Cloud Strife: Mitigating the Security Risks of Domain-Validated Certificates

Borgolte, Kevin; Fiebig, Tobias; Hao, Shuang; Kruegel, Christopher; Vigna, Giovanni

DOI
10.14722/ndss.2018.23327

Publication date
2018

Document Version
Final published version

Published in
proceedings of Network and Distributed System Security Symposium (NDSS)

Cloud Strife: Mitigating the Security Risks of Domain-Validated Certificates

Kevin Borgolte
UC Santa Barbara
kevinbo@cs.ucsb.edu

Tobias Fiebig
TU Delft
t.fiebig@tudelft.nl

Shuang Hao
UT Dallas
shao@utdallas.edu

Christopher Kruegel
UC Santa Barbara
chris@cs.ucsb.edu

Giovanni Vigna
UC Santa Barbara
vigna@cs.ucsb.edu

***Abstract*—Infrastructure-as-a-Service (IaaS), and more generally the "cloud," like Amazon Web Services (AWS) or Microsoft Azure, have changed the landscape of system operations on the Internet. Their elasticity allows operators to rapidly allocate and use resources as needed, from virtual machines, to storage, to bandwidth, and even to IP addresses, which is what made them popular and spurred innovation.**

In this paper, we show that the dynamic component paired with recent developments in trust-based ecosystems (e.g., SSL certificates) creates so far unknown attack vectors. Specifically, we discover a substantial number of stale DNS records that point to available IP addresses in clouds, yet, are still actively attempted to be accessed. Often, these records belong to discontinued services that were previously hosted in the cloud. We demonstrate that it is practical, and time and cost efficient for attackers to allocate IP addresses to which stale DNS records point. Considering the ubiquity of domain validation in trust ecosystems, like SSL certificates, an attacker can impersonate the service using a valid certificate trusted by all major operating systems and browsers. The attacker can then also exploit residual trust in the domain name for phishing, receiving and sending emails, or possibly distribute code to clients that load remote code from the domain (e.g., loading of native code by mobile apps, or JavaScript libraries by websites).

Even worse, an aggressive attacker could execute the attack in less than 70 seconds, well below common time-to-live (TTL) for DNS records. In turn, it means an attacker could exploit normal service migrations in the cloud to obtain a valid SSL certificate for domains owned and managed by others, and, worse, that she might not actually be bound by DNS records being (temporarily) stale, but that she can exploit caching instead.

We introduce a new authentication method for trust-based domain validation that mitigates staleness issues without incurring additional certificate requester effort by incorporating existing trust of a name into the validation process. Furthermore, we provide recommendations for domain name owners and cloud operators to reduce their and their clients' exposure to DNS staleness issues and the resulting domain takeover attacks.

I. Introduction

Over the past ten years, cloud services have grown tremendously. Generally, clouds are comprised of hundreds to thousands of commodity servers, which make up pools of computing resources that are shared by different users. One of the main drivers behind the clouds' rise in popularity is their elasticity: users can acquire and use resources as needed, on demand, and at scale, all while requiring almost no upfront investment. In fact, Amazon Web Services (AWS), Amazon's public cloud, serves over one million active users worldwide [1], Microsoft Azure is gaining 120,000 new customers each month [2], and the global cloud IP traffic has reached 3.9 zettabytes (3.9 billion terabytes) in 2015 already [3]. Unfortunately, as the recent years have shown, the resource pooling and increased popularity of cloud-based deployments also pose severe security issues to the clouds' tenants [4, 5].

With the clouds' increase in popularity and their commoditization, website operators have been empowered to deploy their website themselves instead of relying on more traditional web hosting. At the same time, HTTPS has become basically a requirement for any website operator, not only for dynamic websites trying to protect login credentials, but also for static websites. Unprotected websites are being ranked lower by search engines [6], they are limited in browser features that they can use [7], and they risk having content and advertisements injected, e.g., by wireless access point operators or Internet Service Providers [8, 9]. For HTTP/2, it has become practically mandatory because all major browsers support HTTP/2 over TLS only [10]. Website operators now typically deploy SSL certificates for their domains and use HTTPS to ensure integrity and confidentiality for any communication with their website. For certificates to be trusted by the websites' visitors' browsers, however, they need to be issued by trusted certificate authorities (CAs). Traditional verification approaches involve identity documents, like verifying passports, which incurred high processing overhead. To cope with the high-volume demand for digital certificates, CAs adopted automated approaches to verify and issue certificates, and now heavily rely on domain validation. Having launched only in April 2016, Let's Encrypt has since been dominating the domain-validation part of the certificate authority ecosystem through openly available and well-designed tooling that uses the Automatic Certificate Management Environment protocol (ACME) [11] to validate domain ownership and issue certificates almost transparently for users. Today, Let's Encrypt has issued over 100 million certificates in less than 15 months and their certificates account for 80% of all publicly trusted certificates [12, 13].

Unfortunately, combining the elasticity of cloud infrastructure and the automation of certificate issuance introduces new security vulnerabilities. In this paper, we discover that stale and abandoned DNS entries pointing to cloud IP addresses can be exploited by attackers to deceive domain-based certificate validation and obtain certificates for the victim domains. The problem stems from the ephemeral nature of the cloud resources. More specifically, if a user releases a cloud IP address, but does not remove the corresponding DNS entry before

Network and Distributed Systems Security (NDSS) Symposium 2018
18-21 February 2018, San Diego, CA, USA
ISBN 1-1891562-49-5
http://dx.doi.org/10.14722/ndss.2018.23327
www.ndss-symposium.org

releasing the IP address, an attacker can allocate the same IP address, impersonate ownership of the domain, and request trusted certificates from a CA, like Let's Encrypt. In this paper, we call them IP address use-after-free vulnerabilities, which can enable a variety of attacks and cause harm. Adversaries can leverage the acquired valid certificates for man-in-the-middle attacks, e.g., to intercept the HTTPS traffic to the victim domain on a wireless network. Worse, if an attacker obtains a wild-card certificate, her attack capabilities are significantly enhanced, possibly allowing her to impersonate any sub-domain, including non-existing ones. The obtained certificates can be abused for phishing attacks, by impersonating the legitimate website, including SSL verification and its "trustworthy green lock." Attackers can deface the website, and they might even be able to launch remote code execution attacks, e.g., if JavaScript or native code is being loaded from the domain that was taken over [14–16].

To better understand the prevalence of IP address use-after-free vulnerabilities in the wild, we conduct a large-scale analysis. From passive DNS traffic, we extract over 130 million domains that point to IP addresses of cloud networks. On these domains, we perform regular liveness probes to determine whether their cloud IP addresses are allocated and in use. Our results indicate that over 700,000 domains point to cloud IP addresses that are free, and which are susceptible to domain takeover attacks due to use-after-free vulnerabilities. We further investigate the feasibility of obtaining particularly interesting target IP addresses from cloud services, and we estimate that it would cost attackers less than $1 (USD) to cycle through the necessary unique IP addresses, which renders the attack economically viable for adversaries. Based on our in-depth analysis, we propose to extend the ACME protocol version 2 by including our new trust-based identifier validation challenge, and we provide practical recommendations for domain owners and cloud operators to protect themselves from domain takeover attacks.

In this paper, we make the following contributions:

- We conduct a comprehensive study of IP address use-after-free vulnerabilities, and the domain takeover attacks that these vulnerabilities enable. We show that the scale of the vulnerabilities is considerable: over 700,000 unique domains point to IP addresses that are free and can be abused to take over the respective domains.
- We discover that even well maintained DNS zones can be vulnerable to domain takeover attacks: after releasing cloud IP address resources, an adversary might be able to exploit now outdated zone information in DNS caches to launch attacks.
- We examine the feasibility of launching domain takeover attacks in the real world through cloud IP address re-use, by analyzing their allocation cycles, and we show that it is practical, time-efficient, and cost-efficient for an attacker to launch such attacks.
- We propose a new domain-validation method for automated certificate management environments (ACME) CAs that leverages the existing trust of a name to mitigate domain takeover attacks.

The remainder of this paper organized as follows: First, we provide background detail on DNS, operation of Infrastructure-as-a-Service clouds, and domain validation (see Section II). Next, we analyze and evaluate to what degree IP address use-after-free vulnerabilities pose a security threat (see Section III). Then, we present our mitigation technique, which retains almost all usability benefits of automated domain validation, yet protects against IP address use-after-free (see Section IV). Subsequently, we compare our mitigation to related work (see Section V). Finally, we conclude (see Section VI).

II. Background

We provide a basic introduction to the Domain Name System (DNS), to different operational models in cloud setups, and to the use of domain validation for SSL certificate issuance.

A. Domain Name System and DNSSEC

The Domain Name System (DNS) is a core protocol of the current Internet architecture. It facilitates to use easily identifiable hierarchically organized names instead of IP addresses to access services online. Although the fundamental idea of DNS is straightforward [17], we describe IPv4 and IPv6 resource records (RRs) and DNSSEC as they are essential to our work.

Resolving names to IP addresses via DNS is done by requesting an `A` RR to resolve a name to an IPv4 address, or an `AAAA` RR to resolve to an IPv6 address. The information for a RR is stored in the so-called parent zone. Each record is served by (at least one) DNS server, which is authoritative for that zone. There is, however, no automatic aspect within the DNS ecosystem that guarantees that DNS entries remain "fresh," i.e., a method that ensures that a given RR never becomes "stale," but that it always points to the correct IP address or that it is removed if it should point nowhere.

DNS by itself does not provide authentication, which brings security issues due to response spoofing, and spoofing can allow domain takeover attacks. DNSSEC is one method to provide integrity for the unencrypted DNS ecosystem. Authenticating existing records is a straightforward extension of DNS through a signature record type (`RRSIG`) for each original resource record set (RRset), which is signed with a zone-signing key (`ZSK`). The public key portion of the `ZSK` is hosted in the zone, while the parent zone provides a hash of the `ZSK` in a `DS` RR. The problem of distributing public keys in a trustworthy manner is solved through DNS' hierarchical nature and its existing chain of trust from the root zone to the queried zone. Crucial is that DNSSEC discourages the use of online signing to prevent denial of service attacks against the nameserver and chosen-plaintext attacks against the zone-signing key, as well as deploying the `ZSK` to (hidden) master nameservers to automate signing of updated zone information online [18, Section 5]. Instead, it strongly encourages to publish only zone information that was signed offline in a secure manner, and then deployed to (hidden) masters [19, Section 3.1, Section 9, and Section 12][20, Section 3.4.3]. Furthermore, the current state of the DNSSEC ecosystem shows significant deployment issues, for example, not publishing all required records for validation, incorrectly rolling-over keys, or not rolling keys over in the first place, which indicates a lack of care or tooling when deploying DNSSEC in practice [21].

B. Cloud Models

Cloud Computing has become a widely used concept in Computer Science. Following, we employ the National Institute of Standards and Technology's (NIST) definition of Cloud Computing [22].

Clouds are hardware and software bundles to provide users with five basic characteristics: *on-demand self-service*, *broad network access*, *resource pooling*, *rapid elasticity*, and, *measured services*. Specifically, it means that a cloud must provide services at its users' demand, without requiring any further manual interaction by the cloud operator, it must allow customers to (ideally) automatically scale their resource usage based on their needs, and all operations must be metered precisely and billed accordingly.

