Measuring and Applying Invalid SSL Certificates: The Silent Majority

Taejoong Chung* Yabing Liu* David Choffnes* Dave Levin†
Bruce M. Maggs‡ Alan Mislove* Christo Wilson*

*Northeastern University †University of Maryland ‡Duke University and Akamai Technologies

ABSTRACT

SSL and TLS are used to secure the most commonly-used Internet protocols. As a result, the ecosystem of SSL certificates has been thoroughly studied, leading to a broad understanding of the strengths and weaknesses of the certificates accepted by most web browsers. Prior work has naturally focused almost exclusively on "valid" certificates—those that standard browsers accept as well-formed and trusted—and has largely disregarded certificates that are otherwise "invalid." Surprisingly, however, this leaves the majority of certificates unexamined: we find that, on average, *65% of SSL certificates advertised in each IPv4 scan that we examine are actually invalid.*

In this paper, we demonstrate that despite their invalidity, much can be understood from these certificates. Specifically, we show why the web's SSL ecosystem is populated by so many invalid certificates, where they originate from, and how they impact security. Using a dataset of over 80M certificates, we determine that most invalid certificates originate from a few types of end-user devices, and possess dramatically different properties than their valid counterparts. We find that many of these devices periodically reissue their (invalid) certificates, and develop new techniques that allow us to track these reissues across scans. We present evidence that this technique allows us to uniquely track over 6.7M devices. Taken together, our results open up a heretofore largely-ignored portion of the SSL ecosystem to further study.

IMC 2016, November 14–16, 2016, Santa Monica, CA, USA

ISBN 978-1-4503-4526-2/16/11…$15.00

DOI: http://dx.doi.org/10.1145/2987443.2987454

1. INTRODUCTION

Secure Sockets Layer (SSL) and Transport Layer Security (TLS)[1] are responsible for securing Internet traffic for a variety of common protocols (HTTP, SMTP, IMAP, etc.). Coupled with a Public Key Infrastructure (PKI), SSL provides authenticated identities via certificate chains and private communication via encryption.

The web's SSL certificate ecosystem has been studied extensively [13, 14, 17, 24, 28, 50], with the broad goal of better understanding how resilient websites and browsers are to attacks on end-to-end authentication and confidentiality. As such, the vast majority of these studies (we discuss some exceptions in §3) naturally focus on the *valid* certificates found on the web, that is, the certificates that are well-formed, are within their validity periods, have a certificate chain that verifies at each level, and are rooted in a widely-trusted set of root certificates [8]. The prior studies focused almost exclusively on valid certificates because, after all, if a certificate is not valid, one cannot confidently attribute it to the websites under study.

In this paper, we take another look at the SSL certificate ecosystem, focusing not only on the valid certificates, but also the *invalid* ones. Using a dataset of over 80M certificates collected from 222 full IPv4 scans over three years, we find, surprisingly, that *almost 88% of certificates we observe across all these IPv4-wide scans are invalid.* In other words, most prior studies of the SSL certificate ecosystem have focused on a mere 12% of the overall space of SSL certificates. The broad goal of this paper is to understand why so much of the web's PKI consists of invalid certificates, to evaluate from where these certificates originate, to understand the security implications of these certificates, and to demonstrate the value in this long-overlooked portion of the certificate ecosystem.

We make the following contributions. *First*, we perform a study of all invalid SSL certificates collected from full IPv4 port 443 scans over three years. We find that invalid certificates have considerable differences from

[1] TLS is the successor of SSL, but both use the same certificates. We refer to "SSL certificates," but our findings apply equally to both.

their valid counterparts in terms of validity periods, lifetimes, expiration dates, and sharing of public keys.

Second, we evaluate the origins of these invalid certificates, and find that they largely originate from users' end-devices, including wireless access points, printers, VoIP phones, and cable modems (for which the certificates are used to enable "secure" remote administration). Moreover, we find that invalid certificates tend to be advertised by many fewer hosts, and that they are advertised in a very different portion of the IP address space than valid certificates.

Third, we find that many of these devices periodically reissue new (invalid) certificates, and we evaluate the behavior of such reissues. Tracking reissues of invalid certificates proves to be considerably more difficult than for valid ones, as invalid certificates often have non-unique `Common Name`s or modify their `Common Name` on each reissue (whereas a valid website generally maintains its domain name as its `Common Name` in all of its certificates). We develop a set of techniques that allow us to *link* multiple invalid certificates that are likely to come from a single device.

Fourth, applying our techniques to link together different certificates, we demonstrate that invalid certificates can be used as a means to track millions of user devices as they change IP addresses. Our techniques offer a complementary view to those provided by other device-tracking schemes [1, 2, 37, 44] in that our techniques do not require us to recruit users, and can be performed at scale; we show that we are able to track 6.7M unique end-user devices for over a year.

We make all of our code and data publicly available to the research community at

`https://securepki.org`

The remainder of this paper is organized as follows: we provide background in §2 and an overview of related work in §3. We describe our dataset and methodology in §4. In §5, we evaluate the properties of the invalid certificates and compare them to those of valid certificates. We present and evaluate our methodology for detecting reissues of invalid certificates in §6. In §7, we explore using our techniques to track end-user devices, and we conclude in §8.

2. BACKGROUND

SSL and TLS now secure the vast majority of online communication. Combined with the use of a PKI, they provide authentication between clients and servers, and guarantee the privacy of the communication.

SSL certificates. A certificate is a signed attestation binding a *subject* (called a `Common Name` in practice) to a *public key*. Typically, SSL certificates are issued by a trusted Certificate Authority (CA), which has its own certificate(s); these CA certificates are further signed by other CA certificates, terminating at a small set of self-signed *root certificates*. Thus, there is a logical *chain* of certificates, starting from a *root* certificate through zero or more *intermediate* certificates, to a *leaf* certificate. Each certificate is signed with the private key corresponding to the certificate in the higher level, except the self-signed *root* certificate. When a client connects to an HTTPS-secured site, it must verify that the certificate advertised by the server is valid.

On the Internet, X.509 [8] is the most commonly used certificate management standard. X.509 certificates typically include a subject and public key, a serial number (unique for the issuer), a validity period, acceptable usage of the key, and ways to check whether the certificate has been revoked [28].

Invalid certificates. The X.509 RFC [8] defines a certificate as *invalid* if a client is unable to validate it at some point in time. There are multiple reasons that a client could find a certificate to be invalid: it could be outside of its validity period, it could have been revoked by its CA, its subject could be incorrect, its signature could be wrong, and so on. Because our dataset spans years (§4), we define a certificate as invalid if no client with a standard set of root certificates would *ever* be able to validate it (i.e., we ignore expiry warnings). The most common reason for invalidity that we have observed is certificates signed by an unknown or untrusted root; if the client does not trust the root of a certificate chain, it transitively does not trust the rest of the chain. Specifically, in our dataset, we found that 88.0% of invalid certificates are self-signed (i.e., the root of the chain is the leaf certificate itself) and a further 11.99% are signed by a different, untrusted certificate (i.e., the root of the chain is some other certificate that is not in the set of trusted root certificates).[2]

Internet-connected devices. Internet-connected devices are widely popular today, including end-user routers, printers, cable/DSL modems, IP cameras, VoIP telephones, thermostats, and network-attached storage devices. Many of these devices provide a web server to allow end users to access and manage the device. A recent trend is to enable both HTTP and HTTPS versions of this web server; devices that do so need an SSL certificate for the HTTPS site.

While some devices allow users to upload a certificate, we find that most generate and use an invalid certificate by default. There are several reasons for this behavior. *First*, until recently [29], obtaining valid SSL certificates cost money. Given that many of these devices are not expensive, providing valid SSL certificates might substantially raise costs. *Second*, the `Common Name` of HTTPS certificates are domain names or IP addresses. However, not all users have a domain name or a static IP address to provide for a certificate. *Third*, using an invalid certificate allows the device to function "out of the box," as most users tend to click through "invalid cer-

[2] The other 0.01% are found invalid predominantly due to signature errors and `openssl` parsing errors.

tificate" warnings presented by their browsers [3]; this unfortunately results in weaker security.[3]

3. RELATED WORK

In this section, we discuss related studies of the SSL certificate ecosystem. As we show here, however, we are the first to study invalid certificates in depth.

SSL certificate ecosystem. There is a long thread of work on the SSL certificate ecosystem, ranging from measurements of CAs, certificates they issued, and client root stores [14, 24, 36, 47] to new techniques that can improve the existing SSL ecosystem [27, 32, 39, 42] to alternate architectures to the current CA-based systems [7, 12, 45]. Several closely related papers [13, 28, 49, 50] have explored the patterns of reissuing and revoking certificates, many using the Heartbleed vulnerability incident as a way to obtain visibility into system administrators' behavior.

Our work complements these, as they are focused largely on *valid* certificates. Instead, we focus primarily on the *invalid* certificates, explore how their properties differ, and understand why the ecosystem is dominated by invalid, rather than valid, certificates. Moreover, the dataset we use is considerably broader, including all certificates advertised on public IPv4 address over three years.

