Rusted Anchors: A National Client-Side View of Hidden Root CAs in the Web PKI Ecosystem

Yiming Zhang[1,2], Baojun Liu[1] 🖂, Chaoyi Lu[1,3], Zhou Li[4]
Haixin Duan[1,5] 🖂, Jiachen Li[1] and Zaifeng Zhang[3]
[1] Tsinghua University, China [2] Beijing National Research Center for Information Science and Technology, China
[3] 360Netlab, China [4] University of California, Irvine, USA [5] QI-ANXIN Technology Research Institute, China

ABSTRACT

HTTPS secures communications in the web and heavily relies on the Web PKI for authentication. In the Web PKI, Certificate Authorities (CAs) are organizations that provide trust and issue digital certificates. Web clients rely on public root stores maintained by operating systems or browsers, with hundreds of audited CAs as *trust anchors*. However, as reported by security incidents, *hidden root CAs* beyond the public root programs have been imported into local root stores, which allows adversaries to gain trust from web clients.

In this paper, we provide the first client-side, nation-wide view of hidden root CAs in the Web PKI ecosystem. Through cooperation with a leading browser vendor, we analyze certificate chains in web visits, together with their verification statuses, from volunteer users in 5 months. In total, over 1.17 million hidden root certificates are captured and they cause a profound impact from the angle of web clients and traffic. Further, we identify around 5 thousand organizations that hold hidden root certificates, including fake root CAs that impersonate large trusted ones. Finally, we highlight that the implementation of hidden root CAs and certificates is highly flawed, and issues such as weak keys and signature algorithms are prevalent. Our findings uncover that the ecosystem of hidden root CAs is massive and dynamic, and shed light on the landscape of Web PKI security. Finally, we call for immediate efforts from the community to review the integrity of local root stores.

CCS CONCEPTS

• **Security and privacy** → *Web protocol security*; Authorization; Browser security; • **Networks** → *Web protocol security*.

KEYWORDS

Certificate Authority; Root Certificate; Web PKI

ACM Reference Format:
Yiming Zhang[1,2], Baojun Liu[1] 🖂, Chaoyi Lu[1,3], Zhou Li[4], Haixin Duan[1,5] 🖂, Jiachen Li[1] and Zaifeng Zhang[3]. 2021. Rusted Anchors: A National Client-Side View of Hidden Root CAs in the Web PKI Ecosystem. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3460120.3484768

1 INTRODUCTION

HTTPS provides encrypted communication and authentication mechanisms between web servers and clients, and is increasingly adopted on the Internet. Authentication between peers is powered by digital certificates [19], and the issuance, management and revocation of certificates heavily rely on a set of entities, systems and policies, which are jointly referred to as the Web Public Key Infrastructure (PKI). In the Web PKI trust model, digital certificates are typically issued to websites by organizations called Certificate Authorities (CAs). A small group of CAs (termed as *root CAs*) serve as *trust anchors* in the Web PKI ecosystem, such that certificates signed by them and subordinate organizations can pass verification.

Currently, several mainstream operating systems and browser vendors maintain public lists of trusted root CAs, including Microsoft [2], Mozilla [57] and Apple [40]. The behaviors of public CAs should undergo strict evaluation and regular audit [19, 33, 53], and CAs can be removed from trusted lists upon security incidents like certificate mis-issuance (e.g., ipS [69] in 2009, Trustwave [68] in 2012 and CNNIC [32] in 2015). For digital certificates to be verified, operating systems are usually pre-installed with a *local root CA store* that are copies from the public root programs. As a result, certificate chains that link to root CAs beyond local stores will be rejected by web clients.

However, recent security incidents and studies show that the management of local root stores can be the "Achilles' heel" of Web PKI security. By injecting self-built root certificates into local root stores, local software such as anti-virus and parent-control applications creates a man-in-the-middle proxy to filter SSL/TLS-encrypted traffic [21]. This approach can also be used by government agencies or malware, in order to monitor web users' online behaviors [20, 58]. For instance, reports in 2019 show that citizens in Kazakhstan were forced to import government-built CAs on their devices [58].

In this study, we term root CAs that are not trusted by public root programs as "*hidden*" root CAs, because they are absent from the lists and are not publicly visible. Particularly, we focus on hidden root certificates that *have been imported* into local root stores (i.e., have gained trust from web clients). Certificate issuance of hidden root CAs is usually not audited, allowing them to arbitrarily issue forged certificates and intercept secure connections, which breaks authentication and poses security threats [58]. The ecosystem of hidden root CAs have not been well-studied in literature, because it requires a *client-side view of local root stores*, thus existing certificate datasets (including active datasets [26, 38, 44] and passive datasets [6, 7, 54]) are not suitable for this task.

Research questions. In this paper, we report the first measurement study that sheds light on a nation-wide ecosystem of hidden root CAs. We aim at answering a set of research questions that are critical to understanding their security impact and operational flaws, including: *How many web clients witness and actively trust hidden root CAs? How much HTTPS traffic associates with certificate chains signed by hidden root CAs? Who are operating the hidden root CAs?* And finally, *are there implementation flaws of hidden root CAs and certificates signed by them?*

Our Approach. Seeking answers to these questions is made possible by cooperating with 360 Secure Browser [46], a leading browser vendor in China with more than 100 million monthly active users. Through careful design, the browser collects a small portion of certificate chains together with their client-side verification statuses from web visits of millions of volunteers (see Section 3.1). From certificate data collected during Feb and Jun 2020 (5 months), we manually build filtering criteria from the X.509 standard to identify hidden root certificates. We also design and implement a suite of automated methodology that groups root certificates by their subject names (see Section 3.2), and then classify their usage.

Major findings. Our research reports several critical observations on the hidden root CAs ecosystem. In general, the ecosystem is dynamic and updating, with new hidden root CAs emerging and quickly getting trusted by web clients. We identify *1.17 million hidden root certificates* that have been imported into local root stores of web clients. Based on their subject information, we identify 5,005 certificate groups, and certificates in each group come from the same organization. The impact of hidden root CAs can be profound, as they are witnessed in 0.54% of all web connections, affecting 5.07 million web clients and putting security connections at risk of interception.

Besides self-built root CAs of enterprises and local software, we also uncover a large number of *fake root CAs* that impersonate trusted ones to evade detection. For example, they use homoglyphs to replace characters in authentic names (e.g., `Verislgn` with an "l" and `NetWork` with an upper-case "W"). While not discovered by previous works at scale, we show that fake root CAs are highly trusted by web clients and pose security threats to up to 2.7 million web clients.

As for operational flaws, we find that the security status of hidden root CAs and certificates are worrisome: public key sharing, abuse of wildcards and long validity period are prevalent. More than 87.3% of hidden root certificates and 99.9% of leaf certificates that they sign violate at least one X.509 standard requirements. In particular, 97% of leaf certificates issued by hidden CAs use weak keys, increasing their chances of being compromised.

Scope of study. In this paper, we aim to uncover a nation-wide ecosystem of *hidden root CAs and certificates in the Web PKI* from web clients' perspective. While not included by public root programs, such root certificates have been trusted by web clients (e.g., through importing into local root stores [21] or public key-pinning [30]). Security threats are thus raised since secure web connections can be intercepted. We fill this knowledge gap by analyzing large-scale certificate data and verification statuses.

```
TBSCertificate
    issuer
        countryName = US, organizationName = cPanel, Inc.,
        commonName = cPanel, Inc. Certification Authority
    validity
        notBefore = 20210311000000Z, notAfter = 20210610000000Z
    subject
        commonName = sigsac.hosting.acm.org
    subjectPublicKeyInfo
        subjectPublicKey = D4 06 E9 CE … 8E 33 45 C1 (2048 bits)
    extensions
        subjectKeyIdentifier: keyIdentifier = C0 73 03 … 52 A0 9F
        authorityKeyIdentifier: keyIdentifier = 7E 03 5A … 6A C7 65
        basicConstraints: cA = FALSE
        keyUsage = digitalSignature, keyEncipherment
Signature = 5E AB F4 D0 CD C6 … 9C CE F0 D1 79 26 (2048 bits)
```

Figure 1: Snippet of an X.509 Version 3 certificate

2 BACKGROUND AND RELATED WORK

Below we provide background on the trust model and entities of the Web PKI, particularly on root CAs and certificates. We also provide the definition of the hidden root CA ecosystem, and present previous works relevant to our study.

2.1 Web PKI Infrastructure

The Public Key Infrastructure (PKI) provides authentication mechanisms between communication peers and is heavily relied on by secure Internet protocols (e.g., TLS). Specifically, the Web PKI refers to systems and policies that manage X.509 certificates issued to websites, such as Certificate Authorities (CA).

Trust model and root stores. In the trust model of the Web PKI, Certificate Authorities (CAs) are organizations that issue digital certificates to other entities. On top of the model are *root CAs* that hold self-signed *root certificates*. For flexibility and security considerations, root CAs often delegate their signing abilities by issuing *intermediate certificates* to other organizations [9, 29], which in turn sign *leaf certificates* to websites. The bundle of a leaf certificate and its signing certificates builds a *certificate chain*, linking the leaf certificate to a root certificate.

For web clients, a leaf certificate passes verification only if it has a valid chain to a root certificate that presents in its *local root store*. Local root stores are usually pre-installed copies of public root stores maintained by browsers and operating systems. Due to their significant importance, the included root CAs should undergo strict evaluation (e.g., by public root store policies [33]) and regular audit (e.g., through Certificate Transparency [12]), and can be revoked upon security incidents such as certificate misissuance [32, 68, 69].

In academic and industrial best security practice, public root stores maintained by Mozilla [57], Microsoft [2] and Apple [40] are most commonly adopted [5, 6, 21, 23, 65]. As of Sept. 2020, 590 root certificates are included by at least one of the three programs.

X.509 certificate format. Most certificates on the web adopt the X.509 Version 3 format [19] and Figure 1 shows a snippet. The major fields include distinguished names of its *subject* and *issuer*, its *subject public key* and *signature*, its *validity* period, and *certificate extensions*. Specifically for root certificates, additional requirements should apply in order to ensure proper verification of certificate chains according to the requirements of RFC standards (e.g., `basicConstraints` extension must be set to `TRUE`) [19, 33].

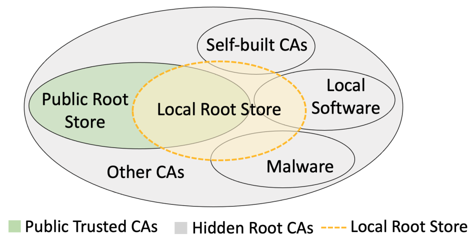


Figure 2: Overview of Hidden Root CA ecosystem.

These requirements are leveraged in Section 3 to filter root certificates in our dataset.