Cloud infrastructures generally have different deployment models, depending on their use case and users: *public* for the general public, *private* for large operators or higher security requirements (e.g., businesses or the government), or *community* for private clouds shared among multiple organizations for cost-savings or security. In this paper, we focus on IP address re-use vulnerabilities in public clouds.

Ultimately, the most distinguishing technical difference for clouds is their respective service model:

Software as a Service (SaaS).
The SaaS model is the most abstract setup. Customers interface with software provided by the operator, either via their web-browser or a standardized program interface (API). Customers do not have access "*the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities [...]*" [22]. Examples include Microsoft Office 365 and the SalesForce Platform.

Platform as a Service (PaaS).
For PaaS clouds, users deploy their own code and applications to run on the cloud. Although the executed code is under the users' control, access to the underlying cloud infrastructure, like network and disk, is similarly restricted as in the case of SaaS clouds. Examples include Heroku and Google App Engine.

Infrastructure as a Service (IaaS).
IaaS clouds, on the other hand, give more control to cloud users. Here, a user can freely request storage, network, memory, processing, and other resources as needed. Commonly, these resources are provided to the user in form of a virtual machine (VM), on which the user can install any operating system and software. Popular examples of IaaS clouds are Amazon Web Services (AWS) EC2 and Microsoft Azure.

In this paper, we investigate IaaS clouds because they allow us to freely and rapidly allocate IP addresses as part of their resource pooling characteristic. Depending on the external interfaces of PaaS clouds, they may also be vulnerable to re-use attacks, which are related to the IP address use-after-free vulnerabilities that we describe in this paper.

C. Domain-Validated Certificates

The HTTPS ecosystem is based on certificate authorities (CAs), which are trusted by operating system and browser vendors. These vendors include the CAs' certificates in their products, and certificates that are presented to clients have to demonstrate a chain of signatures to a certificate of a trusted CA. The job of a CA is to verify that the entity that requests a certificate to be issued is authorized to obtain a signed certificate for the specific domain(s) that the certificate is supposed to be valid for.

Various methods to assert authority over a domain exist. Classical and more expensive methods of identification require a CA to verify that a requesting party conforms to the domain-owning party by checking identity documents, e.g. passports, or company incorporation forms. However, such processes incur significant overhead.

Nowadays, more cost-effective methods of validating domain ownership, or rather establishing that the requesting party is currently controlling the domain, exist, and they have been adopted by all major CAs, mainly to combat operating costs. These methods are generally referred to as issuance of a domain-validated certificate, because only authority over the domain is established. The three most common validation methods are:

DNS Validation.
To validate ownership of a domain via DNS, the certificate requester must set a nonce that she received from the CA in a DNS record, usually a `TXT` record, which the CA will attempt to query and validate. Requiring the requester to change a DNS entry implies that she controls the domain's DNS zone, which is considered a strong indicator for authority over a domain.

Email Validation.
Similarly, to validate a domain via email, the CA sends an email to (a) one of the mail addresses listed in the domain's WHOIS data, or, (b) to one of the common administrative email accounts, like "postmaster," "webmaster,", or "sslmaster." The email includes a unique token that must be send to the CA, or a unique link that needs to be visited to verify ownership of the email address, and, in turn, the domain.

Web-based Validation.
For web-based validation the certificate requester receives a token from the CA that she must make available via HTTP at a CA-specified path on the domain for which the certificate was requested. Once made available, the CA verifies that the token is accessible and contains the correct value, and only then attests ownership of the domain and issues certificate.

Traditionally, CAs were dominated by an enclosed and business-oriented community. CAcert was among the earliest and most prominent approaches to introduce a community driven CA effort [23]. Unfortunately, due to insufficient support by browser and operating system vendors, it never reached widespread adoption. Furthermore, the recent rise of SSL related incidents, e.g., DigiNotar [24] and CAs issuing illegitimate certificates [25], lead to two new developments

trying to disrupt the established CA ecosystem: the widespread introduction and requirement of certificate transparency and the Let's Encrypt CA.

Certificate transparency is a framework that specifies that a CA must publish to a tamper-proof, append-only log, which can be audited by authorized parties [26, 27]. Its purpose is to allow potentially affected parties, e.g., domain owners, to verify that a CA has not issued a certificate for a given domain to an unauthorized party. In an ideal world, all CAs would participate in this scheme and publish certificate transparency logs, but, unfortunately, not all CAs do currently participate. However, some individual CAs have been forced to publish transparency logs by browser vendors, most notably Google, who threatened to void their trust in the CAs and to remove the CAs' certificate from their products if the CA does not comply with Google's request. Without a doubt, the removal of a CA from a major browser, such as Google Chrome, would have severe business and financial consequences for a CA, as it might have to refund cost for already issued certificates and it would likely have difficulty acquiring new customers, which is what forces a CA into compliance and why it is willing to participate in the certificate transparency scheme. One example of such an occurrence is Symantec, who has been required to publish certificate transparency logs after they issued certificates for google.com without Google's authorization [25].

Let's Encrypt, on the other hand, is an effort to make TLS encryption more prevalent on the Internet. They practice a leaner and completely automatic identity verification process, and they only issue certificates with short lifetimes of 90 days, to limit the potential damage of key compromise and mis-issuance, as well as to encourage automation [28]. Contrary to the most other CAs, Let's Encrypt issues certificates free of charge, and identity is verified exclusively via web-based validation and through DNS validation. Thanks to a combination of a browser-trusted certificate, being free of charge, and software tooling openly available to reduce system administrator effort, it has led to a significant increase in the number of systems on the Internet which use validly signed certificates, as well as it increased Let's Encrypt's popularity and market share [29].

III. Problem Analysis

Mitigations to protect from security problems can be implemented with varying degree of complexity, and for problems of varying degree of complexity. However, in practice, these security measures bear performance overhead and have usability drawbacks, which might not be acceptable. In turn, their actual real-world deployment depends on security risk evaluations, operational costs, and human costs. Therefore, before trying to mitigate a non-issue, it is necessary to justify them with supporting data instead of recommending absolutes.

Following, we first discuss the different security issues in respect to use-after-free vulnerabilities for IP addresses in respect to DNS-based domain validation. We then evaluate to what degree those security issues are practical to exploit. Finally, we estimate how many domains might be susceptible to takeovers and whether protecting them is worthwhile.

For our problem analysis, we investigate and interact with systems that are online and in-use by third parties. Naturally, those systems are outside of our control. In turn, our analysis poses ethical challenges to not affect or impact the legitimate users of such systems in any way. We discuss the considerations we undertook for an ethical and appropriate, yet realistic, analysis separately for each experiment in their respective sections.

A. Impact

Domain takeovers bear serious consequences, even temporary takeovers can provide ample opportunity for an attacker (see Section I). Naturally, they way an attacker might cause harm to the legitimate domain operator and domain users varies from case to case and the space of attacks is vast, which is why we only discuss a subset of possible attacks:

Malicious and Remote Code Loading.
Likely the most straight-forward way for an attacker to turn a profit through a domain she took over is by serving malicious code, serving advertisements, or including affiliate marketing [15, 16, 30]. Although considered easier to launch for websites, the attack is not restricted to websites. Instead, an attack could also be launched on mobile or desktop applications, e.g., through remote code loading [31, 32]. Unfortunately, HTTPS and HSTS themselves do not mitigate such an attack.

SSL Certificates.
Another way for an attacker to leverage a domain takeover attack or to increase its success chance is by requesting a SSL certificate that is trusted by operating systems and browsers. Requesting a trusted SSL certificate has become practically feasible because of domain-validated certificates, such as Let's Encrypt. Once she has obtained the certificate, she has increased capabilities for remote code loading attacks over HTTPS, even including HSTS.

Nameservers.
A domain might also point to a nameserver, where the domain server can be for the same domain or different ones. In practice, these cases occur because DNS demands multiple nameservers for redundancy, and if a nameserver does not respond, a client automatically and, transparent to the user, retries queries with fail-over nameservers. Therefore, a domain pointing to a free IP address for a nameserver only incurs a latency penalty and is barely noticeable by the user. However, an attacker could take over the entire domain and even create additional domains. For a domain owner, taking over a domain that is being used as nameserver equates to the worst case scenario. Unfortunately, even entire top-level domains have been vulnerable to nameserver domain takeover attacks [33].

Email Servers.
Similarly, after gaining control over a domain, an attacker might be able to send and receive emails. Importantly, a DNS `MX` record is not required: if a domain has no `MX` record set, then its respective `A` record is being used. Acquiring the capability to send or receive email allows an attacker to abuse a domain for spear-phishing and phishing campaigns, such as CEO email scams, or to recruit victims for fraudulent schemes [34, 35].

Sub-domain Attacks.
Finally, top-level domains are not the only worthwhile takeover targets for an attacker. Sub-domains are at least similarly interesting for attacks, even sub-domains that might have never been used in production, as they could still be abused for authentication bypass vulnerabilities, e.g., like it recently happened to the ride-sharing company Uber [36].

Regarding SSL certificate related attacks, it is sufficient for an attacker to request an ordinary certificate. She does not require a wild-card certificate to launch successful attacks. However, if an attacker can obtain a wild-card certificate, her capabilities are significantly extended. For example, if she can receive a wild-card certificate for "support.example.com," she would then be able to impersonate, intercept traffic to any sub-domain of "support.example.com," and even launch sub-domain related attacks at the main domain "example.com" [36]. Although, currently, wild-card certificates are not supported by free domain-validated certificate authorities, like Let's Encrypt or StartCom, at least Let's Encrypt is planning to support them as early as January 2018 [37]. Furthermore, wild-card certificates are already supported by other mainstream CAs, such as Comodo. While they charge a fee, they allow significantly longer validity periods of up to 3 years, which can make attacks even more disastrous.

B. Taxonomy

For a precise classification of how IP address use-after-free vulnerabilities are being rendered possible, we distinguish four different cases in which a domain points to a free IP address (i.e., the domain is stale) through the following taxonomy:[1]

Early Migration.
A domain-IP mapping is *migrating early* if the domain is in use by the operator, and the records at the authoritative nameserver have been updated to point to the new IP address before the old IP address is being released and available for others to request and use.

Delayed Migration.
Similarly, a domain-IP mapping is *migrating with delay* if the domain is in use by the operator, and the records at the authoritative nameserver have *not* been updated yet, i.e., they point to a released IP address.

Auxiliary.
Differently, a domain-IP mapping is *auxiliary* if the domain is used by the operator, and the domain has multiple records, which point to both current and old IP address, possibly in a way so that the old and free IP address would only be used as in a fail-over scenario and has otherwise no practical impact.

Abandoned.
We define a domain-IP mapping as *abandoned* if the domain is not used legitimately anymore. For example, a company might become defunct and is not operating the service anymore that was previously offered at the domain, but it retains ownership of the domain until its expiration.

[1]Our study focuses on SSL certificates, web servers, domain validation through HTTP, and type `A` DNS records. However, our findings also apply to other record types, e.g., `MX` or `NS`.