Invalid certificates. Many of the studies of the SSL ecosystem discussed above also briefly looked at invalid certificates. For example, Holz et al. [24] demonstrated in 2011 that 40% of certificates were invalid, and Durumeric et al. [14] demonstrated in 2013 that this fraction had grown to 60%. Neither study, however, focused extensively on such invalid certificates (e.g., Durumeric et al. noted that invalid certificates tend to have shorter lifetimes, but did not explore why this was the case).

There has been extensive work on examining the SSL certificate validation code implemented by different non-browser software and native mobile applications [18, 19, 21]. These efforts have demonstrated that many implementations silently accept invalid certificates. Similarly, our recent work [28] has discovered several bugs and omissions when major browsers and operating systems attempt to validate revoked certificates. Another study [3] has found that many users ignore browsers' warnings that invalid certificates are encountered. Huang et al. studied the usage of forged SSL certificates [23], which have been used in man-in-the-middle attacks on Facebook. They showed that 0.2% of the SSL connections analyzed were tampered with forged SSL certificates, mostly by anti-virus software and corporate content filters. Taken together, these results show that, although invalid certificates are ignored by most studies of the SSL ecosystem, they are not ignored by all browsers and users. In this paper, we seek to understand where invalid certificates come from, what role they play in today's SSL ecosystem, and how they can be used as a measurement tool.

4. DATA AND METHODOLOGY

In this section, we describe the datasets we collected and our methodology to isolate the invalid certificates used in the remainder of this study.

4.1 Certificate Dataset

We obtain our collection of SSL certificates from two sets of full IPv4 port 443 scans. Our first dataset was collected by the University of Michigan [46], with 156 scans conducted between June 10, 2012 and January 29, 2014. These scans were not conducted at regular intervals: while there were an average 3.83 days between scans, there were periods of up to 24 days with no scans at all, as well as a set of 42 sequential days during which we have daily scans. Our second dataset was collected by Rapid7 [41], with 74 scans conducted between October 30, 2013 and March 30, 2015 (an average of 7.73 days between scans). Unlike the University of Michigan scans, the Rapid7 scans were almost always conducted seven days apart. On eight days, both datasets have scans, resulting in scans on 222 unique days.

The scans found an average of 28M unique IP addresses responding to SSL handshakes per scan, and a total of 192M unique IP addresses responding to SSL handshakes across all scans (4.49% of the entire IPv4 address space). Across all scans, we observe 80,366,826 unique certificates; 39,147,006 of these are version 3 leaf certificates, 28,997,853 are version 3 CA certificates, and 12,132,294 are version 1 certificates.[4],[5]

Dataset inconsistency. While both datasets claim to be full IPv4 scans, we observed that the two scans actually had different sizes: the Rapid7 scans consistently contained approximately 20% fewer IP addresses than the University of Michigan scans. Looking more closely, we found that the Rapid7 scans were not a strict subset of the University of Michigan scans: there were also a significant number of IP addresses that *only* appeared in the Rapid7 scans. Figure 1 selects one of the days where *both* data sources have a scan, and plots the fraction of hosts in each /8 network that only appear in one of the scans. We can immediately observe that the "missing" hosts from each scan appear to be spread across the entire IP space.[6] After communicating this observation to Rapid7 [25], they were unable to track down the source of this discrepancy.

[3] Invalid certificates make it significantly easier for an attacker to conduct a man-in-the-middle attack, as the browser is unable to verify it is communicating directly with the device.

[4] Valid SSL version 1 certificates are largely deprecated as they cannot distinguish between leaf and CA certificates; the only valid version 1 certificates are a few older root certificates.

[5] We also found 89,667 certificates that contain invalid version numbers, including 2, 4, and 13. We disregarded such certificates.

[6] We provide a more detailed examination of the inconsistency at the level of /24 prefixes at `https://securepki.org`.

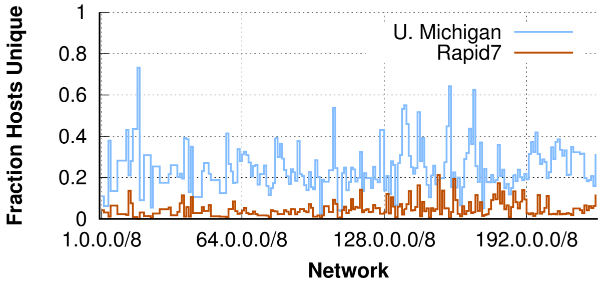

Figure 1: The fraction of hosts unique to each scan, for each /8 network, on a day with both a University of Michigan and a Rapid7 scan. The "missing" hosts in each scan appear to be spread throughout the IP space.

To explore this discrepancy further, we first examine whether the discrepancy might be explained by *blacklisting* by either the scan operators or the target networks (Rapid7 confirmed to us that they have a growing list of networks that requested to not be scanned). If blacklisting were the cause, we would expect to see certain networks (i.e., BGP prefixes) consistently missing from one or both scans.

To test our blacklisting hypothesis, we focus on the eight days where *both* Rapid7 and the University of Michigan performed full IPv4 scans. We group the IP addresses into their advertised BGP prefixes using historic RouteViews data [9]. While there was an average of 285,519 BGP prefixes that were covered by both scans, we found that 1,906 BGP prefixes were *always* missing from the University of Michigan scans but present in the Rapid7 scans, and 11,624 BGP prefixes were *always* missing from the Rapid7 scans but present in the University of Michigan scans. Moreover, these missing BGP prefixes account for much of the discrepancy between the two scans: On average, the University of Michigan scans contained 282,620 IP addresses that the Rapid7 scans did not; on average, 74.0% of these came from the BGP prefixes that Rapid7 never covered. Similarly, on average, the Rapid7 scans contained 84,646 IP addresses that the University of Michigan scans did not; on average, 62.6% of these came from the BGP prefixes that University of Michigan never covered. Thus, it appears that much of the discrepancy is due to blacklisting of different BGP prefixes, either by the scan operators or by the destination network.

4.2 Isolating Invalid Certificates

Recall that a certificate is invalid when it is outside its validity period, it is signed by an untrusted certificate, its signature is incorrect, etc. To isolate invalid certificates, we run `openssl verify` on each certificate. Because the scans and validation processes occurred at different points of time, we ignore certificate validation errors only due to expiration times (i.e., we consider a certificate to be valid if it was valid at *some* point in time). We also configure OpenSSL to trust the set of 222 root CA certificates included by default in the OS X 10.9.2 root store [35]. Finally, we validate all intermediate certificates before validating leaf certificates; this process allows us to construct a valid chain even if the server presented an incorrect chain (so-called "transvalid" certificates [?]).

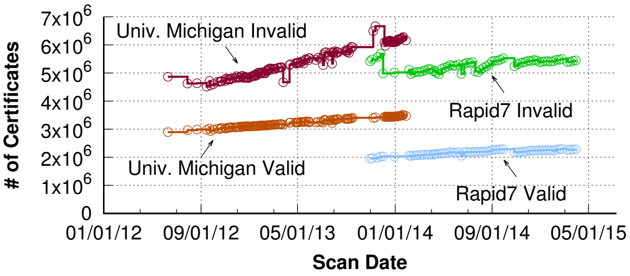

Figure 2: The number of invalid and valid certificates in University of Michigan and Rapid7 datasets.

Through this process, we isolate 70,637,981 invalid certificates (87.9% of all certificates) and 9,728,845 valid certificates (12.1%). Examining `openssl`'s output, we observe that 88.0% of the invalid certificates fail validation because they are self-signed,[7] 11.99% are invalid because they are signed by another, untrusted certificate, and 0.01% are invalid for a variety of other reasons. We use this group of invalid certificates as the basis for the analysis in the rest of our paper.

Inspecting the invalid certificates' `Common Name` lends some insight into their origins; many have domain names corresponding to Internet Service Providers (ISPs, such as FRITZ!Boxes' `fritz.net`) and cloud-accessible storage devices (such as the domain of Western Digital's My Cloud, `wd2go.com`). This provides an initial indication of end-user devices; we explore the origins of invalid certificates more thoroughly in §5.

Figure 2 shows the number of invalid and valid certificates found in the University of Michigan and Rapid7 datasets. The number of invalid certificates is increasing in both of the datasets, which aligns with our observation of the increasing popularity of HTTPS-enabled end-user devices (§2). Looking at each scan, we found that the fraction of invalid certificates varied between 59.6% and 73.7%, with an average of 65.0%. The cause of the disparity between the average fraction of invalid certificates per scan (65.0%) and the fraction across *all* scans (87.9%) will become clear in the following section when we investigate the certificates' lifetimes.

5. COMPARISON TO VALID CERTS

We begin our analysis by comparing the 70M invalid certificates to the 9.7M valid certificates, with the un-

[7] We identified self-signed certificates as those where `openssl` returned error code 19 (self-signed) or where we manually verified the certificate's signature with its own public key. The second step is necessary as `openssl` is known to only generate error 19 if the certificate is self-signed *and* the subject and issuer match.