2.2 The "Hidden" Root CA Ecosystem

Aside from the hundreds of root CAs in public root stores (i.e., Mozilla, Microsoft and Apple), other entities may also function like root CAs and issue digital certificates. We term these as *"hidden"* root CAs and certificates, as they are not visible to public root programs and are difficult to uncover.

Some well-known sources of hidden root certificates are shown in Figure 2, including local software (e.g., VPNs), malware [20], enterprise networks [21] and government agencies [58] self-built CAs. Unlike trusted ones, hidden root CAs typically do not publish policies that are reviewed by other organizations. Meanwhile, their certificate issuance cannot be monitored by systems like Certificate Transparency [12].

For a web client, trusting hidden root certificates can be risky, as their behaviors are not publicly visible. While hidden root certificates should be rejected during certificate chain verification (because they do not present in public root stores), it is possible for them to be *imported* into local root stores (e.g., manually or by local software) [58]. If abused, secure connections can be intercepted and monitored with forged certificates [21].

The use cases of hidden root CAs make them fundamentally different from trusted ones, thus best security practices of certificate issuance may not apply globally. For example, hidden root CAs of malware are built to intercept secure connections, thus their adoption of weak keys and insecure algorithms should not be considered a problem. By contrast, self-built root CAs of enterprise networks and anti-virus software should comply with the security requirements, in order to prevent themselves from being compromised. To address this issue, in our methodology (Section 3) we group hidden root CAs and certificates into different categories and discuss them separately.

2.3 Related work

As HTTPS protocol is increasingly adopted in the web, efforts from the community have been devoted to studying security issues of the certificate ecosystem. Previous works focused on the measurement and analysis of HTTPS interception are most relevant to our study.

To summarize the security threats posed by HTTPS interception, [21] and [67] studied dozens of client-end applications (e.g., antivirus software and proxies) that insert self-build root certificates and perform TLS interception. They evaluated the whole process of certificate generation and validation by code analysis, and uncovered several implementation flaws. Besides, Durumeric et al. further analyzed HTTPS interception from a web-server view and concluded that the presence of middle boxes can heavily reduce the security of HTTPS connections [28].

To describe the prevalence and reasons of HTTPS interception, in 2014, Huang et al. actively sent SSL handshakes towards Facebook web-server and extracted certificates [39]. They found an interception rate of 0.2%, and manually categorized the root certificates of hijacked traffic according to their `Issuer` fields. Following, in 2016, O'Neill et al. conducted similar experiments, detected TLS interception by actively connecting controlled servers, and also discovered HTTPS hijacking from malware and spam tools based on information of `Issuer` fields [56].

Unlike previous works, in this study, we seek to *understanding and characterizing the ecosystem of hidden trust anchors in the client-end local root stores from a national-wide view*, rather than simply detecting TLS interception behaviors from the perspective of a handful of websites. As a result, although previous studies have collected and built various certificate datasets, we find that none of these datasets could be applied in our study due to the lack of user-side certificate verification status.

Typically, certificate data are captured by active scanning or extracted from passive traffic, and could be leveraged to study certificate mis-issuance by CAs [6, 26, 44] and vulnerability assessment [5, 27, 38].

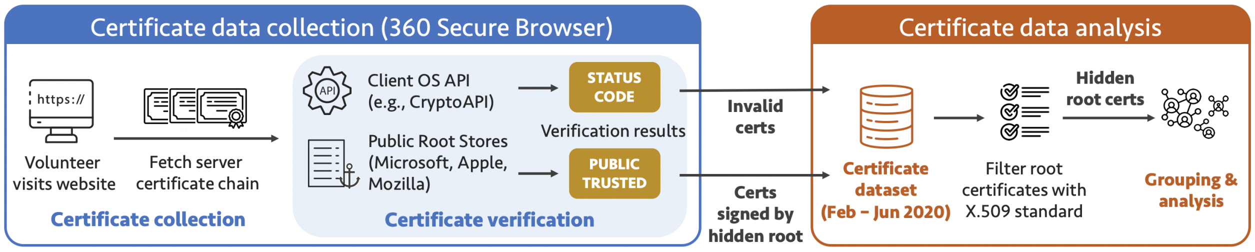
Specifically, as for active certificate datasets, Holz et al. [38] studied the X.509 infrastructure by actively scanned Alexa Top Sites and passively collected traffic from three institutions. Durumeric et al. [26] completed 110 TLS scans on the entire IPv4 address space and studied the status of certificate issuance and HTTPS adoption rates. Following, they published Censys [25], a database constructed by regular Internet-wide scans, including scans of certificates. Censys has been leveraged by to measure HTTPS adoption rate [31] and study certificate revocation strategies [60]. Similarly, in 2016, Cangialosi et al. released another dataset called Rapid7 [10], which actively scans certificates in the IPv4 address space every week.

As for passive certificate datasets, ICSI networking group built SSL Notary [7] that passively extracts certificates from TLS traffic of ten institutions. SSL Notary has been used to study TLS warnings [5] and trust relationships [6]. In 2017, Acer et al. from Chrome analyzed certificate compliance errors reported by browser [4]. Similarly, another recent work was done by Oakes et al. in 2019, who harvested and analyzed residential certificate chains by cooperating with a web monitoring software [54]. In addition, Certificate Transparency (CT) [35] is now widely supported on the web [61], which aims to collect and audit all trusted certificates.

However, these datasets cannot be applied in our study. To identify hidden root certificates in user-side local stores, we need to collect certificate chains together with their *verification statuses* from the view of the client-end. Therefore, datasets collected by active scans are not suitable, and public passive datasets like Notary do not contain the verification statuses. We address this challenge by collaborating with a leading browser vendor and collect certificate data from real visits of volunteers (Section 3).

3 METHODOLOGY

In this section, we elaborate on our workflow of certificate data collection and methodology of root certificate grouping. Then, we also discuss potential ethical concerns before presenting our results.



Certificate data collection (360 Secure Browser)
https://
Volunteer visits website
Fetch server certificate chain
Certificate collection
Client OS API (e.g., CryptoAPI)
Public Root Stores (Microsoft, Apple, Mozilla)
STATUS CODE
PUBLIC TRUSTED
Verification results
Certificate verification
Invalid certs
Certs signed by hidden root
Certificate data analysis
Certificate dataset (Feb – Jun 2020)
Filter root certificates with X.509 standard
Hidden root certs
Grouping & analysis

Figure 3: Methodological overview of certificate data collection and analysis

3.1 Certificate Data Collection

The goal of this study is to identify hidden root certificates in use and to evaluate the real-world impact of those trusted by web clients (i.e., installed into local root stores). To this end, we collaborate with 360 Secure Browser [46], a popular PC browser built over Chromium [13] with over 100 million monthly active users in China mainland [45]. Since 2018, the company has been maintaining its own root certificate store [47] that is shipped into the browsers. To help evaluate CAs in the wild and decide whether they should be included, 360 Secure Browser collects certificate data from volunteer users who opt-in this program. Data collected **from Feb 1, 2020 to Jun 30, 2020** (5 months) is provided to us for research purposes.

Certificate data collection procedure. When a volunteer user visits a website over HTTPS with 360 Secure Browser, the browser first fetches and verifies the web server's certificate chain. Same as Chromium[1], 360 Secure Browser calls underlying operating system APIs (e.g., CryptoAPI on Windows [51]) for certificate verification. The APIs return a double-word STATUS_CODE carrying *all* verification errors that occur. Each bit of the code indicates one type of verification error and we list the mappings in Table 9 of Appendix A. If any bit is set in STATUS_CODE (i.e., an error occurs), the certificate chain is considered *invalid* and the browser terminates the connection, showing an error message to its user.

Particularly, an asserted AUTHORITY_INVALID bit indicates that the root certificate is not trusted by the client's local root store. In addition, the browser examines whether the root certificate includes any of three large public root programs (Microsoft, Mozilla and Apple), and encodes the result in an additional PUBLIC_TRUSTED flag. This extra check is done backstage by matching the public keys of root certificates, and the result is not shown to web users.

For ethical considerations (Section 3.4 discusses more), a certificate chain is collected only when PUBLIC_TRUSTED is unset, or if any bit is set in STATUS_CODE. That is, the browser only collects certificate chains that *link to hidden root CAs*, or are considered *invalid* by the client's operating system. Valid certificate chains signed by root CAs in the three public programs are not recorded and are only counted for statistical purposes. Figure 4 shows the format of a collected certificate chain, including (anonymized) client ID, time of collection, hashes and PEM encoding of each certificate and verification status codes.

Filtering hidden root certificates. Following, we need to filter root certificates from certificate chains for further analysis. Due

[1]Historically, Chrome and Chromium use the local root store of underlying operating systems, but are transitioning to their own root program [14]. 360 Secure Browser does not use the Chrome Root Store.

```
{
   "client_id": 1579968000000, "host": example.com,
   "collected_time": 2020-02-01T00:00:00Z,
   "leaf_sha1": "35 FE 12 … 83 5A FB",
   "parents_sha1": ["A3 A1 … D7 3A", "A8 98 … 54 36"],
   "status_code": [AUTHORITY_INVALID, WEAK_KEY, DATE_INVALID],
   "public_trusted": False
   "certs_pem": [(leaf_cert_pem), (parent1_pem), (parent2_pem)]
}
```

Figure 4: Example chain with three certificates. The root certificate is not trusted by the client's local root store (AUTHORITY_INVALID is set) or the public root programs (PUBLIC_TRUSTED is unset). The chain is considered invalid by the web client (STATUS_CODE has bit asserted).

to web server misconfiguration and network errors [5], the PEM-encoded certificate chain (field CERTS_PEM in Figure 4) can be disordered or incomplete. As a result, simply taking the last certificate in the chain as root can be error-prone. Fortunately, the X.509 RFC standard [19] has special requirements for root certificates and we check all certificates against them.

We identify the root certificate of a chain, if it satisfies *all* of the following criteria:

- The certificate includes a basicConstraints extension with a cA value set to TRUE.
- The certificate includes a keyUsage extension with an asserted keyCertSign bit (the first two criteria indicate that the certificate is held by a CA).
- The distinguished names of subject and issuer are non-empty and identical.
- If included in the certificate, the subjectKey Identifier and authorityKeyIdentifier are identical (the last two criteria are set because root certificates are self-signed).

To further filter *hidden* root certificates, we leverage the PUBLIC_TRUSTED flag drawn from the browser's additional check. If a root certificate is identified in a chain with the above criteria, plus the PUBLIC_TRUSTED flag of this record is unset, we label it as a hidden root certificate. If no root certificate can be found in a chain, we consider the record incomplete and remove it from our dataset. Finally, the AUTHORITY_INVALID flag from certificate verification APIs tells whether the hidden root certificate is rejected by the client, or has been installed into the local root store.