Depending on how the domain becomes stale, the length of the window of opportunity differs. In case of an early migration, an attacker has the shortest window of exploitation: the cache lifetime of the domain IP mapping. Note, however, that the time a domain IP mapping might be cached is not strictly its time to live (TTL) as set by the authoritative nameserver. The mapping can be purged from the cache before its expiration, and a caching nameserver might ignore the TTL entirely and cache entries longer, e.g., for performance reasons, though in violation of the DNS RFC [38]. Theoretically, early migration could prevent IP address use-after-free attacks under the assumption that no intermediate nameservers cache entries longer and that the IP address is released only after the TTL has expired. Practically, unfortunately, human error results in domains not always migrating early and intermediate nameservers might ignore the TTL. Therefore, even those domains migrating early can be at risk of temporary domain takeovers.

From a security standpoint, the remaining three classes are more worrisome. The easiest case to launch a successful attack against is an abandoned domain: the attacker is not rushed by the legitimate operator and she can wait until an opportunity arises. Fortunately, it is also the least interesting case for an attacker because users are not expected to contact the service at the domain regularly anymore but only sporadically (e.g., through an outdated bookmark for a website), thus, the number of potential victims is generally low.

For domains that migrate with delay, the window of opportunity to validate ownership of a domain is fixed in time and often short. While an attacker could miss the window, she can lurk and wait for a target domain migrating with delay by repeatedly trying to allocate the target IP address, which we later show is practical (see Section III-C). More important, once the window of opportunity has passed, the successfully validated domain is not useless to the attacker even though she has no control over the host with the IP address behind the domain-IP mapping anymore (it is now a new IP address, which is not under the attacker's control). For example, in case of domain-validated SSL certificates, once an attacker validated that she owns the domain, she can later leverage the obtained certificate for man-in-the-middle attacks, e.g., for a wireless network at a coffee shop, because the certificate is trusted by major operating systems and browsers. Here, the number of victims is larger than in the case of abandoned domains, but seldom substantial. The core problem with domain-IP mappings that are migrated with delay lies in the long-term capabilities granted to the attacker.

Auxiliary domain-IP mappings are the most troublesome case: they provide a constant window of opportunity and can cause the most havoc. First, an attacker can remain stealthy as a "fail-over" until a viable opportunity arises. During normal operation, the attacker's machine does not respond or it redirects all traffic to a legitimate host. Second, an attacker can force a fail-over to the IP address under his control by launching a denial of service (DoS) attack against the legitimate hosts. However, even without forcing a fail-over, an attacker will see a subset of traffic due to implicit round-robin in DNS, which occurs because DNS records have no implied order. Upon forcing fail-over, the attacker forces a domain-validation service to connect to the host under the control of

the attacker, as no other hosts are responsive. Correspondingly, without forcing a fail-over, the attacker might need to try multiple times until the domain-validation service connects to the address under her control and, in turn, validates her ownership of the domain. The attacker can verify ownership of the domain successfully in both cases, e.g., to request a certificate, and a significant number of users will connect to the attacker's machine (all or a subset due to DNS' round-robin). Overall, auxiliary domain-IP mappings can affect the most victims and it can provide ample opportunity to cause harm, e.g., to visitors of a website by injecting malicious code.

After we classified the reasons for why IP address use-after-free vulnerabilities exist and what their impact can be, the immediate next question becomes: can an attacker actually exploit these vulnerabilities in practice, by allocating the same IP address the victim has freed?

C. IP Address Churn

An attacker can successfully exploit an IP address use-after-free vulnerability in practice if she can get a cloud provider to assign the recently freed IP address to herself within the window of opportunity. Following, we determine whether it is practical for Amazon Web Services (AWS) and Microsoft Azure, the two largest cloud providers today [39].

Specifically, we repeatedly allocate and free IP addresses in succession. To prevent starvation, we are using a slow allocation cycle to not interfere with the clouds' operations: We request 5 IP addresses per region, freeing them immediately, and then sleeping for 10 seconds, i.e., effectively allocating 1 IP address every 2 seconds. We performed our IP address churn experiment from April 29, 2017 01:03 UTC to June 6, 2017 23:27 UTC spanning all regions of the cloud providers at the time for a total cost of $31.06 (USD). Over the course of our measurements, we cycled through a total of 14,159,705 allocations of 1,613,082 unique IP addresses. As we always first released addresses before allocating the next batch, we cannot cause address starvation. This is highlighted by us always receiving an IP address upon issuing an API request.

The success of our technique depends on how fast we can iterate through the pool of free IPv4 addresses for a given availability zone. This depends on the overall size of the pool, and its variance, i.e., how fast addresses are allocated by other users. To illustrate these characteristics for each availability zone, we investigate the churn (see Figure 2) and time between allocation of the same address (see Figure 1). We show only AWS specific plots in the pursuit of brevity and comprehensibility, as Azure is not behaving significantly different.

Using the churn plots in Figure 2, we get an overview of change in the IaaS cloud's IP pools. Figure 2 shows the churn in allocated addresses for AWS, i.e., for each day we allocated addresses we plot the fraction of addresses we previously allocated and the fraction of addresses we did not previously receive as an allocation. Dates without data relate to dates where either the IaaS provider conducted maintenance operations, or our measurement scripts were not yet running for that zone.

The natural pattern we expect for the churn plots is an initially high share of new addresses while the pool is being initially explored. This pattern should then slowly approach a stable socket, which corresponds to those addresses that are handed back to the pool by other tenants. Indeed, we find this pattern in our data. For example, Figure 2(a) and 2(g) show this expected pattern. However, these zones have a relatively large pool of addresses that is free at any given time. Zones like eu-west-2 (see Figure 2(i)) are significantly smaller, hence converge more quickly. This furthermore underlines that the allocation algorithm must, in some form, iterate through the whole pool of addresses, instead of just allocating the (same) first free addresses.

In addition, we also find a couple of interesting events: Zone ap-southeast-2 (see Figure 2(e)) started off similar to eu-west-2. However, at the beginning of week 20 in 2017, a large batch of free addresses was added to the pool, leading to a "restart" of the churn pattern. In eu-west-1 (see Figure 2(i)) and us-east-1 (see Figure 2(k)) we see the effect if several days of not iterating through the pool: As soon as we restart our allocation script, we observe a slight rise in new addresses, which have accumulated during the time we did not perform measurements. We find the last notable pattern in us-west-2 (see Figure 2(n)). Here, a substantial amount of so far unseen addresses is released to the pool in the middle of each week.

Next, we take a look on how long it takes to iterate through the whole pool, i.e., how fast an attacker could obtain a specific address. For this, we look at how much time passes on average, until an address is allocated for the second time. Given our earlier observation that we do indeed circle through the IP pool, we expect the mean to correspond to the point where we iterated through the IP address pool. This is summarized with boxplots (without outliers) in Figure 1. We find that with our ethically restricted approach most pools are exhausted within under a day. Only the largest, like eu-west-1 and us-east-1 reach means significantly over a day.

Although we used a slow allocation cycle to not interfere with the clouds' operations, an attacker is not bound by the same ethical standard. Practically, the attacker will be bound only by the response time of the IP address allocation API endpoint and her network latency to it. Therefore, an attacker can cycle through available IP addresses much more rapidly. In fact, considering the AWS API limit (10,000 requests per second [40]) and the number of requests needed to exhaust pools in our experiments, an attacker would only need between two and 61 seconds to acquire a target IP once the victim has freed it, using a rapid allocation cycle of 5,000 IP allocations per second. In practice, this theoretical limit is not necessary for an attacker to launch a successful attack. For example, DNS cache times are almost always 5 minutes, and often much longer with 60 minutes to multiple hours, thus, allowing an attacker to be successful by exploiting caching effects with rates of less than 50 IP address allocations per second.

D. Affected Domain Names

Considering the worrying high-rate of IP address churn for major cloud providers and low opportunity cost for an attacker to launch an attack, the only question that remains unanswered before we can determine whether temporary stale

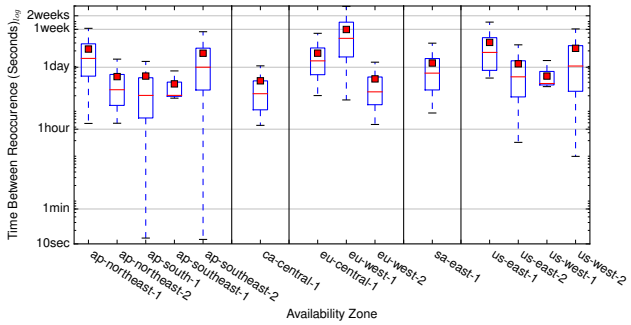

Figure 1: **Time passed between allocations of the same IP address to us.**

domains pointing to readily available IP addresses are a problem in practice is whether a significant number of domains are affected?

For a better understanding of how many domains are affected by IP address churn, we observe DNS traffic through Farsight's passive DNS measurements [41]. The Farsight passive DNS dataset is provided through a continuous datafeed. For our collection and DNS data analysis, we follow established best practices for collecting and handling Internet measurement data [42], we anonymize all incoming data immediately by removing any resolver information, and we only retain successful DNS responses.

Specifically, we collect all DNS responses containing `A` records pointing to the Amazon Web Services (AWS) EC2 cloud, the Microsoft Azure cloud, and the Digital Ocean cloud spanning exactly 120 days from April 11, 2017 0:00 UTC to August 9, 2017 0:00 UTC. Overall, we extract and analyze 130,274,722 unique domains with 767,108,850 unique domain-IP mappings, counting also sub-domains. Including sub-domains is important for completeness, however, it makes an accurate comparison to top domain lists (e.g., Alexa), to estimate the domains' popularity, difficult, because they do not include sub-domains. Matching at the second-level of a domain is similarly problematic due to potentially over-estimating the impact of ephemeral sub-domains and the loss of information on sub-domains of special second-level domains, such as .ac.nz or .co.uk. It remains for future work to evaluate the distribution of DNS zone staleness in regard to domain popularity.

We perform our evaluation on a Kubernetes cluster comprised of 656 processor cores and 3,020 GiB memory, and which is connected at a dedicated 10 Gbps Internet uplink.[2] For each domain, we test every six hours[3] from June 10, 2017 0:00 UTC to August 9, 2017 0:00 UTC (60 days) whether the IP address is still in use or if it might be freed and available:

1) We resolve the domain and check if the IP address the domain points belongs to a network of a cloud provider.[4] If the domain points to a cloud IP, we test if the IP address is responsive and whether it might be free and available to others. If it does not point to a cloud provider or does not exist anymore, we do not perform any further tests.

2) We test if the IP address responds to ICMP ping requests, or responds to any packet sent on 36 of the most frequently used TCP and UDP ports (see Table I) [43] within a two seconds timeout.[5] If we receive a response to any of our requests, we mark the IP address as online and allocated. Correspondingly, if we receive no response until the timeout is reached, we mark the IP address as offline and freed.

