Certified Malware in South Korea: A Localized Study of Breaches of Trust in Code-Signing PKI Ecosystem

Bumjun Kwon[1], Sanghyun Hong[2], Yuseok Jeon[3], and Doowon Kim[4(✉)]

[1] The Affiliated Institute of ETRI, Daejeon, South Korea
[2] Oregon State University, Corvallis, USA
[3] Ulsan National Institute of Science and Technology (UNIST), Ulsan, South Korea
[4] University of Tennessee, Knoxville, USA
doowon@utk.edu

Abstract. Code-signing PKI ecosystems are vulnerable to abusers. Kim et al. reported such abuse cases, *e.g.*, malware authors misused the stolen private keys of the reputable code-signing certificates to sign their malicious programs. This *certified* malware exploits the chain of the trust established in the ecosystem and helps an adversary readily bypass security mechanisms such as anti-virus engines. Prior work analyzed the large corpus of certificates collected from the wild to characterize the security problems. However, this practice was typically performed in a global perspective and often left the issues that could happen at a local level behind. Our work revisits the investigations conducted by previous studies with a local perspective. In particular, we focus on code-signing certificates issued to South Korean companies. South Korea employs the code-signing PKI ecosystem with its own regional adaptations; thus, it is a perfect candidate to make a comparison. To begin with, we build a data collection pipeline and collect 455 certificates issued for South Korean companies and are potentially misused. We analyze those certificates based on three dimensions: (i) abusers, (ii) issuers, and (iii) the life-cycle of the certificate. We first identify that the strong regulation of a government can affect the market share of CAs. We also observe that several problems in certificate revocation: (i) the certificates had issued by local companies that closed the code-signing business still exist, (ii) only 6.8% of the abused certificates are revoked, and (iii) eight certificates are not revoked properly. All of those could lead to extending the validity of certified malware in the wild. Moreover, we show that the number of abuse cases is high in South Korea, even though it has a small population. Our study implies that Korean security practitioners require immediate attention to code-signing PKI abuse cases to safeguard the entire ecosystem.

1 Introduction

The establishment of trust in software distributed over the Internet is challenging due to the nature of software distribution: unknown sources and a high chance

D. Gao et al. (Eds.): ICICS 2021, LNCS 12918, pp. 59–77, 2021.
https://doi.org/10.1007/978-3-030-86890-1_4

of tampering during distribution. To overcome these challenges and to guarantee the authenticity and integrity of software, Code-signing PKI is designed and now becomes a de-facto standard in the software ecosystem. Similar to other PKIs such as the Web's PKI, the code-signing PKI also requires Certificate Authorities (CAs) to attest that a certificate belongs to a legitimate software publisher. The CAs issue code-signing certificates for publishers, after the vetting process. Software publishers sign their software with their issued certificates to warrant the authenticity and the integrity of the software. In turn, clients can establish trust in the signed software by verifying the digital code-signing signature. They can know not only the identity of the publisher but also that the software has not been altered during the distribution.

A security rule of thumb for a user is to only execute or install software that contains valid signatures from reputable software publishers with whom she can establish trust. However, anecdotal evidence has shown that the security rule cannot be guaranteed since software properly signed by legitimate publishers can be severe malware [10,24,31]. For example, the Stuxnet worm included device drivers that had been properly signed with the private keys stolen from two Taiwanese semiconductor companies, located in close proximity [10]. The fact is that the malicious usage of these stolen private keys helps remain undetected for a longer period than the other malware [10]. Furthermore, the abuse of code-signing is also prevalent among Potentially Unwanted Programs (PUPs) [5,16,17,32].

This observation has sparked an interest in the real-world breaches of trust in the code-signing ecosystem. In particular, Kim *et al.* [13,14] conducted a large-scale analysis of code-signing abuse cases in the Windows code-signing PKI ecosystem. However, these studies were mostly conducted from a global perspective; hence, they often left the breaches that would happen in sub-populations overlooked. Local software publishers may mainly target local customers; so in this case, the local publishers should have regional adaptations in their code-signing ecosystem, considering the environmental factors of their countries or regions—*e.g.,* because of law[1]. For instance, regulations may state the qualification of a CA or force how the PKI should be operating. Thus, the characteristics of abuse cases can be different from the previous studies. Moreover, the analysis tools in prior work focus on emphasizing the most prevalent findings in the collected datasets.

