The Broken Shield: Measuring Revocation Effectiveness in the Windows Code-Signing PKI

Doowon Kim and Bum Jun Kwon, ***University of Maryland, College Park;***
Kristián Kozák, ***Masaryk University, Czech Republic;*** **Christopher Gates,** ***Symantec;***
Tudor Dumitras, ***University of Maryland, College Park***

https://www.usenix.org/conference/usenixsecurity18/presentation/kim

This paper is included in the Proceedings of the 27th USENIX Security Symposium.

August 15–17, 2018 • Baltimore, MD, USA

ISBN 978-1-931971-46-1

Open access to the Proceedings of the 27th USENIX Security Symposium is sponsored by USENIX.

The Broken Shield: Measuring Revocation Effectiveness in the Windows Code-Signing PKI

Doowon Kim
University of Maryland

Bum Jun Kwon
University of Maryland

Kristián Kozák
Masaryk University

Christopher Gates
Symantec Research Labs

Tudor Dumitraș
University of Maryland

Abstract

Recent measurement studies have highlighted security threats against the code-signing public key infrastructure (PKI), such as certificates that had been compromised or issued directly to the malware authors. The primary mechanism for mitigating these threats is to revoke the abusive certificates. However, the distributed yet closed nature of the code signing PKI makes it difficult to evaluate the effectiveness of revocations in this ecosystem. In consequence, the magnitude of signed malware threat is not fully understood.

In this paper, we collect seven datasets, including the largest corpus of code-signing certificates, and we combine them to analyze the revocation process from end to end. Effective revocations rely on three roles: (1) discovering the abusive certificates, (2) revoking the certificates effectively, and (3) disseminating the revocation information for clients. We assess the challenge for discovering compromised certificates and the subsequent revocation delays. We show that erroneously setting revocation dates causes signed malware to remain valid even after the certificate has been revoked. We also report failures in disseminating the revocations, leading clients to continue trusting the revoked certificates.

1 Introduction

The code-signing Public Key Infrastructure (PKI) is a fundamental building block for establishing trust in computer software [22]. This PKI allows software publishers to sign their executables and to embed certificates that bind the signing keys to the publishers' real-world identities. In turn, client platforms can verify the signatures and check the publishers, to confirm the integrity of third-party programs and to avoid executing malicious code. A common security policy is to trust executables that carry valid signatures from unsuspicious publishers.

The premise for trusting these executables is that the signing keys are not controlled by malicious actors. Unfortunately, anecdotal evidence and recent measurements of the Windows code-signing ecosystem have documented cases of signed malware [8, 9, 12, 23, 26] and potentially unwanted programs (PUPs) [1, 13, 17, 28], where the trusted certificates were either compromised or issued directly to the malware authors. The primary defense against these threats is to revoke the certificates involved in the abuse. For the better studied Web's PKI, prior measurements have uncovered important problems with this approach, including long revocation delays [6, 29, 30], large bandwidth costs for disseminating the revocation information [19], and clients that do not check whether certificates are revoked [19]. In contrast, little is currently known about the effectiveness of revocations in the code signing PKI. Without this understanding, platform security protections risk making incorrect assumptions about how critical revocations are for end-host security and about the practical challenges for implementing effective revocations in the code-signing ecosystem.

Code signing uses a default-valid trust model, where certificate chains remain trusted until proven compromised. Due to this fact, missing or delayed revocations for a certificate involved in abuse allow bad actors to generate trusted executables until the certificate expires or is successfully added to a revocation list.

Abusive code-signing certificates may also present a security threat beyond their expiration dates, which is an important distinction from the Web's PKI where the expiration date limits the use of a compromised certificate and also puts a limit on how long a revocation for that certificate must be maintained. To avoid re-signing and

Role	Finding	Implication
Discovery of Potentially Compromised Certificates	The mark-recapture estimation for the number of compromised certificates suggests that even a large AV vendor can only see about 36.5% of the population.	There might be malware with compromised certificates that remain a threat for a long time without being detected.
	CAs took on average 171.4 days to revoke the compromised certificates after the malware signed with the certificates appeared in the wild.	Compromised certificates are not discovered and revoked for a long time.
Setting Revocation Date	CAs erroneously set effective revocation dates for 62 certificates, causing 402 signed malware to remain valid.	Wrong effective revocation date setting results in the survival of signed malware although its certificates is revoked.
Dissemination of Revocation Information	788 certificates contain neither CRLs nor OCSP points.	Clients have no way to check the revocation status of the certificates.
	13 CRLs and 15 OCSP servers had reachability issues. OCSP servers responded with *unknown* or *unauthorized* messages. 19 certificates have inconsistent responses from CRLs and OCSP; they are valid from OCSP but are revoked in CRLs.	CAs improperly maintain their CRLs and OCSP servers.
	278 revoked certificates were added and then later removed from 18 CRLs.	Errors in the revocation process are made, and later retracted. CAs misunderstood the code signing PKI and removed expired certificates from CRLs.

Table 1: Summary of findings.

distributing binaries when a signing certificate expires, Windows developers may extend the validity of binaries they release by including a trusted timestamp, provided by a Time-Stamping Authority (TSA), that certifies the signing time of a binary. If a malicious binary is correctly signed and timestamped before the expiration date of the certificate, it will remain trusted even after its certificate expires—unless the certificate is revoked. This means that prompt and effective revocations, even of expired certificates, are critical in the code signing PKI.

An effective revocation process faces additional challenges in the code signing ecosystem. This process involves three roles: (1) discovering certificates that are compromised or controlled by malicious actors; (2) revoking these certificates effectively; and (3) disseminating the revocation information so that it is broadly available.

Unlike in the Web's PKI, where potentially compromised certificates can be discovered systematically through network scanning [6, 29, 30], in the code signing PKI this requires discovering signed malware or PUP samples on end-hosts around the world. Security companies involved in this discovery process cannot observe all the hosts where a maliciously signed binary may appear. This also makes it a challenge to detect the total number of certificates that are actively being used to sign malware, which leads to an incorrect perception about the need and urgency of revocations. Even though a signed malicious binary is discovered, it is difficult to determine the date when a certificate revocation should become effective. Hard revocations that invalidate the entire life of the certificate may invalidate too many benign signed files, while soft revocations that set a revocation date after the issuance date may not cover undiscovered signed malware. Moreover, the CAs also must properly maintain their revocation infrastructure so that the information of compromise can be disseminated to the clients. If the dissemination is not handled as it should be, it may reduce the incentives for revoking code signing certificates. These challenges render the code signing ecosystem opaque and difficult to audit, which contributes to an under-appreciation of the security threats that result from ineffective revocations.

In this paper, we present an end-to-end measurement of certificate revocations in the code signing PKI; in particular, how effective is the current revocation process from discovery to dissemination, and what threats are introduced if the process is not properly done. Our work extends prior works in the code signing PKI; previous studies have focused on signed PUPs [1, 13, 28] and signed malware [12], but there is no study of code signing certificate revocation process yet. Unlike the prior studies in the Web's PKI [2, 6, 7, 10, 19] where TLS certificate can be collected by scanning the Internet, we are unable to utilize a comprehensive corpus of code signing certificates since there is no official repository for code signing certificates. To overcome the challenge, we utilize data sets that are publicly released from prior research [1, 13] and increase our coverage with Symantec's internal repository of binary samples. We extract 145,582 unique leaf code signing certificates from the data sets. From the code signing certificates, we also extract 215 Certificate Revocation Lists (CRLs) used only for code signing certificates, and 131 Online Certificate

Status Protocol (OCSP) points. We periodically probe the collected CRLs to check their status to collect the revocation publication date; the date on which a certificate is revoked by a CA and the revocation information is disseminated.

We highlight the nine findings from our analysis in the revocation process in the three roles and the resulting security implication as depicted in Table 1. To allow the security research community to reproduce and extend our study, we make three data sets publicly available at http://signedmalware.org; (1) Revocation information (D2), (2) Revocation Publication Date List (D3), and (3) CRL/OCSP reachability history (D7) [1].

In summary, we make the following contributions: (1) we collect a large corpus of code signing certificates and the revocation information, (2) we conduct the first end-to-end measurement of the code signing certificate revocation process, (3) we use our data to estimate a lower bound on the number of compromised certificates, (4) we highlight the problems in the three parts of the revocation process as well as new threats that result from those problems, and (5) we discuss suggestions/recommendations to improve the security of the code signing ecosystem.

2 Problem Statement

In this section, we provide a brief overview of the code signing PKI, with an emphasis on certificate revocation. We also discuss the implications of code signing as it currently exists, and highlight the research questions for investigating the effectiveness of the revocation process.

2.1 Code Signing PKI

The code signing PKI provides a mechanism to validate the authenticity of a software publisher and the integrity of a binary executable.

Code signing process. Similar to the Web's PKI (e.g., TLS), the software publishers first ask a Certificate Authority (CA) to issue code signing certificates based on the X.509 v3 certificate standard [4], and they use the certificates to sign their binary files. In the process of signing a binary file, the hash value is first computed, and then the hash value is digitally signed with the software publisher's private key. Finally, the original code is bundled with the signature as well as the public part of the code signing certificate. The end users check the validity of the certificates used to sign the program code when they are first seen, and periodically after that to make sure the certificate is still valid.