3.2 Grouping Hidden Root Certificates

Our next research questions are "*who owns the hidden root certificates*" and "*how are they currently operating*". Recall that hidden root certificates can be held by various parties, such as local software and enterprise networks. One organization can also create multiple root certificates with different public keys but identical or similar subject names (e.g., "Globalsign Root R1", R3 and R6), which often share the same properties (e.g., length of validity period). We aim to identify *certificate groups* held by the same or similar organizations, and then classify their usage.

Inferring certificate ownership. We acknowledge that, due to a lack of ground-truth, identifying the ownership of root certificates is difficult even for those in public root programs. To our best knowledge, Mozilla's Common CA Database (CCADB) [52] provides the most comprehensive and audited ownership information of root certificates. A recent work [48] supplements the CCADB dataset and now labels ownership for 6,846 root certificates [49]. However, as hidden root CAs are invisible to public programs, none of our collected certificates are found in the public datasets.

Without auxiliary information from external data feeds, we follow the practices in previous studies [39, 44, 56] that leverage subject distinguished names to infer certificate ownership. That is, certificates with an identical set of commonName (CN), organization (O) and organizationUnit (OU) in its subject belong to the same organization.

Identifying subject templates. CAs holding multiple root certificates often generate them with *subject templates*, which should be clustered as one certificate group. For example, the subject commonName in root certificates of Whistle (a web debugging tool) follows a template of CN=whistle.[0-9]*. A previous research [23] manually identifies 46 subject templates in regular expressions, and we check our dataset against their list. Unfortunately, only 3 templates hit our data (2 templates match only one certificate).

To comprehensively and automatically discover subject templates from certificates, we are inspired by Server Log Parsing tasks that identify templates from inconsistently formatted logs. Considering subject names as logs, we select the Drain algorithm [37] for this task. Designed over a 5-layer Directed Acyclic Graph (DAG), Drain takes raw log messages as input and outputs structured message templates, and produces 92% to 99% parsing accuracy on 11 log datasets [36]. For each certificate in our dataset, we concatenate all fields in its subject distinguished name (e.g., commonName and organization) into a log string as the input of Drain. For compatibility reasons, special characters that are not common in log messages (e.g., commas and at signs) are replaced with white spaces. In total, Drain outputs 13 subject templates from 1.1 million hidden root certificates (including one discovered by [23]), and no false positives are found during manual verification (see Section 3.3 for evaluation).

Unsupervised Grouping. After identifying subject templates, hidden root certificates that match one of the following criteria are grouped:

- Certificates have identical, non-empty values of commonName (CN), organization (O) and organizationUnit (OU) in their subject distinguished names.
- Subject distinguished names of certificates match the same template.

3.3 Evaluation and Limitations

Evaluating hidden root certificates. During data collection, we use the PUBLIC_TRUSTED flag provided by 360 Secure Browser to identify hidden root certificates, and here we use the Certificate Transparency (CT) database to verify if they are truly beyond public root stores. To monitor the certificate issuance procedure, since June 2016 all CAs in public root stores are required to submit all certificates that they sign to CT for future queries [12]. As a result, hidden root certificates *should not appear* in CT databases.

For evaluation, we query all (1.19 million) hidden root certificates in the CT database provided by our industrial partner's CT monitor and find only 6 hits. After manual inspection, we confirm that these certificates are falsely included in the CT database because several CT nodes use expanded root stores [43]. In the end, we are confident that the PUBLIC_TRUSTED flag of 360 Secure Browser is effective, and that the hidden root certificates we identify are not included by public root programs.

Evaluating grouping algorithm. To group hidden root certificates, we use the Drain algorithm to identify templates from subject distinguished names. To our best knowledge, there have not been other methods designed specifically for this task and we select Drain because it is a state-of-the-art tool and shows good accuracy. Due to a lack of template ground truth in our dataset, we choose to manually evaluate the algorithm on a sample of root certificates. In detail, we inspect a random sample of 10,000 hidden root certificates in our dataset, and manually generate 11 subject templates. Drain identifies all 11 templates, and the other 2 templates from the larger dataset are missed because the number of their associated certificates is low in our random samples. As a result, though not specifically designed for this task, we believe that Drain outputs correct subject templates with high confidence.

Limitations. First, to identify root certificates from certificate chains, we set filtering rules that are driven from the standard requirements of X.509 format (see Section 3.1). We understand that some hidden root CAs (e.g., of local software) may not follow the RFC standard requirements and will thus be overlooked. However, we believe this filtering process also helps us to remove false positives. And we prefer to provide lower-bound numbers rather than inflating them. Besides, if a client is infected by malware, it may also cause the data collection results to be polluted. Although our results theoretically only reflect a lower bound of the hidden root CA ecosystem, as shown in Section 4, we are still able to identify more than 1.19 million hidden root certificates that are witnessed by real clients. Therefore, we believe our dataset is sufficient to shed light on a nationwide ecosystem. Second, due to a lack of audited ownership data, we borrow the approach from previous works and infer certificate ownership from its subject distinguished names. We acknowledge that the subject names may not point to the real organizations behind, for example, self-built CAs of malware may fill deceptive names in the subject fields. While this approach inevitably has limitations, for unidentifiable certificates, we attempt to search in several auxiliary data feeds including threat intelligence and sandbox logs, as a best-effort approach. Third, instead of sensitive PII (Personal Identifiable Information) such as the user's IP address or device ID, 360 Secure Browser utilizes the timestamp when the user joined the data collection program as its client ID (see *client_id* in Figure 4), to better protect user privacy. Although this client ID is less accurate than PII, e.g., one timestamp could be associated with more than one real user and reinstalling the browser will change the ID, we believe this cost of inaccuracy is acceptable to mitigate privacy risks.

3.4 Ethical Considerations

One major ethical concern of this work centers on certificate data collection from users of 360 Secure Browser. To understand the ecosystem of hidden root CAs, we need certificate chains together with their verification statuses at the client-end. This goal cannot be fulfilled through active scanning (e.g., Censys [25]) or passive traffic analysis (e.g., ICSI Notary [7]). The dataset is only collected from volunteer users that join the program in "opt-in" mode. Consent is provided to users of 360 Secure Browser that states the collected data (e.g., certificate chains and verification statuses), purposes and benefits (e.g., help evaluate the 360 Root Certificate Program [47]), and needs explicit agreement. The data collection process is under the supervision of legal departments of the company (providing similar functions as an IRB).

Another concern is the collected certificate data may harm user privacy, as the history of web visits is obtained (e.g., through `hostname` in certificate records), and several signed hosts themselves may also be sensitive. We tried our best to mitigate potential ethical risks in this regard. First, the data collection methodology only captures certificate chains that link to hidden root CAs or that are invalid, only accounting for 0.54% of all web visits (Finding 4.1). Thus the majority of web visits associated with valid certificate chains signed by trusted root CAs are not collected and are only counted for statistical purposes. Meanwhile, we carefully anonymize real users by using the timestamp they joined the data collection process as the client-IDs, which could avoid harvesting any sensitive PII (Personally Identifiable Information) of volunteers. Besides, when analyzing invalid websites and hosts signed by hidden roots, we focused only on insensitive statistics such as the overall scale and percentage of different certificate validation errors. We did not perform any in-depth examinations of the domain content or their access relationship between clients to minimize privacy concerns.

The collected certificate dataset is securely stored on servers that maintain the client-side data by 360 Secure Browser. Researchers in this project obtain access to the dataset via a temporary internship. All analysis programs are run on the company's virtual environments and we do not share the dataset with third parties. In the end, through our best efforts, we believe that operations in this study adhere to ethical conventions.

4 SCALE AND IMPACT ANALYSIS

This section uncovers the basic characteristics (scale, ownership, active patterns, impact) of the hidden CA ecosystem.

4.1 Scale of Hidden Root Certificates

Finding 4.1: Within a five-month period, over 1.19 million hidden root certificates are detected as being used to threaten the security of HTTPS connections, covering 0.54% of all visits.

Dataset overview. Table 1 overviews our dataset. During Feb and Jun 2020, volunteer users of 360 Secure Browser produced over 41 billion web visits over HTTPS. The browser collects over 222 million certificate chains that link to a hidden root, and they account for 0.54% of all web visits. Using the criteria in Section 3, we identify over 1.19 million distinct hidden root certificates (distinguished by certificate hash values) that meet the X.509 standard requirements.

To evaluate their security impact, we then split the hidden root certificates by whether they have been trusted by web clients. Generally, hidden root certificates are invisible to public root programs, thus they should be rejected during verification. However, surprisingly, we find that *only 21 thousand (1.8%) hidden root certificates are rejected by all web clients that witness them*. It suggests that, though not included in public root programs, hidden root certificates have actually been widely installed into local root stores. In the following analysis, we neglect the 21 thousand root certificates that are rejected by all clients, as their security impact is insignificant.

Certificate group overview. We run the grouping algorithm on all 1.17 million hidden root certificates that are imported by clients, and it produces *5,005 certificate groups*. Root certificates in one group are considered to be held by the same organization.

In terms of the size of certificate groups, as expected, the distribution has a long tail: 4,362 groups (87.2%) only include one certificate, and the remaining 12.8% groups account for 99.6% of all certificates. Figure 5 corresponds to each certificate group with their numbers of associated HTTPS connections and web clients that witness them. Dozens of certificate groups (at the top right corner of Figure 5) cause profound impact. For example, the largest group contains 254,412 root certificates that belong to `Certum Trusted NetWork CA 2` (impersonating Certum CA [11]; the authentic CA has a lowercase 'w' in "Network"). Another group contains only 2 certificates that belong to `VeriSlgn trust Network` (impersonating Verisign CA [66] by replacing the 'i' in "Verisign" with 'l') but is witnessed by over 1 million web clients. In Section 4.3 we further discuss the impersonation behaviors of hidden root CAs.

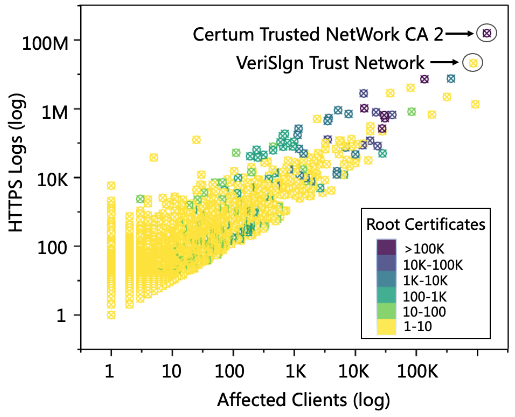


Figure 5: Size and impact of certificate groups

In the following sections, we focus on the top 100 certificate groups that each associate with over 1,000 web clients and 5,000 HTTPS visits. The top 100 groups account for 97.5% of all hidden root certificates that are imported by clients, as well as 98.9% of their associated HTTPS visits.