In this paper, we tackle the prior emphasis on the global perspective and make a first step towards understanding the breach cases in the sub-populations. Specifically, we ask: *What characteristics can we find from an analysis of a specific country?* To answer this question, we give an eye to the Windows code-signing PKI in South Korea. South Korea is known to have its unique PKI ecosystem, developed alongside the digital signature act (DSA), which was established in 1999 [6,15]. DSA states that only a signature is valid if it is endorsed by an accredited CA. South Korean users are also known to be exposed to various "security software" necessary for web activities where identity verification is

[1] PKI in Asia – Case Study and Recommendations: https://fidoalliance.org/wp-content/uploads/FIDO-UAF-and-PKI-in-Asia-White-Paper.pdf.

required *e.g.,* banking, e-commerce [28,29]. The electronic financial transaction act, which became active in 2007, has fostered such an environment. Therefore, we may expect to see unique characteristics reflected in the code-signing PKI ecosystem as well. Nevertheless, little is known about the regional differences; the same applies to the Korean code-signing PKI abuse.

We design a system, which extracts code-signing certificates and identifies Korean certificates that are likely compromised. We utilize information from the certificate and the scanning reports of the binary for identification. We examine the characteristics of Korean signed malicious samples and compromised certificates. Specifically, we investigate how prevalent code-signing abuse is, who are the abusers, who issue the certificates, and whether the compromised certificates are adequately revoked or not.

We found code-signing abuse is prevalent in Korea for its population. The number of signed malicious samples accounted for 1.8% of the total samples, whereas the population is nearly 1% among the global internet users. We also find the unique distribution of the CAs. Thawte dominates the population and a local CA Yessign is observed. Yessign is out of the code-signing business and that could be a potential problem in revocation. Such characteristics might be due to the web environment in Korea, cultivated by its regional PKI laws. Besides, we observe revocation is not done properly in Korea as well. Only 6.8% of the certificates are revoked, and eight certificates have set revocation dates ineffectively. It endangers users of the signed malicious binaries.

Contributions. In summary, we make the following contributions:

- We design a system that collects the malicious programs and compromised certificates from South Korea. We identified 455 certificates that are issued for South Korean companies and are potentially misused by malware authors.
- We highlight the abuse cases in the code-signing ecosystem in South Korea. Using the observations in the previous studies as our baseline, we report the commonalities and differences in our findings.
- Using those differences, we analyze and identify the distinct characteristics of Korean compromised certificates that are fostered by the regional laws.

2 Background and Motivation

In this section, we briefly overview the code-signing PKI; especially, the code-signing process, the distinct characteristic of the code-singing PKI that is mainly different from the Web's PKI, and revocation that can cause extra security threats. We then highlight our motivation why we need to study the unique characteristics of the Korean code-signing PKI ecosystem.

2.1 Overview of the Code-Signing PKI

Code-signing is a security technology that utilizes the digital signature mechanism. It helps authenticate the publisher of a software program and guarantees the software's integrity after signing. It requires creating a digital signature using

the publisher's private key (i.e., signing), and then embed the digital signature into the software. In turn, for clients, when verifying the signed software, they need the public key associated with the publisher's private key to verify the signature.

The code-signing also relies on Public Key Infrastructure (PKI), called the Code-signing PKI. As the nature of the Internet, clients cannot trust any public key transferred over the Internet that claims to be legitimate. It is because public keys do not have any information about the ownership. To mend this problem, third-parties, called Certificate Authorities (CAs), attest that a public key belongs to a particular owner (in this case, a software publisher or developer) who possesses the associated private key. We call this endorsed key a certificate. As long as we trust the CAs, we trust all certificates issued by the CAs except for revoked certificates. This chain of trust starts from the end entity (i.e., publisher) to the root certificate pre-installed in client-side systems such as operating systems or web browsers.

2.2 Code-Signing Process

Like the Web's PKI (e.g., TLS), a software publisher first applies for code-signing certificates to CAs with the applicant's public key. After verifying the publisher's identity, the CA issues a code-signing certificate based on the X.509 v3 certificate standard [8]. The software publisher uses its private key associated with the issued certificate to sign its software. Specifically, in the signing process, the hash value of the software is first computed, and then, the hash value is digitally signed with the publisher's private key. Finally, the digital signature and the chain of the certificates are bundled with the original software. This whole process is illustrated in Fig. 5. In turn, the client has to verify the signature with the public key embedded in the certificate when encountering the signed software. The verification process allows clients to recognize any modifications of the program when verifying the signed software.