Naturally, ingress firewall rules could prevent our test from succeeding and, thus, our estimation is an upper-bound. One might expect it to be a gross over-approximation because cloud virtual machines instances have traditionally received public IP addresses. Nowadays, however, this is not necessarily the case: cloud instances that do not need a public IP address can and generally do live in cloud-only internal networks. Furthermore, by default, many machines respond to ICMP ping requests or allow for secure shell (SSH) access via TCP on port 22. Additionally, a public IP address associated with an instance is freed and can be reused by others if the instance is shutdown, even if it is later powered on again (it receives a new IP address at this point). In turn, it means that we only misclassify machines as offline with heavy ingress filtering that do not provide a service on the top 36 ports (see Table I), and which have not been migrated to an internal network yet, which is becoming scarcer. Therefore, although our estimate remains an upper-bound, we are confident that it is a close estimate.

Over the course of our measurements, we classify 702,180 unique domains (0.539%) as pointing to available and freed IP addresses. Therefore, these domains, most likely, have been vulnerable to a (temporary) domain takeover attack at some point in time. In fact, while the majority of domains migrated delayed (80.31%), a non-negligible amount of domain-IP mappings are abandoned (17.24%) and, fortunately, only a small number of domain-IP mappings are auxiliary (2.45%). Note that we only determine that the domain could be taken over, but its prior purpose remains unknown. Further investigation by future work is required to determine how many of the vulnerable domains have been actively used in the past and what the impact of an attack on them would be, e.g., on a website that is protected through HTTPS and requires a SSL certificate, or a domain that is used to load remote code for a mobile application (see Section III-A). Although the amount of vulnerable domains appears small relatively speaking, in absolute terms, the number of stale domains is quite large. Additionally, due to the nature of our dataset, we only observe domains that are actively being attempted to be accessed: the estimated number of cases that might be vulnerable to domain takeover attacks and could be abused for phishing or scams, but which were not being accessed during our observation period, might be significantly larger.

[2]The cluster is on a network separated from the main network of the institution at which the experiments are performed. The network traffic generated for our evaluation is not subject to packet introspection, which would have had a negative impact on our measurements.

[3]Some tests were up to twelve hours apart because of scheduling delay.

[4]We exclude networks of cloud providers that are used for services other than cloud virtual machine instances, e.g., Load-Balancing-as-a-Service.

[5]We chose a two seconds timeout after we experimented with higher timeouts of five to ten seconds and did not notice any difference in results. A shorter timeouf of one second resulted in a high misclassification rate due to network and system load. The cut-off for no misclassifications was close to 1.4 seconds in our tests. Out of carefulness, we chose a two second timeout.

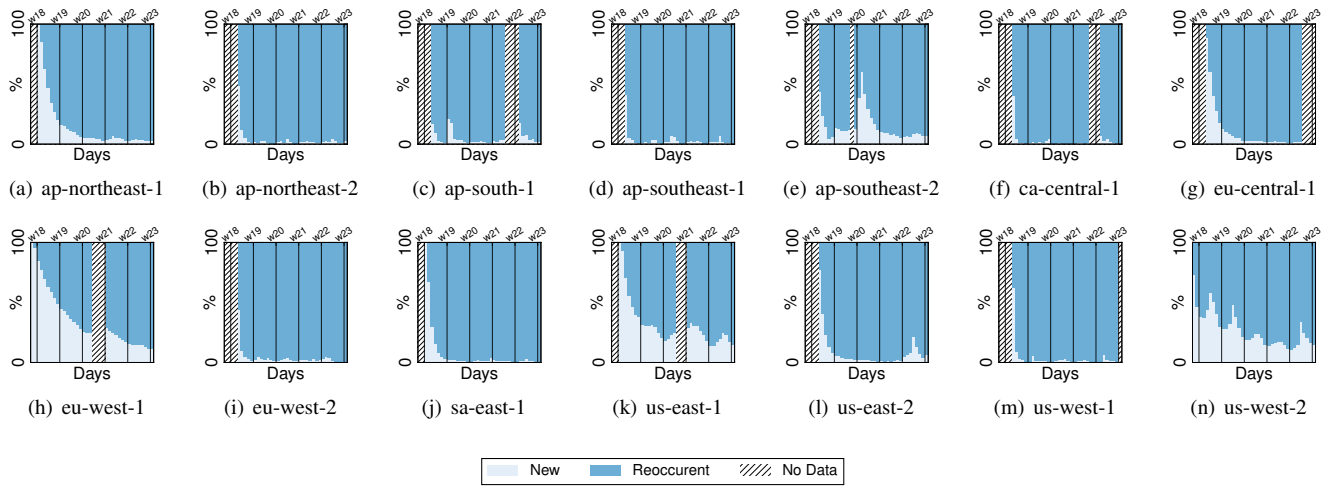

Figure 2: **IP address churn on the Amazon Web Services (AWS) EC2 cloud, i.e., share of newly-observed IP addresses per day per region.**

Protocol (Common)	TCP	UDP	Port(s) ▲
FTP	✓	✓	21
SSH	✓	✓	22, 2222, 22022
Telnet	✓	✓	23
SMTP	✓	✓	25, 587
WHOIS	✓		43
DNS	✓	✓	53
HTTP	✓		80, 8000, 8080
Kerberos	✓	✓	88
POP3	✓	✓	110
IMAP	✓	✓	143
LDAP	✓	✓	389
HTTP (Secure)	✓		443, 8443
SMTP (Secure)	✓	✓	465
LDAP (Secure)	✓	✓	636
Telnet (Secure)	✓	✓	992
IMAP (Secure)	✓	✓	993
POP3 (Secure)	✓	✓	995
MS SQL	✓	✓	1433
CPanel	✓		2082
CPanel (Secure)	✓		2083
CPanel WHM	✓		2086
CPanel WHM (Secure)	✓		2087
MySQL	✓	✓	3306
2Wire RPC	✓	✓	3479
Virtuosso	✓		4643
Postgres	✓	✓	5432
CWMP	✓	✓	7547
Plesk	✓		8087
Webmin	✓		10000
ENSIM	✓		19638

Table I: **Ports and protocols used for IP address liveness checking.**

E. Proof of Concept Domain Takeover

Finally, we show the practicality of domain takeover attacks through a proof of concept certificate request to Let's Encrypt. Certainly, we face the largest ethical challenges with this experiment, as disrupting or having any impact on legitimate users raises ethical concerns. For example, it is impossible to guarantee that we do not interfere with any third party operation that might rely on the domain, or that we do not accidentally receive Personally Identifiable Information (PII) or other confidential data. Therefore, we perform a domain takeover attack for a domain under our control. After obtaining the certificate from Let's Encrypt and verifing that it has been published to certificate transparency logs, we revoke it, and publish the revocation to Let's Encrypt. The time until these actions appear in CT logs serves as an indication of the time that passes before the legitimate owner would be able to notice the attack by monitoring CT logs.

For our experiment, we gained temporary control over the domain "cloudstrife.seclab.cs.ucsb.edu" by attempting to re-allocate the IP address to which the domain points to (34.215.255.68). Note, that the IP address is located in the availability zone us-west-2, which has a high churn that makes takeover attackers more difficult. While this may seem contradictory, as a high churn means that an attacker can allocate more addresses per time-unit, a high churn also indicates a larger IP address pool. Ultimately, we were able to successfully re-allocate the IP address within 27 minutes and 55 seconds with a slow allocation cycle of 2 IP addresses per second (see Section III-C). While anecdotal, it serves as an estimate of the time needed to launch an attack successfully under unfavorable conditions for an attacker (high churn, low allocation rate). We requested a SSL certificate from Let's Encrypt, it appeared in different certificate transparency logs between 34 minutes and 61 minutes later, and we revoked the certificate immediately after certificate transparency log entries had been propagated. Our certificate request was published at the "crt" web-interface under id 250959196; it can be viewed at https://crt.sh/?id=250959196. The certificate that we obtained from Let's Encrypt and a message signed by the respective private key is contained in Appendix A.

Although the incorrect migration of domain-IP mappings is comparatively small on a relative scale, we believe that the absolute numbers speak volumes paired with the practicality of takeovers. Together, they justify looking closer at mitigating IP address use-after-free at its core, however, with a strong requirement to incur as little additional overhead on usability or performance as possible.

IV. Mitigation

In this paper, we address the issue of IP re-use attacks abusing stale DNS records, particular for IP addresses belonging to cloud networks, a topic that has received little attention so

far. To be more specific, we investigate IP address use-after-free vulnerabilities, which can pose severe security threats, and which can be made even more dangerous through domain-validated SSL certificates (see Section III). Current automated domain-validation-based certificate issuance systems are also threatened to be exploited through man-in-the-middle attacks discussed by Gavrichenkov et al. [44]. Existing defenses rely on certificate revocation, which is severely fragmented and cannot be relied on in practice [45, 46]. It became only recently more tractable, e.g., through CRLite [47], but these solutions have not been adopted yet. One core problem is that revocation checks in browsers are not comprehensive: Chrome generally does not verify revocations, its CRLSet is limited to emergency revocations by design [48], and Mozilla's Firefox similarly limits revocation checks through OneCRL to CA intermediate certificates [49]. Certificate revocations in other software and libraries, which rely on the same certificate issuance processes and would also be required to adopt the new revocation checks, are rarely checked in practice [50]. Furthermore, revocations are reactive by nature and they provide a window of opportunity to an attacker by design: the time until the revocation has propagated plus the time until the attacker's certificate has been revoked by the issuing CA on request of the legitimate party, the latter of which is generally a manual process as additional verification is required. We believe that the first line of defense should be with domain-validation-based CAs and it should be preventive. Therefore, we propose an additional layer of protection for domain-validation-based CAs, such as Let's Encrypt, that can efficiently and with negligible overhead prevent these attacks. Our mitigation technique builds on the ACME protocol version 2 [51] and it is complimentary to the certificate transparency framework [26].

A. General Concept and Threat Model

The underlying problem of IP address re-use attacks is that a domain-validated certificate can be requested as soon as an attacker controls the IP address to which a domain points to, and that requesting and receiving a trusted certificate is fully automatic and only a matter of seconds nowadays. An attacker might be able to obtain the IP address legitimately, because the domain record was left stale. To obtain a certificate, she might also be able to perform man-in-the-middle attacks between the authenticating CA and the target system. A similar issue occurs, if she can (temporarily) compromise the DNS (delegation or authoritative servers) for a domain. Then, she can simply change the IP address a record points to, as well as potential CAA or DANE TLSA records [52, 53]. Technically, attacks involving DNS-based attacks should be prevented by DNSSEC [19]. However, if key signing is performed online on the authoritative servers itself (against DNSSEC best practices) [54], and she compromises one of these servers, then she regains full control over the domain. Although, domain takeovers rarely tend to last for extended periods of time, SSL certificate for the domain can later be used by the attacker until the certificate's expiration date, possibly involving other man-in-the-middle attacks.

For all certificate requests that a CA receives, one of the following four cases applies:

1) No certificate has been requested for this domain in the past.
2) A certificate for the domain has been requested in the past, and the domain still points to the same IP address.
3) A certificate for the domain has been requested in the past, but the domain now points to a different IP address.
4) A certificate for the domain has been requested in the past, but it was verified in a more strict manner, possibly using extended validation (EV).