Microsoft Authenticode. In the Windows platforms, Authenticode [21] is the code signing standard designed to digitally sign Windows files including executables (.exe), dynamically loaded libraries (.dll), cabinet files (.cab), ActiveX controls (.ctl, and .ocx), catalogs (.cat) files, etc. The standard relies on Public Key Cryptography Standard (PKCS) #7 [11] that stores X.509 code signing certificate chains, X.509 TSA certificate chains, a digital signature, and a hash value of a PE file, with no encrypted data.

Trusted timestamping. Unlike the Web's PKI, the code signing PKI provides *trusted timestamping*. Trusted timestamping is a way to attest that the code was signed at a specific date and time. The timestamp is issued and signed by Time Stamping Authority (TSA) during the signing process. The trusted timestamp guarantees that the signature is generated within the validity period of a certificate to extend the trust in the signed program code even after the certificate expires. Unfortunately, malware writers also benefit from this mechanism. Properly signed and trusted timestamped malware can be trusted and remain valid even after its certificate expiration date.

Trends of code signing abuse. Digitally signed malware can help to bypass some of the protection mechanisms for end-users such as Windows' User Account Control (UAC) and some Anti-Virus (AV) engines. Therefore, malware authors have abused the code signing PKI and signed their malware code with the certificates either stolen or fraudulently issued to malware authors: for example, Stuxnet, Flame, and Duqu [8, 9, 23]. Kim et al. [12] presented threat models that emphasize three types of weaknesses in the code signing PKI: (1) inadequate client-side protections, (2) publisher-side key mismanagement, and (3) CA-side verification failures. Those weakness can breach the trust in the Windows' code signing PKI.

Moreover, malware authors also use the underground black markets to purchase code signing certificates. According to prior work [14], the certificates are being sold at $350–$1,000 for a code signing certificate and at $1,600–$3,000 for an EV code signing certificate. Also, they reported that about 60% of the compromised certificates in their data sets used to sign malware within the first month after its issue date. They claimed this finding as a new evidence of the growing prevalence of certificates issued for abuse.

[1] Due to the agreement terms, we are unable to publicize the data sets collected in the Symantec internal repository.

2.2 Revocation Process

Certificate revocation is the primary defense against the abuse of code signing. CAs are responsible for revoking certificates for reasons such as: the private key associated with a certificate is made public, the entity behind the certificate becomes untrusted, the certificate is used to sign malware even if the source is unknown, or if a certificate is erroneously issued [12]. The revocation process consists of three roles: (1) promptly discovering compromised certificates, (2) performing an effective revocation of the certificate, and (3) disseminating the revocation information.

Discovery of potentially compromised certificates. It is not clearly stated in the requirements [3] who is responsible for discovering compromised certificates. However, the notification of abuse often comes externally, from Anti-virus (AV) companies, researchers or the companies that own the certificates. Once notified, the CAs, who have issued the certificates, are required to promptly investigate and revoke the abused certificates. The delay between the initial discovery (t_d) and the time when the revocation information is made public (i.e., *revocation publication date* (t_p)) should be as short as possible. Figure 1 depicts the case where the discovery happened after the expiration (t_e). Due to trusted timestamping, the revocation should be performed even after the expiration date of the certificate. The revocation delay can be defined as $t_p - t_d$.

Setting the revocation date. Once the CAs confirm the abuse, in collaboration with the certificate owners, they have to decide the *effective revocation date* (t_r) due to the trusted timestamping. The effective revocation date determines which binaries will be impacted. Suppose we have a code signing certificate valid between t_i (issue date) and t_e (expiration date). We sign a binary with the certificate during its validity period. If a certificate is found to be compromised in some way at t_d (detection date), it must be revoked. At this point the CA also must set t_r (effective revocation date) for the certificate. As shown in Figure 1, any binary signed by the certificate after t_r, regardless of the trusted timestamp, will become invalid. However, a binary signed with a trusted timestamp before t_r remains valid.

Dissemination of revocation information. CAs must then disseminate the revoked certificate information. Unlike the discovery and setting the revocation date, CAs are solely responsible for this part of the revocation process. The two predominant ways to disseminate certificate revocation information are (1) Certificate Revocation List (CRL) [4] and (2) Online Certificate Status Protocol (OCSP) [24].

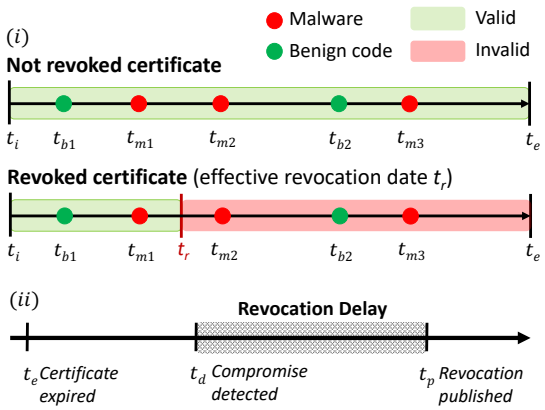

Figure 1: An example of (i) an *effective revocation date (t_r)* that determines the validity of signed malware and (ii) a *revocation delay (t_p - t_d)* (t_i: issue date, t_e: expiration date, t_r: effective revocation date, t_b: signing date of a benign program, t_m: signing date of malware, t_d: detection date, and t_p: revocation publication date). When an effective revocation date is set at t_r, the malware signed at t_{m1} validates continuously as it was signed before t_r.

- *CRLs* contain the revocation information (certificate serial numbers, (effective) revocation date, revocation reason) of certificates that have been revoked. Each CRL is updated based on their CA's issuance policy; for example, they can be issued when a new revoked certificate is inserted, or a specific time of day or a day of month. The location of the CRL is specified at CRL Distribution Point (CDP) of the X.509 certificate. Clients have to periodically download the entire CRL (not just recent changes) to check the latest revocations.
- *OCSP* was introduced to resolve the network overhead problems of CRL. Clients can simply query an OCSP server for a certain certificate, which helps mitigate the network overhead at the server as well as clients. Authority Information Access (AIA), an extension field in a X.509 certificate specifies OCSP point for each certificate.

The TLS CAs are typically not responsible for providing the revocation status of expired certificates. The code signing CAs, however, must maintain and provide the revocation information of all certificates that they have issued including expired certificates due to the trusted timestamp [3, 20]. Since the trusted timestamp extends the life of a signed binary, CAs must maintain the CRLs and OCSP in perpetuity to make revocation information always-available for clients.

2.3 Effectiveness of Revocation Process

In this sub-section we discuss the revocation process. We break this part into four sub-questions:

Q1. How many certificates are being used to sign malware? The revocation process starts from the discovery of compromised certificates. We begin our study by estimating the magnitude of the current threat that should be the target of the revocation process.

Q2. How prompt is the revocation process? When alerted to a certificate problem, CAs have to begin investigating the reports within 24 hours and revoke the compromised certificates and publish the revocation information within seven days or get reasonable cause from the owner of the certificate to delay [3]. The date when the revocation information is available to the public (i.e., added to the CRL or OCSP), is defined as *revocation publication date* (t_p). There are currently some reporting mechanisms in place to allow an outside party, such as an AV company or researcher, to report misuse of certificates to CAs [3]. Due to this adhoc process, there may be delays from initial evidence of compromise (t_d) to the revocation published date (t_p).

Q3. Are effective revocation dates set properly? When revoking a certificate, the CA must set the date when the revocation should be considered active (*effective revocation date* (t_r)). Because of the trusted timestamp, any binary signed with the certificate before the effective revocation date (t_r) is still considered trusted, while any file signed and timestamped after the effective revocation date (t_r) is considered untrusted. Two strategies are used, *hard revocation* where $t_r = t_i$, and *soft revocation* where $t_i < t_r \leq t_e$. Hard revocation has the advantage that all malicious signed files are untrusted, but the side effect is that all benign files also become untrusted. Soft revocation tries to match the date more closely to the date when the certificate was compromised, which means some benign files will still be trusted. If this date is not set correctly, then signed malware (i.e., malware is signed before the date, $t_m < t_r$) may still exist and continue to be trusted as the example shown in Figure 1.

Q4. Is revocation information served properly? Client-side platforms (e.g., Windows) check the validity of both leaf and intermediate certificates used to sign program code. According to the specification [3], a binary should be considered unsigned when it is not possible to check the revocation status. Suppose that a client platform does not follow the specification, but instead applies a *soft-fail* revocation checking policy; the *soft-fail* revocation checking policy is for client platforms to trust certificates when revocation information is unavailable. In this setting, all signed malicious files can remain valid even after the certificate is already revoked if the revocation status information is unavailable. Therefore, it is important to check if the revocation information is properly maintained and disseminated by CAs.

