSEC is an extension on top of the regular DNS protocol [6]. It enables authoritative DNS servers to digitally sign their DNS responses and DNS resolvers to verify whether those responses are valid or not. This is achieved by employing an asymmetric cryptographic system, with private and public keys to sign and verify DNS responses respectively.

A resource record set (RRset) is a collection of resource records, such as A or MX records. With DNSSEC, the entire set is returned to a DNS resolver, as opposed to a single DNS record. The set is digitally signed thus it has a corresponding resource record signature (RRSIG).

DNSKEY records contain the public zone-signing and key-signing keys (ZSK and KSK). These records are also signed and have a corresponding RRSIG.

A delegation signer (DS) record is used to transfer trust from a parent zone to a child zone, establishing the chain of trust. It is the hash of the child zone public KSK and stored in the parent zone.

When a DNS resolver receives a DNS response, the name server also includes the accompanying RRSIG. The resolver then requests the DNSKEY records which contain the public ZSK. The initial RRset can now be validated by verifying the RRSIG with the public ZSK.

When requesting the DNSKEY records, besides the public ZSK and KSK the corresponding RRSIG is also returned. The public KSK is used to verify the RRSIG and thus validate the public ZSK. Note that the public KSK verifies itself here too, which does not provide additional security.

Now, to check the validity of the public KSK, it is hashed by the resolver and compared to the DS record from the parent zone. If this comparison is successful, the public KSK

of the child zone can be trusted which means the public ZSK and all the other resource records in the child zone can be trusted too.

The DS record is also a resource record, so it is also signed and has an RRSIG in the parent. The previous validation process is repeated for every zone until the trusted root is reached.

If all the signatures and keys are verified without a problem, we can ensure that the actual DNS response has not been modified and that it originated from the correct DNS server. Redirecting DNS traffic to attacker-controlled DNS servers by performing BGP hijacks, would not work since the forged responses will not be accepted by DNS resolvers.

To summarize the aforementioned:

- Request the desired RRset, which also returns the corresponding RRSIG record.
- Request the DNSKEY records containing the public ZSK and public KSK, which also returns the RRSIG for the DNSKEY RRset.
- Verify the RRSIG of the requested RRset with the public ZSK.
- Verify the RRSIG of the DNSKEY RRset with the public KSK.

2.3.3. Artemis

Artemis is a real-time detection and mitigation framework for BGP hijacks proposed in 2018 [7]. First, they analyzed the impact and visibility of different types of hijacks through simulations of the Internet routing system. These simulations were performed on an AS-level topology of the Internet with inferred routing policies for peering links. The topology was created with Caida's AS-relationship dataset. Pairs of AS's within the topology graph are chosen as attacker-victim pairs. BGP updates are then originated from the attacker AS and propagated through the simulated network. The monitors in the simulation are defined as the AS's that peer with the infrastructure of the monitoring services, meaning the AS's that peer with route collectors from RIPE and RouteViews are considered monitors. The simulations suggest that all types of hijacks, if they are of significant impact (>2%), can be detected by the monitoring services. Based on their findings they developed a methodology to detect and mitigate such attacks. The detection and mitigation framework was evaluated with the simulations described above to validate their methodology and show the effectiveness.

The effectiveness of their system is also tested through experiments on the real Internet. The PEERING testbed was used for these experiments because it is connected to real Internet networks. It allows one to use ASN's and IP prefixes owned by PEERING to originate routes, so other AS's are not impacted. The monitors here are the route collectors from RIPE and RouteViews that provide the framework with data. With these experiments they were able to determine the time between an attacker announcing a hijacked prefix and ARTEMIS detecting the event as well as the time between the pollution of a monitor and the monitor receiving a legitimate route again.

The impact analysis that is performed considers the impact on the control plane, or route propagation, and thus only looks at affected AS's within the network topology. The effect of the hijack on the data-plane is divided into one of three categories: black holing, man-in-the-middle or impersonation. This is only used to show what types of hijacks in their attack taxonomy can be detected and mitigated. Our work looks at a different form of (possible) impact, such as IP addresses, domain name servers and domains along with what indicators can be found.

3

Problem Description

This chapter will first look at the problems that BGP hijacks pose to various stakeholders, as shown by various examples. Section 3.2 will highlight the research gap and detail the contributions of our research.

3.1. Relevance of the problem

In 2017 there were a total of 13,935 incidents, these include both inadvertent misconfigurations and deliberate prefix hijacks [3]. Over 10% (6,000) of all AS's were affected, and about 5% (3,106) of all AS's were a victim at least once. Despite the numerous possibilities that do exist in terms of securing BGP: securing DNS records (DNSSEC) and authorizing AS's to originate prefixes (RPKI), these numbers show that there is still a lot of work to do to secure BGP.

The Border Gateway Protocol was built on implicit trust, this was fine in the early days of the internet because a limited number of entities were present such as academia and governmental institutions. It does not include security mechanisms and assumes the network operators will secure their systems and not send incorrect data. However, when the internet became widely available to the public this implicit trust could not be guaranteed anymore due to a large number of new entities gaining access.

This still existing trust-based system allows malicious actors to hijack ranges of IP addresses (IP prefixes) they do not own, to redirect traffic destined to those IP addresses. This way DNS traffic destined for certain DNS servers can be redirected to attacker-controlled DNS servers. These rogue DNS servers can then respond with forged DNS responses. If a user navigates to one of the affected domains, they will end up on an attacker-controlled version of said domain.

This issue affects end-users, as stated before they could end up on attacker-controlled websites. These websites are essentially part of a man-in-the-middle attack and the original request can be forwarded to the authentic website for the hijack to stay unnoticed.

Anything that the user transmits to what they think is only the legitimate website, will be transmitted to the attacker. Data can for example include usernames, passwords, email addresses and credit card information. These in turn can be used to compromise accounts, perform credit card fraud or empty online wallets such as for cryptocurrency [8].

The problem for domain registrants themselves is numerous. If their traffic is redirected to IP addresses controlled by the attackers and the attackers don't relay the traffic to the legitimate domain but instead drop it, their service will be interrupted [9]. But if the traffic is relayed to the actual domain the service itself won't go down, thus the hijack can stay undetected for longer and more data can be gathered. If critical system accounts get compromised such as virtual private network accounts, it will provide the attackers with an easy way to enter and compromise the back-end of systems.

IP prefixes belong to autonomous systems (AS) and both are assigned to organisations. These organisations include internet service providers (ISP), universities and also companies. Considering that BGP hijacks reroute traffic for the affected prefixes and the prefixes are owned by said organisations, they have a moral obligation to make sure that their service is not interrupted. If they don't, customers might complain about the state of affairs and even decide to go to a competitor.

On February the 24th, 2008, Pakistan Telecom configured a route for the prefix 208.65.153.0/24 which pointed to nothing. Their goal was to blackhole outgoing traffic to YouTube to block access for their customers [9]. The ISP started to announce the prefix to their upstream provider PCCW Global, which in turn propagated the announcement to its peers. This resulted in large parts of internet traffic destined for YouTube to be redirected towards Pakistan Telecom, instead of only their customers' traffic being blackholed. YouTube had been announcing the prefix 208.65.152.0/22, which is less specific, so the propagation of the hijack was quite large since more specific routes are usually preferred. After an hour and twenty minutes, YouTube began to announce the more specific prefix 208.65.153.0/24, which was also used in the hijack. This meant that the hijacked route was not necessarily preferred anymore because both prefixes were the same length (/24). However, YouTube was still not available for a large part of its users, whose traffic was still taking the path towards Pakistan Telecom. YouTube then started to announce two sub-prefixes: 208.65.153.0/25 and 208.65.153.128/25. These longer prefixes are preferable so the legitimate paths were installed in routing tables and YouTube was reachable again.

MainOne, which is a small ISP in Nigeria, started originating over 200 IP prefixes owned by Google [10] [11] [12]. China Telecom accepted the routes and propagated them to its peers, including Transtelecom and other large ISP's. Due to these large ISP's propagating the illegitimate routes, a large portion of traffic destined for Google was redirected. The prefixes that were hijacked included Google Search and also affected Google Cloud customers such as Spotify. The illegitimate paths also transmitted sensitive communication such as Google's corporate WAN infrastructure and Google VPN, but luckily this traffic is all encrypted. BGP incidents can go undetected until end users begin to report dropped

traffic and unavailable services. In this case MainOne fixed their misconfiguration after 74 minutes.

On the 24th of April, 2018, eNET (AS10297) started to announce five more specific /24 prefixes belonging to Amazon Route 53 DNS: 205.251.192.0/24, 205.251.193.0/24, 205.251.195.0/24, 205.251.197.0/24 and 205.251.199.0/24 [8] [13] [14] [15]. Most likely someone compromised eNET's systems to perform the BGP hijack. The announcements originated from eNET to its peers 1&1 Internet SE, Hurricane Electric, Shaw Communications Inc. and BroadbandOne/WV Fibre. These BGP announcements were recorded by RIPE Stats as well as Isolario. The BGP routes were most likely not globally routed, only just over 15% of Oracle's BGP sources had them in their routing tables. This hijack lasted for around two hours before the original routes were restored. These malicious announcements caused re-routing of Amazon Route 53 DNS traffic to a rogue DNS server hosted on a (compromised) Equinix server in Chicago. The rogue DNS server only responded to queries for myetherwallet.com. Users who used recursive DNS resolvers that accepted the hijacked routes were routed to a phishing website hosted in Russia [16] when they tried to visit myetherwallet.com. DNS observations showed that myetherwallet.com was resolved to 46.161.42.42, registered as being in Germany, but routed out of Luhansk, Ukraine [17]. Users had to click through SSL certificate alerts in their browsers to visit the imposter website.

3.2. CONTRIBUTION

Some solutions, such as RPKI, introduce frameworks to prevent hijacks. There are also real-time detection and mitigation systems such as ARTEMIS. Others limit the impact of hijacks such as DNSSEC. But the latter two do not directly solve the underlying issue. Most of these solutions depend on the willingness of third-parties to be adopted. Since most of the proposed solutions are not implemented on a wide scale, hijacks still do occur as shown in the various examples earlier.

The reported incidents often mention mitigation times exceeding 60 minutes, which is also what Artemis found in their network operator survey. They mention that a possible cause for this could be that detection is not automated and likely based on service unavailability for end-users. The incident analysis usually focuses on the hijacked prefixes and the newly routed IP addresses, or simply that a website has been blackholed. In some cases, such as with the Amazon Route 53 incident, other indicators are examined too such as SSL certificates.

The goal was to develop a methodology that can, in theory, be used in real-time. It does not only focus on the hijacked prefixes, but uses them to determine domains that could be affected. We look at various indicators of compromise to identify what domains were more likely to have been targeted, such as SSL certificates, newly seen IP addresses and AS numbers. We will specifically be looking at hijacks that rerouted DNS traffic.

This research aims to provide data analysis tools that can analyse BGP and DNS hijacking events, which can be used to provide threat intelligence about BGP hijacks and their

impact. This could also be used for designing and developing an incident response system deployable by anybody, without depending on autonomous systems or other third-parties.