4.2 Active Time

We define the active date of a hidden root certificate as the number of days when certificate chains linked to this root are captured

Table 1: Certificate dataset overview

Type of Root Cert	Filtering Condition	# Distinct Root Certs	# Cert Chains	# Leaf Certs	# FQDNs
Hidden CAs	`PUBLIC_TRUSTED=FALSE`, Trusted by at least one client	1,175,145 (98.24%)	222,977,356	59,817,585	1,333,931
	`PUBLIC_TRUSTED=FALSE`, Rejected by all clients	21,010 (1.76%)	263,109	112,946	15,566
Public Trusted CAs	`PUBLIC_TRUSTED=TRUE`, `STATUS_CODE` has bits set	615	241,541,342	3,647,095	1,871,131

by the browser. During the 5-month data collection period, 5,373 (0.4%) hidden root certificates are active for over 100 days. Looking from the angle of the top 100 certificate groups, their active time in the 5-month period can be up to 146 days on average. That is, the vast majority of organizations behind the top groups have at least one active hidden root certificate every day that is trusted by web clients.

Finding 4.2: The ecosystem of hidden root CAs is dynamic and updating, as new emerging CAs and retiring ones are both observed.

To quantify the "stability" of top certificate groups, we calculate the coefficient of variation (CV, the ratio of standard deviation to the mean) of their daily count of associated HTTPS visits. A stable certificate group should have daily traffic volume that does not vary greatly. Empirically, certificate groups with a CV greater than 1 are considered unstable (for reference, the CV of all daily traffic associated with hidden root certificates is 0.33). We find 13 certificate groups that match this criterion, and Figure 6 shows some examples. The first group is only active for one day (Mar 14), which belongs to a web debugging tool. We suppose it is included in our dataset because of temporary tests. The following two certificate groups are *retiring*. The traffic decrease of `CEGN` certificate group is likely caused by a failure of promotion, as the vast majority of clients reported an `AUTHORITY_INVALID` error. The root certificate held by `BJCA` was used to sign `yzt.beijing.gov.cn`, but was replaced on Apr 18 by a trusted certificate chain. The bottom three certificate groups are *emerging*, as their count of root certificates and associated HTTPS traffic both rise (independent of the amount of overall traffic), which suggests a possible growth in web clients that are trusting them.

Interestingly, we find that over 183 thousand (15%) of hidden root certificates are created *during* our 5-month data collection period, and are quickly gaining trust from web clients. The largest batch of creation appeared on Jun 13 with 2,853 new root certificates. Among this batch, 2,505 certificates belong to one fake CA (Sec 4.3 describes more) that impersonates GlobalSign [62].

4.3 Categories of Ownership

Finding 4.3: Three major sources of hidden CAs are identified through manual inspection: Self-built CAs (50 groups), Fake root CAs (11 groups) and Local software CAs (24 groups).

Due to a lack of ground truth, determining the category and purpose of each certificate group is non-trivial. We choose to manually infer this information from the subjects of the top 100 root

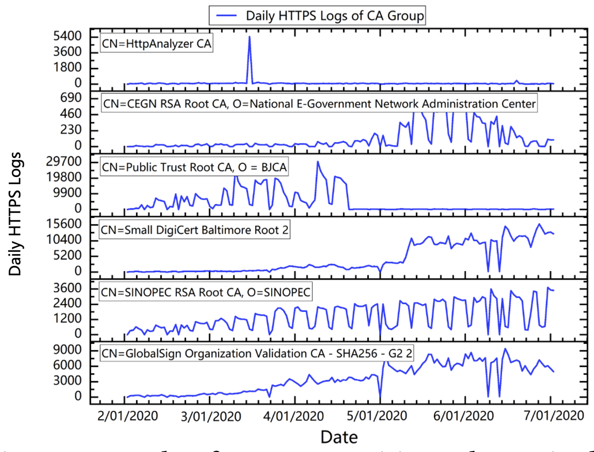


Figure 6: Examples of temporary, retiring and emerging hidden root certificate groups.

certificate groups. For example, subject `O=Venus, OU=Venus VPN` is used by Venustech VPN [42] and categorized as local software. If the subject is not directly identifiable, we search keywords and hash values of the certificate in search engines to infer its ownership. The manual classification is performed by three security researchers, and a certificate is classified only if over two researchers give the same label.

As shown in Table 2, the ownership of 85 certificate groups are identifiable and fall into three categories, including "self-built CAs of organizations", "fake CAs" and "local software". Another 15 groups fall into the "unknown" group due to a lack of identifiers.

Finding 4.4: Self-built CAs are still widely used by organizations such as government agencies and enterprises to issue website certificates. However, over 75% of certificate chains suffer from verification errors (e.g., weak signature algorithms).

Self-built root CAs of organizations. 50 of the top 100 certificate groups are held by self-built root CAs of enterprises, independent organizations (e.g., digital authentication centers) and government agencies. In total, they sign certificates for 3,311 fully-qualified domain names (FQDNs). However, more than 75% of certificate chains that link to self-built root CAs receive verification errors from web clients (i.e., are invalid). The most common error is `WEAK_SIGNATURE_ALGORITHM` (78.3% of all errors), suggesting prevalent adoption of weak security practices.

One may argue that self-built root CAs are for internal use only (e.g., in enterprise networks) and that domains signed by them are not publicly accessible; despite that their security policies are worrisome, their impact is insignificant. However, we perform an

Table 2: Subject category of top 100 hidden certificate groups

Categories	# clusters (groups)	# hidden root certs	# affected connections	# affected clients	Invalid (Authority)	Invalid (Other)	Example of hidden root certificate
Enterprise Self-built	24	48	2,071,344	199,743 (3.94%)	35.54%	75.66%	CN = SZSE ROOT CA, O = Shenzhen Stock Exchange
Digital Authentication	13	18	3,261,905	539,711 (10.65%)	28.37%	96.66%	CN = CFCA ACS CA, O = China Financial Certificate Authentication
Government Self-built	13	16	314,351	62,032 (1.22%)	30.46%	89.67%	O = National E-Government Network Administration Center
Fake Authentications	11	817,532	192,901,548	**2,798,985 (55.21%)**	**0.00%**	0.25%	CN = VeriSlgn Class 3 Public Primary Certification Authority - G4
Packet Filter	11	15,587	3,622,177	73,725 (1.45%)	13.57%	14.39%	CN = NetFilterSDK 2
Proxy/VPN	10	90,131	3,050,138	1,029,648 (20.31%)	2.26%	4.27%	CN = koolproxy.com, O = KoolProxy inc
Security Software	2	7,187	509,645	4,719 (0.09%)	0.01%	0.32%	O = Beijing SkyGuard Network Technology Co., Ltd
Parent Control	1	7,554	70,8129	7,787 (0.15%)	0.00%	0.57%	CN = UniAccessAgentFW 2
Unknown	15	207,957	14,048,377	289,198 (5.07%)	2.89%	4.73%	CN = VRV NDF RootCA 2

additional scanning experiment showing that a majority of self-built root CAs do sign certificates for websites open to the public.

We deploy several controlled Virtual Private Servers on Alibaba-Cloud [18], located in the US, Singapore, Germany and China. On each machine, we use `OpenSSL` to fetch and verify certificate chains of the 3,311 FQDNs that link to self-built root CAs in our dataset. In January 2021 (6 months after the data collection period ended), we are able to fetch certificate chains of 2,439 FQDNs (73.6%) that still link to 36 (72%) self-built certificate groups. We also inspect the remaining 14 organizations that disappear in the scanning experiment, and find 7 of them are for internal use, thus the certificates cannot be reached by our active scanners. The remaining 7 groups are retired by new self-built root CAs or publicly trusted root CAs. In the end, our measurement findings suggest that weak security implementations are common among self-built root CAs, and should be fixed as they widely sign websites that are publicly accessible.

Finding 4.5: Fake CAs which impersonate large trusted CAs with good reputation to evade detection, are becoming emerging security threats. These CAs infecting more than 2.7 million devices and are highly trusted by web clients, as nearly none reports "Authority Invalid" errors.

Fake root CAs. 11 of the top 100 certificate groups are classified as fake root CAs that *impersonate trusted CAs* with deceptive subject names. As shown by examples in Table 3, they replace characters in authentic CA names with homoglyphs (e.g., `Verislgn` with an "l" and `NetWork` with an upper-case "W") or extend their words (e.g., extend `GlobalSign` to `GlobalSignature`). However, none of the certificates' public keys are present in the official lists of large CAs (e.g., public key lists of GlobalSign [62] and Certum [11]).

In Table 2, we show that these fake CAs associate with the most web visits (192M) and certificate chains (2.7M), and almost *all* of them are trusted by web clients (only 0.0001% of connections receive an `AUTHORITY_INVALID` error). Compared to self-built CAs that commonly use weak keys and algorithms, fake root CAs are more secure in implementation, with an invalid rate of only 0.25%. However, fake root CAs still introduce security threats, as they have been found to illegally issue certificates for popular websites [39].

Uncovering the *real* owner behind fake root certificates is challenging, as the subjects are deceptive names that do not provide valuable information. To explore by our best effort, we leverage threat intelligence systems and sandbox logs of malware. We first search all fake root certificates in threat intelligence systems (such as VirusTotal), but only find one hit. The matched certificate is associated with a Trojan [17] and is witnessed by 6.8 thousand web clients in 75 thousand web visits during our data collection period. Following, we also seek for cooperation with two leading security companies and match the certificates in their sandbox logs of malware samples. During malware installation, the sandboxes monitor the local root store to track modifications. We are able to find logs of 43 fake root certificates in our dataset, which associate with "Trojan", "CoinMiner" and "Adware". Again, we are not able to identify the real owner of all fake root certificates in our dataset, but information embedded in threat intelligence and sandboxes show their potential connection to malicious parties. However, considering the large volume of HTTPS connections and web clients that are affected by fake CAs, we speculate that except for malicious software, hidden certs in this category may also come from other sources, including spamming tools and free applications that irregularly intercept user traffic in an impersonal manner.

Table 3: Examples of fake Certificate Authorities

Subject Common Name	# hidden root certs	# connections	# FQDN
Certum Trusted NetWork CA 2	254,414	158.54M	1,137,121
VeriSlgn Class 3 Public Primary Certification Authority - G4	2	21.20M	210
GlobalSign Root CA	1,419	7.61M	6,023
GlobalSignature Certificates CA 2	1	2.85M	74,555
GlobalSign Root CA R3	136,196	1.03M	47,347
Small DigiCert Baltimore Root 2	135,258	0.65M	30,316

Local software. 24 of the top 100 certificate groups are held by local software, including 11 packet filterers, 10 Proxies/VPNs, 2 security software and 1 parent control application. These hidden root certificates are imported at software installation, for purposes such as virus detection, download acceleration and ad blocking. As we do not find direct evidence of malware in this category (which might be identified as fake CAs instead), the software together with its root certificate is likely installed manually by web users (i.e., users want their connections intercepted for benign reasons).