The first case is relatively frequent, and it is indistinguishable from the legitimate first use of domain-validated certificate issuance, which it is impossible to protect against without extended validation, which is itself often deemed too costly or impractical. We also acknowledge that an attacker who has compromised the system to which this domain points to will, in any case, be able to issue a new certificate for the domain, or steal the existing one.[6] Hence, a full system compromise is outside of the scope of our work. What our mitigation technique has to ensure is that a domain-validated certificate is only issued if the CA can verify that there has been no non-cooperative change of authority over either the system the domain points to or DNS zone for the domain.

Concerning our threat model, the attacker does not control a trusted CA, and she has average resources and skills, i.e., she is not a state-level actor and cannot expend significant resources for a successful attack. Her overall objective is to obtain a domain-validated SSL certificate for a target domain that already uses a valid SSL certificate issued by a third party CA. However, she has no administrative access to the machine that the target domain currently points to, she cannot steal the current certificate or factors its keys in a reasonable amount of time, but, instead, she must request a new certificate. Taken into account the current operational model for domain-validating CAs, to achieve her goal, the attacker can: (a) obtain access to an IP address to which a stale A record for the domain points to, (b) perform a man-in-the-middle attack somewhere on the path between the issuing CA and the system to which the target domain points to, or, (c) illegitimately take over authority over the DNS zone for some amount of time.

B. Pre-Signature Certificate Consistency Checks

To ensure that an attacker within our threat model cannot request a new certificate, we must ensure that she cannot show that there has been a cooperative change for: (a) the IP address to which the domain points to, or (b) the DNS zone of the domain. One way to accomplish this task is by requiring each subsequent certificate request for a domain for which a certificate has been issued in the past by trusted CA, or which was covered by a similarly issued wild-card certificate, to be signed with a pre-existing certificate.

1) Pre-Signed Domains: A challenge for a CA receiving a domain-validation certificate request is to determine whether a SSL certificate has been issued to this domain in the past, either by itself, or possibly by another trusted CA.

Fortunately, two approaches to implement these requirements exist that are viable:

[6] Certificate theft can be protected through hardware security modules and may further become a commodity through methods like Intel SGX or ARM's TrustZone, which can be used to entrench certificate handling in a secured enclave.

Federated Approach.
In case of the federated approach, each trusted CA is required to publish its issued certificates in multiple certificate transparency logs, which do not need to be run by the CA itself [26]. This approach has the strong advantage that it utilizes established technology, meaning that the required functionality is readily available and no additional service needs to be deployed and managed. Although certificate transparency logs are not yet required for every CA or every certificate, and not all CAs are publishing certificate logs, Google Chrome is already requiring CT logs to some certificates: for all certificates issued by Symantec, WoSign, and StartCom, as well as for all extended validation certificates (since January 2015). Furthermore, enforcing the requirement for all trusted CAs is expected within the next years [55]. Thus, expected development and policy changes would further empower this approach.

From an algorithmic point of view, a naïve existence check requires lookups for each trusted CA in an aggregated database. Fortunately, by leveraging CAA records via DNS combined with DNSSEC, one can limit lookups to a small set of CAs, e.g., only one or two CAs. Specifically, it is more likely that one of the authorized CAs has issued a certificate for the domain in the past. Once a previously issued certificate has been found that is still valid, then the search can be terminated early, which reduces lookup time. Additionally, CAA records have become mandatory to be honored by CAs in September 2017 [56]. Therefore, due to the increasing adoption and availability of CT and CAA, we consider this approach the most practical and promising one.

Centralized Approach.
Alternatively, a centralized approach is possible. Here, a single authority, possibly IANA, would provide an oracle service. The service would return a boolean answer when queried, confirming whether any CA ever issued a certificate for a specified domain. Before issuing a new certificate, CAs would have to check if a certificate has been issued in the past. Furthermore, they must notify the authority of newly issued certificates. Unfortunately, the centralized approach bears potential trust issues and poses a single point of failure.

C. Domain Takeover Resistant Identifier Validation Challenge

Next, we develop a practical identifier validation challenge that is resistant to domain takeover attacks. Specifically, we target the ACME protocol, which is used by Let's Encrypt and others to automate the process of issuing certificates. To do so, we introduce an additional challenge to the ACMEv2 RFC [51]. No other changes to the RFC are necessary. In turn, it allows our validation challenge to be minimally invasive to the protocol and its subsequent implementations, yet, at the same time, it significantly improves security by mitigating the attacks that we present in this paper. The core idea of our proposed challenge is to leverage existing certificates to form a chain of trust. Implementing a solution that uses existing certificates to sign responses to identification validation challenges triggers various issues with the handling of key material. For example, private keys should not be used outside of the context for which their respective certificate has been issued, which would happen if we naïvely sign a challenge response with a key, for which the respective certificate was issued for handling TLS server connections. Fortunately for us, retrieving the challenge response through over HTTPS eliminates the problem, and verifying the used certificate satisfies all requirements we put forth in the previous sections.

Our challenge works as follows (see Figure 3):

❶ The client sends a certificate request for her domain, e.g., "example.com," to a domain-validating ACME CA.

❷ The CA checks whether a certificate for the domain "example.com" exists, i.e., that one has been issued by a trusted CA in the past. The CA is free to include expired certificates in the check or ignore them according to an agreed-on policy (see Section IV-D).

❸ The CA issues a challenge to the client, which she needs to fulfill to validate ownership of the domain. If a prior certificate exists, the CA sends two challenges: first, our challenge, which is similar to the original HTTP challenge, and which includes a token to be hosted at a specified path at the domain of the requested certificate, and, second, a challenge that is considered more trustworthy than the HTTP challenge, such as a whois-based challenge or a DNS-based challenge. Following the ACMEv2 RFC, a client needs to satisfy only one of the two challenges. If she fails our challenge, which might happen in some cases (see Section IV-D), the more trustworthy challenge must be completed. For more details on how challenges are implemented, we refer to Section 8 "Identifier Validation Challenges"'of the ACME v2 RFC. Alternatively, if no prior certificate exists, the CA is free to send any challenges as defined by the RFC.

❹ Once the client receives our challenge, she will host the nonce from it at the URL specified by the challenge to serve as the verification resource.

❺ The CA will attempt to access the verification resource, and, in turn, verify that the challenge has been completed by the client. Verification requires that the nonce has been placed at the resource, as well as that the HTTPS response is signed with the private key for a certificate of the domain that was previously issued by a trusted CA (see certificate existence check; ❷).

D. Failure Cases

There exist some possible failure scenarios of our challenge, which must be handled gracefully to preserve security of domain validation. However, the simple failure of the process does not (yet) indicate an attack. Furthermore, as soon as a failure has been resolved, the above process can be used to regularly renew certificates automatically because the HTTPS challenge will not fail again for the same reason.

1) Lost Access to Old Certificate or Private Key: Among the most likely non-malicious scenarios for failure is the case of an operator who has lost access to her prior certificate or private key. Here, the HTTPS response cannot be signed and the challenge will fail. From a security standpoint, this case must be treated like a potential attack by the CA because it is impossible to automatically distinguish between a legitimate

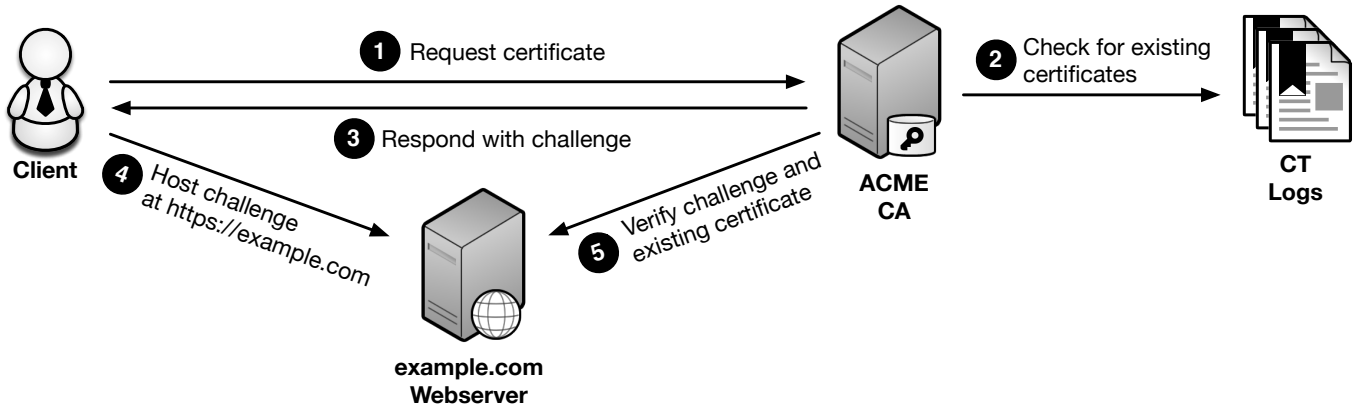

Figure 3: **Certificate request process that mitigates domain takeover attacks.**

lost key, and an attacker not having access to the key in the first place. Instead, the operator should use a DNS-based challenge or whois-based challenge. Note that no additional certificate request is required, but the same certificate request will be used. In fact, instead of issuing the certificate, first, a prompt that additional verification is needed will be shown to the operator, and once she passes the additional challenge (sent along with the first challenge; ❸), only then the certificate will be issued.

2) Expired Certificate: Another common case in which the HTTPS challenge might fail are expired certificates. Operators may simply forget to renew their certificates in time, or a service may be shut down for a longer period, preventing certificate from being renewed. Whether expired certificates should be accepted, and if so, whether their expiration should be limited by a grace period, is a policy decision rather than a technical decision. Basically, two options exist:

1) Accept an expired certificate.
2) Treat it like an attack.

Relaxing the requirement and allowing expired certificates could increase the usability of our approach. However, relaxing requirements for corner cases introduces additional sources for potential errors, and thereby, security issues. Ultimately, we err on the side of caution and default to strong security and treating it as an attack, as also recommended by Fiebig et al. [57].

3) Legitimate Change of Authority: A third legitimate case that might fail is a legitimate change of domain ownership, possibly without the consent of the previous owner. Such cases include but are not limited to seizures because of copyright violations, or court orders, or a simple lapse in renewing the domain itself. Again, such a change in ownership cannot be recognized as legitimate by an automated system, simply because an attack has exactly the same properties. Therefore, similar to the lost private key access, the CA fails the HTTPS challenge and it requests a second challenge to be completed by the client, which any legitimate client can complete easily.

4) Possible Attacks: Following our earlier reasoning, attacks are cases in which the requesting client cannot prove a continuity in authority using previously issued valid certificates, which are considered rare, particular as you renew certificates in the validity period of your current period, and often automatically. Considering prior work (see Section V) and the attacks that we present in this paper, a large portion of attacks are time critical. Therefore, the first aspect in the process of resolving a potential attack should be time. By increasing the time requirement, we increase the likelihood of the enabling attack to be detected. Nonetheless, potential for stale DNS attacks remains. Yet, we can approach this issue by designing an extended process for validating ownership of a domain and the correct delegation to an IP address. Unsurprisingly, CAs already commonly offer such extended validation processes. In addition, this service could also be offered by official institutions or NGOs with a sufficient trust level and the resources to do this. The certificates issued in this process would not even have to be valid for an extended time period. In fact, they can be used as simple seeds to re-initiate the continuous process of retrieving domain validated certificates.