4

METHODOLOGY

The research methodology will be described in this chapter. In section 4.1, the research question will be formulated along with the hypothesis. The research question is divided into individual items that need to be accomplished. An overview of the methodology is also presented. Section 4.2 will explain how BGP hijacks relate to DNS and how we initially identify what domains are potentially affected. Section 4.3 will go over the various data sets that are used. In section 4.4, a methodology is developed that makes use of multiple indicators of compromise, which are used to determine what domains were possibly redirected.

4.1. OVERVIEW

The aim of our methodology is to use the hijacked prefixes as a basis and identify redirected DNS traffic. Based on what DNS traffic was rerouted, we can identify what domains were potentially affected. Lastly, we use multiple indicators of compromise to look for anomalies.

4.1.1. RESEARCH QUESTION & HYPOTHESIS

Following the described problem and research gap in the previous chapter, we define the research question, individual work items and hypothesis as follows:

Research Question: Can we determine what domains are likely to have been targeted in a BGP hijack, based on the IP prefixes that have been rerouted during that hijack?

- We need to determine what Domain Name Servers are located in the hijacked IP address space. For this, we need DNS zone files, because it contains the mappings between IP addresses and domain names (DNS servers).
- With the DNS servers, we can use the DNS zone files again and this time use the NS records to find all the domains resolved by those DNS servers.

- With this list of domains, we need to retrieve their SSL certificates which are to be used as an indicator of compromise (Censys data set). We also need to figure out what certificates attributes are used for this.

- With the same list of domains, we need to retrieve the IP resolutions (RiskIQ Passivetotal). Anomalous IP addresses are also to be used as an indicator of compromise. We need to figure out what method to use for this.

- Those IP addresses can be turned into autonomous system numbers, which would indicate too if rerouting has taken place.

- The previous indicators are combined, which gives me a "likelihood" as to whether or not a given domain was targeted in the BGP hijack.

Hypothesis: With this analysis method, we achieve a list of domains that might have been targeted in a BGP hijack, including a measure of likeliness.

4.1.2. METHODOLOGY

BGP data is used to identify the hijacked IP prefixes. The hijacked IP prefixes are used in conjunction with the DNS zone file to identify the DNS servers within the scope of the BGP hijack. With the DNS servers and the DNS zone file, we identify what domain names they are authoritative for. The SSL certificates are retrieved for those domains. We compare certificates and identify pairs that present certain red flags. From this we gather the domains to work with a smaller, filtered set in the next steps. These domains names are used to retrieve IP prefix data, and a distance measurement is calculated to identify outliers. We also retrieve the A records for the domains and determine the newly occurring IP addresses during the incident. The IP addresses are also combined with IP to AS data to find the newly occurring AS numbers the domain is routed from.

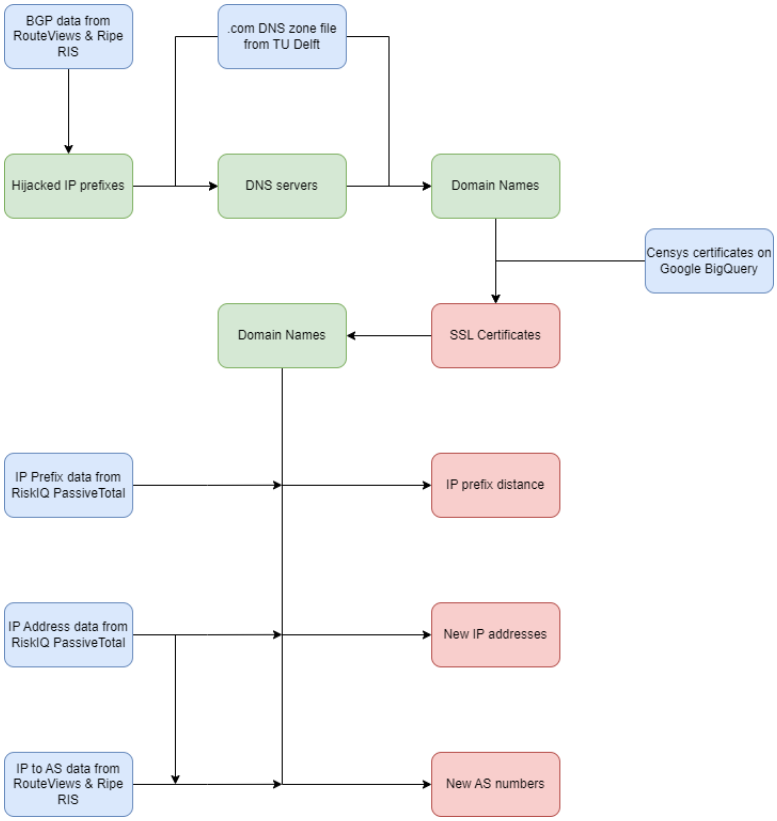

Figure 4.1: Overview of the research methodology.

4.2. IP PROTOCOLS

The hijacked prefixes are used to determine what DNS traffic was rerouted and ultimately what domains could be affected. For this we use DNS zone files, specifically A records and NS records.

4.2.1. BGP

To figure out how BGP is related to DNS, we start looking at the BGP hijacks themselves. This means BGP hijacking events need to be identified with the help of articles, news outlets, and so on. MANRS [18] has a blog post about a BGP hijack targeting Taiwan Public DNS. Oracle [8], Lacnic [17], The Register [14], Internet Society [15] and Oracle Dyn [13] outline BGP hijacks concerning Amazon's Route 53 DNS and a multitude of payment system providers such as Vantiv and Mercury. Note that these incidents seem to target DNS specifically.

BGP data needs to be collected of hijacking events, preferably from a source with a lot of route collectors in different locations such as RIPE and RouteViews. This data would

need to be processed, the result will contain the hijacked prefixes. This data is used to map out the original location of the prefix hijacks, and see what autonomous system numbers are often the culprit. The BGP collectors that receive the BGP hijack update messages are also mapped.

4.2.2. DNS

To find out how the BGP prefix hijacks relate to DNS, the hijacked prefixes need to be referenced against the IP addresses of DNS servers. This way it is determined what DNS servers might be in the scope of the hijack, and thus what DNS traffic might be redirected. DNS zone files are needed for this: they include A records which contain DNS server and IP address tuples.

With a list of possible targeted DNS servers, the NS records corresponding to those servers can be retrieved from the same zone files. NS records contain the DNS server and domain tuples, thus the resulting information will include all domains which are hosted by said DNS servers. From this point, a methodology is devised to determine what domains contain indicators of compromise and were likely redirected.

4.3. DATA SOURCES

The various data sources that were used are described in this section. It provides an overview, arguments in favour of them and why other sources were ultimately not used. This includes the sources for BGP data, DNS zone files, certificate data and DNS records.

4.3.1. RIPE RIS & ROUTE VIEWS

BGP route information data is used from RIPE's Route Information Service raw data set [19] [**RIS2**] and the Route Views project from the University of Oregon [20]. The reason for this is that both provide historical data, are publicly available and use a multitude of collectors situated around the globe. As an example, BGPmon monitors BGP routing information from various connected peers in real-time [21]. It was not used since it does not provide historical data.

4.3.2. DNS ZONE FILES

The supervisor of the project could provide the DNS zone files of the .com top-level domain. These were generated on a day-to-day basis by the University. This was a much easier solution than trying to compile these files ourselves or asking domain name registrars such as Verisign or Namecheap for the zone files.

4.3.3. CENSYS

Censys' data set was chosen to be used for the SSL certificates [22] [23]. They provide research access for free to students and research organisations [24].

4.3.4. DNS RECORDS & IP RESOLUTIONS

The data sets that were considered for the DNS records & IP resolutions were: Securitytrails [25], Forward DNS from Rapid7 [26], DNSDB [27] from Farsight Security, and

Passivetotal from RiskIQ [28].

Securitytrails explanation here.

Forward DNS is where one actively queries domains and records the responses. It turned out that Rapid7 only records responses one day a week, so if a domain was redirected on a different day it was most likely not recorded. Another issue is that they only provide one record for each domain, so it seems they only query it once, and most likely from one location. The problem with this is that the location might not have been within the affected scope of the incident, which also means the redirection might not have been recorded.

DNSDB contains passive DNS data. This works by placing probes at DNS resolvers, these then record DNS responses that pass those resolvers. This would be a better option because Farsight uses a global sensor array. However, DNSDB could not be utilized because the trial API only provides 100 queries per day for one month.

Passivetotal also provides passive DNS data from a global sensor array. The API to access the data set can be queried with curl or the requests library from Python. There are also multiple bindings such as the Python binding provided by RiskIQ themselves. It was decided that the web application would be used instead because it offered a large number of queries each day with a promotion code provided by the University. Also, Harm could provide a bash script that would execute the web query for all the input domains and store the results automatically.

4.4. METHODOLOGY TO DETERMINE REDIRECTED DOMAINS

When researching relevant incidents where domains were redirected, we found that invalid certificates were one of the most obvious indicators. When a web browser deems an SSL certificate as unsafe, it notifies the user with an error message displayed in the browser. This was the starting point for the idea to start with SSL certificates.

The idea is that SSL certificates are issued for domains the attacker plans to redirect. So if an incident occurs on a particular day, one can analyse the recently issued SSL certificates for the possibly affected domains. Censys makes use of public certificate transparency servers.

Censys provides a few different ways of querying their certificate data, a decision had to be made as to what method was going to be used. The web application could not be used, since only a handful of queries are allowed each month. Since almost all of the processing was done in Python, Censys' Python binding was looked into after. API access was easy to set up and the API calls were straightforward to construct. It was very simple to specify the filter in the API calls since the same syntax was used for the web application of Censys. However, the number of domains for which certificates needed to be retrieved exceeded the number of API calls that could be performed per month. Such a large number of queries could not be condensed into one query (or a few) due to the

limited query size imposed by the Python binding.

The data set was also available via Google BigQuery [29], so we decided to look into this next. The aforementioned problem about not having enough API calls was not present here. We could apply for enough research credits. Since the maximum query length for its web application is 12 MB, it was not feasible to manually run multiple queries, and then manually export and download the results. This is especially the case if you consider the effect on scalability if more domains need to be queried in the future. The Python binding for BigQuery provides the same maximum query length of 12 MB, but it allows for automatic query execution as well as exporting and downloading of the results. This way the query itself and the input of the queries need to be specified manually, and then they will be executed one by one and their results exported and downloaded automatically. This took quite a while to set up because it requires quite a bit of programming including formulating a working SQL query, but provides excellent scalability and ease of use in return.

Once the certificates are retrieved, they need to be analysed. The domain certificates are first sorted by issue date and then by subject. This way for any given domain the certificates can be compared in a pairwise manner. The assumption is that issued certificates stay mostly consistent over time. So the key fingerprint of the certificate stays the same. When the fingerprint changes, it means the signing key pair that signs the certificate has changed. This could indicate that a malicious party had a certificate issued. Even with the availability of key rotation, this is still worthwhile to check, since key rotation is not used all that frequently Reference. If a certificate within a comparison is self-signed, this is an immediate red flag. Self-signed means the subject of the certificate is also the issuer, there is no chain of certificates leading back to a trust anchor (a certificate authority). Certificate Authority certificates are intended to sign other certificates, they are not intended to be used by websites. Thus if a certificate for one of the domains is suddenly issued as a CA certificate, this is a reason for suspicion. Lastly, the certificate data also contains information about the chain of certificates. Apple, Google, Microsoft and NSS assess whether or not the chain of certificates leads back to a trust anchor. So in the case that the certificate is not self-signed, but these assessments state there is no path leading to a trust anchor, it will still be flagged because the adversary might have used a rogue CA. This is used as a filter to narrow down the list of domains as much as possible.