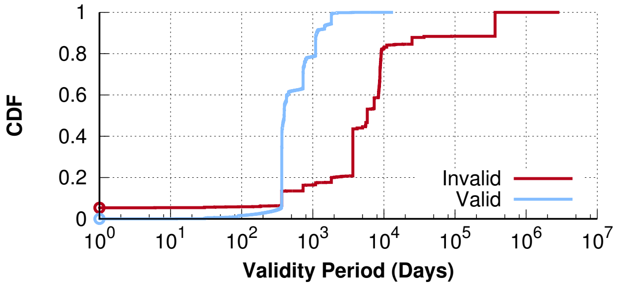

Figure 3: Cumulative distribution of the validity periods for both valid and invalid certificates. The validity periods of invalid certificates differ significantly, with 5.38% of them having a *negative* validity period, and others lasting decades.

derlying goal of understanding why so much of the PKI consists of invalid certificates. We start by evaluating how invalid certificates differ from their valid counterparts across several different properties, including key lengths, lifetimes, and geographic distribution.

5.1 Certificate Longevity

One possible explanation for why there are so many more invalid certificates than valid ones across our years-long dataset is that some hosts replace invalid certificates more frequently. We investigate this by examining two characteristics of certificates: their validity periods and their lifetimes.

First, we examine the certificates' *validity period*, or the time between the certificates' `Not Before` and `Not After` [20] dates (the first and last dates, respectively, during which the certificate should be considered valid). Figure 3 shows the cumulative distribution of all valid and invalid certificates' validity periods. The distributions are starkly different. Valid certificates typically have narrow validity periods, with a median of 1.1 years and a 90^{th} percentile of 3.1 years. Conversely, invalid certificates have an exceedingly large range, with a median of *20 years*, a 90^{th} percentile of 25 years, and some with `Not After` dates that end in the year 3000 or beyond, resulting in validity periods greater than 1M days. Moreover, we observe that 5.38% of invalid certificates have `Not After` dates prior to `Not Before` dates, which results in negative validity period (not shown in the graph, but note the *Invalid* line starts at $y = 0.0538$). The differences between these distributions indicate that the manner in which valid and invalid certificates are created also differs significantly.

Second, we examine the certificates' *lifetime*, or the (inclusive) time between the first scan and the last scan where we saw the certificate. For example, if we only saw the certificate on a single day, we calculate the lifetime to be one day; if we saw the certificate on two scans a week apart, we calculate the lifetime to be 8 days.[8] Figure 4 shows the cumulative distribution of all valid and invalid certificates' lifetimes. Here, too, we see drastically different distributions: the median lifetime for valid certificates is 274 days, while the median lifetime for invalid certificates is *one day*! This means that the majority of invalid certificates were observed in only a single scan, implying that they are *ephemeral* (i.e., the device that served these certificates likely reissues its certificate on a regular basis). An alternative explanation is that users are manually updating the certificates in their devices at least once a week, on average, but we believe this to be unlikely.

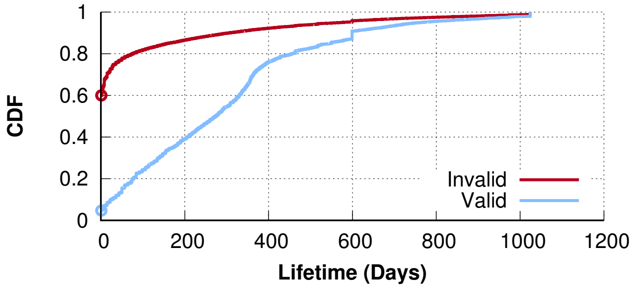

Figure 4: Cumulative distribution of the lifetimes of both valid and invalid certificates. The lifetime of invalid certificates is far shorter than its validity period: 60% of invalid certificates' lifetimes are a single day.

To provide additional evidence that these ephemeral certificates are likely reissued, we compare each ephemeral certificate's `Not Before` date to the day we observed the certificate. We anticipate that a `Not Before` date typically has the same day as when its certificate was created; assuming this, if most of the `Not Before` dates are close to the scan dates, it suggests that the certificates were generated right before our scan. The cumulative distribution of this difference is shown in Figure 5. Interestingly, we notice a bi-modal distribution: 70% of ephemeral certificates show a difference of less than four days, while 20% show a difference over

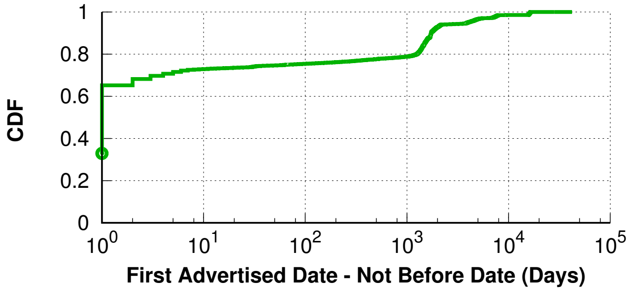

Figure 5: Cumulative distribution of the difference between ephemeral certificates' first advertised and `Not Before` dates for all invalid certificates. Note that y-axis starts at 32.9%: For 30% of the certificates, these were the same date; for 2.9%, the `Not Before` date was *after* the first advertised date (negative values not shown in the plot). The long tail reaches to 42,091 days.

[8] Our calculation of a certificate's lifetime is a lower bound on its true lifetime, due to the periodic nature of our scan data.

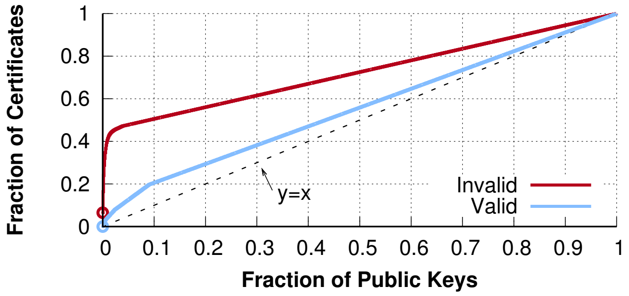

Figure 6: Fraction of public keys (x) needed to cover a given fraction of valid and invalid certificates (y). There is a significant difference in the two distributions, with invalid certificates showing significantly higher rates of sharing keys.

1,000 days. This indicates that ephemeral certificates are largely from devices that reissue their certificates on a regular basis.

In §6, we present techniques that allow us to "link" these reissued certificates together, enabling us to track a given device's reissued certificates. In the remainder of this section, we seek to understand the root cause and origins of invalid certificates.

5.2 Key Diversity

In principle, each unique SSL certificate should carry the `Public Key` from a *unique* key pair, as the certificate binds a subject (e.g., a domain name) to a `Public Key`. If `Public Keys` are shared across certificates owned by different domains, either party could impersonate the other.

Figure 6 shows the relationship between the fraction of certificates and the fraction of public keys they span. In an ideal case—wherein each certificate has a unique public key—this would result in a linear relationship with $y = x$. However, given that a certificate can have at most one public key, it must be the case that $y \geq x$ in this plot; the larger this inequality, the more certificates there are that share the same public key.

Figure 6 shows that neither valid or invalid certificates exhibit a perfect $y = x$ line. This result is somewhat expected: Zhang et al. [50] found that nearly half of all reissues of valid certificates in the Alexa top-1M are done with the same public key.[9]

However, the invalid certificates exhibit *significantly* less diversity of public keys. To our surprise, we observe that over 47% of invalid certificates share their `Public Key` with another certificate. As a point of comparison, Heninger et al. [?] found that over 60% of hosts scanned in 2012 were observed to share keys with another host; the discrepancy between their results and ours is likely to due to looking at *hosts* (as they do) versus *certificates* (as we do).

Top Issuers of Valid Certificates	*Num.*
`Go Daddy Secure Certification Authority`	1,869,701
`RapidSSL CA`	969,879
`PositiveSSL CA 2`	511,894
`Go Daddy Secure Certificate Authority - G2`	436,055
`GeoTrust DV SSL CA`	435,477

Top Issuers of Invalid Certificates	*Num.*
`www.lancom-systems.de`	4,691,873
`192.168.1.1`	2,438,776
(Empty string)	925,579
`remotewd.com`	881,406
`VMware`	748,937

Table 1: The top five issuers of valid and invalid certificates. Our results for valid certificates align with prior work [14]; our results from invalid certificates suggest that end-user devices are prevalent.

In fact, we found that one particular public key is shared by 4,586,469 certificates (6.5% of all invalid certificates in our dataset)! We manually inspected these certificates and found that they are all issued by *Lancom Systems*, a German company manufacturing home routers and wireless access points. This observation indicates that a large number of Lancom devices use the same key pair making them potentially vulnerable to impersonation or snooping if an attacker is able to extract the private key from one of these devices[10].