2.4 Our Goal and Non-Goal

In the Web's PKI (e.g., TLS), the security issues of certificate revocation have been well-understood [6, 19, 30]. In contrast, little is known about code signing certificate revocation: in particular, the revocation process (1) promptly discovering compromised certificates, (2) revoking the compromised certificates effectively, and (3) disseminating the revocation information. In this paper, our goal is to systemically measure the problems in the revocation process and new threats introduced by these problems. Our non-goals include fully characterizing (1) CA's internal infrastructure problems, (2) their internal revocation policies, and (3) Windows platforms internal revocation checking policies.

2.5 Challenges for Measuring Revocation

In our study, the challenges for measuring code signing certificate revocation are (1) visibility and (2) timing. *Visibility* is an issue because, unlike on the open Internet, there is no easy way to identify all the certificates that are actively being used in the wild. Instead, we have to find data from sources that provide as wide a view of the ecosystem as possible. *Timing* is a problem because if we observe only a single version of the CRL, we can only see the *effective revocation date* (t_r), which helps define which files should be untrusted, but not when the trust was lost. To see the *revocation publication date* (t_p), when a certificate appears on a revocation list, we must actively monitor the CRLs over an extended period of time.

3 Data Collection

There are no publicly available datasets that are used to perform research on code signing certificates. In this section we describe our data collection methodology and how we measure the revocation process for code signing certs.

	Malsign	*Malcert*	*Symantec*	*WINE*	**Total***
PKCS #7	2,171	801,995	149,840	11,108	965,114
CS certs.**	2,106	1,121	145,411	1,137	145,582
CRL URLs	55	60	403	49	413
OCSP URLs	24	24	130	16	131

Table 2: Summary of the fundamental data. (*: total number of unique data, **: CS stands for code signing – some certificates have parsing errors.)

3.1 Fundamental Data (D1 – D2)

The code signing certificates are the seed to collect additional information since they include the revocation distribution points (CRLs and OCSP points) and other information that we monitor. Here we describe how we collect the code signing certificates and the revocation information. Table 2 shows the breakdown of the fundamental data.

Code signing certificates (D1). There is a publicly available corpus of TLS certificates at *Censys.io* [2], collected by scanning all IPv4 network address. In contrast, there is no large public corpus of code signing certificates observed in the wild. We use multiple data sets that are publicly released from prior research [1, 5, 13] and a proprietary repository of binary samples. The data sets are:

- *Malsign.* Kotzias et al. [13] evaluated signed malicious PE files and they publicly released the 2,171 leaf code signing certificates used to sign the PE files.
- *Malcert.* Alrawi et al. [1] examined 3.3 million samples collected from a commercial feed of a private company, and they shared 801,995 signed PE samples. The reason for the large reduction from PKCS #7 to CS certs for Malcert in Table 2 is that most of the PKCS #7 files were duplicate code signing certificates used to sign binaries with different hashes.
- *Symantec data set.* Symantec has an internal repository of binary files, from which they extracted a sample of 149,840 PKCS #7 files for analysis.
- *Samples from WINE [5] and VirusTotal.* To get more code signing certificates, we also select around 300 PE files for each CA from WINE (c.f., Section 3.3) and download the samples from VirusTotal using the download API; 11,108 PE samples are collected. The details of VirusTotal will be explained in Section 3.3.

A PKCS #7 [11] file includes code signing certificate chains, TSA certificate chains, a signature, and a hash value of a PE file. The data sets consist of PKCS #7 files except for the *Malsign* data set that provides only leaf code signing certificates. First, we extract only a leaf certificate from each PKCS #7 file by filtering out intermediate certificates and TSA certificates, and we select only code signing certificates using the keyword of "Code Signing" in the *extendedKeyUsage* extension field. We are unable to parse 1,989 leaf certificates due to parsing errors. 145,582 unique leaf code signing certificates (extracted from 965,114 binary samples) legitimately issued from CAs remain after we remove duplicate leaf certificates (85.2% leaf certificates are duplicate) and two self-signed certificates. Table 3 shows the number of code signing certificates for the top-ten most popular CAs in our data set (D1).

CA	Leaf Certificates	
Verisign	44,014	(30.23%)
Thawte	26,884	(18.47%)
Comodo	24,780	(17.02%)
GlobalSign	12,079	(8.30%)
Symantec	8,913	(6.12%)
DigiCert	8,300	(5.70%)
Go Daddy	7,376	(5.07%)
WoSign	3,796	(2.61%)
Certum	1,874	(1.29%)
StartCom	1,830	(1.26%)
Other	4,281	(2.94%)
Total	145,582	(100%)

Table 3: Top 10 Code signing Certificate Authorities. The top 10 CAs account for 97% of the certificates in our data set (D1).

The D1 data set is used for (1) the trend of revocation setting policy (Section 5.1), (2) the certificates without CRL and OCSP (Section 6.3), (3) the inconsistent responses from CRLs and OCSP (Section 6.3), and (4) the unknown or unauthorized responses from OCSP (Section 6.3).

Revocation information (D2). The CRLs and OCSP points (URLs) are specified at the *CRLDistributionPoints* and *AuthorityInfoAccess* extensions respectively. We extract the CRL and OCSP points from 145,582 leaf code signing certificates that we find in the four data sets. Most (137,027, 94.1%) certificates contain both CRL and OCSP points; only CRL points are specified in 7,794 (5.3%) certificates and only OCSP points are expressed in 98 (0.06%) certificates. We observe a total of 413 unique CRLs, however CRLs can be used for other purposes such as TLS. Therefore, we manually search *Censys.io* for each CRL and filter out CRLs used for other purposes. Eventually, 215 CRLs that are used only for code signing remain. We observed 131 unique points

[2] https://censys.io

for OCSP. This D2 data set is used to examine the problems in effective revocation date setting (Section 5.1), the transient certificates in CRLs (Section 6.3), and the no longer updated CRLs (Section 6.3).

3.2 Revocation Publication Date List (D3)

A CRL contains the serial numbers of revoked certificates, revocation date, and reason code. The *revocation date* field is *effective revocation date* (t_r) (c.f., Section 2.2) that determines the validity of signed program code. In other words, the revocation information in CRLs does not contain the date on which the certificates become revoked. Therefore, we devise a system, called *revocation publication date collection system* that collects revoked serial numbers once a day from our CRL data set in order to detect the *revocation publication date* (t_p), when the certificate is added to CRL or OCSP servers. This information can be used to measure the revocation delay between a malicious signed binary appearing in the wild and a CA revoking the compromised certificate. From the 215 CRLs, we observe 2,617 unique certificates added to the CRLs between Apr. 16th, 2017 to Sept. 10th, 2017. This D3 data set is used to examine the revocation delay (Section 4.2).

3.3 Binary Sample Information (D4 – D6)

Among our measurements, there exist several research questions which require information about the signed binaries. For example, to measure the malware which is still valid due to the ineffective revocation date setting, we need a view of the binaries signed with a revoked certificate and information to determine their maliciousness and their signing date. Therefore, we collect information about the signed binaries from three data sets: WINE, Symantec, and VirusTotal.

Worldwide Intelligence Network Environment (WINE) (D4). WINE [5] provides security telemetry submitted from 10.9 million Symantec customers around the world that opt into this data sharing. Among the various data sets in WINE, we use the binary reputation data that contains metadata of binary files that are seen on endpoints. We extract the following information from this data set: the SHA256 hash value of the file, the server-side timestamp, and the names of the publisher and the CA which are extracted from the code signing certificate. Note that detailed information of the certificate (e.g., a serial number of the certificate, CRL) is not provided in WINE. Also, WINE does not provide the actual binary. This D4 data set is used to examine the problems in revocation date setting (Section 5.1).

Symantec metadata telemetry (D5). For the revoked certificates observed by our *revocation publication date collection system*, we also received meta information about the binaries signed by the 2,617 code signing certificates from Symantec, using the serial numbers of the certificate to identify the set of the affected binaries. The information is similar to WINE, but for a more recent time period than what is in WINE (from Jan. 1st, 2016 to Sept. 10th, 2017) so that we could observe information related to more recent certificates and revocations that we track in D3. The data consist of the serial number of the signing certificates, the SHA256 hash of the binary, the first seen timestamp. Symantec provided us ground truth for identifying malware among these signed binaries as well. With the ground truth, we identify the certificates used on signed malware. This D5 data set is used to estimate malware signing certificates in the wild (Section 4.1), and to examine the revocation delay (Section 4.2).

VirusTotal (D6). Because the previous two data sets do not provide actual binaries, we use VirusTotal [27] to find specific binaries and to perform further analysis. VirusTotal provides a service that analyzes potentially malicious binary files and URLs using up to 63 different antivirus engines. The analysis is triggered when a sample is submitted, the report is kept in a database and exposed externally via an API. We use the private API to collect the following information from these reports: the signed date of the binary, the number of AV engines detected the file as malicious, and the first submission timestamp to VirusTotal.