4.4.1. DNS RECORDS

Once we processed the domain certificates, we have determined a filtered list of domains for which certificate anomalies were found. The next step is to determine whether or not forged DNS responses were issued for these domains. The plan is to first collect historical DNS data for each domain in the filtered list. As mentioned before, Passivetotal from RiskIQ [28] will be used to gather this data. The IP resolutions will also be retrieved. Next, records will be identified which indicate a domain was hosted on a new IP address, outside of the regular IP space. The IP address that has the largest average distance to the other addresses is deemed the anomalous IP, for the given domain.

4.4.2. IDENTIFYING INDICATORS

If the signing keys and/or algorithm are inconsistent across certificates, it may indicate a different entity has issued the certificate with a different key pair and/or algorithm. So this might be a certificate issued to redirect the domain. However, this might also just be a case of key rotation or using a different certificate authority to issue the certificate, so it serves as a baseline and additional attributes are considered for the decision to flag a certificate comparison.

If an end-entity certificate is self-signed, it lacks a trust chain. Generally speaking, this is suspicious because it cannot be verified back at the trust anchor. An end-entity certificate which happens to be an intermediate CA certificate is also strange. This is because intermediate CA certificates are designed to issue other certificates, end-entity certificates are not (they are leaves). So these enhance the initial baseline of the key inconsistency.

The validity of the chain of trust is something else that can be investigated. This could be done by retrieving the parent certificates, but these are stored as hashes inside a list. However, the hash lists increase the amount of processed data a lot and to get these certificates more data queries would need to be done. This is not feasible with the available BigQuery credits. The certificate transparency information provides whether the certificate had a trusted path, which means whether or not it had a valid chain of trust to a trust anchor. Let's say the certificate is not self-signed and it is not designated as a CA certificate, this can still give information whether it was issued legitimately or by for example a Rogue CA.

In all the certificates, the blacklisted, whitelisted and revoked attributes are false. So they don't give additional information to include them in the certificate comparisons, as opposed to the had_trusted_path attribute as explained before.

4.4.3. DISTANCE BETWEEN THE DIFFERENT IP PREFIXES

We assume a domain is generally hosted on a particular IP address, a range of IP addresses or even a couple of prefixes. By using the IP resolutions from DNS records, we can determine the address that, on average, is the furthest away from the others (logically speaking). The idea is to find anomalous IPs where the domain could have been redirected to.

4.4.4. COMPARING IP ADDRESSES AND AS NUMBERS BEFORE AND DURING THE BGP HIJACK

Similarly to the certificates being compared to identify chronological changes, it was also decided to look at the IP addresses and AS numbers of the domain names. The idea behind this is that there is a particular set of IPs/As's before the BGP hijack that host a particular domain. When we compare this with the set of IPs/As's during the BGP hijack, we may find out a domain is now hosted somewhere else, which gives us another indicator as to whether or not a domain has been redirected as a result of the BGP hijack.

5

IMPLEMENTATION

In this chapter, the implementation of the previously discussed methodology will be described in detail. Section 5.1 and 5.2 will go over how the BGP prefix hijacks are used in conjunction with DNS zone files to obtain the set of DNS servers and domains in the impact scope. Then in section 5.3, the process of data retrieval for the SSL certificates is described, including what attributes are used to compare certificates to each other. How the weights are determined for those attributes is also explained in this section. The prefix distance measurement is examined in section 5.4. The comparison of the IP addresses & AS numbers will be the focus of sections 5.5. Then, in section 5.6 the various indicators will be combined. The last section will detail the process of the BGP data inspection.

5.1. BGP DATA

The first step is to download the BGP update messages of a particular hijack incident. These are stored as .gz files on the website of RIPE NCC [19] in five-minute intervals. We used the Python library requests by Kenneth Reitz [30] and the Beautiful Soup library by Leonard Richardson [31] to retrieve the web pages of RIPE and parse the HTML content respectively. This web scraper navigates through the web page to retrieve all the route collector pages and identifies the links to the desired year/month pages.[1] Then, it retrieves the year/month pages and identifies the links containing the correct date and timestamp. It also checks the file names to make sure they are the .gz BGP update files.[2] The complete links are used to define the output folder and file name. Lastle, the files are downloaded to the defined destination.[3]

Since the files are in MRT format, we need to convert them to plain text for actual processing. For this conversion we use the C library libBGPdump[32], which is maintained

[1] iterate_website() https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/downloadmrt.py
[2] get_mrt_links(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/downloadmrt.py
[3] download_mrt_data(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/downloadmrt.py

by RIPE NCC. It takes MRT dumps as input and then outputs the BGP updates in plain text.[4]

The directory which contains all the converted files is then traversed, and the update messages containing the hijacked prefixes are selected. The time of the messages, the path of autonomous systems and the announced prefixes are then retrieved from those update messages.[5]

5.2. DNS ZONE FILES

From this point on, we use the zone file as described in section 4.3.2. The figure below depicts the overall structure of such a file.

```
; The use of the Data contained in Verisign Inc.'s aggregated
; .com, and .net top-level domain zone files (including the checksum
; files) is subject to the restrictions described in the access Agreement
; with Verisign Inc.

$ORIGIN COM.
$TTL 900
@ IN    SOA     a.gtld-servers.net. nstld.verisign-grs.com. (
                                1524499341 ;serial
                                1800 ;refresh every 30 min
                                900 ;retry every 15 min
                                604800 ;expire after a week
                                86400 ;minimum of a day
                                )
$TTL 172800
 NS A.GTLD-SERVERS.NET.
 NS G.GTLD-SERVERS.NET.
 NS H.GTLD-SERVERS.NET.
 NS C.GTLD-SERVERS.NET.
 NS I.GTLD-SERVERS.NET.
 NS B.GTLD-SERVERS.NET.
 NS D.GTLD-SERVERS.NET.
 NS L.GTLD-SERVERS.NET.
 NS F.GTLD-SERVERS.NET.
 NS J.GTLD-SERVERS.NET.
 NS K.GTLD-SERVERS.NET.
 NS E.GTLD-SERVERS.NET.
 NS M.GTLD-SERVERS.NET.
COM. 86400 DNSKEY 257 3 8 AQPDzldNmMvZFX4NcNJ0uEnKDg7tmv/F3MyQR0lpBmVcNcsI
VmcQ1GLkC6NlYvG3HwYmynQv6oFwGv/KELSw7ZSdrbTQ0HXvZbqMUI7BaMskmvgm1G7oKZ1YiF
COM. 86400 DNSKEY 256 3 8 AQOz+lBqxZtCKBBqKsO/l9JVchZ2Z1pFCWnj+pFHJl3uPWiY
COM. 86400 NSEC3PARAM 1 0 0 -
COM. 900 RRSIG SOA 8 1 900 20180430160221 20180423145221 36707 COM. Lms+34
F07lJiV33cOR5kgqVwE9gusafC9B+UlFq3g=
COM. RRSIG NS 8 1 172800 20180430044536 20180423033536 36707 COM. a5+WasLE
9dG9XrtYNl/D1PcMdAldjlOCCiurfA3+Y=
COM. 86400 RRSIG NSEC3PARAM 8 1 86400 20180430044536 20180423033536 36707
hvq7yUcL3JZpgnH5UuxXFAhuniN1GjDjIfmJ9fbQQYEoUQ=
COM. 86400 RRSIG DNSKEY 8 1 86400 20180502182533 20180417182033 30909 COM.
uez+dbM24XQr1CHCZyUzgZ+WXXn0uOAIU4eMTQHO7veGwTdWHbC6aVcoIoXo+C6kzoAv6cDYxV
ijlBrYqTw==
KITCHENEROKTOBERFEST NS NS1.UNIREGISTRYMARKET.LINK.
KITCHENEROKTOBERFEST NS NS2.UNIREGISTRYMARKET.LINK.
KITCHENFLOORTILE NS NS1.UNIREGISTRYMARKET.LINK.
KITCHENFLOORTILE NS NS2.UNIREGISTRYMARKET.LINK.
KITCHENTABLESET NS NS1.UNIREGISTRYMARKET.LINK.
KITCHENTABLESET NS NS2.UNIREGISTRYMARKET.LINK.
```

Figure 5.1: The zone file of the top level domain .com.

We first get the A records from the DNS zone files. This includes all the DNS servers and their IP addresses.[6]

[4] process_mrt_data(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/downloadmrt.py
[5] iterate_website(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/processmrt.py
[6] retrieveDNSservers(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/processDNSzone.py

```
0DB1.0DB A 67.171.129.222
568BWDNSEXT01.SCHOLASTIC A 208.156.34.11
98.TORIDE A 160.16.238.13
AAADEL.MANTRAONLINE A 202.56.230.6
AAS-NETSTATION1.STATESMAN A 169.137.124.61
NS13.DUROCOM A 216.53.144.31
ABC.BBS-LA A 216.144.232.8
NS1.ABOVEWEB A 209.177.153.167
ABRAXAS.SCENE A 216.231.37.202
KINDA.IFFY A 208.201.229.60
AC3.CORENETWORKS A 24.222.78.5
AC.PUB.MSANET A 204.28.31.132
NS1.ACCELERATED A 208.69.76.3
ACCESSAIG.AIG A 167.230.227.139
NS2.NETAXS A 207.59.153.241
```

Figure 5.2: The DNS A records of the DNS servers.

Then for each record, the IP address is referenced against the hijacked prefixes. If the IP address is contained within the hijacked prefix, the DNS server is written to a file. This gives us a file listing the possible targeted DNS servers.[6]

```
NS1.ASIS
NS.WARNERBROS
NS2.EMPIRETOWERS
NS1.DYNAMIC
NS1.FCBIS
NS2.FCBIS
NS2.MICRON
NS1.MICRON
NS1.INTACS
NS2.INTACS
DNS.KYONPY
NS2.TOOWARDS
NS2.BROADPOOL
NS1.GLPALS
NS2.PRIMALITY
```

Figure 5.3: The DNS servers after checking the IP addresses.

Then, all the NS records of those DNS servers are retrieved from the zone file and written to another file. These NS records contain the domains for which the DNS servers are responsible.[7]

```
WBKIDSGO NS NS.WARNERBROS
IDIGTHIS NS NS.WARNERBROS
SKULLISLANDTHEMOVIE NS NS.WARNERBROS
SKULLISLANDMOVIE NS NS.WARNERBROS
WBINTL NS NS.WARNERBROS
THRILLERMAX NS NS.WARNERBROS
THEWATERDIVINER NS NS.WARNERBROS
B-THREE NS NS.WARNERBROS
ARTOFWARMOVIE NS NS.WARNERBROS
HARRYPOTTER-ONLINE-MINISITE NS NS.WARNERBROS
ARABISOFT NS NS.WARNERBROS
DEFTONESRECORDS NS NS.WARNERBROS
NFLFILMSDVD NS NS.WARNERBROS
MEBEFOREYOUMOVIE NS NS.WARNERBROS
3D-HQ NS NS.WARNERBROS
```

Figure 5.4: The NS records of the filtered DNS servers.