Trust Timestamping. The distinct difference between the Web's PKI and the code-signing PKI is *trust timestamping*. The trust timestamping guarantees when a binary file is signed, and if a binary is signed before the certificate's expiration date, the validity extends after the certificate expires, which is different from the Web's PKI where the validity of a domain is no longer ensured after the certificate expires.

As illustrated in Fig. 5, when a binary file is signed, the hash value of the binary file is sent to a Time Stamping Authority (TSA), and the TSA issues a trusted timestamp. The TSA signs the timestamp and the hash value with its certificate. This so-called trust timestamp is sent back to the publisher. Then the software publisher embeds the trust-timestamp signature and the TSA's certificate into the signed software.

2.3 Revocation

Another important role for CAs besides issuing certificates is to revoke the compromised certificates that they have issued. There are various reasons for CAs

to revoke their issued certificates; (1) when the private key associated with a certificate is stolen and used to sign malware samples [13], (2) when a weak cryptographic key is used to generate a certificate [33], (3) when CAs are hacked and compromised, and then issue certificates for adversaries [24], and (4) when a certificate is issued under the name of a shell company or through impersonations, etc. [13].

There are two primitive ways for CAs to disseminate the revocation status information; (1) Certificate Revocation List (CRL) and (2) Online Certificate Status Protocol (OCSP). In CRL, clients need to download the revocation lists periodically and to check if the certificate is on the lists. If the certificate's serial number is on the list, clients can consider the certificate is revoked and no longer valid. OCSP is the successor to CRL, and it allows clients to query a CA for the revocation status of a certain certificate rather than downloading a bulk of the serial numbers using CRLs. Both CRLs and OCSP responses are signed with CAs' certificates to guarantee their integrity.

Erroneous Revocation Data Setting. When revoking certificates, CAs must set the effective revocation date (c.f., Sect. 2.3). Kim et al. [14] have examined the security problems of the current code-signing revocations. If CAs erroneously set an effective revocation date, all signed programs (including malware) signed before the effective revocation date can remain valid even though the certificate is revoked. It is due to the trust timestamp mechanism.

2.4 Motivation

Code-Signing Abuse. Recent measurement studies [5,13,14,16–18] have reported that adversaries have attempted to compromise the code-signing PKI for their malicious purpose; their main purposes are 1) to efficiently distribute their malware and 2) to lure clients into installing their malware. Attackers can make a bold move of stealing the private keys of benign software companies and use the keys to sign their malware, which makes a much powerful attack. The signed malware now looks like a legitimate product from a benign software company, which misleads clients to believe the signed malware is safe to execute. Furthermore, adversaries incorporate shell companies and use this fake company information to get issued code-signing certificates legally and legitimately from the code-signing CAs.

Motivation for a Regional Study. Previous measurement studies have been conducted from a global perspective considering software publishers and CA as global entities. However, this global perspective analysis can lead to misunderstanding or neglecting local characteristics because it mainly focuses on global-scale cases. In other words, the code-signing abuse cases may vary depending on the locality of the attackers and their targets. Thus, to enhance the security of the code-signing PKI ecosystem, we need to understand 1) the local characteristics of the code-signing abuse cases and 2) adversaries who compromise local software publishers targeting local victims. Moreover, in terms of data collection,

the previous methods and results may often be biased to the majority population and a limited number of countries. Thus, a regional target attack campaign with a small number of malware samples could have been neglected or overlooked.

South Korean Web Environment. We focus on South Korea for this study. South Korea has a unique environment fostered by the regulations. Two acts played as the dominant factor. The digital signature act (DSA) established in 1999 has restricted the "valid" form of digital signatures [6,15]. It only concedes signatures issued from "accredited CA"s to be legitimate. There are six accredited CAs, including KFTC, KICA, Koscom, KECA, KTD, and Initech [4]. Among them, KFTC once served as a code-signing CA under the name of "yessign" (https://www.yessign.or.kr/). KICA and KECA act as a distributor of the global code-signing CA. KICA (https://www.kicassl.com/) is a relay of Comodo; KECA (https://cert.crosscert.com/) offers Digicert and Thawte products. Next, the electronic financial transaction act, which became active in 2007, is known as the main cause of the notorious Korean web environment. Due to this act, Korean users have been forced to install various "security software" such as keylogger detection for web activities [28,29]. The flood of these mandatory "legitimate" software, which are digitally signed, may have introduced side effects that incapacitate the defense mechanism of code-signing. For instance, a survey was conducted on Korean adware victims [1], which reported that only 2.8% consciously clicked "allow install" the adware. Moreover, anecdotes [2,3] show that South Korean software companies have become an attractive target for adversaries. Specifically, many South Korean software companies were stolen the private keys of their code-signing certificates, and the stolen private keys were misused to sign malware. Therefore, we believe South Korea is an attractive candidate for studying the local characteristics of the code-signing PKI, and understanding such characteristics may help improve the security of the entire code-signing PKI ecosystem.