VirusTotal also allows users to apply rule-based matching on the incoming submissions, which can help researchers find a specific type of malware. This platform is called VirusTotal Hunting [3], and it uses YARA [4] to define rules. We write a YARA rule that triggered when a binary was signed and at least 10 AV engines convict the binary. From each report, we extract the SHA256 hash of the binary, the first submission date, and the serial number of the leaf code signing certificate. The data collection began on Apr. 18th, 2017. The extracted data set is used in the estimation of malware signing certificates (Section 4.1), and to examine the revocation delay (Section 4.2).

We also use the VirusTotal download API to download the actual binary of a given hash when necessary (e.g., to collect the certificate to extract the CRL/OCSP information).

[3] https://www.virustotal.com/#/hunting-overview
[4] http://virustotal.github.io/yara/

3.4 CRL/OCSP Reachability History (D7)

CRL reachability history. For the list of CRLs we have in our data set, we check the reachability of the CRLs daily from Aug. 10th, 2017 to Sept. 10th, 2017. If a CRL is unreachable, we record the timestamp and the reason of failure to the log. This D7 data set is used for measuring the unreachability of CRLs (Section 6.2).

OCSP reachability history. Similar to the reachability checker for CRLs, we also develop an OCSP reachability checker. The checker tests the reachability of each OCSPs we found from the four data sets every 30 minutes. Rather than simply pinging the domain, it queries each OCSP points with the certificates that contain the OCSP point over the OCSP protocol using *Openssl*. Similarly, the timestamp and the reasons are logged if not reachable. It has been running with 131 unique OCSP points from Aug. 10th, 2017 to Sept. 10th, 2017. This D7 data set is used for measuring the unreachability of OCSP points (Section 6.2).

4 Discovery of Potentially Compromised Certificates

There are many reasons for revoking a code signing certificate, and in general it is difficult to determine whether and when a certificate should have been revoked. However, one situation warrants a prompt certificate revocation: when the corresponding private key has been used to sign malicious code [3]. We therefore compute a conservative estimate of the number of certificates used to sign malware in the wild, and we compare it with the coverage of a major security company to assess the odds of discovering all the potentially compromised certificates (Section 4.1). Furthermore, after a signed malware sample has been discovered, the information must reach the principal responsible for revoking the code signing certificate, and the principal must add the certificate to Certificate Revocation List (CRL). We therefore analyze the delay between the time when this information is available to the community and the time when the certificate appears on a CRL (Section 4.2).

4.1 Mark-recapture Population Estimation

The process of revocation starts from discovering the certificates used in malware. To understand how effective the discovery phase is, we need to answer our first research question, *Q1. How many certificates are used to sign malware in the wild?* However, there exists no official repository for code signing certificates and the signed binaries. To overcome this problem, we employ the mark-recapture analysis [15]. This technique was originally developed for measuring wildlife populations. The goal of mark-recapture is to estimate the size N of a population that cannot be observed in its entirety. In our case, N is the number of certificates employed by digitally signed malware. The technique requires two separate samples drawn, with replacement, from the population. The first sampling results in the capture of n_1 subjects. These subjects are marked and released in the wild. The second sampling results in the capture of n_2 subjects, among which p bears the marks from the previous sampling. In other words, p is the size of the intersection of the two samples, denoting the subjects that have been recaptured. An estimator $\hat{N}$ for the total population N can then be computed as:

$$\hat{N} = \frac{n_1 n_2}{p} \tag{1}$$

We apply the mark-recapture technique to the malware signing certificates from two different data sets: Symantec telemetry (D5) and VirusTotal (D6). We consider that each data set is a sample of the total population of potentially compromised certificates. Specifically, n_1 and n_2 represent the numbers of certificates that should have been revoked, as they are known to sign malware, from the Symantec and VirusTotal data sets respectively.

Assumptions and interpretation. Mark-recapture makes three assumptions about the population and the sampling process that may not hold in our case. First, the subjects in the population should have an equal chance of being captured; in other words, the population is *homogeneous*. However, the certificate population is unlikely to be homogeneous. For example, a certificate used by a popular software company would have a higher chance of appearing in our datasets. Second, the samples from the population should be *independent*. That is, the initial capture should not affect the likelihood of recapture. This assumption ensures that the proportion of recaptured subjects in the second sample p/n_2 is the same as the proportion of marked subjects out of the total population n_1/N, which leads to Equation 1. However, security companies share malware feeds with each other, which raises the probability of recapture for the potentially compromised certificates captured in the first sample. Third, the population should be *closed*. A population is closed when its size does not fluctuate due to the birth and death of its members. However, our population changes over time, as certificates are issued and revoked.

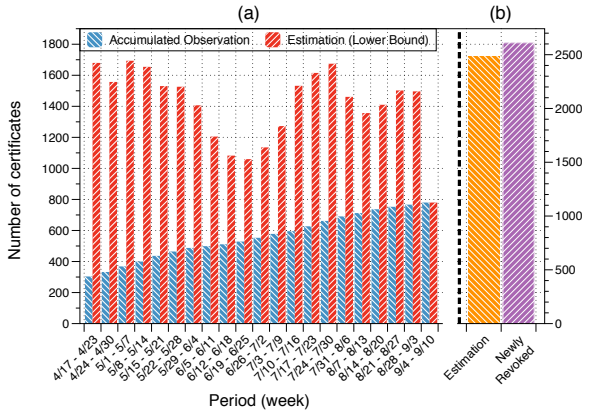

Figure 2: (a) Trend in malware signing certificates (mark-recapture estimation as red and observed number as blue) over time (b) comparison between the estimation and the total number of newly revoked certificates during $(4/18/17 - 9/10/17)$ (the label starts from $4/17$ since it is the start of that week).

To minimize the impact of the last issue, we estimate $\hat{N}$ separately for each day. We set the birth date for each certificate as the first seen timestamp in the Symantec telemetry and the first submission date for VirusTotal, as this is when they join the population of potentially compromised certificates. Using the same reasoning, we consider that a certificate leaves the population on its revocation publication date (t_p). Because CRLs are updated daily, our population of interest is approximately closed within each day.

To mitigate the impact of a non-homogeneous population, we compute our daily estimates between 4/18/17 and 9/10/17, the collection period for D6. While the two data sets include certificates issued before April 2017, malware signed with these older certificates may have a lower probability of occurring in the VirusTotal Hunting. Furthermore, some certificates may have a low prevalence, for example because they are only used in targeted attacks and may not occur in either data set. The existence of such certificates would imply that $\hat{N}$ underestimates the real population N. Similarly, dependencies between the two data sets would lead to an increase of the intersection p, which would also result in an underestimation of N. Our estimation in this section should be interpreted as a *lower bound* for the true population of potentially compromised certificates.

Results. Figure 2(a) shows the average of our daily estimations $\hat{N}$, for each week during our measurement period. We also compare these estimations with the number of potentially compromised certificates that we actually observe, which is the union of the sets of certificates observed daily from the Symantec telemetry (D5) and from VirusTotal (D6). Excluding the last week (9/4–9/10), we estimate that at least 1,004–1,786 code signing certificates were used to sign malware in the wild and had not been revoked by the date of the estimation.[5] On average, the estimated population is 2.74× larger than the observed number of certificates. This suggests that even a major security company like Symantec and an information aggregator like VirusTotal do not observe a large portion of the potentially compromised certificates.

To illustrate the effect of the inefficient discovery process on the revocations, in Figure 2(b) we compare the mark-recapture estimation on all the certificates observed during the measurement period (4/18–9/10/17) with the actual number of newly revoked certificates, which revocation publication date (t_p) is between 4/18/17 and 9/10/17, from data set D3. The number of the estimated population of potentially compromised certificates during this period represents 95.1% of the code signing certificates added to the CRLs. While the CRLs do not indicate the reason for the revocations, our close estimation could indicate that most revocations are done in response to the discovery of signed malware. We note that, because our estimation is a lower bound, the number of potentially compromised certificates may be much larger in reality. However, even if all the certificates that sign malware in the wild are eventually revoked, this does not imply that the security threat is mitigated effectively, as the revocations may correspond to older discoveries. We next investigate the delay between the discovery of potentially compromised certificates and their revocation.

4.2 Revocation Delay

Kim et al. [12] estimated that 80% of the compromised code-signing certificates remain a threat for over 5.6 years after they are first used to sign malware. Their estimation included certificates that were never revoked and used an approximation for the revocation publication date. We take a data driven approach to explore the revocation process. As discussed in Section 2.3, CAs must revoke a certificate within seven days after they are alerted that the certificate has been used to sign malicious code. Therefore, our second research question is *Q2. After the signed malware is discovered, how promptly is the corresponding certificate revoked?*

[5] Because the Symantec telemetry dataset was collected starting from the certificates we observed on CRLs (D3), all the certificates in D5 were revoked by the end of our observation period. During the last week $n_1 = 1$, which prevents us from making an accurate estimation.