From here, all the records are iterated, the domains are retrieved from them, the top-level domain (.com) is incorporated and then they are written to a file. Now we have a file that lists all the possible targeted domains, that are hosted by the aforementioned DNS servers. This file will later be used as the input for Google BigQuery. The domains are added to a list and each time the list contains 25.500 domains, the entire list is written to a separate file and then emptied. This was done to construct the input files for RiskIQ Passivetotal, due to their query limits.[8]

[7] retrieveDomains(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/processDNSzone.py
[8] getUniqueDomains(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/processDNSzone.py

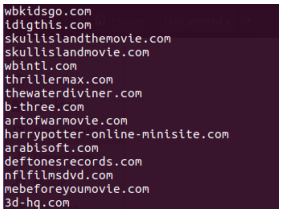

Figure 5.5: The domain names that are hosted by the DNS servers within the scope of the hijack.

5.3. SSL CERTIFICATES

This section will describe the implementation of the certificate retrieval, as well as the comparison process.

5.3.1. CERTIFICATE RETRIEVAL

First, we need Python to perform an oAuth 2.0 authorization flow, for the application to have access to the data on BigQuery. For this, the program needs the key of the Google account used on BigQuery that has the required resources. This key is called an OAuth client ID and is stored in a JSON file. The program also needs the scope that it wants to access. The scopes, in this case, are the BigQuery resources that contain the Censys data set and Google Cloud Storage which is used to export query results to and download them from. The authorization flow returns credentials, which are used to create a BigQuery client object that can perform the various tasks.[9]

The application opens all the files with the possible targeted domains, the domains are iterated over and put in a list with a maximum length of 300.000. This limitation is imposed by the maximum size the query parameters can be. The list is provided as a query parameter and the destination table for the results is set. The query is then constructed with those properties and executed.[10]

For exporting the query results from the destination table to Google Cloud Storage, we provide the destination table to the export method. We also need to specify the destination, which is the Cloud Storage URI. This URI has the format of "gs://bucket/filename". The bucket is a container or folder within Google Cloud Storage, where all the exports are going to be stored. The filename is based off the domain sequence. We also set the destination format to newline delimited JSON, which writes every row of the destination table (every certificate) on a separate line. This makes it easy to iterate over certificates later. Then, the extract table method is executed with these parameters. Now we have our results in a JSON file on Google Cloud Storage from where we can download them.[11]

After all the queries have been executed and all the results have been exported, the download method is called. We use the same credentials as before to create a storage

[9] oauth(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/bigquery.py
[10] bigquery_request(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/bigquery.py
[11] export(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/bigquery.py

client, which is used to perform actions within Google Cloud Storage. We specify the bucket where we stored all our exports and list all the files in there. Then we iterate over the files, download each file and subsequently delete it from the bucket.[12]

5.3.2. CERTIFICATE COMPARISON

As mentioned in the methodology in chapter 4, the certificates need to be sorted by issue date and then by subject. The reason for this is that for any given domain, we can then compare the certificates pair by pair, chronologically, and identify changes in the certificates over time. Each line in the SQL query results is iterated and the certificates are put in a list. This list is first sorted on the key start, which is the issue date. The resulting list is then sorted based on the key common name, which is the subject. Now that our list of certificates is sorted the way we need it to be, we can iterate it one more time to write each certificate back to a file.[13]

The file that contains the sorted certificates is opened and the certificates are iterated over. From the current certificate, we retrieve the common name, the key fingerprint, the self-signed boolean and the is_ca boolean. The Apple, Google, Microsoft and NSS booleans, that assess whether or not the certificate had a trusted path to a trust anchor, are also retrieved. These properties are also retrieved from the previous certificate, which is stored in a separate variable that is continuously updated. Then we compare the properties, as described in the methodology. If the key fingerprint changes and at least one of the self_signed, is_ca, Apple, Google, Microsoft and NSS booleans as well, we deem this a certificate comparison of interest and store them in a dictionary, before putting them in a list. The common name is also checked to be the same for both certificates because we are comparing certificates of a specific domain. This check removes the edge cases where the domain certificates of a new domain appear in the iteration. Finally, the previous certificate is updated to the current certificate for the next iteration. After all this has been done, the list of flagged certificate comparisons is stored in a file.[14]

5.3.3. CERTIFICATE ATTRIBUTE WEIGHTS

As described in the methodology in chapter 4, various attributes of the certificates are used as an indicator of compromise. The idea is that changing certificate attributes signify that the domain for which it was issued might have been redirected. To define how valuable the attributes are in being an indicator of compromise, weights need to be determined. We decided to look at a large set of popular domains and their certificates, and see how often an attribute has the indicative value. The top 500.000 domains of the Majestic Domain Ranking [33] and their certificates from Censys were used for this. Majestic ranks websites based on the number of referring subnets. The assumption is that most of these certificates are valid, so the less frequently an indicative value occurs, the more telling it will be.

The top 500.000 domain list was downloaded as a .csv file. The .csv file is opened, a

[12] download_blobs(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/bigquery.py
[13] sortCerts(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/sortCerts.py
[14] process(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/processCertificates.py

CSV reader is created that parses the file, delimiting on columns. The first row is skipped since it contains column headers, then for each next row in the file, the domain from the second column is written to an output file.[15]

The certificates of the top 500.000 domains are retrieved in a similar way as described in section 5.3.1. First, the counters for the various attributes and a counter for the number of processed certificates are defined. Then, the files in the directory that contains the Majestic domain certificates are iterated over, they are opened and their lines are iterated over as well. From each line, the certificate is retrieved. If 'example.com', 'synology.com' or 'fritz.box' is found within the certificate, the certificate is skipped. If this is not the case, we continue and increment the counter for the processed certificates. Now, we check for every attribute whether or not it has its desired value, if it does its corresponding counter is incremented as well. In the end, the ratio between the attribute occurrence and the number of processed certificates is calculated.[16]

After determining the frequency of the attributes with the indicative values, there were a couple of attributes with a frequency as high as 0.8, which was suspected to be incorrect. We then listed all the domains whose certificates contributed to the occurrences of the attributes and how often this was the case for each attribute. We found three domains that were responsible for almost all occurrences of attributes: example.com, synology.com and fritz.box. The reason for this is that certificates issued for these domains are issued for testing purposes or personal connections such as connections to routers or network storage solutions. When we excluded these three domains, the frequency of the attributes went down considerably, which are the semi-final weights.

Attributes can influence one another, for example, a self-signed certificate would never have a path leading back to a trust anchor, hence the trusted path attributes would always be false. To identify this dependence, conditional probabilities are calculated, which are then used to adjust the previously mentioned attribute weights. The files that contain the certificates are defined and each line is traversed to retrieve the certificate from. If attribute A has the desired value, it is checked if B has the desired value as well. This determines P(B|A) and this is done for all combinations of attributes.

We define counters for each attribute and the files that contain the SSL certificates from the BGP incident. Each line of the files is iterated and each certificate is put in a dictionary, from which the The different attributes are retrieved. Then it is checked if a particular attribute A has its indicative value, if this is the case, its counter will be increased. Only then will the other attribute B also be checked for their indicative value and their corresponding counter incremented as well. In the end, the counter of B will be divided by the counter of A and multiplied by 100 to get the probability of B given A. This process is essentially the same for all individual conditional probabilities.[17]

[15] Top1mDomains(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/processDNSzone.py
[16] attributeWeights(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/DetermineWeights.py
[17] probabilities(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/DetermineWeights.py

P(A|1)	self_signed	is_ca	apple_path	google_path	microsoft_path	nss_path
self_signed	-	26.982	100	100	100	100
is_ca	99.768	-	100	100	100	100
apple_path	8.471	2.291	-	99.800	99.800	100
google_path	8.488	2.295	100	-	100	100
microsoft_path	8.488	2.295	100	100	-	100
nss_path	8.471	2.291	100	99.800	99.800	-

The probabilities in the table above show that the attribute apple_had_trusted_path represents the set of certificates that is the super set, in comparison to the sets of certificates represented by the other attributes. This in turn means that only this attribute provides information about the redirection of the accompanying domain because all other attributes are contained within it.

5.4. IP PREFIX DISTANCE MEASUREMENT

Now that we have a list of the certificate comparisons which display anomalies, we need to extract the unique domains from them. We used the certificates as a filter to narrow down our set of possibly targeted domains. We are going to query the DNS records for this new set.

We open the file that contains the flagged certificate comparisons and iterate over each comparison. For each comparison we retrieve the alternative subject names, and we check if they are not in the dictionary that keeps track of duplicates. If they are also in the set of possible targeted domains from section 5.2, it is written to a file and put in the dictionary to signify we have seen this particular domain.[18]

With this new set of domains, we can start querying their DNS records. As mentioned before, Harm Griffioen provided a bash script that can do this for us. The script run.sh[19] makes a directory to house all the outputs of the queries. It then takes the input file with the domains and reads every line. For each line or domain, it creates an output file in the output directory to put the query output of that specific domain. Then the script createcommand.sh[20] is called with the domain as the input, and the result is stored in the aforementioned output file. Before starting the next iteration the program sleeps for two seconds as not to overburden the RiskIQ servers.

The script createcommand.sh takes the curl command from curlcommand.txt.[21] The curl command can be retrieved from the RiskIQ Passivetotal dashboard. When you search for the IP resolutions of a domain, the curl command can be obtained by going to the network tab of the developer tools (Ctrl + Shift + I or F12 in Chrome or Firefox). We need the resolutions query with the property "includeFacets" set to true, otherwise, you will only get the number of IP resolutions instead of the actual results. In the curl com-

[18] DomainsOfCertsToIncidents(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/processCertificates.py
[19] run.sh, https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/run.sh
[20] createcommand.sh, https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/createcommand.sh
[21] curlcommand.txt, https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/curlcommand.txt

mand, a few things are noteworthy: "query=google.com" is constantly replaced by the domain we want to query, "start=" and "end=" are used to specify the period for which we want to query. "pageSize" is set to a large number so we make sure all available results are shown in the response. We also include a retry and delay option as a fail-safe, if the query fails. Createcommand.sh then evaluates the command, and thus performs the query, and stores the result as mentioned earlier.