In previous studies of HTTPS interception, benign local software accounts for the most intercepted traffic. For example, [23] reports that anti-virus software performs 53% of all HTTPS interception, and the ratio for firewalls in [56] is 69%. However, in our dataset the ratio of intercepted connections by local software is lower, only 3.58%. As for the reasons, on the one hand, local software involved in our study may be not implemented with identifiable subject information, resulting in their classification as unknown at best. On the other hand, we observe the hidden CA ecosystem from a different, passive perspective than previous active measurements, which may better reflect the nature phenomenon. That is, "user-informed" or "regulated" (e.g., security software and well-identified agents) interceptions may not predominate as commonly expected.

Other unknown owners. From manual subject classification, we cannot identify 15 certificate groups, holding a total of 207K (17.6%) hidden root certificates. Most of them use neutral keywords to create certificate subject names, such as `root`, `trust`, `tech` and `ca`, and we do not have search results of their public keys in search engines or threat intelligence databases.

4.4 Impact on Web Hosts and Web Clients

Finding 4.6: On average, we observed 1.48 million HTTPS traffic from 300 thousand clients associated with hidden root CAs per day, demonstrating the broad impact on web clients.

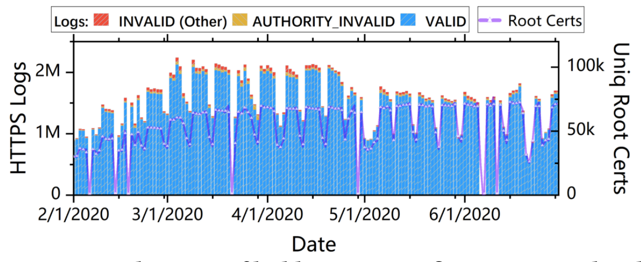


Figure 7: Daily count of hidden root certificates captured and associated logs of HTTPS web visits.

Scale of web visits. Figure 7 shows the daily count of hidden root certificates witnessed by clients and their associated web visits. On average, 1.48 million secure connections use certificate chains that link to a hidden root. During the 5-month data collection period, 222 million certificate chains that link to a hidden root are recorded, accounting for 0.54% of all web traffic. We also find that up to 96.4% of traffic do not receive *any* verification errors from local operating systems. Only 1.5% received `AUTHORITY_INVALID` errors, and the remaining 2.1% of certificate chains are invalid because of other issues (e.g., `DATE_INVALID` which means the certificate is not within its validity period). This result echoes with our earlier findings that hidden root certificates are widely imported by web clients (Findings 4.1 and 4.5).

Hosts signed by hidden roots. As presented in Table 1, a total of 1.3 million fully-qualified domain names (FQDNs) were found to be issued certificates from hidden roots in the period of data collection. Further, we examined the popularity of these domains with Alexa Top Rank [41]. As shown in Table 4, 815K (61.11%) of FQDNs are within the top 1M rank. Also, there are a significant number (519K, 38.89%) of long-tail domains (not ranked within top 1M) being affected by hidden roots. Besides, in addition to hosts in the subject fields, certificates are also valid for entities in `subjectAlternativeName` (SAN) extensions. As defined in the X.509 standard [19], hosts embedded in these extensions are also threatened by hidden root CAs. By extracting the two most common formats, `dnsName` and `ipAddress` from SAN extensions in leaf certificates, we totally identify that 1.54 million FQDNs under 792K Second-Level Domains (SLD) and 12,496 distinct IP addresses covering 48 countries are "potentially" affected. This result also illustrates the widespread impact of hidden roots on web servers.

Due to ethical concerns, our data analysis of affected hosts is limited to the above statistics. In order to protect user privacy, we neither analyzed the specific content of the hosts nor did any in-depth exploration of the domains accessed by individual customers to avoid exposing sensitive information.

Table 4: FQDNs signed by hidden root certificates grouped by Alexa rankings. One domain can have multiple FQDNs.

Alexa Rank	1-100	100-10K	10K-1M	>1M
# FQDN	386K (28.95%)	309K (23.13%)	120K (9.03%)	519K (38.89%)
# connections	110M (49.42%)	68M (30.42%)	16M (7.32%)	29M (12.84%)

Scale of web clients. We use the anonymized client ID in the collected records to give an estimation[2] of web clients that witness hidden root certificates. In total, 360 Secure Browser captures hidden roots from 5.07 million volunteer users during the 5-month data collection period, with an average of 300 thousand clients per day. 4.67 million (92.1%) web clients have trusted at least one hidden root CA (i.e., the `AUTHORITY_INVALID` bit is cleared for at least one hidden root certificate), again echoing with our findings that hidden root certificates are widely trusted. We also find that 95% of such clients only have one hidden root certificate through analysis of their verification codes.

Although on average 0.54% of daily web traffic is covered by hidden roots, the proportion per individual client varies widely. For more than 95% of clients, the percentage is less than 0.01%, while 0.28% of clients have more than 90% of their web visits impacted. To figure out why certain clients were impacted so heavily, We sampled 104 cases who had more than 500 web visiting records and an affected rate of over 99% for further analysis. One may attribute this high percentage to interceptions from Local Software like proxies and packet filters, but we find this situation appeared on only 10.58% clients (11 of 104). On the contrary, hidden roots from Fake Authentication (64 clients, 61.54% of 104) lead the pack. By examining the traffic timestamps of those clients, we also find that, hidden roots from Fake Authentication would be constantly updated on the client-side, possibly to avoid detection. Specifically, 3 of the 104 cases had successively installed more than 20 hidden root certificates from the same issuer, and the average lifetime (the period they appeared in traffic logs, rather than the validity period) of each root did not exceed 1 day.

Finding 4.7: By further exploring trust relationships between hidden CA groups and affected clients, we identified fake CA groups that may come from the same malware family, and unknown groups potentially being associated with fake CAs.

The trust relationship between hidden CA groups could provide insights into "what actually happened to affected clients". To explore whether the same set of hidden CA groups are trusted by the

[2]The browser logs the timestamp when a volunteer joins the data collection program as client ID (see Section 3.4). We roughly correspond a client ID to one volunteer user.

same set of clients, we clustered clients that trusted hidden CAs from at least three identical groups at the same time. 127 such sets were found in this way along with several interesting findings. For example, three fake CA groups hijacked traffic of the same set of 309 clients in different orders, suggesting that they may belong to the same family and successively infected this set of clients. We also found one government-owned CA, one digital authentication, and the CA from one VPN simultaneously installed on 195 clients, with the associated traffic flowing mainly to the intranet or government-related hosts. These clients are likely used within government-related agencies that install the hidden CAs for working purposes. Besides, three groups marked as "unknown" were found installed on the same set of clients with two or three fake authentication groups, implying possible relationships between their upstream operators.

5 IMPLEMENTATION FLAWS

As trust anchors of the Web PKI model, root certificates are heavily protected and should follow strict implementation requirements. In this section, we provide the analysis of the implementation flaws of hidden root certificates.

5.1 Certificate Misuse

Finding 5.1: Over 97% of chains in hidden CA ecosystem are signed directly by root certificates instead of intermediate ones.

Direct signing with root certificates. For security considerations, root certificates should be heavily protected and stored offline [9, 29], and leaf certificates are usually signed by intermediate certificates. Previous studies[3] report that the typical length of valid certificate chains signed by trusted root CAs is 2 to 3 (excluding the root certificate) [54]. By contrast, for hidden root CAs, we find that most chains (97.55%) are signed *directly* by root certificates without intermediate ones. For local software (e.g., VPN) that needs to sign certificates in real-time, this option is reasonable for flexibility. However, we also find that 41.4% of hidden root CAs owned by government agencies and enterprises exhibit direct signing behaviors. The direct signing behavior of hidden root certificates increases their risk of being compromised.

Finding 5.2: Public key sharing between root certificates is prevalent in the hidden CA ecosystem, with 144 groups suffering from this threat.

Public key sharing. Among the 5,005 certificate groups, 643 include over one root certificate and we find public key sharing is common. Figure 8 shows the count of public keys and certificates in each group. Surprisingly, 144 groups (22.4%) use one key to issue *all* hidden root certificates, and 36 of them hold over 10 certificates. For a CA, sharing public keys among its root certificates increases the risk of being compromised [21], particularly when the root certificates are directly used to sign leaf certificates (Finding 5.1). While the certificate groups typically belong to fake root CAs, two kinds of security software are also found to use the same public key for over 7 thousand certificates that they hold.

[3]We do not perform a comparison with trusted certificates in our dataset, because only invalid ones are collected and the results can be biased.

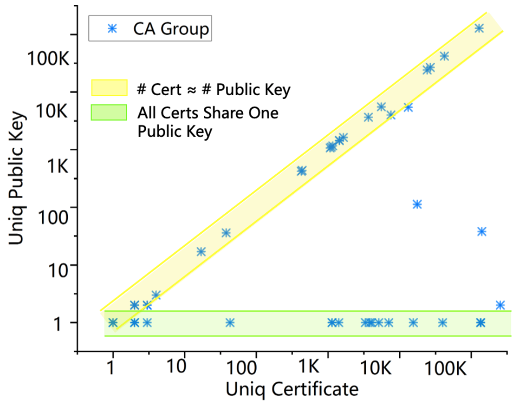


Figure 8: Count of hidden root certificates and public keys in each group.

Finding 5.3: Wildcard is overly used (over 75%) in the issuance of hidden CAs, which is not recommended for security considerations.

Abuse and misuse of wildcard. As shown in Table 1 and discussed in Section 4.4, hidden roots are found to have issued certificates for over 1.54 million FQDNs (both in subjects and SANs) under 792K Second-Level Domains (SLD) in our dataset. Among them, we find a prevalent usage of *wildcards* (e.g., `*.example.com`): over 75% of leaf certificates signed by hidden root CAs use wildcard domains. Although allowed, wildcard is actually discouraged due to the frequent incorrect implementations [24, 50]. As an "inferior case" of certificate practice, a measurement study on HTTPS adoptions of government websites [59] showed their usage of wildcard certificates was about 39.21%, while still lower than that of hidden CA deployments. In addition, wildcard violations are identified in 59 leaf certificates, such as appearing in the non-left-most label (e.g., `violation.*`) or multiple labels (e.g., `*.*.violation`) of a domain.

Long validity period. Figure 9 plots the creation dates and expiration dates of hidden root certificates. We first find dozens of outliers that have unreasonable dates (e.g., 1 root certificate created in the year 1899 and 3 root certificates expiring in year 9999) and remove them from further validity related analysis.

Finding 5.4: Over 79% of hidden root certificates are valid for over 60 years, significantly longer than best practices. Security risks may last long for clients trusting them.

Zooming into root certificates created during year the 1980 and 2020, we find that most of them have a valid period *significantly longer* than best practices: over 935 thousand (79%) are valid for over 60 years, 317 of which are even valid for over 100 years. To compare, all root certificates in public programs are valid for less than 40 years, with a median value of 20 years. Meanwhile, recent practices of operating systems and root CAs [34, 55] suggest that root certificates adopt a validity period between 6 months to 16 years based on the strength of their public keys.