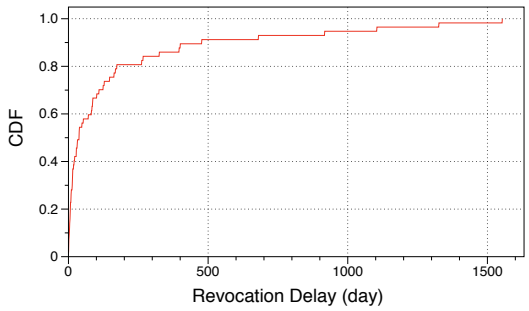

Figure 3: Revocation delays between the dates on which the malware signed with compromised certificates and the dates on which CAs revoke the compromised certificate.

To answer this question, we need an accurate estimation of the revocation publication date (t_p). This is provided by our *revocation publication date collection system* (D3). We focus only on certificates that have been revoked; D3 includes 2,617 code signing certificates, with t_p between Apr. 16th, 2017 and Sept. 10th, 2017.

Our next challenge is to determine the discovery date for the corresponding signed malware. We use Symantec metadata telemetry (D5) to identify a set of hashes for binaries files that are signed with the revoked certificates from D3. Of the 2,617 revoked certificates, we find 468 (17.9%) revoked certificates in the D5 data set, and 146,286 hashes signed with the revoked certificates. Since Symantec does not collect these binaries we rely on VirusTotal (D6) and *AVClass* [25] to get a report of the binary and label the signed malware using consensus results. From the VirusTotal reports we also retrieve the first submission timestamp of the binaries. In total we find 19,053 unique samples in VirusTotal, and 254 unique certificates used to sign the samples.

For each certificate, we use the earliest detection date of a signed malware sample as the discovery date (t_d). As multiple anti-virus vendors were aware of the abuse, this represents a conservative estimate for the date when the security community started suspecting that the certificate was likely compromised. We compute the *revocation delay* ($t_p - t_d$) as the difference between this date and the revocation publication date (t_p), when the certificate was added to its CRL.

Results. The revocation delay ranges from one day to 1553 days; Figure 3 shows a cumulative distribution. The average delay is 171.4 days (5.6 months) (std 324.9 days, median 38 days). The long delays imply that CAs either do not receive the information in a timely manner or do not strictly follow the minimum requirements set by the CA/Browser Forum Code Signing Working Group [3]. In consequence, users remain exposed to this threat for over five months, on average, after the discovery of the signed malware.

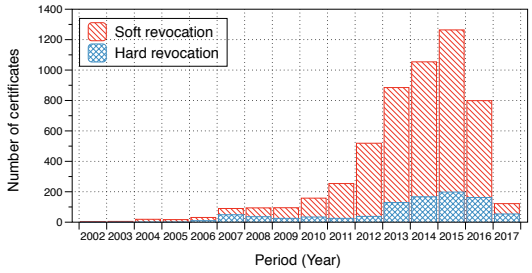

Figure 4: Effective revocation date setting trends: Number of revoked certificates (stacked).

5 Setting the Revocation Date

Even if potentially compromised certificates could be discovered efficiently, the CA must determine a proper revocation date (we call this the *effective revocation date*) to cover the period when trust in the certificate is compromised.

5.1 Problems in Revocation Date Setting

To have an effective revocation process, the next question we have to answer is *Q3. Are effective revocation dates set properly?* As described in Section 2.3, CAs must set revocation dates when revoking the certificates that they have issued. CAs can set t_r (effective revocation date) to t_i (issue date), called *hard revocation*. On the other hand, t_r can be set to any date between t_i and t_e (expiration date), called *soft revocation*. The trust in a signed binary depends on the effective revocation date, and so a CA generally tries to set t_r (effective revocation date) close to the oldest t_m (the date on which the certificate signed malware). We examine CAs' revocation date setting policies to better understand how the CAs set the effective revocation date (e.g., hard or soft), and how the trend is changed over time, using our data set (D1). We also identify the security problem led by the wrong effective revocation date setting in soft revocation.

Trend of effective revocation date setting. We examine how the CAs set the effective revocation date when they revoke the certificates using our collected 145,582 code signing certificates (D1). First, we check the certificate's

	$< t_i$	$= t_i$	$\leq t_e$	$> t_e$	Total
Comodo	0	426	1,437	17	1,880
Thawte	0	74	1,055	39	1,168
Go Daddy	2	14	672	18	706
Verisign	2	59	430	51	542
Digicert	1	161	323	3	488
Starfield	0	3	153	2	158
Symantec	0	33	89	1	123
Wosign	0	57	17	0	74
Startcom	0	0	47	0	47
Certum	0	1	9	0	10
Other	0	96	117	1	214
Total	5	924	4,349	132	5,410

Table 4: Effective revocation date setting policy for top 10 CAs (t_i: issue date, t_e: expiration date).

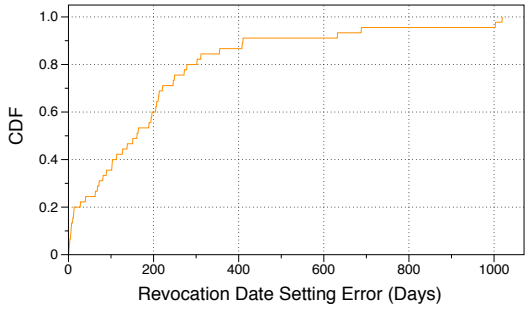

Figure 5: CDF of the revocation date setting error ($t_r - t_m$): difference between the effective revocation date and the first malware signing date of a certificate.

revocation status using CRL points, specified at its certificate extension field. Table 4. shows the breakdown of the effective revocation date setting policy. We observe that 5,410 (3.7% out of 145,582 certificates) certificates are explicitly revoked. Of those, 96% (5,196) certificates have been issued by the top 10 CAs; most (1,880, 34.8%) revoked certificates are issued by Comodo, followed by Thawte (1,168, 21.6%). Most (4,481, 82.8%) revoked certificates take *soft revocation* while only 17.2% certificates perform *hard revocation*.

Most CAs apply both *hard revocation* and *soft revocation* when revoking a certificate. Soft revocation is more common than hard revocation in all CAs except for Wosign; in particular, Startcom has never performed hard revocation in our observation. Interestingly, three CAs (Go Daddy, Verisign, and Digicert) set the effective revocation date to before their certificates' issue date. The two certificates of Go Daddy were set to one day before their issue date, and the one certificate of Digicert was set to five days before its issue date. However, other two certificates of Verisign were set to around five months and nine months respectively before their issue date. It is considered hard revocation; therefore, there are no security threats to clients. Figure 4 presents the total number of soft and hard revocations. The total number of revocation has made a drastic increase since 2012. It is also worth noting that the numbers for 2016 and 2017 are not yet final, as we have already seen in the previous section, due to revocation delay these numbers should continue to grow in the future.

Ineffective revocation date setting. So far we have seen the dominance of soft revocation among the CAs. As mentioned in Section 2.3, soft revocation may result in the survival of signed malware even after a certificate has been revoked if a CA sets the wrong effective revocation date. As shown in Table 4, most CAs have set the effective revocation dates even after its certificate expiration date. In this case, the effective revocation dates become ineffective. In other words, the revoked certificates should not affect any properly signed and timestamped sample including malware should remain valid.

We measure how many CAs erroneously set effective revocation dates, and how many signed malware still remains valid even after the certificates used to sign are revoked. To examine the erroneous effective revocation date problems, the information (e.g., signing date) of binary samples signed with the revoked certificates is necessary. We use WINE data set (D4), and query VirusTotal with the 12,351,946 signed hashes from WINE. Only 4,729,023 (38.3%) samples have *sigcheck* information in its VirusTotal report; and the 4,729,023 samples are signed with 45,613 unique certificates. We are unable to directly obtain the effective revocation dates of the 45,613 certificates because of the following two reasons. First, the search index service of VirusTotal supports only 80TB of data, or about a month of samples so that we cannot query VirusTotal for all old samples. Second, the VirusTotal reports contain neither CRLs nor OCSP points to check the revocation status and to obtain effective revocation dates. Therefore, we query the CRLs we have collected (D2) to check whether or not the certificate is revoked and to obtain effective revocation dates (t_r) if revoked. This process gives us 1,022 revoked certificates (out of 45,613 certificates).

We find that CAs applied the soft revocation policy to revoke 891 (87.2%) certificates. Of those, the effective revocation date (t_r) of 45 (5.1%) certificates were erroneously set by CAs. The affected CAs are summarized in Table 5. We also measure how many malware signed with the certificates are still valid due to the ineffective revocation dates. We first use AVClass [25] to

label malware using the VirusTotal reports. For the labeled malware sample, we extract the signed date t_m. If we find a signed malware with $t_m < t_r$, we say the effective revocation date is erroneously set and the malware remains valid. We find that 250 malware (5.3% out of the 4,716 malware) signed with the 45 certificates still remain valid. The still-valid signed malware should be revoked, but due to the CAs' error, they remain valid and a security threat to clients. The number of still-valid signed malware is relatively small in our data sets, but we believe that more still-valid signed malware can be found in the wild since our data sets are limited, and do not cover all samples in the wild. Figure 5 shows the difference between the effective revocation date (t_r) and the oldest signing date of signed malware (t_m) of the certificate. The shortest difference is one day, and the longest difference is 1019 days (2.8 years). Clients may execute or install the still-valid malware because the executions of the malware do not trigger any warnings for clients even though its certificate is already revoked.