The program walks through the current directory and iterates over all the files. If a filename ends with ".com" we want to evaluate it. First, we create a dictionary to store the set of IP prefixes for the current domain. We open the file, iterate over the lines and try to retrieve the IP resolutions. This is done with a try and except block because sometimes a domain will not have any resolutions. Then we iterate over the IPs in the resolutions and retrieve the IP objects and the IP prefix from there. If the prefix is not in the previously mentioned dictionary, and the prefix is not "None" then we store it. After this has been done, the method prefixDistance is called with the dictionary and the filename, to determine the prefix distance measurements.[22]

In prefixDistance, we retrieve the prefixes from the dictionary as a list. Then by using the combinations method from the itertools package, we can determine all the possible pairs of those prefixes. We also create the dictionary averageDistDic to store the average distance to all other prefixes, for a given prefix. For each pair in the list of pairs, retrieve both prefixes. If those are not yet present in the dictionary, we put them in there. The prefixes are split on the forward-slash that designates the prefix length, this length is then stored. From those, we pick the shortest, least specific length. We create an IPNetwork object from the prefixes to easier manipulate it. The IP address is taken from it, converted to bits where we remove the dots from the representation. Lastly, we only care about the least specific length that we found earlier. So if we have a /24 and /22 prefix we only look at the first 22 bits of both. The binary string is then converted to a binary integer, this is needed because of the XOR operation that is used next. Both binary integers are XOR'd and converted to binary again where we ignore the binary designation 0b. We then prepend the result with zeroes until we have the same length as the input binary strings. We do this because we are interested in the number of leading zeroes which indicates how many bits of the two prefixes are the same. The resulting string is then split on the first occurrence of a 1, and we pick the first section which contains the leading zeroes. With a theoretical address size of 32 bits, the distance is then (32 - number of leading zeroes) * 1/32. This distance is then averaged over the total amount of distances the prefix has and is added to the prefix entries in the averageDistDic.

After we have the average distances of all the prefixes of a particular domain, we pick the maximum average distance and identify the accompanying prefix (key). This key and value pair is put in a dictionary called sub result, which in turn is put in a dictionary called finalresult where the key is the corresponding domain / filename. This is done for the IP resolutions of every domain, so eventually finalresult will contain every domain,

[22] riskIQ(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/cidrfrequency.py

along with the prefix and the maximum average distance value it has.[23]

5.5. COMPARISON OF THE IP ADDRESSES AND AS NUMBERS BEFORE AND DURING THE NETWORK INCIDENT

As explained in the previous chapter, the IP addresses and AS numbers for a particular domain are compared before and during the BGP hijack. For this, a similar curl command is executed to get the output for the input domains. In this case, it are the actual DNS records that are being retrieved.[24] The program walks through the current directory and iterates over all the files. If a filename ends with ".com" we want to evaluate it. We open the file, the line is read and converted to a python dictionary and then the results object is retrieved. Each record in the results object is iterated and checked if the recordType or resolveType is equal to A, meaning only DNS A records are looked at. Then the resolved IP is collected from the record. The input domain will be the key in the dictionary and the IP will be added to the value which is an empty set after initialization. Afterwards, new IP's for the same domain are added to this same set. Based on the directory of the domain file, a different intermediary dictionary is used. One for the results before the incident (dictionary zero), one for the results during the incident (dictionary one).[25]

This same process is done for the AS numbers, except that the IP addresses are looked up in an IP-to-AS database, and the AS number will be stored in the set instead.[26] To create this database we used the pyasn Python library by Hadi Asghari and Arman Noroozian and the input for this is routing information base data which was downloaded from Ripe. Because we are comparing the AS numbers before and during the network incident we need the data from before and during the incident. The RIB data from all the available collectors was downloaded. The downloaded archives are of the .gz format and the library only accepts .bz2 archives, this was easily done with a renaming script (picture). Then all the .bz2 archives were converted using pyasn_util_convert.py from the library, but within a script to convert everything at once (picture). The resulting .dat files were all written to a .txt file using the cat command and '>' operator, resulting in a database file for before the incident and one for during the incident.

So, now we have two dictionaries containing domain names and sets of IP addresses, and two dictionaries containing domain names and sets of AS numbers. The following process is identical for both pairs of dictionaries. Every domain in dictionary one is iterated, the value is retrieved, which is the set of IP addresses. If that same domain is in dictionary zero, that set of IPs is also retrieved. The difference between set one and set zero is determined to find any new IP addresses that are present. If there are, this new set is stored with the domain as a key-value pair in a result dictionary. This dictionary is later written to a file.

[23] prefixDistance(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/cidrfrequency.py
[24] curlcommandRecords.txt, https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/curlcommandRecords.txt
[25] compareRecords(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/CompareDNSRecords.py
[26] compareASN(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/CompareDNSRecords.py

5.6. COMBINING THE DIFFERENT DATA POINTS

Now, with these various indicators, it is time to combine them for each domain name. An 'indicator' dictionary is initialized where the default value is another dictionary. This entails that if a key (domain) is not yet in the dictionary, it will be put in there and its value will be another dictionary. In this nested dictionary, the keys are identifiers for the indicators and the values are the indicators themselves. This allows to add domains, the identifiers and the indicators with less convoluted code.

The files containing the indicator results are opened. The line containing the JSON is read and then converted to a Python dictionary. Each domain is iterated, and the value is retrieved (set of IPs, set of AS's, prefix distance data) and put in the dictionary where the key is the correct identifier.

However, the certificate comparison data is a little bit different since the comparisons were done based on the common names. The file containing the data is opened, the line is read and the JSON is turned into a Python dictionary again. Each common name is iterated, certificate comparisons are retrieved and iterated over as well. Then we go through the domain names in the parsed names object and if a domain is found to be from the Route53 domain set (from the initial possibly targeted domain set), the certificate comparison data we are currently at is added to the indicator dictionary.[27]

5.7. BGP DATA ANALYSIS

To say something about the propagation of the BGP incidents and the hijacked prefixes, we decided to look at the BGP collectors that receive hijacked prefix messages of the incidents and the peer AS's that provided the collectors with them. We used pybgpstream[28] which is a Python binding for libbgpstream to retrieve BGP messages that satisfy a specified filter. We retrieved the hijacked messages of the BGP incidents by using the BGPStream method of this library. The method parameters from_time and until_time determine the time period for which messages are retrieved, these are chosen according to the date and time of the BGP hijack. The projects parameter allows one to specify the data source which will be both Route Views and RIS RIPE. What type of records are retrieved is determined by record_type, in our case BGP update messages. Lastly, the filter is specified to only collect messages from the correct origin AS and with the hijacked prefixes.

Each record and element of that record is iterated. The prefix, the collector that received the message as well as the peer AS that forwarded the message are collected. The prefix is added to a dictionary and the value contains a list of the collector, which will be expanded as more messages are processed. Another dictionary stores the prefix along with a list of the peer AS, which is also expanded the same way. Each prefix is iterated, the length of the collector list is determined and appended to the list. Both lists are then sorted mainly for readability. Lastly, the dictionaries are written to separate files.[29]

[27] method(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/Combine.py
[28] pybgpstream, https://github.com/caida/pybgpstream
[29] pybgpstream, https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/prefixinmrt.py

Now we have the prefixes along with the BGP collectors that received the hijack messages and the prefixes along the peer AS's that provided collectors with the hijack messages.

The input files are defined first, these contain the prefixes and the collectors that received a BGP hijack message for those prefixes. We also initialize a counter for each internet exchange. Each line of the input files is iterated, which contains the JSON string of the dictionary, and is then loaded into a Python dictionary. Each prefix is iterated and the list of collectors is retrieved. The collectors are grouped by their respective internet exchanges by using if statements. If a collector of a particular internet exchange is found within the list of collectors, the counter for that internet exchange is incremented. Afterwards, we will have determined the number of unique prefixes that each internet exchange has observed.[30]

We define the input files first again, this time they contain the prefixes and the peer AS's that propagated a BGP hijack message to a collector. A default dictionary is initialized too with the default value being used 0. Each line of the input files is iterated, which contains the JSON string of the dictionary as before, and is then loaded into a Python dictionary. Each prefix in the dictionary is iterated and the list of peer AS's is retrieved. Now we iterate over each peer AS, retrieve its value from our default dictionary and increment the counter by one. If it is new, the default dictionary will store the peer AS with the default value of 0 before incrementing it. When every file and each prefix has been processed, we will have the number of unique prefixes propagated to collectors per peer AS.

X coordinates are defined depending on the number of peer AS's in the dictionary. Two lists are also defined for the Y coordinates and the labels on the X axis. Each peer AS is then iterated and the AS number is appended to the list of labels, the value of the peer AS (number of unique prefixes) is appended to the Y coordinate list. A bar graph is then plotted with those X coordinates, Y coordinates, labels and alternating colors.[31]

If an AS originates a particular prefix, to a set of peer AS's, the more peer AS's this set contains, the further the message should propagate. Thus, an IX (or the collectors on that IX) which connects to a lot of peer AS's should receive more of these prefix messages. If an IX processes a lot of traffic, I'd expect the collectors to receive more prefix messages due to the sheer volume. These two characteristics are used to average the data to make a fairer comparison.

[30] uniquePrefixes(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/processBGPsensors.py
[31] PeerASNDiagrams2(), https://github.com/Nilsbijlsma/BGPThreatIntelligence/blob/main/processBGPsensors.py

6

RESULTS

In this chapter the results will be discussed. First, we reiterate the research goals. Section 2 will go over the results of the BGP prefix hijack impact analysis. The third section will go into detail about the propagation of hijacks to BGP route collectors.

6.1. RESEARCH GOALS

BGP was designed during the early days of the internet when only a few select institutions were connected, like the government and academia. BGP assumes implicit trust between the route-exchanging network points. Since then, internet connectivity has become almost essential in daily life and this huge influx of internet users made it so this implicit trust started to get abused by malicious actors, resulting in BGP hijacks to redirect network traffic.

There are proposed solutions to help prevent BGP hijacks, but they still do occur since most of these solutions are not implemented on a wide scale for various reasons. In terms of threat intelligence, it is not always entirely clear what the impact is of such networking incidents, beyond the redirection of network traffic.

The developed framework aids in analysing BGP and DNS hijacking events and can produce a list of domains with indicators of compromise to lift the curtain in terms of the impact and scope of such incidents.

6.2. INDICATOR RESULTS

We will be looking at the results of each BGP hijack separately. The Route 53 AWS incident and the Payment System Provider incident. For each domain we detail what nameserver is responsible, the IP address of the nameserver, which prefix hijack it was a part of, the category and which indicators were found. Each domain name contains up to four indicators of compromise: unique IP addresses resolved during the incident in comparison with the time before the incident, unique autonomous system numbers resolved

during the incident in comparison with the time before the incident, a distance measurement of the most outlying IP prefix and the compared SSL certificates that presented a red flag.

6.2.1. ROUTE 53 HIJACK

For the Route 53 AWS incident, the number of domains within the scope of the hijack is 1.010.804. The number of domains we found indicators of compromise for is 3.668. This was the result of 5 BGP prefix hijacks. We also looked at how many domains have 1, 2, 3 or 4 indicators. This is 2.702, 928, 35 and 3 domains respectively and is shown in figure 6.1. Most domains have either 1 or 2 indicators, it is logical that domains with more indicators appear less often.

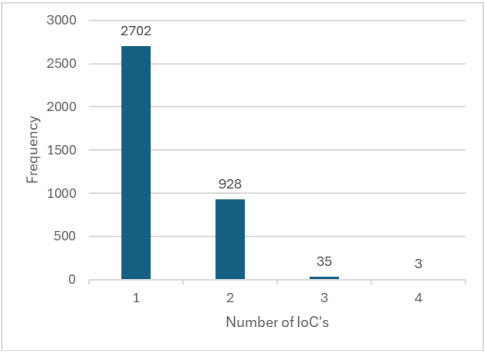

Figure 6.1: Number of domains that have a certain number of indicators.