What's worse, because hidden root CAs lack certificate revocation and incident handling, security risks can last for a long time for clients that already import them into local root stores. To check this point, we also manually collected a list of 116 root certificates that have been publicly announced as being revoked by operating systems [3], browsers [63] or technical reports [1]. 23 roots on this list were observed active in the hidden CA ecosystem in our data, trusted by over 34,018 clients and affecting 263,500 HTTPS connections in 5 months. One of them, CFCA, which is a digital authentication CA, even has a long validity period ending at the year 4749.

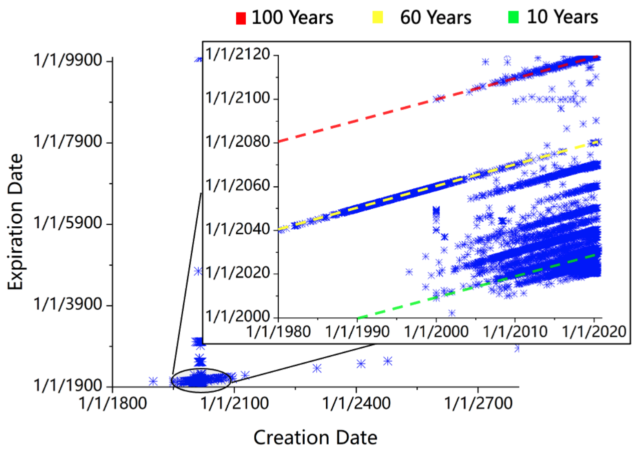


Figure 9: Creation dates and expiration dates of hidden root certificates.

5.2 Non-compliant Certificate Content

In addition to issuance behavior and usage flaws, we further identify content in-compliance of hidden CA ecosystem at scale by Zlint [44]. Zlint is a certificate linter that checks whether a certificate meets implementation requirements specified by the X.509 standard, CAs and browsers. We run it on all 1.17 million hidden root certificates and 59.8 million leaf certificates signed by them. The tool inspects certificates with 266 lints of different security levels, i.e., ERROR (implementation violates a MUST in standards), WARNING (implementation violates a SHOULD in standards) and NOTICE (other low-risk information). We focus on all ERROR messages output by Zlint, as they are violations of basic implementation requirements.

Finding 5.5: The implementation of most hidden root certificates is problematic, with over 87% of them violating at least one basic requirement of X.509 standards.

Problematic content of root certificates. Of 1.17 million hidden root certificates, Zlint reports 1,201,189 ERROR messages of 73 lints. Over 87.3% of all hidden root certificates have been reported at least one ERROR message, as shown in Table 5. Unfortunately, considering average performance, CAs from fake authentications exhibited the best with 0.84 errors per root, while the weakest were self-built CAs with an average error number of 3.25. Specifically, most root certificates miss critical extensions such as key identifiers, which can cause certificate verification errors. Invalid and vulnerable field values (e.g., negative serial numbers and weak public keys) are also prevalent. In addition, 10.9% hidden root certificates only receive WARNING and NOTICE messages. While they do not violate basic requirements, the implementation is also considered flawed and should be fixed [19, 33, 44].

Table 5: Zlint ERROR messages of hidden root certificates

Error Type	# Lints	# Cert Errors	Example
Missing extensions	15	954,453 (79.46%)	Missing key identifier
Invalid values	49	121,745 (10.14%)	Negative serial number
Missing fields	3	89,763 (7.47%)	Missing CA organization
Vulnerable values	6	35,228 (2.93%)	RSA key < 2048 bits

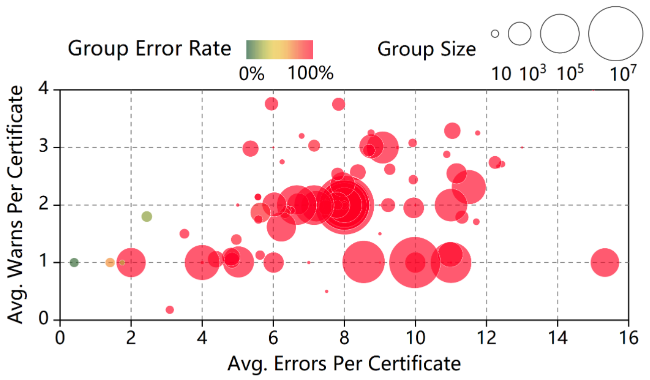


Figure 10: Zlint ERROR message ratio of leaf certificates associated with the top 100 certificate groups.

Finding 5.6: The implementation of leaf certificates signed by hidden CAs is even more worrisome. 85 of the top 100 root certificate groups sign every leaf certificate with implementation errors.

Problematic content of leaf certificates. The implementation of leaf certificates signed by hidden CAs is even worse. On average, Zlint reports 8.14 ERROR and 1.93 WARNING messages for *each* leaf certificate. Compared to the root certificates, the ERROR messages in leaf certs come from 114 lints, with an increased ratio of vulnerable fields (22.8%). After inspection, we confirm that it's caused by a prevalent usage of weak 1024-bit RSA keys (which would be detailed discussed in Sec 6). However, similar to the case of root certificates, we found leaf certs from fake authentications contribute the least to the violations. The top 10 groups with the highest average number of errors in their issued leaf certificates include 2 VPNs, 1 Proxy, 4 Enterprise CAs, 2 Government CAs and 1 group of unknown category. The correlation between the scale of leaf certificates signed by each hidden CA group and their errors is shown in Figure 10. The size of each bubble shows the scale of leaf certificates and the color shows the certificate error rate. 85 of the top 100 groups sign *every leaf certificate* with implementation errors (i.e., the error rate is 100%). Only 3 certificate groups with a smaller scale of leaf certificates have an error rate of less than 50%. The results are *significantly more worrisome* than leaf certificates signed by public trusted CAs, where only 0.02% received ERROR messages in 2018 [44].

6 VALIDATION ERRORS FROM WEB CLIENTS

We have spotted several implementation flaws of hidden CAs, such as extra-long validity periods and violating wildcards in Sec 5. However, as for real-world web visits, the validity rules checked by clients are not exactly the same as the above discussed security guidelines documented in standards and could be analyzed based on scaled certificate data alone. For example, it could be impossible for the client to check whether the root certificate issued the chain shares the public key with other roots, but instead, it could reject the chain based on local and contextual information such as local trust lists (AUTHORITY_INVALID), timestamps (DATE_INVALID) and the host it accesses (COMMON_NAME_INVALID).

This section benefits from the STATUS_CODE of each record in our collected data, which reports the actual errors detected by the web client at the time it received the chain. It could provide a complement to the validity of hidden CA issued chains from the perspective of web clients in the wild. Table 9 of Appendix A lists all the reported error types. Besides, recall that 360 Secure Browser also collects invalid certificate chains signed by public trusted CAs (see Table 1), which provides us the opportunity to perform comparisons with public CAs.

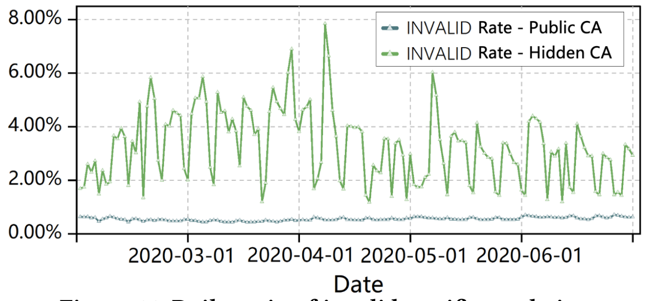


Figure 11: Daily ratio of invalid certificate chains

Finding 6.1: On average, clients would report more errors when encountering chains issued by hidden CAs than public CAs in the wild, with an overall error rate of 3.59%.

Overall error rate. The daily ratio of invalid certificate chains (over the number of all certificate chains) is shown in Figure 11. As expected, certificate chains signed by hidden CAs show a higher invalid ratio: an average of 3.59% are invalid, while the number for public trusted roots is 0.55%. It is also worth noting that, several prevalent implementation flaws analyzed in Sec 5 are currently not in the checking list of clients, probably for better usability. While, it also leaves the users in some blinded state, i.e., hard to be aware of the presence and potential risks of hidden CAs in their daily web visiting traffic. Besides, unfortunately, we found the potentially least reputable member of the hidden CA ecosystem, the chains issued by fake roots had the lowest invalid rate, while self-built CAs signing public websites presented the highest error rate on the contrary (see Table 2). Table 6 also shows the top 5 verification errors occurring in the hidden CA ecosystem and their main contributors. The most serious error reported by clients is WEAK_SIGNATURE_ALGORITHM, i.e., the signature algorithm is weaker than the requirements of the verification API, suggesting that hidden CAs, especially the self-built roots, may have weaker cryptographic practices.

Finding 6.2: Cryptographic flaws are the most serious errors of hidden CAs validated by clients. Weak algorithms like SHA-1 are still used and weak keys like 1024-bit RSA are found in over 97% of their issued leaf certificates.

Table 6: Top 5 verification errors occurring in certificate chains signed by hidden root CAs.

Error Type	% traffic	Main Source
WEAK_SIGNATURE_ALGORITHM	2.4645%	Self-built (81.38%)
COMMON_NAME_INVALID	2.2392%	Self-built (74.81%)
AUTHORITY_INVALID	1.5260%	Self-built (55.90%)
DATE_INVALID	0.4832%	Fake Root (38.33%)
INVALID	0.1715%	Unknown (97.80%)

Table 7: Signature algorithms of hidden certificates (weak algorithms are marked red).

Signature Algorithm	Root Certs	Leaf Certs
SHA-256 with RSA	1,041,118 (88.59%)	59,553,038 (99.56%)
SHA-256 with ECDSA	128,857 (10.97%)	190,241 (0.32%)
SHA-512 with RSA	102 (0.01%)	640 (0.00%)
SHA-1 with RSA	5,005 (0.43%)	69,419 (0.12%)
Other	63 (0.01%)	4247 (0.01%)

Table 8: Public key strength of hidden certificates (weak keys are marked red).

Key Type	Root Certs	Leaf Certs
RSA (2048-bit)	1,038,534 (88.37%)	1,451,858 (2.43%)
ECDSA (256-bit)	128,857 (10.97%)	190,114 (0.32%)
RSA (4096-bit)	781 (0.07%)	661 (0.00%)
RSA (1024-bit)	6,905 (0.59%)	58,165,008 (97.24%)
RSA (256-bit)	-	5,905 (0.01%)
Other	68 (0.01%)	4039 (0.01%)

Cryptographic practice issues. We then take a closer look at the cryptography flaws of hidden CAs. Table 7 and 8 show the distribution of their signature algorithms and public key strengths. From recommendations by NIST [8], algorithms and public keys that are considered weak and dangerous are marked in red. As shown in Table 7, a small number of hidden CAs (including 1,679 roots of Proxy/VPN) still use vulnerable algorithms as SHA-1, which should be banned since 2016 [33]. As for public keys, 0.59% hidden root certificates (20.73% of self-built CAs) use weak keys, while up to *97% leaf certificates signed by them use weak public keys*. Comparing to leaf certificates signed by public trusted CAs, a study in 2013 [26] find that nearly 90% already had a key strength of 2048-bit RSA or above.