These results provide further evidence that *invalid certificates represent a fundamentally different point in the certificate landscape*, and are not simply valid certificates that have "gone bad." This raises the question: who issues the invalid certificates in the first place?

5.3 Issuer Diversity

We evaluate who issues invalid certificates by investigating two properties. *First*, we consider the diversity of the keys used to sign the invalid certificates. Prior studies have identified the relatively low diversity in the set of issuers for valid certificates [14, 24]. Our results confirm this, showing that just *five* signing keys (out of 1,477 observed valid keys) is enough to span half of all valid certificates.

When we consider the invalid certificates, the results are quite different. Recall that over 88.0% of invalid certificates are self-signed, immediately making it appear as if they have significant diversity. However, even if we focus on the non-self-signed certificates that list their `Authority Key ID`, we find that invalid certificates appear to have greater parent key diversity than valid certificates: the top five keys only cover 37% of the certificates. Moreover, we observe a total of 1.7M unique parent keys (as opposed to only 1,477 unique parent keys for valid certificates).

[9] This is an acceptable security practice as long as the private key has not been compromised. However, in the case of Heartbleed (which potentially exposed private keys), Zhang et al. [50] found that 4.1% of reissues were still done with the same key.

[10] Additionally, we found that these devices do not support Perfect Forward Secrecy (PFS) [26], meaning that their historic traffic is also vulnerable to decryption.

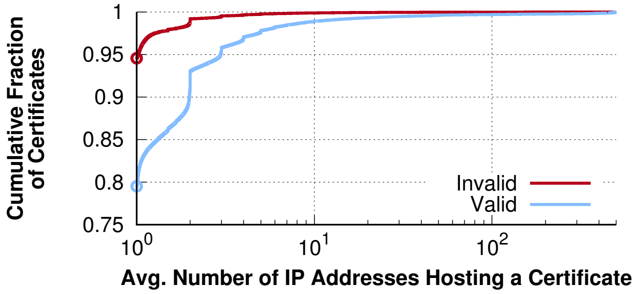

Figure 7: Cumulative distribution of the average number of IP addresses advertising each certificate across all scans. Valid and invalid certificates exhibit significantly different distributions. Note that the *y*-axis starts at 0.75; the long tail for invalid certificates extends to over 3.6M IP addresses.

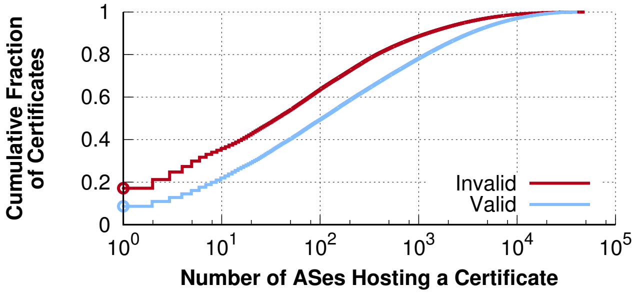

Figure 8: Cumulative distribution of the number of autonomous systems from which valid and invalid certificates are served. Although invalid certificates far outnumber valid ones, they originate from fewer ASes. 18% of all invalid certificates originate from a single AS.

Second, we examine the most frequent `Common Names` of the certificates used to sign valid and invalid certificates in Table 1. The most common issuers of valid certificates are well-known CAs, such as GoDaddy and RapidSSL (similar to findings in prior studies [14]). The invalid certificates' issuers are far less standard. We see several device manufacturers, including Lancom Systems, and Western Digital. Additionally, a large fraction of invalid certificates are issued by malformed `Common Names`, including private IP addresses (e.g., 3,353,464 invalid certificates were issued with a common name from `192.168.0.0/16`) or empty strings.

Ultimately, these results indicate a bimodal distribution of the source of invalid certificates: many originate from a small set of issuers while the majority are self-signed. In other words, *invalid certificates are largely issued by the very hosts who serve them.* Next, we turn our attention to investigating these hosts directly.

5.4 Host Diversity

We expect that invalid certificates do not represent globally replicated websites. To evaluate this hypothesis, we consider the diversity of hosts who serve invalid versus valid certificates. We measure the average number of unique IP addresses that advertise the certificate during each scan.

IP address diversity. Figure 7 shows the cumulative distribution of the average number of IP addresses advertising each certificate per scan, for all invalid and valid certificates. We observe that the majority of both valid and invalid certificates are advertised by just a single host (note that the *y* axis starts at 0.75). However, we observe that invalid certificates are, overall, advertised by far fewer hosts. The 99th percentile of invalid certificates are served by 2.0 hosts, while the same 99th percentile of valid certificates are served by 11.3 hosts. In fact, we observe valid CA certificates that are served by over 3.6M IP addresses!

These results show that, across our entire dataset, there is a (nearly) one-to-many mapping between IP addresses and invalid certificates. It may thus be feasible to *track end-user devices* based on their invalid SSL certificate. We explore this application in §7.

AS diversity. One might expect that because invalid certificates far outnumber valid certificates, they come from a much larger set of IP addresses, and thus perhaps a more diverse set of ASes. On the other hand, the results from Figure 7 show that a single valid certificate can be served from many distinct IP addresses, and thus valid certificates may come from a larger set of ASes. To resolve this, we show in Figure 8 the diversity of ASes for both valid and invalid certificates, by mapping the IP addresses advertising certificates into ASes using CAIDA's historic RouteViews dataset [9] Both show a surprising number of certificates originating from a small number of ASes; 10% of all valid certificates and 18% of all invalid certificates originate from a single AS. Beyond this small number of popular ASes, the invalid certificates exhibit significantly less diversity in their origins than valid ones; a set of 165 ASes accounts for 70% of all invalid certificates, while it takes a set of 500 ASes to account for the 70% of valid certificates.

To explore *which* ASes the certificates come from, we classified the ASes using CAIDA's AS classifications dataset [10]. Table 2 shows the resulting distribution; we can immediately observe that valid certificates come

AS Type	*% of Valid*	*% of Invalid*
Transit/Access	46.6%	94.1%
Content	42.9%	4.7%
Enterprise	7.8%	1.5%
Unknown	2.6%	1.7%

Table 2: Breakdown of AS types where certificates are advertised, based on CAIDA's classification [10]. Most invalid certificates come from transit/access networks, while most valid certificates are advertised from transit/access and content networks. Content networks are the most likely to advertise a valid certificate instead of an invalid one.

Top ASes Hosting Valid Certificates		*Num.*
#26496	GoDaddy.com, LLC (USA)	836,521
#46606	Unified Layer (USA)	224,806
#14618	Amazon, Inc. (USA)	171,689
#36351	SoftLayer Technologies (USA)	168,285
#16509	Amazon, Inc. (USA)	151,048

Top ASes Hosting Invalid Certificates		*Num.*
#3320	Deutsche Telekom AG (DEU)	12,115,594
#7922	Comcast Cable Comm., Inc. (USA)	2,889,282
#3209	Vodafone GmbH (DEU)	2,525,880
#6805	Telefonica Germany GmbH (DEU)	1,808,687
#4766	Korea Telecom (KOR)	1,795,298

Table 3: The top five ASes (with AS number and country) from which valid and invalid certificates are hosted.

from both Content networks (e.g., large websites) and Transit/Access networks (e.g., ISPs, and major ISPs who are hosted primarily on ASes other than their own), while invalid certificates almost exclusively come from the latter.

To underscore this difference, Table 3 shows the ASes from which valid and invalid certificates are most frequently hosted. The top ASes for valid certificates are all in the United States, and constitute a combination of popular hosting services. Conversely, invalid certificates are more geographically dispersed, with many hosted in Germany. Moreover, the hosting ASes of invalid certificates correspond largely with end-user home ISPs.

Device diversity. Finally, we investigate the types of the actual devices hosting the invalid certificates. We did this manually by looking up model numbers included in certificates and by simply loading the web pages from the IP addresses hosting invalid certificates and inspecting them. Because this was a manual process, we restricted this investigation to the certificates corresponding to the top 50 issuers.

Table 4 shows the breakdown of invalid certificates by device type. From this subset of the most common issuers, we see that the greatest contributor of invalid certificates is home routers and cable modems. In general, invalid certificates tend to be served by Internet devices, such as routers, VPNs, firewalls, and remote storage devices. There is also a relatively wide range of other devices, such as IPTVs and IP phones. Based on these results, we anticipate that, as these devices become increasingly popular—and particularly with the growing trend towards an Internet of Things—the number of invalid certificates will continue to grow.

5.5 Summary

We began this section with the goal of better understanding why so much of the SSL ecosystem consists of invalid certificates. We found that the large portion of invalid certificates is caused by devices that reissue their certificates on a regular basis, thereby inflating the number of invalid certificates overall (valid certificates are typically used for a year or more, whereas many invalid certificates have lifespans of days). We also found significant differences between valid and invalid certificates, including that invalid certificates are more likely to share keys, be advertised by fewer hosts, and be advertised from ASes that provide end-user Internet connectivity. In the next section, we take a closer look at the invalid certificates to see if we can “link” together those that originate from the same device.