We also took a look at the occurrence of each indicator itself. 69 domains have the new IP address indicator, 61 have the new AS number indicator, 2260 have the prefix distance indicator and 2285 domains have the certificate comparison indicator. The indicator that was found most frequently is the certificate indicator. Figure 6.2 shows the occurrences of the IP and AS indicators, figure 6.3 shows the same for the prefix distance and certificate indicator.

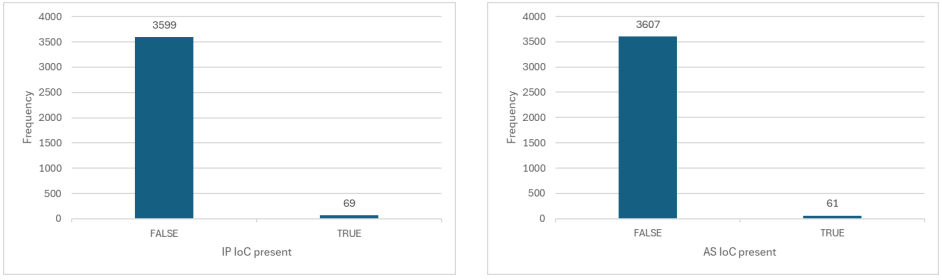

Figure 6.2: Left: Number of domains having the IP indicator. Right: Number of domains having the AS indicator.

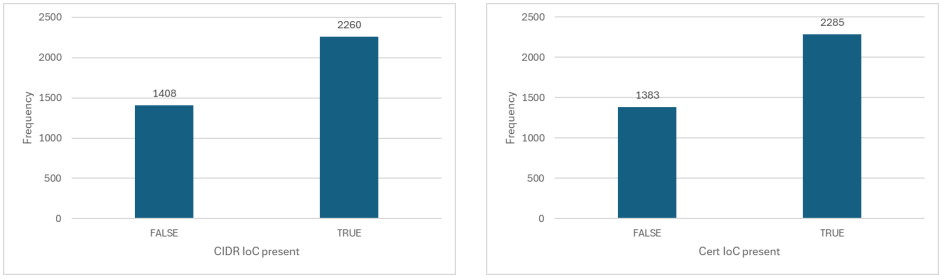

Figure 6.3: Left: Number of domains having the CIDR indicator. Right: Number of domains having the certificate indicator.

The categories for this incident are a lot more widespread. This is most likely due to the fact the Route 53 BGP hijack affected a larger set of DNS servers. This results that more domains are in the scope of the BGP hijack. The categories that were seen the most were: technology, travel, sports, medical, financial, education, automotive, law, marketing, e-commerce, restaurant, gaming and cryptocurrency. It is suspected that this BGP hijack had a very specific target, MyEtherWallet, and that the other redirected domains were simply collateral. The distribution of categories is shown in figure 6.4, this includes clearly defined categories that are the most prevalent. It does not include each and every domain and some domains could not be categorized.

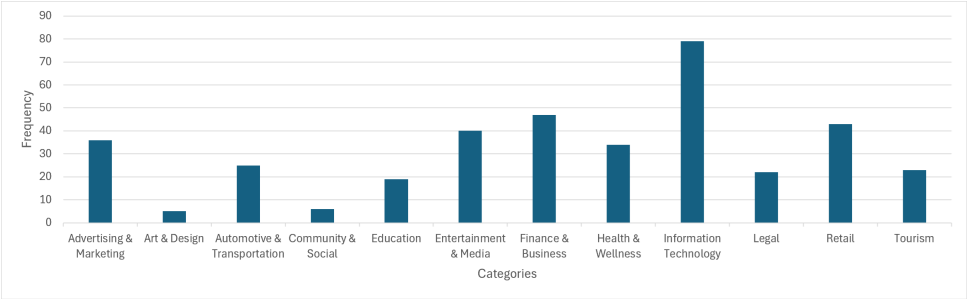

Figure 6.4: Number of domains that belong to a particular category.

6.2.2. PAYMENT SYSTEM PROVIDER HIJACK

For the Payment System Provider incident, there were significantly less domains potentially affected: 24.945. The same is true for the number of domains with indicators of compromise, which was just 71. This was the result of 6 prefix hijacks. We found 28, 42 and 1 domain with 1, 2 and 3 indicators respectively. There were no domains with 4 indicators, as can be seen in figure 6.5.

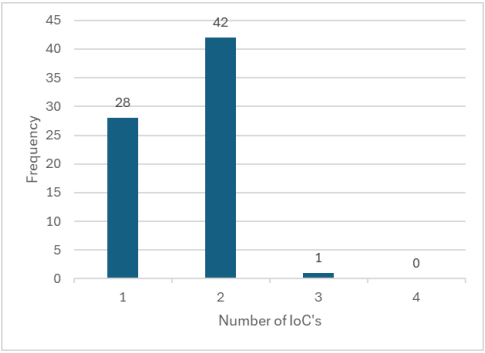

Figure 6.5: Number of domains that have a certain number of indicators.

For this particular incident, 60 domains have the new IP address indicator, 7 have the new AS number indicator, 4 have the prefix distance indicator and 44 domains have the certificate comparison indicator. The certificate indicator occurred quite often, just like the Route 53 incident. Interestingly, the IP indicator was the most frequent of the 4. Figure 6.6 and 6.7 show the frequency of each indicator.

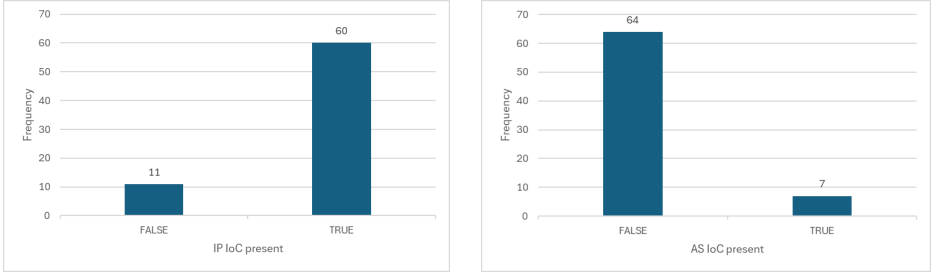

Figure 6.6: Left: Number of domains having the IP indicator. Right: Number of domains having the AS indicator.

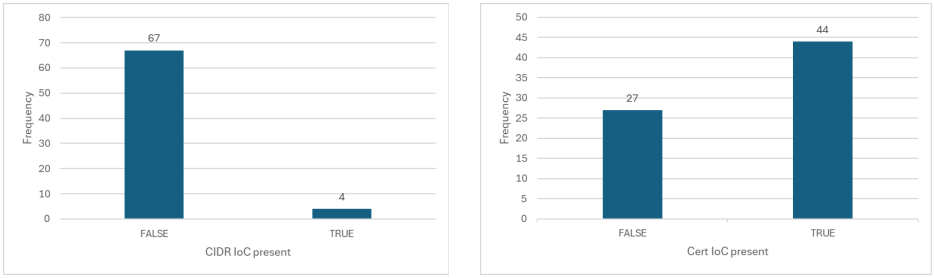

Figure 6.7: Left: Number of domains having the CIDR indicator. Right: Number of domains having the certificate indicator.

Almost all domains are related to the automotive industry, mainly dealerships. One domain was related to the medical field. Another domain could not be categorized. When looking into it, it seems to be redirecting to the Salesforce website which provides enterprise software. It is surprising that almost all domains are related to the same category and most of them are routed to new IP addresses during the incident. Given the fact they are resolved by the same nameserver with a second nameserver as backup, it is likely that they belong to the same organization or registrant. This could not be verified however because the whois information of the domains is anonymous. It is possible that the new IP addresses for these domains was an intentional routing change. Figure 6.8 details how often each category occurred.

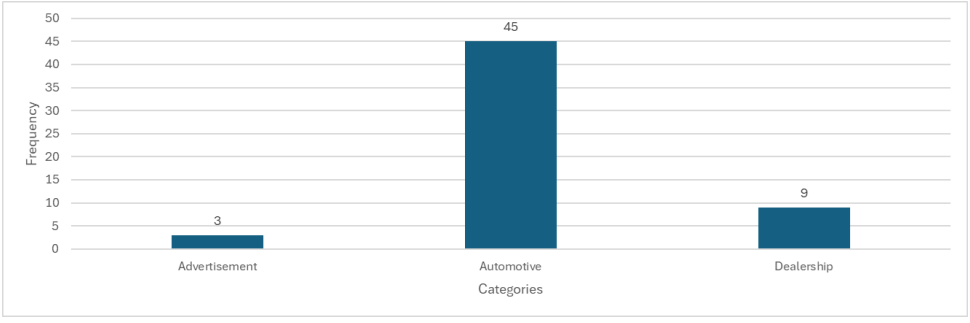

Figure 6.8: Number of domains that belong to a particular category.

BGP data is used to identify the hijacked IP prefixes. The hijacked IP prefixes are used in conjunction with the DNS zone file to identify the DNS servers within the scope of the BGP hijack. With the DNS servers and the DNS zone file, we identify what domain names they are authoritative for. The SSL certificates are retrieved for those domains. We compare certificates and identify pairs that present certain red flags. From this we gather the domains to work with a smaller, filtered set in the next steps. These domains names are used to retrieve IP prefix data, a distance measurement is calculated to identify outliers. We also retrieve the A records for the domains and determine the newly occurring IP addresses during the incident. The IP addresses are also combined with IP to AS data to find the newly occurring AS numbers the domain is routed from.

Our **research question** was: *Can we determine what domains are likely to have been targeted in a BGP hijack, based on the IP prefixes that have been rerouted during that hijack?* Based on the previously described results that followed from the methodology, we can indeed determine what domains might have been targeted in a BGP hijack, based on the hijacked IP prefixes and the indicators that were found. An example of such a domain and its indicators can be found in figure 6.9.