E. Transitioning Techniques

One of the biggest problems when introducing new technique is the transitioning phase. However, for the adoption of our challenge, this is not an issue. The certificate ecosystem already makes extensive use of validity periods, generally certificates are set to expire within 1-3 years, and even as early as 3 months in case of Let's Encrypt. If our challenge would be adopted, we can also make use of extensive CT logs, which contain over hundreds of millions of domains already. For domains for which no entry exist in CT logs, we realize that our challenge is based upon "trust on first use" [58]. However, this does not leave domains for which certificates are already issued with less security than today, but it strictly increases security. Furthermore, CAs may add domains for which they previously signed certificates to certificate transparency logs voluntarily. Therefore, we believe that our system provides a robust and painless transition toward an increase of security for domain-validated certificates within the diverse certificate ecosystem.

F. Best Practices

Beyond directly addressing the root cause of the presented problems in the certificate issuing process, we suggest that cloud providers deploy mitigations as well. These mitigation

techniques aim to prevent attackers from allocating specific addresses, e.g., by rate-limiting IP address allocation and release operations, using disjoint sets of IP addresses for different tenants to reduce attack surface, and perhaps even by monitoring their networks for (non-scanning) inbound traffic to unallocated addresses to warn previous users of those addresses.[7] Finally, for cloud tenants, we strongly suggest keeping old addresses allocated when migrating IP addresses, at least until the TTL of the record has expired out, preferably until one can be reasonable sure that it is not cached anymore (preferably from a day to a week). Furthermore, we can only stress the importance of maintaining DNS zones properly and to remove obsolete records as quickly as possible to not fall victim to domain takeover attacks.

V. Related Work

We discuss related work, specifically in the areas of cloud security, DNS security and measurements, and the security of domain-based certificate and trust validation.

A. DNS Security

DNS is a critical service in the Internet ecosystem and prior work has studied DNS security extensively. Bell and Britton hold a patent in which they describe how a host can be taken over by assigning the same IP address to a virtual interface on another system [59]. Yadav et al. report on domain-flux practices in botnets, a technique in which a domain generation algorithm is used to generate many domains, of which the operator only needs to control one to remain in control of her botnet [60]. However, to some degree as the dual of exploiting stale DNS records, one can register a single or multiple of those domains to take over a botnet, and it has successfully been done by Stone et al. [61]. Liu et al. conducted a study similar to our work [62]. However, methodological challenges and limitations of their datasets lead them to an under-estimation of the impact of stale DNS records in cloud scenarios. Indeed, contrary to them, we find that the problem of stale DNS records is amplified by multiple orders of magnitude. We further systematically analyze the practicality of acquiring the previously-used cloud IP addresses, discover use-after-free attacks based on DNS caches, and we propose a usable mitigation technique to automatically validate certificate issuance.

Instead of relying on correct DNS responses, bit-squatting exploits random bit-flips in DNS requests to lure clients to malicious or phishing websites [63]. Different from our attack, bit-squatting relies on integrity errors that occur at random and thus is not as targeted as our attack. Furthermore, exploiting integrity errors, it can be mitigated easily via hardware and software, e.g., by adopting DNSSEC and leveraging its integrity guarantees. Similar to our technique, typo squatting can be used to lure clients on malicious websites [64–66]. It remains important to note that in a typo-squatting attack, the attacker needs to register a new domain and hope that users visit that domain. For our attack, although the window of opportunity might be shorter, the attack is significantly more severe: it is impossible for users to tell whether they are in fact being attacked, as domains and IP addresses have residual trust, and any connection might be marked trusted by the browser due to domain-validated SSL certificates. Indeed, Zdrnja et al. demonstrated an approach to detect typo-squatting attacks from mined DNS data [67]. Different from prior work, our study focuses on the vulnerabilities of stale DNS records pointing to cloud IP addresses, we conduct comprehensive measurements, and we propose a mitigation to retain the convenience of domain validation for certificate issuance.

[7]The noise-to-signal ratio might impractical for monitoring because of Internet-wide scanning efforts, and filtering scanning traffic from other traffic might be too costly for a supplemental warning service.

B. IP Address Squatting

Taking over IP addresses has been a well-known problem in security. The most common and well discussed attack method aims to take over entire network prefixes using BGP, which can be easily observed and will be scrutinized quickly [68]. Wählisch et al. demonstrated a method to detect such takeovers using RPKI [69]. Ballani et al. conducted a study investigating prefix hijacking in 2007 [70], while Zhang et al. developed first defense methods against such attacks [71]. In 2015, Gavrichenkov demonstrated that modern domain validated SSL certificates (and thereby HTTPS in general) can be broken using prefix hijacking [44]. Attackers with more powerful capabilities on the network path between a client requesting a certificate and a CA do not even have to perform prefix hijacking, but instead can easily exploit IP address squatting, as they are already on the path. Our work, on the other hand, details a new attack vector to conduct IP address squatting, which is practical, and time and cost efficient to launch.

C. Certificate Validation Security

The security threats we studied in this paper tie in with modern, domain-based, certificate authorities and their surrounding security challenges. Various efforts currently track the adoption of Let's Encrypt [72, 73]. In general, the security implications of domain-based certificate validation are widely accepted. In their comprehensive analysis of the HTTPS/SSL trust ecosystem, Clark et al. [74] place great trust in DANE [53], to mitigate this issue. Apart from DANE, Certificate Transparency [26, 27] is considered the ideal mitigation for maliciously and wrongfully obtained certificates and has received significant attention recently. The DNS certificate authority authorization (CAA) record might reduce the impact of IP use-after-free attacks to some degree [52], as it limits the CAs that are allowed to issue a certificate for a specific domain, and, thus, force an attacker to request a certificate from these CAs. However, our analysis shows that current domain validation in trust ecosystem is susceptible to use-after-free attacks regardless of CAA records. In fact, the only way to defend against use-after-free attacks through CAA is to restrict certificate issuance in its entirety, which then raises problems when the certificate expires while also relying on automatic certificate renewal setups, such as those recommended by Let's Encrypt, in which case automatic DNS zone updates are required (which become difficult in the presence of DNSSEC). Overall, relying on CAA would require numerous compromises in terms of certificate lifetime management and DNS zone maintenance, while still providing a potential (small) window of opportunity of an attacker whenever the CAA record needs to be relaxed to allow certificate renewal. We

introduced a mitigation that incorporates existing trust of a name into the validation process and can protect against these attacks.

D. Cloud Security

Concurrent with the increasing adopting of cloud services, cloud security has drawn more research attention. Chen et al. provided a contemporary summary and analysis of cloud security issues [75], and indicated problems of shared resources. Similarly, Subashini and Kavitha provided a comprehensive analysis of security challenges in cloud scenarios [76]. Their analysis of IaaS platforms only includes similar issues to those approached by Ristenpart et al. [4]. Specifically, Ristenpart et al. exploit shared resources in IaaS environments to facilitate cross-VM side-channel attacks. However, they focus on physical computing resources and they do not investigate issues induced by logical resource sharing, e.g., access to the same IP address pool. Jensen et al. focus on classical web attacks, especially in SaaS (Software-as-a-Service) scenarios [77]. Takabi et al. discuss the overall issue of IP squatting that is related to secure handling of provisioning and multi-domain cloud platforms with shared resource pools [78]. Zhang et al. investigate access control and trust management in the context of multi-tentant environments [79]. Our work and mitigation approaches are orthogonal to prior cloud security research, and we focus on the certificate ecosystem vulnerabilities as it is being used in combination with cloud services.

VI. Conclusion

In this paper, we have shown that it is practical, time-efficient, and cost-efficient for an attacker to (temporarily) takeover domains by exploiting so-called IP address use-after-free vulnerabilities on, currently, the two largest Infrastructure-as-a-Service clouds (Amazon AWS and Microsoft Azure).

In our study, we discovered that attacks are practical on public clouds because of their instances' ephemeral nature and the "throw-away culture" of development operations concerning immutable instances and service migration. In turn, it is not necessary to takeover the IP address to which a domain points to, but IP address migration occurs regularly and sometimes is outside of the control of the cloud user (e.g., reboot or shutdown of the hypervisor because of an update), thus freeing the previously assigned IP address and making it available for re-use by others. Here, a slightly incorrect DNS domain record migration strategy can immediately render domains vulnerable to IP address use-after-free attacks. In fact, the problem is even further amplified for so-called "spot" instances, which are significantly cheaper instances, but which can be terminated at any point and without notice to the cloud user, and for which he cannot protect himself from temporary domain takeovers.

We have examined the reasons of why and how IP address re-use domain takeover attacks can occur in practice, and we classify them according to what their potential impact in practice is. Particularly, we investigated their impact on domain-validated SSL certificate issuance, such as through automatic certificate management environments (ACME), e.g., Let's Encrypt. Based on our findings, we then developed best practice recommendations for cloud operators as well as domain owners and cloud users, which can reduce vulnerability to the aforementioned attacks.

Finally, we introduced a new mitigation techniques that addresses the issue of domain takeover attacks for trust-based domain-validation services, focusing on the real-world case of automatic certificate issuance. Our mitigation technique protects against IP address use-after-free attacks with negligible operational overhead and only requires manual intervention in disaster-recovery scenarios, thus, rendering it practical for real-world deployment even under strict performance and usability requirements of services like Let's Encrypt.

Acknowledgements

We thank the anonymous reviewers for their helpful suggestions to improve the paper. We also thank David Choffnes and Martina Lindorfer for their valuable feedback.

This material is based on research sponsored by the Defense Advanced Research Projects Agency (DARPA) under agreement number FA8750-15-2-0084, the Office of Naval Research (ONR) under grant N00014-17-1-2011 and N00014-15-1-2948, the National Science Foundation (NSF) under grant DGE-1623246 and CNS-1704253, and a Google Security, Privacy and Anti-Abuse Award to Giovanni Vigna.

The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

Any views, opinions, findings, recommendations, or conclusions contained or expressed herein are those of the authors, and do not necessarily reflect the position, official policies, or endorsements, either expressed or implied, the U.S. Government, DARPA, ONR, NSF, or Google.

References

[1] Ingrid Lunden. *Amazon's AWS Is Now A $7.3B Business As It Passes 1M Active Enterprise Customers*. Oct. 2015. URL: https://techcrunch.com/2015/10/07/amazons-aws-is-now-a-7-3b-business-as-it-passes-1m-active-enterprise-customers/.

[2] Haje Jan Kamps. *Microsoft Celebrates Strong Azure Adoption at Build 2016*. Mar. 2016. URL: https://techcrunch.com/2016/03/31/azure-growth/.

[3] C. Public. *Cisco Global Cloud Index: Forecast and Methodology, 20152020*. White paper. 2016.

[4] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. "Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-party Compute Clouds". In: *Proc. ACM Conference on Computer and Communications Security (CCS)*. 2009.

[5] K. Razavi, B. Gras, E. Bosman, B. Preneel, C. Giuffrida, and H. Bos. "Flip Feng Shui: Hammering a Needle in the Software Stack". In: *Proc. USENIX Security Symposium (SEC)*. 2016.