7 DISCUSSION

Comparison with HTTPS interception studies. Although previous studies of HTTPS interception seem to be the most relevant to this work, the scope of research is different: traffic from hidden roots is not always covered by interception (e.g., visits to websites issued certificates from government self-built roots), and intercepted traffic is also not covered by hidden roots only (e.g., mis-issuance from public trusted CAs). Thus we consider several results of previous works, such as the HTTPS interception rate [23, 39, 58, 64], are not directly comparable with our results. Besides, since a considerable proportion of our data also comes from interceptions, some of our findings, e.g., the hijacking behavior of fake CAs and the implementation flaws in local software, have also been discussed previously [15, 21, 22, 28, 56]. However, we are the first to report the systematic evaluation of their impact on real-world clients. Besides, the insights we provide into the nature of the hidden root ecosystem are not covered by previous works, e.g., its dynamic update properties.

Distribution channels of hidden root CAs. In this work, we study over 1.19 million hidden root certificates. While they can be imported or pinned to local operating systems, we cannot learn exactly how they got trusted by web clients (i.e., their distribution channels) solely from certificate data analysis. For websites that actively use self-built root CAs, instructions on how to import non-public certificates into local root stores are often shown to web users (e.g., `www.rootca.gov.cn` and `gat.hunan.gov.cn`). Typically, users may follow the instructions in order to dismiss certificate errors and visit the websites. For root certificates of fake CAs, we find several of them hitting threat intelligence or associated with malware according to sandbox logs. They may be maliciously installed by client-side malware or spamming tools in order to inspect secure connections. For others, they can be imported by local software (e.g., VPNs and proxies). While we are still unsure of the exact reason, considering the widespread impact and various implementation flaws of hidden CAs as discussed in this work, web users and security software are recommended to check carefully, or even prompt alert messages when installing software with such behaviors.

Recommendations. This paper reveals the large-scale and widespread impact of hidden root certificates. Meanwhile, the inherent "non-transparency" of this ecosystem makes it hard to be well regulated. It also leaves serious security risks as reported in this work. We acknowledge that it could be impractical to rigorously check and block all roots outside of the public trust list, while at least several parties could cooperate to mitigate the potential security risks. Below lists our recommendations.

• **Operating system: To regulate root store modification.** It is allowed to install hidden roots into the local trust store, while the modification process could be regulated by the operating system in a stricter way through the following steps: 1) Compliance check before import. Unlike public CAs that are effectively reviewed by CTs, hidden roots can be generated and installed locally with non-audit content. Therefore, the OS should take the responsibility of the regulator, i.e., check the compliance of hidden roots (at least their certificate content, as Sec 5 and Sec 6 in this work) before importing them and reject the flawed ones. 2) Monitor and log trust store modifications. Any insertion into the local trust store should be monitored and logged in detail, including which application/process requests the insertion. 3) Provide explicit risk notification to users. Importing hidden roots into the local trust store should be considered as a high-risk operation, thus the OS should prompt informative security alerts (instead of simply notifying "local configuration is being modified") to users.

• **Browser: To enhance UI design and notification.** Currently, except for Mozilla [57] which maintains its own root store, the vast majority of browsers rely on the trust list provided by the platform they are running on. That is, certificate chains issued by hidden roots imported in local trust lists would be indicated as authenticated just as public trusted ones, e.g., with a (green) padlock shown in the address bar, and leaving the users even unaware of the presence of hidden roots. However, as found in this work, the security properties of such links are questionable (e.g., malicious interceptions and vulnerable chains from flawed hidden roots). Therefore, we recommend browsers adjust the alerting mechanism by: 1) Maintaining a public trusted list itself (or transition to its own root store, as Chrome [16] plans to do) to distinguish hidden roots. 2) Differentiating safety icons for such links, e.g., by turning the padlock orange or tagging it.

• **Application: To standardize certificate implementation.** We find non-standard certificate implementations are very common, especially with ambiguous subject information, which raises great difficulties in identifying the ownership and usage of hidden roots. In this work, despite the best efforts, 15 out of the top 100 hidden root groups are still indistinguishable and temporarily marked as unknown. Therefore, we suggest that, at least for local applications that intercept traffic for non-malicious purposes, as well as the internal roots of governments and enterprises, provide more precise subject information and normalize certificate content for better identity verification and security regulations.

8 CONCLUSION

Root CAs serve as trust anchors in the Web PKI security. However, client-side local root CA stores could be manipulated by vendors, local software or even malware. Suffering from hidden root CAs that are not trusted from public root programs into local stores, web clients are exposed to a series of security risks, such as HTTPS downgrade and TLS interception.

In this study, we provide a national client-side view of large-scale hidden root certificates by cooperating with a leading browser vendor in China. We performed a comprehensive measurement study, and identified 1.17M hidden root certificates, affecting 5.07 million web clients and 222 million (0.54%) HTTPS connections. Following, we also classified their usages and analyzed their distribution sources, including self-built CAs, local software and fake authentications CAs. Numerous implementation and operation flaws have been demonstrated, for example, public key sharing, abuse of wildcards and long validity period are prevalent.

In general, our study suggests that the community should immediately review the security of the local root store, and seek a security best practice to notify client users in the event of root store modification.

ACKNOWLEDGEMENT

We thank our shepherd Frank Li and all the anonymous reviewers for their valuable comments to improve this paper. This work was supported in part by the National Natural Science Foundation of China (U1836213, U1636204) and the BNRist Network and Software Security Research Program (Grant No. BNR2019TD01004). Baojun Liu is partially supported by the NSFC 62102218 and Shuimu Tsinghua Scholar Program. Zhou Li is partially supported by gift from Cisco and Microsoft.

REFERENCES

[1] 2014. Revoke China Certs. https://github.com/masquey/RevokeChinaCerts.
[2] 2019. List of Participants - Microsoft Trusted Root Program. https://docs.microsoft.com/en-us/security/trusted-root/participants-list.
[3] 2021. Release notes - Microsoft Trusted Root Certificate Program. https://docs.microsoft.com/en-us/security/trusted-root/release-notes.

[4] Mustafa Emre Acer, Emily Stark, Adrienne Porter Felt, Sascha Fahl, Radhika Bhargava, Bhanu Dev, Matt Braithwaite, Ryan Sleevi, and Parisa Tabriz. 2017. Where the wild warnings are: Root causes of Chrome HTTPS certificate errors. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1407–1420.

[5] Devdatta Akhawe, Bernhard Amann, Matthias Vallentin, and Robin Sommer. 2013. Here's my cert, so trust me, maybe? Understanding TLS errors on the web. In *Proceedings of the 22nd international conference on World Wide Web*. 59–70.

[6] Bernhard Amann, Robin Sommer, Matthias Vallentin, and Seth Hall. 2013. No attack necessary: The surprising dynamics of SSL trust relationships. In *Proceedings of the 29th annual computer security applications conference*. 179–188.

[7] Bernhard Amann, Matthias Vallentin, Seth Hall, and Robin Sommer. 2012. *Extracting certificates from live traffic: A near real-time SSL notary service*. Technical Report. Citeseer.

[8] Elaine B Barker and Quynh H Dang. 2015. Recommendation for Key Management Part 3: Application-Specific Key Management Guidance. (2015).

[9] Doug Beattie. 2018. What Are Subordinate CAs and Why Would You Want Your Own? https://www.globalsign.com/en/blog/what-is-an-intermediate-or-subordinate-certificate-authority.

[10] Frank Cangialosi, Taejoong Chung, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, and Christo Wilson. 2016. Measurement and analysis of private key sharing in the https ecosystem. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 628–640.

[11] Certum. [n. d.]. Root certificates. https://www.certum.eu/en/cert_expertise_root_certificates/.

[12] Chromium. [n. d.]. Certificate Transparency. https://chromium.googlesource.com/chromium/src/+/master/net/docs/certificate-transparency.md#Certificate-Transparency-For-Enterprises.

[13] Chromium. [n. d.]. The Chromium Projects. https://www.chromium.org/.

[14] Chromium. 2020. The Chromium root program. https://www.chromium.org/Home/chromium-security/root-ca-policy.

[15] Taejoong Chung, David Choffnes, and Alan Mislove. 2016. Tunneling for transparency: A large-scale analysis of end-to-end violations in the internet. In *Proceedings of the 2016 Internet Measurement Conference*. 199–213.

[16] Catalin Cimpanu. 2020. Chrome will soon have its own dedicated certificate root store. https://www.zdnet.com/article/chrome-will-soon-have-its-own-dedicated-certificate-root-store/.

[17] Cisco. 2018. Weekly Threat Intelligence Report - 2018.04.13. https://www.cisco.com/c/dam/global/zh_cn/products/security/talos/Threat_Roundup-for-April.pdf.

[18] Alibaba Cloud. [n. d.]. Alibaba Cloud. https://www.alibabacloud.com/.

[19] David Cooper, Stefan Santesson, Stephen Farrell, Sharon Boeyen, Russell Housley, and W. Timothy Polk. 2008. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. *RFC* 5280 (2008), 1–151. https://doi.org/10.17487/RFC5280

[20] Mikhail Davidov and Darren Kemp. 2015. DUDE, YOU GOT DELL'D: PUBLISHING YOUR PRIVATES. https://duo.com/decipher/dude-you-got-dell-d-publishing-your-privates.

[21] X de Carné de Carnavalet and Mohammad Mannan. 2016. Killed by proxy: Analyzing client-end TLS interception software. In *Network and Distributed System Security Symposium*.

[22] Xavier de Carné de Carnavalet and Paul C van Oorschot. 2020. A survey and analysis of TLS interception mechanisms and motivations. *arXiv preprint arXiv:2010.16388* (2020).

[23] Xavier de Carné de Carnavalet. 2019. *Last-Mile TLS Interception: Analysis and Observation of the Non-Public HTTPS Ecosystem*. Ph. D. Dissertation. Concordia University.

[24] Zheng Dong, Kevin Kane, Siyu Chen, and L Jean Camp. 2016. The New Wildcats: High-Risk Banking From Worst-Case Certificate Practices Online. https://www.researchgate.net/profile/L-Camp/publication/317722542_The_New_Wildcats_High-Risk_Banking_From_Worst-Case_Certificate_Practices_Online/links/59ee56060f7e9b3695759f90/The-New-Wildcats-High-Risk-Banking-From-Worst-Case-Certificate-Practices-Online.pdf.

[25] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J Alex Halderman. 2015. A search engine backed by Internet-wide scanning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 542–553.