Pct.	*Device Type*
45.3%	Home router/cable modem
32.0%	Unknown
6.04%	VPN
5.70%	Remote storage
4.27%	Remote administration
1.92%	Firewall
1.78%	IP camera
2.62%	Other (IPTV, IP phone, Alternate CA, Printer)

Table 4: Device types based on manual analysis of the top 50 issuing CAs. Invalid certificates tend to be served from end-user and enterprise devices.

6. LINKING CERTIFICATES

The results from our comparative study of valid and invalid certificates indicate that a relatively small number of end-user devices regularly reissue new invalid certificates. In this section, we develop techniques to “link together” distinct invalid certificates as having originated from the same device.

6.1 Challenges

In the context of valid certificates, the process of obtaining a new version of an existing certificate is referred to as *reissuing* the certificate. Tracking valid certificates reissues is relatively straightforward, as one can generally match on `Common Names`; we find that it is considerably more difficult with invalid certificates.

There are two key challenges to linking together invalid certificates. First, a single certificate can be shared across many different devices—this is also relatively common with valid certificates, largely due to services replicated across multiple physical machines or locations. We discuss in §6.2 how we distinguish legitimate duplicates from inconsistencies in the scanning methodology. The second challenge is that even for certificates that are only ever advertised from a single device at any one time, surprisingly, there is often no one clear certificate field that makes it obvious whether two different invalid certificates were generated by the same device. Moreover, across the entire dataset, many certificate values are shared across many distinct devices, even those we expected to be unique (such as the `Common Name`). In §6.3, we present heuristics to choose and link together certificate features and we evaluate in §6.4.

6.2 Scan Duplicates

Before we attempt to link two distinct certificates *across* certificate scans, we first seek to map a given certificate to a device within a *single* scan. At first glance, it would seem trivially straightforward to treat the IP address from which a certificate is advertised as a unique identifier for the device hosting the certificate. However, there is a complicating factor: the scans are not instantaneous, as it takes up to 10 hours [15] to scan and collect all certificates in the entire IPv4 space. Additionally, ZMap's implementation scans IP addresses in a random order [15] to prevent a sudden surge in traffic to different networks. As a result, if a device changes its IP address *during* the scan, it is possible that it may end up being scanned twice from different IP addresses (or more, if it changes more than once and the scan contacts IP addresses in the right order). Thus, if we are not careful, certificates that are unique to a single device may be incorrectly classified as shared.

To address this issue, we leverage the fact that most known dynamic IP assignment policies lease IP addresses to users on the order of days [30]. Thus, the case of a device changing its IP address *multiple times* during a scan—and the scan discovering these three or more IP addresses in order—should be rare. We therefore set our threshold for certificate uniqueness to two: if we observe a certificate advertised by no more than two IP addresses during each scan[11], we declare the certificate to be unique; if we ever observe the certificate advertised by more than two IP addresses, we declare the certificate to be non-unique.

Thus, for the 1.6% of invalid certificates that are advertised from more than two IP addresses (shown in more detail in Figure 7), we cannot be sure that this certificate is not shared across multiple devices. We exclude these; for the remainder of this section, we consider the remaining 69,481,047 invalid certificates.

6.3 Linking Certificates Across Scans

After accounting for scan duplicates, we obtain from each scan a one-time snapshot of a device-to-certificate mapping. We next seek to use the certificates to "link" devices from one scan to another. We would ideally like to link using both features of the certificate (e.g., the `Common Name`) and features that can be observed from the network connection used to collect the certificate (e.g., the initial TCP window size). Unfortunately, the certificate scan data contains only the certificates themselves; thus, in the remainder of this section, we focus on using only features from certificates and leave other features to future work.

% Non-unique	*Feature*
67.7%	`Not Before`
67.5%	`Common Name`
61.4%	`Not After`
47.0%	`Public Key`
19.6%	`Subject Alternate Name List`
4.2%	`Issuer Name & Serial No. (IN + SN)`

Table 5: Features from invalid certificates and the percentage of them that are not unique across all scans in our dataset.

For valid certificates, tracking a device across scans is mostly straightforward. Valid certificates tend to have long lifetimes (§5.1), and they generally have unique `Common Names`: a certificate for `example.com` in one scan is likely to appear in the subsequent scan, as well. However, invalid certificates are often short-lived (§5.1) and have non-unique `Common Names`, with `192.168.1.1` being one of the most common (§5.4).

Our insight is driven by our findings from §5: because many invalid certificates are self-signed, we anticipate that the *combination of features* of invalid certificates will uniquely define a device (or at least a particular device vendor). To this end, we examine different fields within each SSL certificate to determine whether they provide clues for linking.

6.3.1 Linkable Features

To determine that two given certificates correspond to the same device, we seek features of the certificates that *remain the same* between two scans, but do not appear in other certificates (i.e., they uniquely identify a single device). Table 5 lists the certificate features we considered, as well as the percentage of invalid certificates that have a non-unique value for each feature. For linking two certificates, non-uniqueness is a necessary condition—if two certificates share *no* common values, we cannot link them as coming from the same device. However, non-uniqueness is not a sufficient condition; for instance, 2,438,776 certificates share the common name of `192.168.1.1` (see Table 1), but clearly these do not map to a single device. Similarly, as shown in §5.2, many certificates share the same public keys.

In addition to the features listed in Table 5, we also considered certificate extensions that help clients access the "parent" certificate and ensure the validity of the given certificate: `Certificate Revocation List` (CRL) endpoints, `Authority Information Access` (AIA) locations, `Online Certificate Status Protocol` (OCSP) responders, and `Object IDs` (OIDs). Although these are present in 95% of valid certificates, we find that they are rarely used in invalid certificates: 99.2% of invalid certificates have no `CRL`, 99.3% have no `AIA` location, 99.9% have no `OCSP` responder, and 99.9% have no `OID`. As a result, they link only a small fraction of certificates, and they link very few uniquely.

[11] The only exception to this policy is if we see the certificate advertised by exactly two IP address in *every scan*; since IP addresses are scanned in random order each time, this strongly suggests that there are two devices with the certificate. In this case, we declare the certificate to be non-unique.

6.3.2 *Linking Methodology*

To successfully link certificates despite the challenges that the different fields present, we devise a methodology to eliminate many false positives by leveraging the *lifetime* of each certificate (where the lifetime is defined as the time between the first scan and the last scan where the certificate was observed). Our methodology is based on the observation that if a device reissued its certificate, then the lifetime of the old certificate must end and the lifetime of a new certificate must begin.

Thus, our methodology proceeds as follows. We first group certificates by shared field values: for each field type and each value of that field, we collect all certificates that have that value (i.e., all certificates with a `Common Name` of "WD2GO 293822", or all certificates with a certain public key). Then, we consider this set of certificates to be linked (from the same device) so long as the lifetimes of any pair of these certificates do not overlap by more than a single scan.

If any pair of the certificates' lifetimes do overlap, then we do not allow any of these certificates to be linked using this field (of course, subsets of this group may be linked using other fields). The reason we allow the overlap of certificates on a single scan is because devices may change IP addresses (and reissue their certificate) during a single scan.

To show the intuition behind this methodology, we present a few examples in Figure 9. Each box in this figure represents a certificate we observed from a given IP address on a given scan. Across multiple scans, we can group certificates by the public keys they share; each color in the figure represents a different group of certificates, each of which shares a given public key. For the certificates sharing PK$_1$, we observe no overlap, that is, PK$_1$ is only ever advertised from at most one IP address on any given scan (note that we did not observe it in the third scan). Therefore, by our methodology, the PK$_1$ certificates are linkable across multiple scans. Likewise, although we see PK$_2$ advertised from several different IP addresses, certificates 3 and 4 only overlap on a single scan; thus these certificates are also linkable. However, for PK$_3$, we see *two* scans with overlapping yet distinct certificates, and we therefore conclude that these certificates are, with high probability, served from different devices. Therefore, the certificates with PK$_3$ in this example are not linkable.

This methodology eliminates most of the issues with the certificate fields identified in §6.3.1. In particular, for fields that are shared by different devices (e.g., the *Lancom* public key), our technique marks them as non-linkable, as the certificates have significant overlap.

6.4 Evaluation

We now evaluate our proposed linking methodology. Below, we first describe our approach to evaluating our linking procedure before presenting the results.

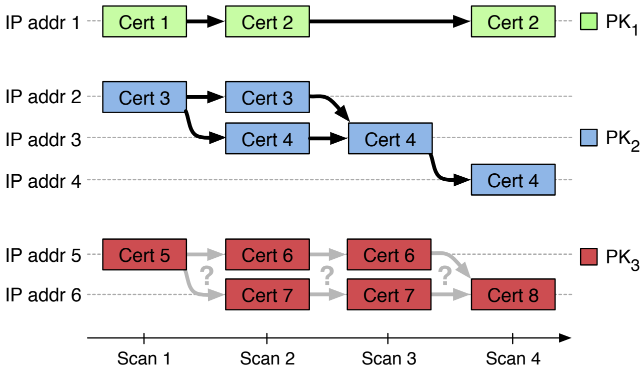

Figure 9: Timeline of example certificate lifetimes for three groups of certificates. Each box represents a certificate with a given `Public Key` (PK) seen in a given scan. Our methodology links the certificates with public keys PK$_1$ and PK$_2$ across their respective scans, but it marks certificates with PK$_3$ as *not* linkable, as they overlap on more than one scan.