The idea is that a domain with all four indicators present is more likely to have been redirected as a result of the hijack. The problem with this, however, is that it cannot be verified properly. Certificates can be issued and signed with a different key pair, for example when key rotation is used. It can also be a coincidence that a domain is routed to a different IP address, within another autonomous system. It is entirely possible for hosting information to change and have nothing to do with the hijack. Lastly, even if we assume this is not the case, it is still difficult to make a statement about the redirection of such a domain. It is practically impossible to track down the owner of each domain and verify the findings.


```
first-responder.com
{'IP': [u'67.207.93.81'], 'AS': [14061], 'PrefixDistance': {u'67.207.80.0/20': 0.9687499999999998}, 'CertComparison': [{u'currcn': [u'first-responder.com'], u'currgp': False, u'currgw': False, u'currab': False, u'prevnw': False, u'currss': False, u'prevnr': False, u'currca': False, u'prevnp': True, u'currgb': False, u'curraw': False, u'currgr': False, u'currap': False, u'currar': False, u'currnames': [u'first-responder.com', u'www.first-responder.com'], u'prevnb': False, u'prevar': False, u'currsub': u'CN=first-responder.com', u'prevap': True, u'prevnames': [u'first-responder.com', u'www.first-responder.com'], u'prevaw': False, u'prevgb': False, u'currnb': False, u'prevab': False, u'prevcn': [u'first-responder.com'], u'prevsub': u'CN=first-responder.com', u'prevgr': False, u'currnw': False, u'currnp': False, u'currstart': u'2018-04-24 05:20:28 UTC', u'currnr': False, u'prevgw': False, u'prevnw': False, u'currfp': u'11237e7d5fcc007a75a884a17953570546378c2313c0801ec44f8a1163ad6aaf', u'prevmp': True, u'prevmd5': u'686afd4c928ac755a8aa114eedf06de0', u'prevgp': True, u'prevmb': False, u'previss': u"C=US, O=Let's Encrypt, CN=Let's Encrypt Authority X3", u'currmd5': u'705c545470de32903c0b9a667a9a374e', u'likelihood': 0.971360052651, u'prevss': False, u'currmb': False, u'prevfp': u'6161cac3ea3e66d7003f9a4423dbf29385c0eb62e8d0c016a3ceb91fe2a1d6e3', u'curriss': u'CN=Fake LE Intermediate X1', u'prevca': False, u'currnp': False, u'currmr': False, u'prevstart': u'2018-04-24 05:17:21 UTC', u'currnw': False, u'prevnr': False}, {u'currcn': [u'first-responder.com'], u'currgp': True, u'currgw': False, u'currab': False, u'prevnw': False, u'currss': False, u'prevnr': False, u'currca': False, u'prevnp': False, u'currgb': False, u'curraw': False, u'currgr': False, u'currap': True, u'currar': False, u'currnames': [u'first-responder.com', u'www.first-responder.com'], u'prevnb': False, u'prevar': False, u'currsub': u'CN=first-responder.com', u'prevap': False, u'prevnames': [u'first-responder.com', u'www.first-responder.com'], u'prevaw': False, u'prevgb': False, u'currnb': False, u'prevab': False, u'prevcn': [u'first-responder.com'], u'prevsub': u'CN=first-responder.com', u'prevgr': False, u'currnw': False, u'currnp': True, u'currstart': u'2018-06-23 11:54:03 UTC', u'currnr': False, u'prevgw': False, u'prevnw': False, u'currfp': u'75d964cd3fb608dd42f75dc9b77061af85c7ee27a5d8eddc208495e8f62ffa4d', u'prevmp': False, u'prevmd5': u'705c545470de32903c0b9a667a9a374e', u'prevgp': False, u'prevmb': False, u'previss': u'CN=Fake LE Intermediate X1', u'currmd5': u'11a357b1a9e2ddbcb493378644fbd586', u'likelihood': 0.971360052651, u'prevss': False, u'currmb': False, u'prevfp': u'11237e7d5fcc007a75a884a17953570546378c2313c0801ec44f8a1163ad6aaf', u'curriss': u"C=US, O=Let's Encrypt, CN=Let's Encrypt Authority X3", u'prevca': False, u'currnp': True, u'currnr': False, u'prevstart': u'2018-04-24 05:20:28 UTC', u'currnw': False, u'prevnr': False}]}
```


Figure 6.9: An excerpt from the indicator results.

The table below shows an excerpt from the results, indicating for each domain what name server is authoritative for it, what BGP hijack it was a part of and what indicators are present.

Domain name	Name server	IP address	Hijacked prefix	Hijacking AS	Incident	Cert IoC	IP IoC	AS IoC
ganleyfiatbedford	ns1.dealer	204.74.108.248	204.74.108.0/24	38182	payment system provider	False	True	False
hartevw	ns1.dealer	204.74.108.248	204.74.108.0/24	38182	payment system provider	False	True	True
subaruofsterling	ns2.dealer	204.74.109.248	204.74.109.0/24	38182	payment system provider	True	True	False
site	dns01.salesforce	204.74.108.235	204.74.108.0/24	38182	payment system provider	True	False	False
site	dns02.salesforce	204.74.109.235	204.74.109.0/24	38182	payment system provider	True	False	False
familyautos	ns2.dealer	204.74.109.248	204.74.109.0/24	38182	payment system provider	False	True	False

6.3. BGP DATA RESULTS

To say something about the propagation of the BGP incidents and the hijacked prefixes, it was decided to take a look at the BGP data itself. We decided to look at the BGP collectors that receive hijacked IP prefix messages and the peer AS's that provided the collectors with them. The collectors were grouped by their respective internet exchanges and we determined the unique prefixes that were observed on each exchange, and the total amount of hijack messages that were observed on each exchange. This was normalized for the number of peers the collectors have. This information was retrieved from RIS RIPE's peer list[1] and the RouteViews collectors map[2]. The data was also normalized for the average Gbps of traffic on the internet exchanges which was retrieved from each respective exchange's website.

With the data grouped per exchange and normalized, we noticed a lot of inconsistency.
SFMIX has 14 unique prefixes over 17 peers, which is 0.82 unique prefixes per peer.
CIXP has 2 unique prefixes over 19 peers, which is 0.105 unique prefixes per peer.
Equinix Singapore has 23 unique prefixes over 70 peers, which is 0.33 unique prefixes per peer.
France-IX has 12 unique prefixes over 76 peers, which is 0.15 unique prefixes per peer.
IX Brazil Sao Paulo has 7 unique prefixes over 170 peers, which is 0.04 unique prefixes per peer.
This is quite dissimilar, and we suspect this is since it depends on what set of peers are providing the collectors with BGP messages. To provide reasoning for this suspicion, we looked at the peer AS's that propagated the hijacked prefix messages to the collectors.

I have looked at the peer AS's that propagated the most prefixes to the collectors: 38182, 553, 18106, 31424 and 47962. They have 100-2000 peers and 30.000 - 2.800.000 IP's. When looking at the peer AS's that propagate only a few prefixes to the collectors, they are 56730, 3252, 38001 and 50300. They have 3-900 peers and 7.000 - 1.000.000 IP's. This could mean that AS's with a smaller presence, might receive fewer hijack messages, and propagate fewer hijack messages to collectors. It could also be the case that they already had a more preferable route, or that they are not connected with a lot of collectors on the internet exchanges.

However, also a few of the smaller AS's (with 100-300 peers) such as 38182 and 31424 propagate a lot of prefixes to collectors. This could be explained by improper filtering. This is kind of hinted at by the following: the Taiwan Public DNS BGP hijack was performed by 268869 and this AS only has 4 peers. However, both of the hijacked prefixes were provided to the collectors by 200+ unique peer AS's. This indicates those 4 peers (and the peers of those) have mostly accepted the hijacked routes because of misconfiguration, which leads to a large pool of AS's providing the collectors with the hijacked prefixes. This same principle could hold for small AS's propagating a lot of prefixes to collectors due to misconfiguration. Size is not the only reason why an AS would propagate a lot or not at all. Its location in the network graph also influences how many messages

[1] https://www.ris.ripe.net/peerlist/all.shtml
[2] http://www.routeviews.org/routeviews/index.php/map/

it passes on. If it only has upstream providers it will only pass on hijacked messages it originates. If it is more situated in the middle with lots of peers it will receive and pass on a lot more messages.

This tells us that the question "it depends on what peers are providing the collectors with BGP messages" can be answered with: AS's with a larger presence on the internet, or AS's that don't filter very well (misconfigurations) will result in a higher propagation of prefixes to the collector. AS's with a smaller presence or AS's that filter their messages properly will result in lower propagation of prefixes to the collector.

The set of peers, that provides the collectors with BGP messages, thus influences the number of unique prefixes measured (and causes the inconsistency) that can be seen at the various exchanges.

7

CONCLUSION

In this concluding chapter, we will present the conclusions of this research, discuss some limitations of the proposed methods and possibilities for future work.

7.1. SUMMARY

The border gateway protocol is used to exchange routing information between routers in autonomous systems. Network routes are advertised to each other and the routes are stored in routing tables. The routing information includes the destination, the network path to said destination, as well as additional information to be used in routing policies. This system essentially allows the internet to work, by making data routing from point A to point B possible. During this research project, we have focused on BGP hijacks and their possible impact.

A problem with this system is that it assumes the participating BGP routers can be trusted: there are almost no security mechanisms present. Nothing is stopping a malicious actor from advertising routes to IP prefixes they do not own and altering routing paths, with the intention to redirect the network traffic to those IP addresses. Such an announcement is then installed in the routing tables of receiving routers and subsequently forwarded to other BGP peers. So by abusing the implicit trust system of BGP, one can perform a BGP hijack to take control of an IP prefix.

Various technologies have been proposed and developed as countermeasures. Namely DNSSEC and RPKI. DNSSEC allows DNS providers to digitally sign the DNS data itself, whose authenticity can then be verified by DNS resolvers. This prevents rogue DNS servers from giving out forged DNS responses, if DNS traffic would get redirected to them. RPKI is a public key infrastructure that connects internet resource information such as IP addresses to a trust anchor, similarly to how the certificate system works. It allows owners of IP address space to authorize autonomous systems to originate routes to that address space. This route origin authorization can be verified with the accompanying public key, and thus prevent BGP route hijacking because those routes would not

be accepted by BGP routers. The issue with these preventative solutions is that they depend on their adoption rate, they need to be widely implemented to make a widespread impact.

There are detection methods that aim to detect BGP hijacks, for example by using a database of autonomous systems that originate certain IP prefixes, which is constructed over time. This is similar to the RPKI, but without the centralized infrastructure and is detective rather than preventative. It essentially detects when an IP prefix is originated by a different AS than usual, but it can only detect when a hijack takes place. The scope of the impact of such a hijack is not considered.

Regarding these issues about prevention and detection, we established the following research question:

- Can a methodology be developed to determine what domains are likely to have been targeted in a BGP hijack, based on the IP prefixes that have been rerouted during the hijack?
- What features are feasible to be used as an indicator of compromise in this methodology?

Various BGP hijack reports were used to identify BGP hijacking events, as a basis to start from. The BGP data was collected from RIPE and processed to obtain the hijacked prefixes. The prefixes were used to determine what DNS servers were in the scope of the hijack, this was done with DNS zone files provided by the university. Using the DNS servers and the same zone files, the NS records were used to retrieve the domain names. The indicators of compromise consist of SSL certificates, a distance measurement between IP prefixes a domain resolved to, IP address resolutions and AS numbers compared before and during the BGP hijack. The domains were filtered first based on the certificate indicator before the other indicators were retrieved.

The following results are obtained:

- It is possible to determine what domains might have been targeted in a BGP hijack, by using the IP prefixes that were announced during the hijack.
- The four indicators of compromise that were ultimately found and used are unique IP addresses resolved during the incident in comparison with the time before the incident, unique autonomous system numbers resolved during the incident in comparison with the time before the incident, a distance measurement of the most outlying IP prefix and the compared SSL certificates.
- A set of domain names that contain up to four of the previously mentioned indicators of compromise, detailing what could indicate malicious activity.

We believe this framework serves as a tool to be used for threat intelligence. It uses the basic ideas of BGP hijack detection to look deeper into the impact of such incidents. Instead of only looking at the hijacked prefixes themselves, the DNS servers and domains

that are within the scope of the hijack are also considered along with what indicators they might present. If a BGP hijack takes place, it often takes multiple hours to get resolved. Afterwards, the incident is analysed and it is determined what happened. One of the reasons for this long winded process is that only a relatively small part of the internet is usually affected, so it takes a while before things get noticed. This framework could also be used as a foundation for a detection and response system, such that domains that might be targeted and redirected during such a hijack can act in time.