6 Dissemination of Revocation Information

After compromised certificates are properly revoked and the appropriate effective revocation dates are decided, the next step for CAs is to make the revocation public and maintain its availability. We first take a look into the enforcement of the Windows platforms[6] since clients can be affected depending on the enforcement policies in client-side platforms for checking revocation status information. Then, we examine the security problems in dissemination of revocation status information and try to answer our last research question *Q4. Is revocation information served properly?*

6.1 Enforcement in Windows

Client-side platforms must check the validity of code signing certificates when a signed binary is encountered. When there is a failure or inconsistent state at some point in the revocation infrastructure, it matters how the endpoint, where the binary is being executed, handles that failure. Windows considers binary samples signed with revoked certificates as unsigned samples and displays "unknown publisher" in a security warning message. Windows also typically follows the *soft-fail* policy to allow execution with no prompts unless the revocation is explicitly found, for all unknown and unexpected cases the assumption is that it is safe to proceed. We observed that some of the problems in the revocation information dissemination, when combined with the enforcement policy of Windows, could allow binaries with revoked certificates to be executed without security warning messages.

[6] According to Net Market Share (`https://www.netmarketshare.com`), since more than 75% of Windows clients use Windows 7 and Windows 10, we focus on only these two platforms.

6.2 Unavailable Revocation Information

In the code signing PKI, CAs must maintain the revocation information indefinitely since the trusted timestamp extends the life of the certificate for an unknown length much longer than the certificates lifetime. This is an important difference between code signing and Web's PKI. This means that revocation status information has to be always-available and updated much longer than the life of the certificates [20]. There are several cases when the revocation status information for a certificate is not available for clients. The results and affected CAs are summarized in Table 5.

Certificates without CRL and OCSP. The first problem arises when there are no CRL or OCSP points embedded in certificates. Code signing certificates that follow the X.509 v3 standard must include CRLs and OCSP points for clients to check revocation status. However, we observe that 788 (0.5% out of 145,582) certificates contain neither CRLs nor OCSP points from the corpus of leaf code signing certificates (`D1`). This means that clients have no way to check the revocation status for these certificates. Of the 788 certificates that contain neither CRLs nor OCSP points, most (676, 85.8%) were issued by *Thawte*, and they were issued before 2003. Recently, in 2014, *iTrusChina* issued a code signing certificate to *Huawei* without revocation information; thus, the problem does persist. The 788 certificates with no CRLs and OCSP points have already expired. Therefore, no new binaries can be signed with these certificates. However, old binaries (including malware) already signed with the certificates can be valid as long as it contains a trusted timestamp.

We also examine how it affects the Windows platforms. We download several samples signed with the certificates from VirusTotal. We then inspect the certificate of the samples on Windows 7 and 10 to observe how the Windows platforms check the validity of the sample. In both versions of Windows, a message saying "The revocation function was unable to check revocation for the certificate" is displayed if you manually inspect the certificate (seen in Figure 6 in the appendix), but the certifi-

	Comodo	Thawte	Go Daddy	Verisign	Digicert	Starfield	Symantec	WoSign	Startcom	Certum
Ineffective revocation date	●	●	●	●	●	○	○	○	○	○
Certs. without CRLs and OCSP points	○	●	○	○	○	○	○	○	○	○
Unreachable OCSP or CRLs points	●	○	○	●	○	○	○	○	●	●
Inconsistent responses from CRLs and OCSP	○	○	●	○	○	●	○	○	○	○
Unknown or Unauthorized OCSP response	○	○	○	○	○	○	○	○	○	●
Transient certs. in CRLs	●	○	○	○	●	○	○	○	○	●

● = Issues found, ○ = Issues not found

Table 5: Mismanagement issues found across the top 10 CAs.

cate appears trusted due to the soft-fail revocation checking policy of Windows. In fact, when clients attempt to execute such a file, the prompt presents a normal trusted file as seen in Figure 7 in the appendix even though the revocation status information of the certificate is unavailable (at worst, it might be compromised and already revoked).

Unreachable CRLs and OCSP server. We now examine the unreachability of the CRLs and OCSP points in our data set. Recall that we record the unreachability of the CRLs (D7). During our observation period (Apr. 16th, 2017- Sept. 10th, 2017), we observe that 55 CRLs are unreachable at least in one day. However, a few times, there were networking issues for our institution's network which caused issues that were probably localized to our monitoring system. After removing the CRL URLs that were generally reachable, we are left with 13 CRLs that were never available during our observation period.

Of the 13 CRLs, 5 (38.4%) CRLs are unreachable due to HTTP 404 Not Found Error. For example, two CRLs points (http://crl.globalsign.net/ObjectSign.crl, http://www.startssl.com/crtc2-crl.crl) produce HTTP 404 error, which indicates that the CA has removed the CRL from the address but a server still exists at that domain.

One domain has been bought by a domain reseller, which means the CRL point is no longer available. The certificates with this CRL were issued by a certificate reseller; however the reseller shut down that part of its business and let the related domain lapse. We do not provide too many details because at this time the domain can still be purchased, which could have serious implication; either explicitly revoking all certificates for this CA or never revoking them even if they are used to sign malicious files. We suggest that for this case, the root or intermediate CAs should take over and maintain CRLs or OCSP servers if their resellers are no longer operated.

We also measure the unreachability of OCSP severs (D7). As we have experience some network and storage problems on our institution internal infrastructure, we have unreachable 15 OCSP URLs operated by eight CAs (AOL, Verisign, Comodo, StartSSL, WoSign, GlobalTrustFinder, Certum, and GlobalSign) after removing the affected OCSP URLs. The unreachability can be caused by *bad hostname*, *timeout*, *forbidden*, and *method not allowed*. For example, in the case of *bad hostname*, AOL used to be a CA, and operate both one CRLs and two OCSP servers. However, the AOL's servers are currently no longer maintained, and its clients who try to verify program code signed with the certificates are unaware where to query for revocation status information.

Unreachable CRLs and OCSP points are common, since there are many valid reasons to not have a network connection, and so Windows handles these failures quietly. However, this means that when the CRL and OCSP are permanently gone, then the failure also happens quietly. Any binary, including malware, signed with this type of certificate can remain valid due to the Windows soft-fail revocation checking policy.

6.3 Mismanagement in CRLs and OCSPs

Here we highlight some mismanagement issues we found while observing the CRLs and OCSPs during the period from Apr. 16th, 2017 to Sept. 10th, 2017. The affected CAs are summarized in Table 5.

No longer updated CRLs. Recall that CRLs should be re-issued at least once a week, and the next update timestamp at the *nextUpdate* field should be less than ten days from *thisUpdate* field [3]. We examine how often they update and re-issue their CRLs. Of 215 CRLs,

57 CRLs are never updated at all since their *nextUpdate* timestamps are not changed in the observation period of our *revocation publication date collection system*. Most (34 of 57, 59.6%) CRLs are issued by Shanghai Electronic CA, and well-known CAs' CRLs are not found in the 57 CRLs. Moreover, most (130, 89.7% out of 145 CRLs except for unreachable CRLs and not-updated-CRLs) CRLs are updated and re-issued every day. It indicates that CAs re-issue their CRLs when revoked serial numbers are added.

Transient certificates in CRLs. Recall that code signing CAs must maintain and provide the revocation status information of all certificates including expired ones because of the trusted timestamp. However, we find that 278 certificates are added and then later removed from 18 CRLs. The CRLs are maintained by ten different CAs including *GlobalSign*, *Certum*, *Entrust*, *Digicert*, and *Comodo*. Most removed serial numbers are never re-added to its CRL. However, one serial number of *Digicert* is re-added to the CRL after 106 days.

We reach out to the CAs to try and understand the factors that go into a decision to remove a revocation from the CRLs. One CA replied that they had a flaw in their revocation system that removes certificates after the certificate expired, and they fix the flaw to keep the certificates on the CRL indefinitely thanks to our report.

The disappeared serial numbers from CRLs are unlikely to affect the Windows platforms as long as certificates have both CRLs and OCSP points since in Windows, OCSP is always preferred over CRL to check revocation status. However, when code signing certificates contain only CRL points, Windows must rely on only the CRL mechanism. In our data set (D1), the 28,386 leaf code signing certificates (19.4% out of 145,582) contain one of the 18 CRL points that have experienced serial numbers disappearance. Most certificates (82.8%) have both the CRL and OCSP points, but the 4,878 (17.2%) certificates issued by *GlobalSign* include only CRL. Therefore, the Windows platforms must rely on only the specified CRL points to check revocation status. If revoked serial numbers are removed from CRLs, any program code including malware signed with one of the 4,878 certificates can remain valid even though the certificate is already revoked.

Inconsistent responses from CRLs and OCSP. Since CAs are distributing revocation information through CRLs and OCSP, and one is a fallback mechanism for the other. We expect that the state in the CRL and OCSP would be consistent; for example, when the serial number of a revoked certificate is found in a CRL, the corresponding OCSP will also return that the certificate is revoked.