6.4.1 *Approach*

When trying to evaluate how well our certificate linking methodology works, we face a significant challenge: we do not have ground truth on whether different certificates are actually from the same device. Thus, we use the information we do have available to us to serve as a rough version of ground truth: the IP address where we saw the certificate advertised.

Specifically, to evaluate whether two linked certificates are from the same device, we ask whether they were advertised from the same IP address (we refer to this as *IP-level consistency*). If they were, we have some confidence that they are actually from the same device (recall that we only link groups of certificates that share a common field and do not have overlapping lifetimes). Thus, the most likely explanation for a set of certificates that have non-overlapping lifetimes, share a common field that no other certificates have, and are advertised from the same IP address is that they are from the same device.

However, it is well-known that many ISPs implement dynamic IP address assignment policies, meaning devices' IP addresses may change over time [**?**]. As a result, if a device were to change its IP address, it should be considered a valid linking, but our rule may incorrectly mark it as a false positive. To counteract this effect, we repeat the same analysis, examining whether the certificates were advertised by IP addresses in the same /24 network or whether the certificates were advertised by IP addresses in the same AS, referred to as */24-level consistency* and *AS-level consistency*. Of course, looking beyond exact IP address matches may introduce false positives—and assuming that address reassignment policies stay within the same /24 may introduce false negatives—but we can get an idea of how the linking procedure performs overall. For example, if we observe cases where the linking procedure has low IP-level consistency but high AS-level consistency, we can

	Public Key	Not Before	Common Name	Not After	IN + SN	SAN	CRL	AIA	OCSP	OID
Total linked	23,276,298	16,301,321	8,576,231	6,235,419	4,193,744	2,484,652	389,264	377,310	3,352	593
Uniq. linked	11,798,203	5,296,175	1,794,118	1,197,317	955,764	123,740	4,912	3,192	185	121
IP-consistency	41.9%	53.5%	51.1%	51.2%	48.2%	52.2%	85.8%	85.7%	52.2%	83.9%
/24-consistency	46.1%	54.3%	53.3%	52.9%	49.6%	55.0%	87.2%	87.1%	55.0%	86.6%
AS-consistency	98.0%	63.0%	96.6%	58.2%	89.3%	97.5%	95.2%	95.1%	97.5%	92.6%

Table 6: The linking performance of different SSL certificate fields (IN + SN is Issuer Name & Serial No.). The first two rows show the total number of certificates linked by each field, and the total number of certificates linked *only* by that field. The bottom three lines show the IP-level, /24-level, and AS-level consistency.

infer that this may be a case where the AS has a dynamic IP address assignment policy.

IP addresses as Common Names. 33,145,677 (46.9%) of certificates' Common Names appear to be an IPv4 address. As our objective is to link certificates regardless of the IP address of the hosts, we intentionally *disregard* the certificates whose Common Name appears to be an IP address when trying to link using Common Names for the fairness of our evaluation.

Example. Consider the group of certificates in Figure 9 that all share the public key PK_2. Recall that our methodology would link these certificates, as they do not overlap with one another on multiple scans. We observe these certificates across three distinct IP addresses; suppose that IP addresses "2" and "3" reside in the same /24, and that all three of them are owned by the same AS. The most commonly appearing IP address (IP address 2) is seen on two of the four scans; thus the IP-level consistency is 0.5. The most commonly appearing /24 shows up three of the four times, and thus the /24-level consistency is 0.75. Finally, since they all reside in the same AS, the AS-level consistency would be 1.0. This example shows that, generally, our IP-level consistency is the most conservative, and we would expect it to be low in ASes that implement dynamic IP address assignment policies.

6.4.2 Results

Table 6 presents our evaluation results when linking on each certificate field. The first two rows of the table present the number of certificates linked by that field, and the number of certificates linked *only* by this field. The bottom three rows present the IP-level consistency, /24-level consistency, and AS-level consistency that results when we use each field to link.

Below, we take a closer look at the performance of each of the fields; in §6.4.3, we more closely examine the characteristics of the linked groups.

Public Key. We observe that Public Key can link the most certificates among all fields, and does so with 98% AS-level consistency, but only 41.9% IP-level consistency. To take a closer look at this discrepancy, we manually examined the certificates for which IP-level consistency is less than the average. We find that the certificates with one particular SAN—[fritz.fonwlan.box][12]—represent 12,078,402 (51.9%) of the public-key-linked certificates, but the consistency of linking of this group is 27% at the IP-level and 99% at the AS-level. Thus, these devices account for much of the low IP-level consistency of linking on public key; if we remove them, we find that the consistency jumps to 69.4% at the IP address level. We found 82.6% of these devices to be deployed in the German ISPs Deustche Telekom, Vodafone, and Telefonica Germany. These ASes are known to change end users' IP addresses frequently [?, 33], explaining why we see such low IP-level consistency. Thus, there are a large number of devices that regenerate their certificates, but keep the same, unique Public Key over time; this enables us to link these certificates together and track the devices.

Linking on Public Key represents a case where a few ASes give specific devices to users and implement a dynamic IP address assignment policy. Thus, our methodology is correct in linking these certificates together (i.e., the AS-level consistency is the correct one to consider).

Not Before and Not After. These fields are both able to link large numbers of certificates—16M and 6M certificates respectively—but their consistency has poor performance in both IP- and AS-level. We tried to investigate the reason behind the poor performance by manually investigating the group of certificates linked together, but we are unable to observe any specific patterns. However, we found that the sizes of the linked groups is almost always two or three certificates; we conjecture that Not Before and Not After incorrectly link certificates that happen to share the same value but are not actually related. This is also bolstered by the fact that the size of the set of potential values (i.e., recent timestamps) for Not Before and Not After is dramatically smaller than the size of the set of potential values for Common Name and Public Key. As a result, we exclude Not Before and Not After from consideration.

Common Name. We observe that Common Name performs similarly to Public Key, with 96.6% AS-level consistency, but only 51.1% IP-level consistency. We observe that 5,561,069 (21.0%) of the linked certificates have URL-formatted Common Names. To investigate which domains are used for the Common Name, we grouped them by second-level domain. Interestingly,

[12] These devices are FRITZ!Box devices mentioned in §6.3.1.

895,775 (16%) of these certificates (the largest portion) share the common second-level domain `myfritz.net`, which is the dynamic DNS feature provided by FRITZ!Box. Similarly, we observe that 445,585 (8%) of these certificates contain "dyndns" or "selfhost" in the `Common Name`, which also indicates that the device uses dynamic DNS.

Issuer Name and Serial Number. Similar to `Public Key`, these fields exhibit 89.3% AS-level consistency, but only 48.2% IP-level consistency. To investigate this phenomenon, we manually examined the linked certificates. Interestingly, we find 965,366 (23.1%) of certificates are advertised from "PlayBook" devices, a tablet computer developed by BlackBerry. These certificates share the same format for `Issuer Name`: "PlayBook: [MAC-ADDRESS]". We conjecture that these devices are connected through Blackberry's mobile network, which frequently re-assigns their IP address. If we disregard the PlayBook devices, the IP-level consistency jumps to 71.9%.

Subject Alternate Names. We observe that `SAN`s can link 2.4M certificates in total, but only about 123K certificates uniquely. Also, similar to `Public Key`, `SAN`s show high consistency at the AS-level, but low consistency at the IP-level. Closely investigating these, we find that 1,658,575 (66.8%) of the linked certificates are from FRITZ!Box devices (the same devices that we found with `Public Keys`), explaining why `SAN` links few devices uniquely. As expected, the IP-level consistency among these is 24.3%, while the IP-level consistency for the non-FRITZ!Box devices is 85.0%. Thus, the behavior of linking on `SAN`s is similar to that of `Public Key`.

CRL, AIA, OCSP, and OID. We found common characteristics among `CRL`s, `AIA`, `OCSP`, and `OID`. But, because they are rarely used, they link few certificates.

6.4.3 *Linked Groups*

Based on the results above, we view the fields `Not Before`, `Not After`, and `Issuer Name` and `Serial Number` as having insufficient consistency to be used to link (i.e., less than 90% consistency in AS-level); we do, however, consider all other fields for linking. Specifically, we consider each field in Table 6 (other than `Not Before`, `Not After`, and `Issuer Name` and `Serial Number`) in decreasing order of AS-level consistency. We iteratively link all certificates each field i, remove the linked certificates from consideration, and continue with field $i + 1$.