[6] G. I. Zineb Ait Bahajji. *HTTPS as a ranking signal*. Aug. 2014. URL: https://webmasters.googleblog.com/2014/08/https-as-ranking-signal.html.

[7] K. Basques. *Why HTTPS Matters*. Sept. 2017. URL: https://developers.google.com/web/fundamentals/security/encrypt-in-transit/why-https.

[8] P. Venezia. *Code injection: A new low for ISPs*. May 2015. URL: http://www.infoworld.com/article/2925839/net-neutrality/code-injection-new-low-isps.html.

[9] E. Mill. *The Web Is Deprecating HTTP And It's Going To Be Okay*. May 2015. URL: https://motherboard.vice.com/en_us/article/wnjyay/the-web-is-deprecating-http-and-its-going-to-be-okay.

[10] D. Stenberg. *TLS in HTTP/2*. Mar. 2015. URL: https://daniel.haxx.se/blog/2015/03/06/tls-in-http2/.

[11] R. Barnes, J. Hoffman-Andrews, and J. Kasten. *Automatic Certificate Management Environment (ACME)*. Internet-Draft draft-ietf-acme-acme-latest. Work in Progress. Internet Engineering Task Force, June 2017. URL: https://ietf-wg-acme.github.io/acme/draft-ietf-acme-acme.html.

[12] Josh Aas. *Milestone: 100 Million Certificates Issued*. June 2017. URL: https://letsencrypt.org//2017/06/28/hundred-million-certs.html.

[13] Dan Cvrcek. *Lets Encrypt in the spotlight*. June 2017. URL: https://dan.enigmabridge.com/lets-encrypt-in-the-spotlight/.

[14] K. Borgolte, C. Kruegel, and G. Vigna. "Meerkat: Detecting Website Defacements through Image-based Object Recognition". In: *Proc. USENIX Security Symposium (SEC)*. Vol. 24. Aug. 2015.

[15] N. Nikiforakis, L. Invernizzi, A. Kapravelos, S. V. Acker, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. "You Are What You Include: Large-scale Evaluation of Remote JavaScript Inclusions". In: *Proc. ACM Conference on Computer and Communications Security (CCS)*. 2012.

[16] D. Kumar, Z. Ma, Z. Durumeric, A. Mirian, J. Mason, J. A. Halderman, and M. Bailey. "Security Challenges in an Increasingly Tangled Web". In: *Proc. World Wide Web Conference*. 2017.

[17] P. Mockapetris. *Domain Names - Implementation and Specification*. RFC 1035 (Internet Standard). RFC. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2673, 2845, 3425, 3658, 4033, 4034, 4035, 4343, 5936, 5966, 6604, 7766. RFC Editor, Nov. 1987. URL: https://www.rfc-editor.org/rfc/rfc1035.txt.

[18] S. Weiler and J. Ihren. *Minimally Covering NSEC Records and DNSSEC On-line Signing*. RFC 4470 (Proposed Standard). RFC. RFC Editor, Apr. 2006. URL: https://www.rfc-editor.org/rfc/rfc4470.txt.

[19] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. *DNS Security Introduction and Requirements*. RFC 4033 (Proposed Standard). RFC. Updated by RFCs 6014, 6840. RFC Editor, Mar. 2005. URL: https://www.rfc-editor.org/rfc/rfc4033.txt.

[20] O. Kolkman, W. Mekking, and R. Gieben. *DNSSEC Operational Practices, Version 2*. RFC 6781 (Informational). RFC. RFC Editor, Dec. 2012. URL: https://www.rfc-editor.org/rfc/rfc6781.txt.

[21] T. Chung, R. van Rijswijk-Deij, B. Chandrasekaran, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson. "A Longitudinal, End-to-End View of the DNSSEC Ecosystem". In: *Proc. USENIX Security Symposium (SEC)*. 2017.

[22] P. Mell, T. Grance, et al. "The NIST Definition of Cloud Computing". In: (2011).

[23] CAcert. *Welcome to CAcert*. URL: http://www.cacert.org/.

[24] J. Prins and B. U. Cybercrime. *DigiNotar Certificate Authority Breach Operation Black Tulip*. 2011.

[25] B. Budington. *Symantec Issues Rogue EV Certificate for Google.com*. 2015. URL: https://www.eff.org/deeplinks/2015/09/symantec-issuesrogue-ev-certificate-googlecom.

[26] B. Laurie. "Certificate Transparency". In: *Queue* 12.8 (2014).

[27] B. Laurie, A. Langley, and E. Kasper. *Certificate Transparency*. RFC 6962 (Experimental). RFC. RFC Editor, June 2013. URL: https://www.rfc-editor.org/rfc/rfc6962.txt.

[28] J. Aas. *Why ninety-day lifetimes for certificates?* Nov. 2015. URL: https://letsencrypt.org/2015/11/09/why-90-days.html.

[29] M. Aertsen, M. Korczyski, G. Moura, S. Tajalizadehkhoob, and J. van den Berg. "No domain left behind: is Let's Encrypt democratizing encryption?" In: *Proc. of the ACM Applied Networking Research Workshop (ANRW)*. 2017.

[30] K. Borgolte, C. Kruegel, and G. Vigna. "Delta: Automatic Identification of Unknown Web-based Infection Campaigns". In: *Proc. ACM Conference on Computer and Communications Security (CCS)*. Vol. 20. Nov. 2013.

[31] M. Neugschwandtner, M. Lindorfer, and C. Platzer. "A View To A Kill: WebView Exploitation". In: *Proceedings of the 6th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*. 2013.

[32] T. Luo, H. Hao, W. Du, Y. Wang, and H. Yin. "Attacks on WebView in the Android System". In: *Proc. ACM Annual Computer Security Applications Conference (ACSAC)*. 2011.

[33] M. Bryant. *The .io Error Taking Control of All .io Domains With a Targeted Registration*. July 2017. URL: https://thehackerblog.com/the-io-error-taking-control-of-all-io-domains-with-a-targeted-registration/.

[34] B. Krebs. *FBI: $2.3 Billion Lost to CEO Email Scams*. Apr. 2016. URL: https://krebsonsecurity.com/2016/04/fbi-2-3-billion-lost-to-ceo-email-scams/.

[35] S. Hao, K. Borgolte, N. Nikiforakis, G. Stringhini, M. Egele, M. Eubanks, B. Krebs, and G. Vigna. "Drops for Stuff: An Analysis of Reshipping Mule Scams". In: *Proc. ACM Conference on Computer and Communications Security (CCS)*. Vol. 22. Oct. 2015.

[36] A. Swinnen. *Authentication Bypass on Uber's Single Sign-On via Subdomain Takeover*. June 2017. URL: https://www.arneswinnen.net/2017/06/authentication-bypass-on-ubers-sso-via-subdomain-takeover/.

[37] J. Aas. *Wildcard Certificates Coming January 2018*. July 2017. URL: https://letsencrypt.org//2017/07/06/wildcard-certificates-coming-jan-2018.html.

[38] J. Pang, A. Akella, A. Shaikh, B. Krishnamurthy, and S. Seshan. "On the Responsiveness of DNS-based Network Control". In: *Proc. ACM Internet Measurement Conference (IMC)*. 2004.

[39] C. Coles. *AWS vs Azure vs Google Cloud Market Share 2017*. URL: https://www.skyhighnetworks.com/cloud-security-blog/microsoft-azure-closes-iaas-adoption-gap-with-amazon-aws/.

[40] Amazon Web Services, Inc. *Throttle API Requests for Better Throughput*. Aug. 2017. URL: http://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-request-throttling.html.

[41] Farsight Inc. *Farsight - Security Information Exchange (SIE)*. URL: https://www.farsightsecurity.com/solutions/security-information-exchange/.

[42] M. Alllman and V. Paxson. "Issues and Etiquette Concerning Use of Shared Measurement Data". In: *Proc. ACM Internet Measurement Conference (IMC)*. 2007.

[43] Network Sorcery Inc. *Well known SCTP, TCP and UDP ports*. URL: http://www.networksorcery.com/enp/protocol/ip/ports00000.htm.

[44] A. Gavrichenkov. "Breaking HTTPS with BGP Hijacking". In: *Black-Hat Briefings* (2015).

[45] S. Helme. *Revocation is broken*. June 2017. URL: https://scotthelme.co.uk/revocation-is-broken/.

[46] A. Langley. *No, don't enable revocation checking*. Apr. 2014. URL: https://www.imperialviolet.org/2014/04/19/revchecking.html.

[47] J. Larisch, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson. "CRLite: A Scalable System for Pushing All TLS Revocations to All Browsers". In: *Proc. IEEE Security & Privacy*. 2017.

[48] The Chromium Project. *The Chromium Project: CRLSets*. URL: https://dev.chromium.org/Home/chromium-security/crlsets.

[49] M. Goodwin. *Revoking Intermediate Certificates: Introducing OneCRL*. 2014. URL: https://blog.mozilla.org/security/2015/03/03/revoking-intermediate-certificates-introducing-onecrl/.

[50] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov. "The Most Dangerous Code in the World: Validating SSL Certificates in Non-browser Software". In: *Proc. ACM Conference on Computer and Communications Security (CCS)*. 2012. URL: http://doi.acm.org/10.1145/2382196.2382204.

[51] R. Barnes, J. Hoffman-Andrews, and J. Kasten. *Automatic Certificate Management Environment (ACME)*. Internet-Draft draft-ietf-acme-acme-07. http://www.ietf.org/internet-drafts/draft-ietf-acme-acme-07.txt. IETF Secretariat, June 2017. URL: http://www.ietf.org/internet-drafts/draft-ietf-acme-acme-07.txt.

[52] P. Hallam-Baker and R. Stradling. *DNS Certification Authority Authorization (CAA) Resource Record*. RFC 6844 (Proposed Standard). RFC. RFC Editor, Jan. 2013. URL: https://www.rfc-editor.org/rfc/rfc6844.txt.

[53] P. Hoffman and J. Schlyter. *The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA*. RFC 6698 (Proposed Standard). RFC. Updated by RFCs 7218, 7671. RFC Editor, Aug. 2012. URL: https://www.rfc-editor.org/rfc/rfc6698.txt.

[54] PowerDNS. *PowerDNS Online Signing*. URL: https://doc.powerdns.com/md/authoritative/dnssec/#online-signing.

[55] G. C. Team. *Certificate Transparency in Chrome*. May 2016. URL: https://github.com/GoogleChrome/ct-policy/blob/master/ct_policy.md.

[56] K. Hall. *[cabfpub] Results on Ballot 187 - Make CAA Checking Mandatory*. Mar. 2017. URL: https://cabforum.org/pipermail/public/2017-March/009988.html.

[57] T. Fiebig, F. Lichtblau, F. Streibelt, T. Krueger, P. Lexis, R. Bush, and A. Feldmann. "SoK: An Analysis of Protocol Design: Avoiding Traps for Implementation and Deployment". In: *arXiv preprint arXiv:1610.05531* (2016).