We observe that 19 certificates have inconsistent responses from CRLs and OCSP from our data set (D1); the certificates are valid according to the OCSP, but are revoked in the corresponding CRLs [7]. To examine how the inconsistency between OCSP and CRLs affects the Windows platforms we download the binary samples signed with these certificates from VirusTotal and check its revocation status in the Windows platforms. These downloaded samples are classified as malware by most AV vendors and their certificates are explicitly revoked in the CRL. Therefore, the samples must be invalid and not be executed. However, the Windows platforms present these signed malware as valid, due to the inconsistency between OCSP and CRLs. The Windows policy is to first check the OCSP. If the response from the OCSP indicates the certificate is valid, then Windows does not double-check the status using CRLs. To prevent this sort of threats caused by mismanagement issues, Windows should double-check certificate revocation status using both OCSP and CRLs.

The 19 certificate were issued by Go Daddy; three certificates were issued by Starfield Technologies (related to Go Daddy). We believe that Go Daddy and Starfield Technologies may share the same infrastructures for revocation information repositories; the infrastructures may cause the inconsistency problem. It indicates that CAs must keep monitoring the consistence between CRLs and OCSP responses.

Unknown or unauthorized responses from OCSP. According to the OCSP specification, the OCSP responders (servers) should return three statuses for a certificate; *good*, *revoked*, and *unknown* [24]. The *unknown* state indicates that the responder is unaware of the status of the certificate being requested. Surprisingly, in our data set (D1), the three OCSP servers (Certum, Shanghai Electronic CA, and LuxTrust) respond that they are unaware of the status of their 669 certificates; almost all of the certificates (658, 98%) are issued from Certum; the rest of them (2%) are issued by Shanghai Electronic CA and LuxTrust.

OCSP responders may also respond with an error message. The error message has the five types; *malformedRequest*, *internalError*, *tryLater*, *sigRequired*,

[7] We consider only this case where the responses from OCSP indicate the certificates are valid, but revoked in CRLs since only this case can lead to security threats where Windows users are allowed to execute the binary samples with revoked certificates. However, the reversed inconsistent responses (revoked in OCSP and valid in CRLs) do not affect Windows in terms of security as Windows believes certificates are revoked when the responses from OCSP indicate revoked, and it displays warning messages for Windows users.

and *unauthorized*. The *unauthorized* response means that; (1) the client is not authorized to query the OCSP server, or (2) the OCSP server is unable to respond authoritatively [24]. In the OCSP server-side case, OCSP responders return an *unauthorized* error message when (1) they are not authorized to access the revocation records for the certificate, or (2) when they remove the revocation records of expired certificates and are unable to locate the records for requested certificates. We examine how many OCSP servers return the error messages for the requested certificates that they have issued. In our data set (D1), we observe that 2,129 certificates (1.5% out of 145,582) have the *unauthorized* error messages; most certificates (1,515, 71.2%) are issued by Go Daddy. To figure out whether client or server-side causes the problem, we check the revocation status of the certificates through OCSP using OpenSSL, and using *SignTool* on the Windows platforms. Both tools receive the *unauthorized* error messages, which indicates that this problem results from the server-side, not the client-side.

The *unknown* or *unauthorized* responses from OCSP may not affect Windows platforms in terms of security since they also check CRLs if they receive those responses. However, it indicates that CAs improperly maintain their OCSP servers.

7 Limitation

Data sets collection. Due to the nature of how signed binaries are distributed (various distribution mechanisms), there is no easy way to collect all signed binaries and code signing certificates in the wild. For example, some binaries come directly from websites, but others come after running installers or updaters or from external storage. More importantly malicious binaries often are targeted and the samples are hard find or only available for a short time. This is an important difference between the code signing PKI and the Web's PKI as it relates to measurement studies. TLS certificates collected through network scanners provide a view of the publicly accessible Web's PKI, however our collected code signing certificates may not be representative of the entire code signing PKI ecosystem as the collected data sets do not cover all certificates and signed samples in the wild. Therefore, we attempt to collect the broadest view of code signing certificates, and also try to approximate how large compromised code signing certificates are with the mark-recapture estimation.

Mark-recapture population estimation. As we discussed in Section 4.1, the characteristics of the data violates the assumptions of Mark-recapture algorithm: 1) the population should be homogeneous, 2) the samples should be independent, and 3) it should be a closed population. It results in underestimating the true population of the potentially compromised certificates. Therefore, the actual severity of the threat might be much more significant. However, the results suggest that even with the underestimation, the number doubles the number of malware-signing certificates observed by Symantec and VirusTotal combined (which is a precise measurement, not an estimate). This puts the challenge of discovering compromised certificates into perspective, as a major security company and an information aggregator cannot see most of these certificates. Additionally, it provides a possible explanation for the long revocation delays we report.

8 Discussion

The findings from our measurement study (Section 4–6) suggest the current revocation systems based on CRLs and OCSP are facing several problems including (1) difficulties in discovering compromised certificates, (2) revocation delay, (3) ineffective revocation dates, and (4) improper maintenance of the revocation information. We discuss several preliminary recommendations for the effective code signing PKI and how a new design could address the current problems in revocation.

Recommendation. We suggest the following properties for the revocation system:

- *Publicize the issuances of certificates and signed binaries.* As depicted in Section 4, CAs have difficulties in discovering compromised certificates that they have issued due to the nature of the code signing PKI. If CAs or owners of certificates are informed and aware that their certificates are abused, CAs would promptly and properly revoke the compromised certificates. For this goal, similar to TLS certificate transparency [18], we suggest a new certificate transparency system for the code signing PKI. In this system, CAs should log the issuances of code signing certificates when issuing new certificates. The distinct feature from TLS certificate transparency is that publishers are required to log the history of when/what binaries (to be publicly distributed) are signed with their private keys. Along with code signing certificates, the hash values of signed binaries are logged in the proposed system. The system should be available to the public so that anyone can audit and monitor the logs. Using the logs, CAs and owners are able to know the first date of when a certificate becomes compromised, which results in a proper effective revocation date.

- *Better dissemination of revocation information.* The CAs should better understand the code signing PKI and properly maintain their revocation systems (CRLs and OCSP servers) to have better availability and consistency that can help clients correctly check the revocation status of certificates. Moreover, rather than maintaining their own separate infrastructures only for dissemination of revocation information, they may use our proposed code signing certificates transparency to log their revocation information.
- *More conservative Windows' checking policy.* Windows should double-check the revocation status of code signing certificates for the inconsistent responses from OCSP and CRLs. Moreover, Windows should apply the *hard-fail* revocation checking policy for better security.

9 Related Work

We discuss related work in two key areas: identifying the code signing PKI abuse and measuring revocation problems in the Web's PKI.

Code signing PKI abuse. Sophos [28], Kotzias et al. [13], and Alrawi et al. [1] examined the signed malicious PE files. They found that the most malicious PE files were PUP, and they were signed with code signing certificates legitimately issued from CAs. On the contrary, Kim et al. [12] focused on the breaches of the trust in the code signing PKI ecosystems; many certificates associated with stolen private keys were used to sign malware. These studies briefly introduced a few of the revocation problems, but they did not make a distinction between the effective revocation date (t_r) and the revocation publication date (t_p) and only measured the former. This may result in an inaccurate estimation of the revocation delay. In contrast, we measured t_p by periodically collecting CRLs. Additionally, we analyzed the revocation process from end-to-end and we report new findings regarding the discovery of compromised certificates and the dissemination of revocation information.

Revocations problems in the Web's PKI. Compared to the code signing PKI, the Web's PKI ecosystems has been well studied since many network scanners have been introduced to collect data: e.g., Zmap [7]. Zhang et al. [30] and Durumeric et al. [6] have found that the number of revocations increased after the *Heartbleed* announcement. However, the majority of the compromised certificates were not revoked even after new certificates were re-issued. Liu et al. took a close look at the TLS certificate revocation [19]. They found that a large fraction of TLS revoked certificates are served. Web browsers often failed to check the revocation status due to the expensive revocation status checking in terms of bandwidth and latency. Kumar et al. [16] measured the mismanagement of OCSP and CRLs in the Web's PKI: specifically endpoint availability, uptime, and error responses.

10 Conclusion

Certificate revocation is the primary defense against the abuse in the code signing PKI. An effective certificate revocation process consists of three roles: (1) discovering compromised certificates, (2) revoking the compromised certificates with a meaningful date, and (3) disseminating the revocation information. However, we found that the revocation processes can have security problems, and new security threats can be introduced by the problems. In the discovery phase, CAs take on average 5.6 months to revoke the compromised certificates after the certificates was used to sign a known malicious binary. The mark-recapture estimation of compromised certificates point to the fact that it is difficult to find abusive certificates in the wild. The validity of a signed sample is determined by the effective revocation date, but CAs improperly set effective revocation dates. The inaccurate effective revocation dates mean that signed malware remains valid even after its certificate is revoked. Although CAs properly and promptly revoke the compromised certificates, clients can be exposed to signed malware attacks due to CAs' mismanagements of CRL and OCSP. There are many cases that we have seen where clients are unable to check certificate revocation status due to (1) missing CRLs and OCSP points, (2) unreachable CRLs and OCSP points, (3) CRLs that are no longer updated, (4) revoked certificates that are mistakenly removed from a CRL, (5) inconsistent responses from CRL and OCSP, and (6) unknown or unauthorized responses from OCSP. These discoveries highlight various properties of the code signing PKI and its revocation process that should be monitored more actively due to the security implications that they create.