Ultimately, we are able to link 27,373,584 certificates (39.4% of all invalid certificates) into 2,980,746 groups. Figure 10 shows the cumulative distribution of the sizes of these groups, both in aggregate and depending on the field that they were linked on. We find that public key can group the largest number of certificates: 62% of linked groups are composed of more than 2 certificates, and there exist groups that link up to 413 certificates. In contrast, for certificates linked via `CRLs`, 90% of groups are only two certificates. Interestingly, in our linking process, even though we link the certificates via the `SAN` field after `Common Names`, the average number of certificates in `SAN` groups (5.10) is larger than that of `Common Name` (2.60), which suggests that each field has different potential for linking certificates.

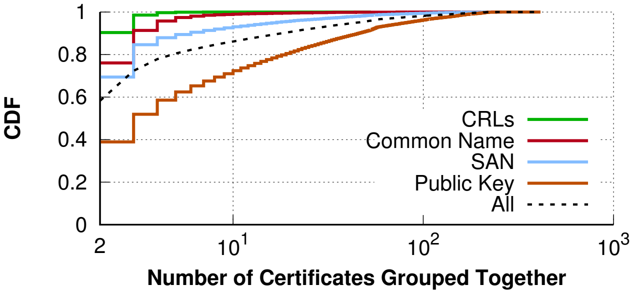

Figure 10: Cumulative distribution of the number of certificates grouped by the major fields. The x-axis starts at 2 and the long tail extends to 413 certificates.

6.4.4 *Comparison with Original Set*

As a final evaluation, we examine how our methodology changes the set of devices that we are able to track. To this end, we combine the linked invalid certificates (27M certificates in 2.9M groups) and unlinked invalid certificates (42M) together and examine how the lifetime of certificates changes after they are linked. We observe that the portion of unlinked certificates that appear in only a single scan drops from 61% to 50.7% and mean lifetime increases from 95.4 days to 132.3 days. This indicates that our methodology has linked together many of the ephemeral certificates, and has given us a better understanding of the behavior of the device that generated them.

7. TRACKING END-USER DEVICES

The linking methodology we developed in §6 permits us to associate the reissued certificates from a given device. Moreover, recall from §5.4 (and in particular Figure 7) that over 94% of invalid certificates are hosted by a single IP address on any given scan. This indicates that, once we determine the set of certificates that originated from a given device, we can *track end-user devices* as they move through the IP address space, purely by observing the (invalid) SSL certificates they serve on port 443.

In this section, we explore two applications of end-user device tracking: monitoring device movement (across ASes and geographically), and inferring IP address reassignment policies. Neither of these studies are complete, and both leave open many interesting questions for future work; however, they show that, by applying our linking methodology, invalid certificates can lend considerable insight into the Internet at breadth.

7.1 Background

Longitudinal measurements of the Internet are crucial for understanding past trends as well as making future predictions. For example, researchers may wish to study how the end-user devices attached to the Internet are changing, as users upgrade devices or change ISPs. Unfortunately, IP addresses—the most straightforward form of identification—cannot be used to uniquely identify devices over long timescales for several reasons: ISPs often use dynamic IP address assignments [48] or carrier-grade network address translators (NAT) [6], and users and devices can move between ISPs.

In long timescales, researchers have given users monitoring software [43] or dedicated hardware [37], enabling long-term measurements regardless of IP address changes. These approaches, however, are also difficult to scale, as it is necessary to recruit users to participate. Thus, being able to track Internet-connected devices over long timescales, both at large-scale and high fidelity, remains a challenge. Most related work relies either *application-level* or *network fingerprints*. For the former, there is work [5, 34, 38, 51] that aims to track users via their web browser configurations, such as cookies [4, 16], `User-Agent`, plugins, JavaScript [31] etc. In fact, recent work [51] proposes generating stable device IDs using inaudible sounds. For the latter, Greenwald et al. [22] have presented a method to understand the potential for, and to prevent, network device fingerprinting. Often, network device fingerprinting can reveal the type of device (or software), but is unable to identify *individual* devices uniquely.

The techniques we present in this section differ fundamentally from prior approaches, as they do not require users to deploy hardware [37,44], install additional software, or interact with a server [1,2]. Instead, we use *latent features* within the certificates that users' devices already publicly host as evidence that two different certificates measured days or weeks apart may correspond to the same physical device.

7.2 Trackable Devices

For the analysis below, we define a *trackable* device as one we observe for longer than a year. Before applying our linking methodology, we can only track the 5,585,965 devices that advertise the same distinct certificate on every scan. After applying our methodology from §6, we find a total of 6,750,744 trackable devices, representing an increase of 17.2%. This highlights that our linking methodology is able to significantly increase the number of devices that can be tracked over time. Moreover, it shows how we are able to gain a far wider view into devices than would be possible with explicit user participation or hardware deployment.

7.3 Tracking Device Movement

Using our set of tracked devices, we observe 718,495 devices that change ASes at least once (among the 6,750,744 total tracked devices), as well as a total of 1,328,223 AS transitions. We can observe that most of these devices (69.7%) change their AS only once, while others changed their AS more than 100 times (and are likely mobile devices like the Playbook).

There are two common reasons why a device may move from one AS to another. *First*, the device's ISP could have transferred its IP addresses to another AS. To detect this, we look for instances wherein at least 50 devices switch from one AS to another between scans; this happens 1,159 times in our dataset, covering 343,687 devices movements between 500 ASes. In some instances, this is due to the ISP simply transferring its IP addresses between ASes the ISP owns. For instance, Verizon (AS19262) transferred a large portion of its IP address space to MCI Communications (AS701), acquired by Verizon, twice (once with 14,453 devices and the other with 10,880). We see similar such movements with AT&T in September, 2013 (12,739 IP addresses).

Second, a device could move from one AS to another because the user chose to switch ISPs or to physically move to a different geographic location. We observe 45,450 devices move across countries.[13] For example, we observe 9,719 devices moved out of the U.S. and 7,868 devices moved into the U.S.

These findings show that much can be inferred from applying our linking methodology (§6) to invalid certificates longitudinally. Considering the growing number of Internet of Things devices using port 443, we believe increasingly many devices will use invalid certificates for their secure communications and our approaches can be directly applicable. There are also many interesting avenues of future work, but each comes with potential ethical concerns; for instance, it may be possible to track mobile devices with these techniques, which risks tracking personal movements over time.

7.4 Inferring IP Reassignment Policies

Being able to track individual devices and their assigned IP addresses naturally allows us to observe how a given ISP reassigns IP addresses to its customers. To investigate reassignment policies, we group IP addresses by AS and exclude any AS that has fewer than 10 tracked devices in our dataset; this leaves 4,467 ASes. Although we cannot distinguish with certainty slow-changing from static IP addresses, we classify an address as statically assigned if it is mapped to one device across our entire dataset and we have observed it for at least one year. Figure 11 shows the distribution of ASes' static IP address assignment policies.

Surprisingly many ASes assign static IP address pervasively. We observe from Figure 11 that 2,517 ASes (56.3%) assign static IP addresses to 90% of their de-

[13] We rely on CAIDA's historical AS-to-organization dataset [11] to determine the country in which each AS is located. It is worth noting that the dataset has a resolution of 3–4 months; we choose the entry that is closest to each of our scans.

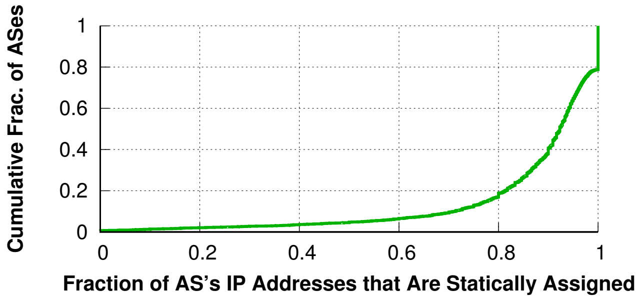

Figure 11: The fraction of devices with statically assigned IP addresses, as a distribution over ASes. ASes tend to have long-lived IP address assignments.

vices. For example, three of Comcast Cable Communications' ASes do not reassign IP addresses to 90% (25,178) of the devices we are able to track, likewise with AT&T Internet Services (9,058 devices, or 88.9%, across 14 of their ASes).

Conversely, relatively few other ASes exhibit highly dynamic IP address reassignment policies. In particular, we found 15 ASes who assign a new IP address to at least 75% of their hosts between *every scan.* For example, 182,536 (76.3%) of the devices in Deutsche Telekom had their IP address changed between every scan. Similarly, the 29,385 (99.6%) devices in *Telefonica Venezolana* from Venezuela, 820 (97.0%) devices *Tim Celular* from Brazil, and 206 (95.3%) devices in *BSES TeleCom Limited* from India did so as well.

This application demonstrates the benefits of being able to track devices across a wide range of populated ASes, made possible, surprisingly, by the widespread deployment of invalid certificates.