[58] T. Ylonen. “SSH–secure login connections over the Internet”. In: *Proc. USENIX Security Symposium (SEC)*. Vol. 37. 1996.

[59] *Host Identity Takeover Using Virtual Internet Protocol (IP) Addressing*.

[60] S. Yadav, A. K. K. Reddy, A. N. Reddy, and S. Ranjan. “Detecting Algorithmically Generated Domain-Flux Attacks with DNS Traffic Analysis”. In: *IEEE/ACM Trans. Networking (TON)* 20.5 (2012).

[61] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. “Your Botnet is My Botnet: Analysis of a Botnet Takeover”. In: *Proc. ACM Conference on Computer and Communications Security (CCS)*. 2009.

[62] D. Liu, S. Hao, and H. Wang. “All Your DNS Records Point to Us: Understanding the Security Threats of Dangling DNS Records”. In: *Proc. ACM Conference on Computer and Communications Security (CCS)*. 2016.

[63] N. Nikiforakis, S. Van Acker, W. Meert, L. Desmet, F. Piessens, and W. Joosen. “Bitsquatting: Exploiting Bit-flips for Fun, or Profit?” In: *World Wide Web*. 2013.

[64] Y.-M. Wang, D. Beck, J. Wang, C. Verbowski, and B. Daniels. “Strider Typo-Patrol: Discovery and Analysis of Systematic Typo-Squatting”. In: *SRUTI* 6 (2006).

[65] J. Szurdi, B. Kocso, G. Cseh, J. Spring, M. Felegyhazi, and C. Kanich. “The Long “Taile” of Typosquatting Domain Names”. In: *Proc. USENIX Security Symposium (SEC)*. 2014.

[66] M. T. Khan, X. Huo, Z. Li, and C. Kanich. “Every Second Counts: Quantifying the Negative Externalities of Cybercrime via Typosquatting”. In: *Proc. IEEE Security & Privacy*. 2015.

[67] B. Zdrnja, N. Brownlee, and D. Wessels. “Passive Monitoring of DNS Anomalies”. In: *Proc. SIG SIDAR Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*. Springer. 2007.

[68] H. Yan, R. Oliveira, K. Burnett, D. Matthews, L. Zhang, and D. Massey. “BGPmon: A real-time, scalable, extensible monitoring system”. In: *Proc. IEEE Conference For Homeland Security—Cybersecurity Applications & Technology (CATCH)*. 2009.

[69] M. Wählisch, O. Maennel, and T. C. Schmidt. “Towards Detecting BGP Route Hijacking Using the RPKI”. In: *ACM SIGCOMM Computer Communication Review* 42.4 (2012).

[70] H. Ballani, P. Francis, and X. Zhang. “A Study of Prefix Hijacking and Interception in the Internet”. In: *ACM SIGCOMM Computer Communication Review*. Vol. 37. 4. 2007.

[71] Z. Zhang, Y. Zhang, Y. C. Hu, and Z. M. Mao. “Practical Defenses Against BGP Prefix Hijacking”. In: *Proc. ACM CoNEXT*. 2007.

[72] M. Aertsen, M. Korczyski, G. Moura, S. Tajalizadehkhoob, and J. v. d. Berg. “No Domain Left Behind: Is Let’s Encrypt democratizing Encryption?” In: *arXiv preprint arXiv:1612.03005* (2016).

[73] A. Manousis, R. Ragsdale, B. Draffin, A. Agrawal, and V. Sekar. “Shedding Light on the Adoption of Let’s Encrypt”. In: *arXiv preprint arXiv:1611.00469* (2016).

[74] J. Clark and P. C. van Oorschot. “SoK: SSL and HTTPS: Revisiting Past Challenges and Evaluating Certificate Trust Model Enhancements”. In: *Proc. IEEE Security & Privacy*. 2013.

[75] Y. Chen, V. Paxson, and R. H. Katz. “What’s New About Cloud Computing Security”. In: *University of California, Berkeley Report No. UCB/EECS-2010-5 January* 20.2010 (2010).

[76] S. Subashini and V. Kavitha. “A Survey on Security Issues in Service Delivery Models of Cloud Computing”. In: *Journal of Network and Computer Applications* 34.1 (2011).

[77] M. Jensen, J. Schwenk, N. Gruschka, and L. L. Iacono. “On Technical Security Issues in Cloud Computing”. In: *Proc. IEEE Conference on Cloud Computing Technology and Science (CloudCom)*. 2009.

[78] H. Takabi, J. B. Joshi, and G.-J. Ahn. “Security and Privacy Challenges in Cloud Computing Environments”. In: *IEEE Security & Privacy* (2010).

[79] Y. Zhang and J. Joshi. *Access Control and Trust Management for Emerging Multidomain Environments*. Emerald Group Publishing, 2009.

APPENDIX

For our proof of concept experiment (see Section III-E), we obtained a valid certificate for the domain “cloud-strife.seclab.cs.ucsb.edu.” The obtained certificate is shown in Listing 2. The respective entry in the certificate transparency log can be found at: https://crt.sh/?id=250959196. We revoked the certificate after the certificate has propagated to certificate transparency logs, i.e., shortly after issuance. In face of often ignored revocation checks, we opt not to publish the private key. Instead, we prove ownership of the certificate by signing a unique message (see Listing 3 and Listing 4). We did not use the certificate for any purpose besides signing the message. It can be verified as follows:

```
# Copy Listing 2 to certificate.pem
# Copy Listing 4 to message.txt.dgst.b64

# Create message.txt
$ echo -n "Cloud Strife: Mitigating the Security Risks of Domain
     ↝  Validated Certificates" > message.txt

# Convert the full certificate to raw PEM:
$ openssl x509 -pubkey -noout -in certificate.pem >
     ↝ certificate_raw.pem

# Base64 decode the signature
$ base64 -d message.txt.dgst.b64 > message.txt.dgst

# Verify the message
$ openssl dgst -sha256 -verify certificate_raw.pem -signature
     ↝ message.txt.dsgt message.txt
```

Listing 1: **Instructions to verify the signature.**

```
-----BEGIN CERTIFICATE-----
MIIFHzCCBAegAwIBAgISA3XAEcaykugGaCy9tCoCdJWKMA0GCSqGSIb3DQEBCwUA
MEoxCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBFbmNyeXB0MSMwIQYDVQQD
ExpMZXQncyBFbmNyeXB0IEF1dGhvcml0eSBYMzAeFw0xNzExMDkyMzA4NTVaFw0x
ODAyMDcyMzA4NTVaMCkxJzAlBgNVBAMTHmNsb3Vkc3RyaWZlLnNlY2xhYi5jcy51
Y3NiLmVkdTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAONF0TzeAA6N
q5Li7e9h6+Y//d8Zy2gbWN465t3MPVlz1lSLqCZvT4e3IDjuyQ/gx+yWnndtQrhs
zHt+GigQbBcAFM5YohIVrTr7M8ozZVZhu1x11xmPZYJ9hAi8NO6p2uoZMNwiHh35
XVFQs5LFG6QpPGBWoNtu1t5zwLYF01STlMS/hNn0P/KlrnAzs2tSX//OxxaY+jos
KQCl9LrXKhOXcmaZMXFe7t8uglFsjbEvM9TRFqeENROik/TLjRlyb3BM5HtKVnno
tDh6078qCgwMzZyh5YRy2uOGHCp13TdZQtOELq0qfGNjVClwRENo+AWlK8fPnw9L
S49OpBwzx2MCAwEAAaOCAh4wggIaMA4GA1UdDwEB/wQEAwIFoDAdBgNVHSUEFjAU
BggrBgEFBQcDAQYIKwYBBQUHAwIwDAYDVR0TAQH/BAIwADAdBgNVHQ4EFgQUKnFO
hVGO9fXAoSDpoRiztZhSYo4wHwYDVR0jBBgwFoAUqEpqYwR93brm0Tm3pkVl7/Oo
7KEwbwYIKwYBBQUHAQEEYzBhMC4GCCsGAQUFBzABhiJodHRwOi8vb2NzcC5pbnQt
eDMubGV0c2VuY3J5cHQub3JnMC8GCCsGAQUFBzAChiNodHRwOi8vY2VydC5pbnQt
eDMubGV0c2VuY3J5cHQub3JnLzApBgNVHREEIjAggh5jbG91ZHN0cmlmZS5zZWNs
YWIuY3MudWNzYi5lZHUwgf4GA1UdIASB9jCB8zAIBgZngQwBAgEwgeYGCysGAQQB
gt8TAQEBMIHWMCYGCCsGAQUFBwIBFhpodHRwOi8vY3BzLmxldHNlbmNyeXB0Lm9y
ZzCBqwYIKwYBBQUHAgIwgZ4MgZtUaGlzIENlcnRpZmljYXRlIG1heSBvbmx5IGJl
IHJlbGllZCB1cG9uIGJ5IFJlbHlpbmcgUGFydGllcyBhbmQgb25seSBpbiBhY2Nv
cmRhbmNlIHdpdGggdGhlIEN1cnRpZmljYXRlIFBvbGljeSBmb3VuZCBhdCBodHRw
czovL2xldHNlbmNyeXB0Lm9yZy9yZXBvc2l0b3J5LzANBgkqhkiG9w0BAQsFAAOC
AQEAIj1W4ZzHlsaj6ccWccGyVahfk9JDhImMQLDUR02FYqtHLPjyM1JIIyYHP9xE
S2JZBbzMlrr2SjfxC3IQhDkUIjyPEeLv6WVT0hFbbzu3QAYjW5yigctpuggx/v7c
rhbWpmY9TJRU2QAsADF9NIeSXo+3zp15QAvrss2l+qtEK3uLgQ12+antYaI85wkc
P6MGHVV52asshcjy+v2wHxJDONmtzCHQbYXA7nhSUfspnVax8EfraGWF5XobZyLw
p91BZjOB1D+HD3ubtbk2PjlW/Eld7jgv2pCEM0iXk5suidCnG47jmZQA892iUVVf
tx4z5/ntnkiw7Gwwzm+o34fMmQ==
-----END CERTIFICATE-----
```

Listing 2: **Proof of concept certificate, signed by Let’s Encrypt.**

```
Cloud Strife: Mitigating the Security Risks of Domain Validated
     ↝ Certificates
```

Listing 3: **Proof of concept message (one line, no trailing new line).**

```
Bc99Sl5FwjqYLJl/jS1gPC9fyI9XiS/ex7QVg+zIFZpJ+aPCYcsGm4fGkJxathte
w4i0p3q3lSmnkukRoRNVSvMJdfJRm5QvRQr43HsC6iT+N2xZI/QLcH0nMGUftpR2
HuEiY8LwIalNuxOOjTZJwfTTSRM+NdCjSa39RDpqQLU5LGKjBpSTT/jfg0RwrX0w
MhDnq+iqqrW0kDg08bxARWUfY7tHUAvPpiyyEhnfyThliHFkrKUjAGtH6f+6fKFe
8pZO0XJHRoMuhq4OXMjOWKJZYu7XwQXn3GDoo1bwIwykwmIpUu9wGAjlimtTY5eW
uM0tg2PkmbuZi3JaGsczuQ==
```

Listing 4: **Signature for the proof of concept message.**