3 Data Collection

To better understand the landscape of code-signing abuse in South Korea, we first need to capture signed malware and PUPs in the wild and extract code-signing certificates. From the code-signing certificates, we need to obtain information such as publisher names (common names), locality addresses, issue dates, expiration dates, issuers (CAs), and more. However, due to the nature of software distribution, it is significantly challenging to collect all signed malicious samples and their code-signing certificates in the wild. Whereas in the Web's PKI, a comprehensive list of TLS certificates can be readily collected by scanning the entire IPv4 addresses with a network scanner (e.g., ZMap [9]). This is because signed malware samples can be distributed through a pre-installed updater/installer tool; or others can be distributed from external storage or directly from websites. To overcome these challenges, we present a new collection pipeline for Korean code-signing certificates that are likely compromised, as illustrated in Fig. 1.

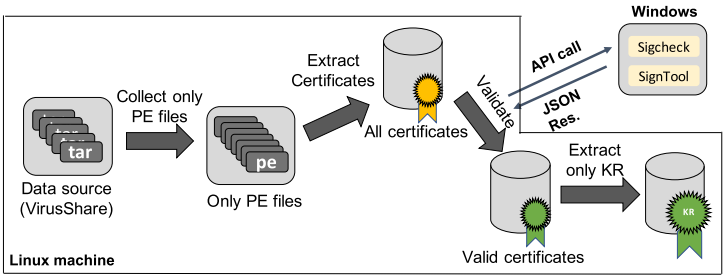

Fig. 1. The overview of our Korean compromised certificates collection pipeline. (1) Malicious files are collected from VirusShare, (2) filter out non-PE files, (3) extract code signing certificates from PE files, (4) validate PE files and certificates using the Windows SigCheck and SignTool, and (5) extract only Korean certificates.

3.1 Data Source

We utilize *VirusShare* (http://virusshare.com), the large corpus of malware, to collect signed malware and to extract Korean compromised code signing certificates from the corpus. We also utilize VirusTotal (http://virustotal.com) to label the collected signed malware samples.

VirusShare and VirusTotal. We collect malicious binaries from VirusShare that is one of the most extensive sets of malware samples available to the public. Since the data sets are freely downloadable, many security research works have utilized them. The malicious samples consist of not only Windows Portable Executable (PE) files, but also HTML files including malicious JavaScript code. We sample 57 tar files (out of 312 tar files) from VirusShare. Each tar file contains either 131,072 or 65,536 malware samples. We collect a total of 5,934,399 malicious files.

To classify the malicious samples, we use VirusTotal. VirusTotal is a Web service where users can freely upload executable samples (including malware and benign samples) and analyze the samples to classify with up to 63 different Anti-Virus (AV) engines. The service provides a report containing the number of AV engines that detect the samples as malicious and the corresponding labels. In our work, we utilize that information to classify the collected samples (c.f., Sect. 3.3).

3.2 System Overview

In this section, we describe our new system. As presented in Fig. 1, the new system is a pipeline for identifying digitally signed PE files and extracting compromised Korean code-signing certificates.

Identifying PE Files. We first filter out non-PE files from the total of 5,934,399 malicious files (out of 57 VirusShare tar files) since the files include not only PE files, but also JavaScript code. The 5,934,399 samples are fed into our

system shown in Fig. 1 to exclude non-PE files and non-signed PE files. When a PE file is signed, the size of the IMAGE_DIRECTORY_ENTRY_SECURITY field is non-zero. 525,071 digitally signed PE files remain after this step, which accounts for 8.8% of the original data set. We now move next to extract code-signing certificates from signed PE files.

Extracting Certificates. We utilize the Python PE module[2] to locate the PKCS #7 SignedData structure that contains code-signing certificates and to dump the structure into a file encoded in Distinguished Encoding Rules (DER). Of 525,071 signed PE files, only 495,124 PKCS #7 files are extracted due to parsing errors. Then, we extract all code-signing certificates (including TSA certificates and code-signing intermediate certificates) from the DER-encoded PKCS #7 files, and then we filter out non-leaf code signing certificates using the keyword of "CodeSigning" in the *extendedKeyUsage* extension field and the "Basic Constraints" field.