8. CONCLUSION

In this paper, we presented the first in-depth study of the *invalid* certificates in the web's PKI. Although typically ignored by measurement studies and browsers (and end users who proceed anyway), we have demonstrated that invalid certificates merit study, as they constitute the vast majority (65.0% in daily average, and 87.9% in total measurement period) of all certificates in the web's PKI over the past four years. Our investigation into the origins of invalid certificates led us to observe significant differences with valid ones, and to ultimately conclude that invalid certificates originate largely from end-user devices that regularly regenerate new, self-signed certificates. Surprisingly, a large number of invalid certificates stem from a rather small set of device manufacturers and ISPs. Moreover, we presented techniques that use the latent features of invalid certificates to track devices over long periods of time, permitting a broad, longitudinal study of IP address reassignment policies and user mobility without any explicit user interaction.

Future work can strengthen the results in this paper. Primarily, we lack a ground truth against which to validate our techniques for tracking end-user devices, largely because our technique applies to devices that do not appear in standard device-tracking datasets, such as RIPE Atlas [40] and SamKnows [44]. We believe that the approach that we have taken in this paper (using IP-level, /24-level, and AS-level consistency as a proxy for accuracy) in fact forms a *lower* bound of the true accuracy, as nearly half of all IP address changes result in a different /16 [?], but a more direct validation is in order. Despite these shortcomings, this paper provides strong evidence that invalid certificates, though long overlooked, can provide valuable insights into the certificate ecosystem, IP address changes, and end-user devices in general. We believe that many areas of future work can benefit from incorporating them, and to this end, we make our data and code publicly available at `https://securepki.org`

Acknowledgments

We thank the anonymous reviewers and our shepherd, kc claffy, for their helpful comments. This research was supported by NSF grants CNS-1409249, CNS-1421444, CNS-1563320, and CNS-1564143, and by the NSA as part of a Science of Security lablet.

9. REFERENCES

[1] Cisco IP Device Tracking. 2015. `http://www.cisco.com/c/en/us/td/docs/switches/lan/Denali_16-1/ConfigExamples_Technotes/Config_Examples/Device_Tracking/ip_device_tracking.pdf`.

[2] SolarWinds User Device Tracker. 2015. `http://cdn.swcdn.net/creative/v12.7/pdf/datasheets/SW_UDT_datasheet.pdf`.

[3] D. Akhawe and A. P. Felt. Alice in Warningland: A Large-scale Field Study of Browser Security Warning Effectiveness. *USENIX Security*, 2013.

[4] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz. The Web Never Forgets: Persistent Tracking Mechanisms in the Wild. *CCS*, 2014.

[5] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel. FPDetective: Dusting the Web for Fingerprinters. *CCS*, 2013.

[6] S. Alcock, R. Nelson, and D. Miles. Investigating the impact of service provider NAT on residential broadband users. *INFOCOM*, 2010.

[7] A. Bates, J. Pletcher, T. Nichols, B. Hollembaek, and K. R.B. Butler. Forced Perspectives: Evaluating an SSL Trust Enhancement at Scale. *IMC*, 2014.

[8] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, IETF, 2008. `http://www.ietf.org/rfc/rfc5280.txt`.

[9] CAIDA Routeviews Prefix to ASMappings Dataset. `http://www.caida.org/data/routing/routeviews-prefix2as.xml`.

[10] CAIDA ASClassifications Dataset. `http://www.caida.org/data/as-classification/`.

[11] CAIDA ASOrganizations Dataset. `http://www.caida.org/data/as-organizations/`.

[12] Convergence. `http://convergence.io`.

[13] Z. Durumeric, J. Kasten, D. Adrian, J. A. Halderman, M. Bailey, F. Li, N. Weaver, J. Amann, J. Beekman, M. Payer, and V. Paxson. The Matter of Heartbleed. *IMC*, 2014.

[14] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman. Analysis of the HTTPS Certificate Ecosystem. *IMC*, 2013.

[15] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-Wide Scanning and its Security Applications. *USENIX Security*, 2013.

[16] S. Englehardt, D. Reisman, C. Eubank, P. Zimmerman, J. Mayer, A. Narayanan, and E. W. Felten. Cookies That Give You Away: The Surveillance Implications of Web Tracking. *WWW*, 2015.

[17] EFF SSL Observatory. `https://www.eff.org/observatory`.

[18] S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, and M. Smith. Why Eve and Mallory Love Android: An Analysis of Android SSL (in)Security. *CCS*, 2012.

[19] S. Fahl, M. Harbach, H. Perl, M. Koetter, and M. Smith. Rethinking SSL Development in an Appified World. *CCS*, 2013.

[20] D. W. S. Ford and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. IETF RFC 3280, IEFT, 2013.

[21] 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. *CCS*, 2012.

[22] L. G. Greenwald and T. J. Thomas. Understanding and preventing network device fingerprinting. *Bell Labs Technical Journal*, 12(3), 2007.

[23] L.-S. Huang, A. Rice, E. Ellingsen, and C. Jackson. Analyzing Forged SSL Certificates in the Wild. *IEEE S&P*, 2014.

[24] R. Holz, L. Braun, N. Kammenhuber, and G. Carle. The SSLLandscape – AThorough Analysis of the X.509PKIUsing Active and Passive Measurements. *IMC*, 2011.

[25] H.D. Moore. Personal Communication.

[26] D. C. D. Harkins. The Internet Key Exchange (IKE). IETF RFC 2409, IETF, 1998.

[27] T. H.-J. Kim, L.-S. Huang, A. Perring, C. Jackson, and V. Gligor. Accountable Key Infrastructure (AKI): A Proposal for a Public-key Validation Infrastructure. *WWW*, 2013.

[28] Y. Liu, W. Tome, L. Zhang, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, A. Schulman, and C. Wilson. An End-to-End Measurement of Certificate Revocation in the Web's PKI. *IMC*, 2015.

[29] Let's Encrypt. `https://letsencrypt.org`.

[30] G. Maier, A. Feldmann, V. Paxson, and M. Allman. On Dominant Characteristics of Residential Broadband Internet Traffic. *IMC*, 2009.

[31] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham. Fingerprinting Information in JavaScriptImplementations. *W2SP*, 2011.

[32] S. Matsumoto, P. Szalachowski, and A. Perrig. Deployment Challenges in Log-based PKI Enhancements. *EuroSec*, 2015.

[33] G. C. M. Moura, C. Gañán, Q. Lone, P. Poursaied, H. Asghari, and M. van Eeten. How Dynamic is the ISPs Address Space? Towards Internet-Wide DHCP Churn Estimation. *Networking*, 2015.

[34] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting. *IEEE S&P*, 2013.

[35] OS X Yosemite: List of available trusted root certificates. `https://support.apple.com/en-us/HT202858`.

[36] H. Perl, S. Fahl, and M. Smith. You Won't Be Needing These Any More: On Removing Unused Certificates from Trust Stores. *FC*, 2014.

[37] Project BISmark. `http://projectbismark.net`.

[38] F. Roesner, T. Kohno, and D. Wetherall. Detecting and Defending Against Third-Party Tracking on the Web. *NSDI*, 2012.

[39] M. D. Ryan. Enhanced Certificate Transparency and End-to-End Encrypted Mail. *NDSS*, 2014.

[40] RIPE NCC Annual Report 2015. `https://www.ripe.net/publications/docs/ripe-665`.

[41] Rapid7 SSL Certificate Scans. `https://scans.io/study/sonar.ssl`.

[42] P. Szalachowski, S. Matsumoto, and A. Perrig. PoliCert: Secure and Flexible TLS Certificate Management. *CCS*, 2014.

[43] M. A. Sánchez, J. S. Otto, Z. S. Bischof, D. R. Choffnes, F. E. Bustamante, B. Krishnamurthy, and W. Willinger. Dasu: Pushing Experiments to the Internet's Edge. *NSDI*, 2013.

[44] Samknows & Ofcom UK broadband performance testing. `http://www.samknows.com/broadband/ofcom_and_samknows`.

[45] The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698, IETF, 2012. `https://tools.ietf.org/html/rfc6698`.

[46] University of Michigan HTTPS Ecosystem Scans. `https://scans.io/study/umich-https`.

[47] N. Vallina-Rodriguez, J. Amann, C. Kreibich, N. Weaver, and V. Paxson. A Tangled Mass: The Android Root Certificate Stores. *CoNEXT*, 2014.

[48] Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber. How Dynamic are IP Addresses? *SIGCOMM*, 2007.

[49] S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage. When Private Keys Are Public: Results from the 2008 Debian OpenSSL Vulnerability. *IMC*, 2009.

[50] L. Zhang, D. Choffnes, T. Dumitraş, D. Levin, A. Mislove, A. Schulman, and C. Wilson. Analysis of SSL certificate reissues and revocations in the wake of Heartbleed. *IMC*, 2014.

[51] Z. Zhou, W. Diao, X. Liu, and K. Zhang. Acoustic Fingerprinting Revisited: Generate Stable Device ID Stealthily with Inaudible Sound. *CCS*, 2014.