[26] Zakir Durumeric, James Kasten, Michael Bailey, and J Alex Halderman. 2013. Analysis of the HTTPS certificate ecosystem. In *Proceedings of the 2013 conference on Internet measurement conference*. 291–304.

[27] Zakir Durumeric, Frank Li, James Kasten, Johanna Amann, Jethro Beekman, Mathias Payer, Nicolas Weaver, David Adrian, Vern Paxson, Michael Bailey, et al. 2014. The matter of heartbleed. In *Proceedings of the 2014 conference on internet measurement conference*. 475–488.

[28] Zakir Durumeric, Zane Ma, Drew Springall, Richard Barnes, Nick Sullivan, Elie Bursztein, Michael Bailey, J Alex Halderman, and Vern Paxson. 2017. The Security Impact of HTTPS Interception.. In *NDSS*.

[29] Let's Encrypt. 2021. Chain of Trust - Let's Encrypt. https://letsencrypt.org/images/isrg-hierarchy.png.

[30] Chris Evans, Chris Palmer, and Ryan Sleevi. 2015. Public Key Pinning Extension for HTTP. *RFC* 7469 (2015), 1–28. https://doi.org/10.17487/RFC7469

[31] Adrienne Porter Felt, Richard Barnes, April King, Chris Palmer, Chris Bentzel, and Parisa Tabriz. 2017. Measuring HTTPS adoption on the web. In *26th USENIX Security Symposium (USENIX Security 17)*. 1323–1338.

[32] Dennis Fisher. [n. d.]. Google, Mozilla Drop Trust in Chinese Certificate Authority CNNIC. https://threatpost.com/google-drops-trust-in-chinese-certificate-authority-cnnic/111974/.

[33] CA/Browser Forum. [n. d.]. Baseline Requirements Documents (SSL/TLS Server Certificates). https://www.mozilla.org/en-US/about/governance/policies/security-group/certs/policy/.

[34] Aaron Gable. 2020. Let's Encrypt's New Root and Intermediate Certificates. https://letsencrypt.org/2020/09/17/new-root-and-intermediates.html.

[35] Google. [n. d.]. Certificate Transparency. https://certificate.transparency.dev.

[36] Pinjia He, Jieming Zhu, Pengcheng Xu, Zibin Zheng, and Michael R Lyu. 2018. A directed acyclic graph approach to online log parsing. *arXiv preprint arXiv:1806.04356* (2018).

[37] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R Lyu. 2017. Drain: An online log parsing approach with fixed depth tree. In *2017 IEEE International Conference on Web Services (ICWS)*. IEEE, 33–40.

[38] Ralph Holz, Lothar Braun, Nils Kammenhuber, and Georg Carle. 2011. The SSL landscape: a thorough analysis of the x. 509 PKI using active and passive measurements. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. 427–444.

[39] Lin Shung Huang, Alex Rice, Erling Ellingsen, and Collin Jackson. 2014. Analyzing forged SSL certificates in the wild. In *2014 IEEE Symposium on Security and Privacy*. IEEE, 83–97.

[40] Apple Inc. 2018. Lists of available trusted root certificates in macOS. https://support.apple.com/en-us/HT202858.

[41] Alexa Internet Inc. 2020. Alexa Top Sites. https://www.alexa.com/topsites.

[42] Venustech Group Inc. 2020. VenusTech VPN. https://www.venustech.com.cn/.

[43] Nikita Korzhitskii and Niklas Carlsson. 2020. Characterizing the Root Landscape of Certificate Transparency Logs. In *2020 IFIP Networking Conference (Networking)*. IEEE, 190–198.

[44] Deepak Kumar, Zhengping Wang, Matthew Hyder, Joseph Dickinson, Gabrielle Beck, David Adrian, Joshua Mason, Zakir Durumeric, J Alex Halderman, and Michael Bailey. 2018. Tracking certificate misissuance in the wild. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 785–798.

[45] Qihoo 360 Technology Co. Ltd. 2019. Financial Reports of 360 for 2019. https://news.qudong.com/article/590463.shtml.

[46] Qihoo 360 Technology Co. Ltd. 2020. 360 Secure Browser. https://browser.360.cn/.

[47] Qihoo 360 Technology Co. Ltd. 2020. Root CA Program of 360 Browser. https://caprogram.360.cn/#plan.

[48] Zane Ma, Joshua Mason, Manos Antonakakis, Zakir Durumeric, and Michael Bailey. 2021. What's in a Name? Exploring CA Certificate Control. In *30th USENIX Security Symposium (USENIX Security 21)*.

[49] Zane Ma, Joshua Mason, Manos Antonakakis, Zakir Durumeric, Michael Bailey, Sascha Fahl, Jörg Schwenk, Sebastian Schinzel, Adam Doupé, Gail-Joon Ahn, et al. 2020. CA Transparency. https://github.com/zzma/ca-transparency.

[50] Christopher Meyer and Jörg Schwenk. 2013. SoK: Lessons learned from SSL/TLS attacks. In *International Workshop on Information Security Applications*. Springer, 189–209.

[51] Microsoft. [n. d.]. CryptoAPI System Architecture. https://docs.microsoft.com/en-us/windows/win32/seccrypto/cryptoapi-system-architecture.

[52] Mozilla. [n. d.]. Common CA Database by mozilla. https://www.ccadb.org.

[53] Mozilla. 2021. Mozilla Root Store Policy. https://www.mozilla.org/en-US/about/governance/policies/security-group/certs/policy/.

[54] Edward Oakes, Jeffery Kline, Aaron Cahn, Keith Funkhouser, and Paul Barford. 2019. A Residential Client-side Perspective on SSL Certificates. In *2019 Network Traffic Measurement and Analysis Conference (TMA)*. IEEE, 185–192.

[55] Yvette O'Meally. 2018. Recommendations for PKI Key Lengths and Validity Periods with Configuration Manager. https://techcommunity.microsoft.com/t5/configuration-manager-archive/recommendations-for-pki-key-lengths-and-validity-periods-with/ba-p/272758.

[56] Mark O'Neill, Scott Ruoti, Kent Seamons, and Daniel Zappala. 2016. TLS proxies: Friend or foe?. In *Proceedings of the 2016 Internet Measurement Conference*. 551–557.

[57] Mozilla Project. 2021. Mozilla Included CA Certificate List. https://wiki.mozilla.org/CA/Included_Certificates.

[58] Ram Sundara Raman, Leonid Evdokimov, Eric Wurstrow, J Alex Halderman, and Roya Ensafi. 2020. Investigating Large Scale HTTPS Interception in Kazakhstan. In *Proceedings of the ACM Internet Measurement Conference*. 125–132.

[59] Sudheesh Singanamalla, Esther Han Beol Jang, Richard Anderson, Tadayoshi Kohno, and Kurtis Heimerl. 2020. Accept the Risk and Continue: Measuring the Long Tail of Government https Adoption. In *Proceedings of the ACM Internet Measurement Conference*. 577–597.

[60] Trevor Smith, Luke Dickinson, and Kent Seamons. 2020. Let's revoke: Scalable global certificate revocation. In *27th Annual Network and Distributed System*

Security Symposium, NDSS.

[61] Emily Stark, Ryan Sleevi, Rijad Muminovic, Devon O'Brien, Eran Messeri, Adrienne Porter Felt, Brendan McMillion, and Parisa Tabriz. 2019. Does certificate transparency break the web? Measuring adoption and error rate. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 211–226.

[62] GlobalSign Support. [n. d.]. GlobalSign Root Certificates. https://support.globalsign.com/ca-certificates/root-certificates/globalsign-root-certificates.

[63] The Bugzilla Team. 2020. Bug List of CA Certificate Root Program. https://bugzilla.mozilla.org/buglist.cgi?component=CA%20Certificate%20Root%20Program&product=NSS&bug_status=__open__.

[64] Narseo Vallina-Rodriguez, Johanna Amann, Christian Kreibich, Nicholas Weaver, and Vern Paxson. 2014. A tangled mass: The android root certificate stores. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*. 141–148.

[65] Benjamin VanderSloot, Johanna Amann, Matthew Bernhard, Zakir Durumeric, Michael Bailey, and J Alex Halderman. 2016. Towards a complete view of the certificate ecosystem. In *Proceedings of the 2016 Internet Measurement Conference*. 543–549.

[66] VeriSign. [n. d.]. VeriSign Root Certificates. https://www.websecurity.digicert.com/content/dam/websitesecurity/digitalassets/desktop/pdfs/repository/root-certificates.pdf.

[67] Louis Waked, Mohammad Mannan, and Amr Youssef. 2018. To intercept or not to intercept: Analyzing tls interception in network appliances. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. 399–412.

[68] Sebastian Wiesinger. 2012. Remove Trustwave Certificate(s) from trusted root certificates. https://bugzilla.mozilla.org/show_bug.cgi?id=724929.

[69] Kathleen Wilson. 2009. IPS Action items re IPS SERVIDORES root certificate. https://bugzilla.mozilla.org/show_bug.cgi?id=523652.

APPENDIX

A CERTIFICATE VERIFICATION CODES

To verify a certificate chain, 360 Secure Browser calls APIs of underlying operating systems (e.g., `cryptoAPI`) which return all verification errors that occur in a double-word value. The errors are then recorded as `STATUS_CODE` in the collected certificate records. Table 9 shows all certificate verification errors in `STATUS_CODE` that are captured in our dataset.

Table 9: Certificate verification errors reported by `STATUS_CODE` (captured in our dataset).

#	Verification Error	Description
1	`COMMON_NAME_INVALID`	`commonName` of leaf certificate does not match the given hostname.
2	`DATE_INVALID`	Certificate is out of its validity period.
3	`AUTHORITY_INVALID`	Root certificate is not trusted by local operating system.
4	`ERR_CERT_CONTAINS_ERRORS`	Certificate is malformed.
5	`UNABLE_TO_CHECK_REVOCATION`	Cannot check the revocation status of certificate.
6	`REVOKED`	Certificate is revoked.
7	`INVALID`	Invalidation due to other reasons.
8	`WEAK_SIGNATURE_ALGORITHM`	Certificate uses a signature algorithm weaker than the API's requirements.
9	`WEAK_KEY`	Certificate uses a key weaker than the API's requirements.
10	`PINNED_KEY_MISSING`	Pinned public key is not in the certificate chain.
11	`NAME_CONSTRAINT_VIOLATION`	CA signs certificate to subjects out of its name constraints.
12	`VALIDITY_TOO_LONG`	Certificate's validity period is longer than the API's requirements.
13	`CT_COMPLIANCE_FAILED`	Certificate has compliance failures with CT policies.
14	`CERTIFICATE_TRANSPARENCY_REQUIRED`	Certificate is not properly logged in CT.
15	`SYMANTEC_LEGACY`	Root certificate belongs to distrusted Legacy Symantec PKI.