Acknowledgement

We thank the anonymous reviewers and our shepherd, Mohammad Mannan, for their feedback. We also thank VirusTotal for access to their service and Symantec for making data available through the WINE platform. This research was partially supported by the National Science Foundation (award CNS-1564143) and the Department of Defense.

References

[1] ALRAWI, O., AND MOHAISEN, A. Chains of distrust: Towards understanding certificates used for signing malicious applications. In *Proceedings of the 25th International Conference Companion on World Wide Web* (Republic and Canton of Geneva, Switzerland, 2016), WWW '16 Companion, International World Wide Web Conferences Steering Committee, pp. 451–456.

[2] CANGIALOSI, F., CHUNG, T., CHOFFNES, D., LEVIN, D., MAGGS, B. M., MISLOVE, A., AND WILSON, C. Measurement and analysis of private key sharing in the https ecosystem. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2016), CCS '16, ACM, pp. 628–640.

[3] CODESIGNINGWORKINGGROUP. Minimum requirements for the issuance and management of publicly-trusted code signing certificates. Tech. rep., 2016.

[4] COOPER, D., SANTESSON, S., FARRELL, S., BOEYEN, S., HOUSLEY, R., AND POLK, W. Internet X.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC 5280, RFC Editor, May 2008. `http://www.rfc-editor.org/rfc/rfc5280.txt`.

[5] DUMITRAŞ, T., AND SHOU, D. Toward a standard benchmark for computer security research: The Worldwide Intelligence Network Environment (WINE). In *EuroSys BADGERS Workshop* (Salzburg, Austria, Apr 2011).

[6] DURUMERIC, Z., KASTEN, J., ADRIAN, D., HALDERMAN, J. A., BAILEY, M., LI, F., WEAVER, N., AMANN, J., BEEKMAN, J., PAYER, M., AND PAXSON, V. The matter of Heartbleed. In *Proceedings of the 2014 Conference on Internet Measurement Conference* (New York, NY, USA, 2014), IMC '14, ACM, pp. 475–488.

[7] DURUMERIC, Z., WUSTROW, E., AND HALDERMAN, J. A. ZMap: Fast internet-wide scanning and its security applications. In *Proceedings of the 22Nd USENIX Conference on Security* (Berkeley, CA, USA, 2013), SEC'13, USENIX Association, pp. 605–620.

[8] FALLIERE, N., O'MURCHU, L., AND CHIEN, E. W32.Stuxnet dossier. Symantec Whitepaper, February 2011.

[9] GOODIN, D. Stuxnet spawn infected kaspersky using stolen foxconn digital certificates, Jun 2015.

[10] HOLZ, R., BRAUN, L., KAMMENHUBER, N., AND CARLE, G. 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* (2011), ACM, pp. 427–444.

[11] KALISKI, B. PKCS #7: Cryptographic message syntax version 1.5. RFC 2315, RFC Editor, March 1998. `http://www.rfc-editor.org/rfc/rfc2315.txt`.

[12] KIM, D., KWON, B. J., AND DUMITRAŞ, T. Certified malware: Measuring breaches of trust in the windows code-signing pki. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017), CCS '17.

[13] KOTZIAS, P., MATIC, S., RIVERA, R., AND CABALLERO, J. Certified pup: Abuse in authenticode code signing. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA, 2015), CCS '15, ACM, pp. 465–478.

[14] KOZÁK, K., KWON, B. J., KIM, D., GATES, C., AND DUMITRAŞ, T. Issued for abuse: Measuring the underground trade in code signing certificate. In *17th Annual Workshop on the Economics of Information Security (WEIS)* (2018).

[15] KREBS, C. J., ET AL. Ecological methodology. Tech. rep., Harper & Row New York, 1989.

[16] KUMAR, D., WANG, Z., HYDER, M., DICKINSON, J., BECK, G., ADRIAN, D., MASON, J., DURUMERIC, Z., HALDERMAN, J. A., AND BAILEY, M. Tracking certificate misissuance in the wild. In *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 288–301.

[17] KWON, B. J., SRINIVAS, V., DESHPANDE, A., AND DUMITRAS, T. Catching worms, trojan horses and pups: Unsupervised detection of silent delivery campaigns. In *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017* (2017).

[18] LAURIE, B., LANGLEY, A., AND KASPER, E. Certificate transparency. RFC 6962, RFC Editor, June 2013.

[19] LIU, Y., TOME, W., ZHANG, L., CHOFFNES, D., LEVIN, D., MAGGS, B., MISLOVE, A., SCHULMAN, A., AND WILSON, C. An End-to-End Measurement of Certificate Revocation in the Web's PKI. ACM Press, pp. 183–196.

[20] MICROSOFT. Code-Signing Best Practices. Tech. rep., 2007. `https://msdn.microsoft.com/en-us/library/windows/hardware/dn653556`.

[21] MICROSOFT. Windows Authenticode portable executable signature format, Mar 2008. `http://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/Authenticode_PE.docx`.

[22] PARNO, B., MCCUNE, J. M., AND PERRIG, A. Bootstrapping trust in commodity computers. In *IEEE Symposium on Security and Privacy* (2010), pp. 414–429.

[23] RESEARCH, K. L. G., AND TEAM, A. The duqu 2.0 persistence module, Jun 2015.

[24] SANTESSON, S., MYERS, M., ANKNEY, R., MALPANI, A., GALPERIN, S., AND ADAMS, C. X.509 internet public key infrastructure online certificate status protocol - ocsp. RFC 6960, RFC Editor, June 2013. `http://www.rfc-editor.org/rfc/rfc6960.txt`.

[25] SEBASTIÁN, M., RIVERA, R., KOTZIAS, P., AND CABALLERO, J. Avclass: A tool for massive malware labeling. In *International Symposium on Research in Attacks, Intrusions, and Defenses* (2016), Springer, pp. 230–253.

[26] SWIAT. Flame malware collision attack explained, Jun 2012.

[27] VIRUSTOTAL. `www.virustotal.com`, 2017.

[28] WOOD, M. Want My Autograph? The Use and Abuse of Digital Signatures by Malware. *Virus Bulletin Conference September 2010*, September (2010), 1–8.

[29] YILEK, S., RESCORLA, E., SHACHAM, H., ENRIGHT, B., AND SAVAGE, S. When private keys are public: Results from the 2008 debian openssl vulnerability. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement* (New York, NY, USA, 2009), IMC '09, ACM, pp. 15–27.

[30] ZHANG, L., CHOFFNES, D., LEVIN, D., DUMITRAS, T., MISLOVE, A., SCHULMAN, A., AND WILSON, C. Analysis of ssl certificate reissues and revocations in the wake of heartbleed. In *Proceedings of the 2014 Conference on Internet Measurement Conference* (New York, NY, USA, 2014), IMC '14, ACM, pp. 489–502.

Appendix

A Screenshots

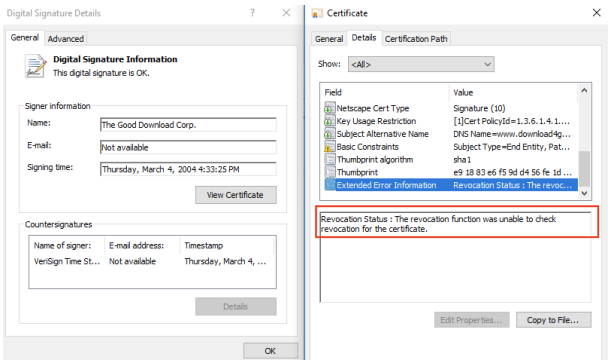

Figure 6: Screenshot of Windows 10 when a certificate without revocation information.

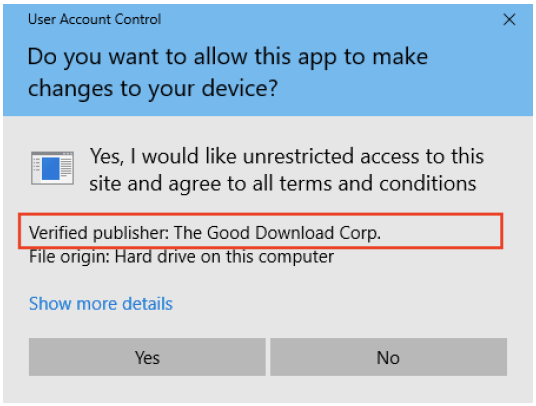

Figure 7: Screenshot of the prompt displayed in Windows 10 when executing a signed binary file with missing certificate revocation information. In this case, nor CRL or OCSP information is provided in the certificate.