Valid Korean Certificates. The last part is where we obtain the set of Korean certificates that are valid. We can specify the certificates belonging to a particular country by looking at the country code in the leaf certificate's subject field. If the country code of a leaf certificate is "KR," we know that the certificate is issued for a Korean publisher. Using this concise but effective method, we identify 844 certificates issued to Korean identities and 8,815 malicious PE files signed with those certificates. The number of PE files accounts for 1.8% of the initial data set.

Now we explain the verification process. Only valid certificates remain after this step. We first verify the digital signatures and code-signing certificates embedded in PE files using both SignTool[3] and SigCheck[4] tools in the Windows Sever 2016. SignTool returns error code with a message for the scanned certificate. Table 4 enumerates the error code returned by SignTool and the associated messages. We consider the three messages of "*Successfully Verified*," "*0x800B0101*," and "*0x800B010C*" valid since the two error code, "*0x800B0101*" and "*0x800B010C*" are returned only when PE samples have been properly signed. Specifically, *0x800B0101* returns when a PE file has not been trust-timestamped and its certificate expires, and when a certificate is revoked, *0x800B 010C* returns. Detailed information is described in Table 4. In the end, we have 783 valid Korean certificates and 8,093 PE files signed with the valid Korean certificates as described in Table 5(left). In other words, 94.2% of signed samples have a proper PKCS#7 structure.

3.3 Binary Labeling

Samples from VirusShare may contain false positives. Here, we describe the line of efforts we made to reduce the false positives. First, we re-scan the malicious

[2] https://github.com/erocarrera/pefile.
[3] https://docs.microsoft.com/en-us/windows/desktop/seccrypto/signtool.
[4] https://docs.microsoft.com/en-us/sysinternals/downloads/sigcheck.

samples using VirusTotal. It is known that AV engines' labels may change over time as more evidence is gathered. Thus, some samples may be re-labeled as benign. We observe 234 PE samples among the malicious samples signed with Korean code-signing certificates are no longer malicious after the re-scan.

Next, we set a threshold to filter out samples with less confidence. For each signed PE sample, we define c_{mal} as the number of AV engines in VirusTotal that label the sample as malware. We consider a signed PE file as malware when $c_{mal} \geq 10$. For example, $c_{mal} \geq 10$ means that about 15% of AV engines (out of more than 60 AV engines in VirusTotal) detect the samples as malware. This approach is presented in prior works [13,19]. After this step, we now have 455 valid Korean certificates used to sign malicious samples detected by more than 10 AV engines in VirusTotal.

As a final step, we utilize a malware labeling tool, called AVClass [30] to label our malicious samples and classify them into malware and Potential Unwanted Program (PUP).

4 Code-Signing PKI Abuse in Korea

Table 5(right) summarizes the breakdown of PE malicious files, signed PE malicious files, Korean compromised certificates, and Korean malicious PE files signed with the Korean certificates. With this data, we investigate the characteristics of code-signing PKI and the abuse within Korea. Here, we try to answer the following research questions.

1. *Q1: How prevalent is code-signing abuse in Korea?*
2. *Q2. Who abuses the code-signing in Korea?*
3. *Q3: Who issued the certificate?*
4. *Q4: Are the certificates issued with safe cryptographic guarantees?*
5. *Q5: How long do the abusive certificates survive in Korea?*

The final goal for these questions is to ask the main research question we raised in the introduction: *Q: What characteristics can we find from an analysis of a specific country?*

4.1 Abusers

We answer a couple of questions *Q1. How prevalent is code-signing abuse in Korea?* and *Q2. Who abuses the code-signing in Korea?* in this section. We initiate with simple statistics to answer the first question. For the second question, we investigate the problem from two different angles 1) the malicious sample family based on their labels and 2) the publisher's information stated on the certificate.

Prevalence. As presented in Table 5(right), signed malicious binaries with Korean certificates are 844 in numbers. It accounts for 1.8% of the data set, which is a global collection. The Korean internet population is about 5 million,

Table 1. Top 10 Malware/PUP label breakdown. SINGLETON is labeled when AVCLass is unable to find a family name for a malware sample such as generics. On average, a Korean certificate is used to sign 2.5 different family of malicious samples. Bold malware families are considered as trojan or severe threats.