7.2. LIMITATIONS

One data source problem that would become more apparent in a real-time detection and response system, concerns the BGP sensors. As stated earlier, only around 15% of RIPE's BGP sensors detected the prefix hijacks from the Route 53 incident. This could mean only 15% of the internet was affected, but it could also mean that RIPE's monitoring visibility is sparse. This visibility issue also extends to any data provider that might be used for indicator data, because they might not have visibility on all parts of the internet. As an example, malicious SSL certificates were found for a domain that was redirected during the BGP hijack. However, the IP addresses and autonomous system numbers where the redirected domain was hosted, were not found in the data.

It was not feasible to use certain data providers, such as Securitytrails and DNSDB for historical IP addresses, simply because of the large amount of queries and the associated cost. This issue is also present to a lesser extent at other data providers. This meant it was not feasible to retrieve the prefix, IP and AS data for all the domains within the hijacks. This is why we chose to work with the intermediate result from the certificate comparisons. In reality, there are probably more domains with the prefix distance, new IP address and new AS number indicators.

When looking at the indicators themselves, there are some limitations there as well. The used certificate data from Censys does include an attribute to indicate if a certificate is revoked or not. However, there were no certificates designated as revoked according to this attribute. Revoked certificates would add a layer of feedback or certainty to the certificates we labelled as malicious. Another point is the IP prefixes where a domain was routed from and their distances to each other, to identify outliers. The distances between prefixes are simply calculated by how many bits are common and how many are different. This does not tell a complete story, as the same Autonomous System can have vastly different IP ranges under the same administrative umbrella.

Evaluating the effectiveness of the proposed solution is difficult. We have discussed looking at false positives and false negatives to determine how accurate the solution is. The main issue is that to determine false positives and false negatives, a labelled data set is required. For our purpose, domains would be labelled as redirected or not. If a domain is not in the results but should be according to its label, it is a false negative. If a domain is in the results but should not be, it is a false positive. However, it simply is not possible to label with certainty which domains were redirected and which were not. Contacting

web administrators is not an option due to the sheer amount of domains, even if we assume they would cooperate.

We have discussed creating a labelled data set ourselves but ultimately decided against it. This labelled data set would have consisted of three parts: the domains, the associated indicators and the labels themselves as described previously. However, if we would decide a certain domain was redirected, the indicators would need to reflect that. The same holds for creating the indicators first, the label of the domain needs to make sense with respect to those indicators. This means there is a 1-to-1 relationship of the indicators and the label, whereas the indicator analysis and redirection verdict is meant to be evaluated. Another problem is that such a created data set would not be an accurate representation of real-world data.

7.3. FUTURE WORK

To alleviate some of the issues surrounding the visibility of prefix hijacks, the BGP sensors of Route Views could be included besides RIPE. Another option would be to construct one's own infrastructure in a wide variety of locations to increase the monitoring visibility of such a hypothetical system. As mentioned in the previous section, indicator data was sometimes not found in the data sources that were used. A possible solution for this could be using various data sources for the same type of data. If one data source does not contain something, another might. A similar option as before is to keep track of historical data yourself in something like a database, such as IP addresses and AS numbers, instead of relying on third party organizations to provide the data. This would also need to be deployed on a wide variety of servers to increase the overall visibility. The problem with data providers' query limits and their cost, would also be reduced by these two methods.

Incorporating something similar to the certificate transparency system could be useful, to give the indicator more context. To more accurately predict outlying IP prefixes, IP addresses and Autonomous Systems, some type of clustering could be applied to the data. By grouping similar data points and identifying patterns, it would be easier to determine those outliers.

Evaluating such a framework is difficult in terms of quantifiable metrics. One way to do this is with a network simulation, where BGP hijacks can be simulated leading to redirected network traffic. Browser traffic from clients could also be simulated, to simulate DNS traffic and the redirection of it. DNS servers would need to be included to reply to the redirected DNS traffic. Domains could be simulated along with their IP addresses and AS numbers and keep track of the data before and after the hijack. Some domains could have malicious certificates just like in the real-world, in an attempt to trick users. With such a network simulation, one could compare what should happen with what actually happens in the testbed.

BIBLIOGRAPHY

[1] Juniper Networks. *BGP Overview*. 2021. URL: https://www.juniper.net/documentation/us/en/software/junos/bgp/topics/topic-map/bgp-overview.html#id-understanding-bgp.

[2] Noction Network Intelligence. *BGP Hijacking overview. Routing incidents prevention and defense mechanisms*. 2018. URL: https://www.noction.com/blog/bgp-hijacking.

[3] Andrei Robachevsky. *14,000 incidents - routing security in 2017*. APNIC. 2018. URL: https://blog.apnic.net/2018/01/24/14000-incidents-routing-security-2017/.

[4] John Hawkinson and Tony Bates. *Guidelines for creation, selection, and registration of an Autonomous System (AS)*. RFC editor. 1996. URL: https://datatracker.ietf.org/doc/html/rfc1930.

[5] Matt Lepinski and Stephen Kent. *An Infrastructure to Support Secure Internet Routing*. RFC editor. 2012. URL: https://datatracker.ietf.org/doc/html/rfc6480.

[6] Scott Rose et al. *DNS Security Introduction and Requirements*. RFC editor. 2005. URL: https://datatracker.ietf.org/doc/html/rfc4033.

[7] Pavlos Sermpezis et al. "ARTEMIS: Neutralizing BGP Hijacking Within a Minute". In: *IEEE/ACM Transactions on Networking*. Vol. 26. 2018, pp. 2471–2486. DOI: 10.1109/TNET.2018.2869798.

[8] Doug Madory. *BGP Hijack of Amazon DNS to Steal Crypto Currency*. Oracle Dyn. 2018. URL: https://medium.com/oracledevs/bgp-hijack-of-amazon-dns-to-steal-crypto-currency-a90dd29cb3ab.

[9] Uli Bornhauser and Peter Martini. "About prefix hijacking in the Internet". In: *IEEE Conference on Local Computer Networks*. 2011, pp. 143–146. DOI: 10.1109/LCN.2011.6115172.

[10] Dan Goodin. *Google goes down after major BGP mishap routes traffic through China*. Ars Technica. 2018. URL: https://arstechnica.com/information-technology/2018/11/major-bgp-mishap-takes-down-google-as-traffic-improperly-travels-to-china/.

[11] Ameet Naik. *Internet Vulnerability Takes Down Google*. ThousandEyes. 2018. URL: https://www.thousandeyes.com/blog/internet-vulnerability-takes-down-google.

[12] Jeremy Kirk. *Who Hijacked Google's Web Traffic?* BankInfoSecurity. 2018. URL: https://www.bankinfosecurity.com/who-hijacked-googles-web-traffic-a-11699.

[13] Doug Madory. *Recent Routing Incidents: Using BGP to Hijack DNS and more*. Lacnic. 2018. URL: https://www.lacnic.net/innovaportal/file/3207/1/dougmadory_lacnic_30_rosario.pdf.

[14] Shaun Nichols. *AWS DNS network hijack turns MyEtherWallet into ThievesEtherWallet*. The Register. 2018. URL: https://www.theregister.com/2018/04/24/myetherwallet_dns_hijack/.

[15] Aftab Siddiqui. *What Happened? The Amazon Route 53 BGP Hijack to Take Over Ethereum Cryptocurrency Wallets*. Internet Society. 2018. URL: https://www.internetsociety.org/blog/2018/04/amazons-route-53-bgp-hijack/.

[16] URL: https://blog.cloudflare.com/bgp-leaks-and-crypto-currencies/.

[17] Doug Madory. *BGP/DNS Hijacks Target Payment Systems*. Oracle Dyn. 2018. URL: https://web.archive.org/web/20190611040109/https://dyn.com/blog/bgp-dns-hijacks-target-payment-systems/.

[18] Aftab Siddiqui. *Public DNS in Taiwan the latest victim to BGP hijack*. MANRS. 2019. URL: https://www.manrs.org/2019/05/public-dns-in-taiwan-the-latest-victim-to-bgp-hijack/.

[19] RIPE NCC. *Route Collection Raw Data: MRT Files*. 2017. URL: https://ris.ripe.net/docs/mrt/.

[20] University of Oregon. *University of Oregon RouteViews Project*. 2017. URL: http://www.routeviews.org/routeviews/.

[21] Colorado State University. *BGPmon*. 2017. URL: https://www.bgpmon.io/.

[22] Censys. *Certificates Google BigQuery Dataset*. 2017. URL: https://search.censys.io/#.

[23] Censys. *Certificates Google BigQuery Dataset*. 2017. URL: https://support.censys.io/hc/en-us/articles/360056772812-Introduction-to-Certificates.

[24] Censys. *Certificates Google BigQuery Dataset*. 2017. URL: https://support.censys.io/hc/en-us/articles/360038761891-Research-Access-to-Censys-Data.

[25] SecurityTrails. *DNS records*. 2017. URL: https://securitytrails.com/.

[26] Rapid7. *Forward DNS*. 2017. URL: https://opendata.rapid7.com/sonar.fdns_v2/.

[27] Farsight Security. *DNSDB*. 2017. URL: https://www.farsightsecurity.com/solutions/dnsdb/.

[28] RiskIQ. *Passivetotal*. 2017. URL: https://community.riskiq.com/.

[29] Google. *Google BigQuery*. 2017. URL: https://cloud.google.com/bigquery/.

[30] Kenneth Reitz. *Requests: HTTP for Humans*. Version 2.25.1. Dec. 16, 2020. URL: https://requests.readthedocs.io/en/latest/.

[31] Leonard Richardson. *Beautiful Soup*. Version 4.11.1. Apr. 8, 2022. URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/.

[32] RIPE NCC. *libBGPdump*. Version 1.6.2. June 7, 2020. URL: https://github.com/RIPE-NCC/bgpdump.

[33] Majestic. *The Majestic Million*. 2017. URL: https://majestic.com/reports/majestic-million.

[34] Xin Hu and Z. Morley Mao. "Accurate Real-time Identification of IP Prefix Hijacking". In: *IEEE Symposium on Security and Privacy*. 2007, pp. 3–17. DOI: 10.1109/SP.2007.7.

[35] Jian Qiu et al. "Detecting bogus BGP route information: Going beyond prefix hijacking". In: *IEEE International Conference on Security and Privacy for Emerging Areas in Communications Networks*. 2007, pp. 381–390. DOI: 10.1109/SECCOM.2007.4550358.

[36] Mathew J. Schwartz. *Criminals, Nation-States Keep Hijacking BGP and DNS*. Bankinfosecurity. 2019. URL: https://www.bankinfosecurity.com/hijacking-bgp-dns-persists-despite-available-fixes-a-12028.

[37] Jinjing Zhao et al. "The Relation on Prefix Hijacking and the Internet Hierarchy". In: *IEEE International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. 2012, pp. 415–420. DOI: 10.1109/IMIS.2012.40.