Family label	PE	Certificate
Kraddare	2,850 (41.70%)	198 (43.52%)
Onescan	798 (11.68%)	50 (10.99%)
SINGLETON*	418 (6.12%)	191 (41.98%)
Sidetab	298 (4.23%)	5 (1.10%)
Hotclip	177 (2.59%)	6 (1.32%)
Openshopper	169 (2.47%)	7 (1.54%)
Delf	158 (2.31%)	32 (7.03%)
Viruscure	243 (3.23%)	36 (6.79%)
Adkor	135 (1.98%)	63 (13.85%)
Hebogo	121 (1.77%)	7 (1.54%)
Total	6,835 (100%)	1,123 (246.81%)

Table 2. Top 10 common name, issuer, and region breakdown. N/A in region means that neither province nor locality name information are specified.

Common Name	PE	Cert.
cloudweb Inc	1,040 (15.22%)	3 (0.66%)
nbiz Ltd.	702 (10.27%)	5 (1.10%)
UCF	489 (7.15%)	4 (0.88%)
NKsolution Corp.	358 (5.24%)	5 (1.10%)
Akorea	306 (4.48%)	5 (1.10%)
SearchLink Co., Ltd.	263 (3.85%)	3 (0.66%)
TGSM Inc.	194 (2.84%)	5 (1.10%)
JE communication	166 (2.43%)	4 (0.88%)
씨큐미디어	161 (2.36%)	2 (0.44%)
OPEN.co., ltd	158 (2.31%)	4 (0.88%)
Total	6,835 (100%)	455 (100%)

Issuer	Cert.
Thawte	336 (73.85%)
VeriSign	74 (16.26%)
YesSign	19 (4.18%)
eBiz Networks	10 (2.20%)
Symantec	7 (1.54%)
GlobalSign	6 (1.32%)
COMODO	3 (0.66%)
Total	455 (100%)

Region	Cert.
Seoul	267 (58.68%)
Busan	63 (13.85%)
Gyeonggi-do	63 (13.85%)
N/A	20 (4.40%)
Incheon	11 (2.42%)
Gyeongsangbuk-do	7 (1.54%)
Daegu	7 (1.54%)
Gyeongsangnam-do	6 (1.32%)
Jeollanam-do	2 (0.44%)
Ulsan	2 (0.44%)
Total	455 (100%)

which occupies about 1% of all internet users worldwide[5]. Compared to its population, code-signing abuse is quite prevalent in Korea. It may imply that Koreans tend to be vulnerable to code-signing abuse and attackers are exploiting it. Such a tendency might have been formed due to its web usability environment, as mentioned in Subsect. 2.4.

Malicious Sample Family. To better understand what kind of malware family used Korean code-signing certificates, we utilize the VirusTotal reports of our collected malicious samples and AVClass [30] to label the samples. We identify 278 different malicious sample families, and we break down the top ten malware

[5] Internet world stats: https://www.internetworldstats.com/stats3.htm.

and PUP labels as described in Table 1. The SINGLETON label indicates that AVClass is unable to classify malicious samples.

About 41% malicious samples signed with Korean certificates is kraddare. This family is considered PUP/PUA, which redirects to unwanted homepages without user action, changes the browser settings, shows unwanted advertisements using pup-ups [11]. Microsoft Defender Antivirus [22] classifies the malicious sample as a "severe" threat and removes the sample as encountered. The following malware family is onescan. The family is considered as a "severe" threat by Windows Defender Antivirus, and called "fakeAV [23]." The malware pretends to scan victims' computers, and reports to them that their computers have been infected by any malware, and then asks them to pay for cleaning up the reported malware. However, the victims' computers are not infected by any malware, and nothing is actually done by the malware, but victims pay for it. Delf [21] is a trojan that redirects Web traffic, downloads malicious programs, etc. On average, a Korean certificate is misused to sign 2.5 different families of malware/PUP samples. It would imply either 1) a couple of malware groups share a code-signing certificate to sign their malware or 2) a malware group produces a couple of malware families. However, we have little evidence to specify which.

Publisher. In Windows, when executing/installing a signed PE file, a client is prompted a request that shows the publisher name of the PE file by the system. Only after the client accepts the request, the signed PE files will be executed. Details about the publishers' information are available when clients look at the certificates since certificates include publishers' information such as the company/individual name, physical address (country, province, street address, and zip code), etc.

We start the investigation from the publisher's name stated in the Common Name (OID: 2.5.4.3) field. The common name is a required field in Subject of the X.509 v3 standard. It is used to identify the legal name of a publisher. The Legal names can be specified in the field only when verified by CAs using notarized documents or legal documents from attorneys. Unlike TLS, where the common name should have a domain name to be verified, in the code-signing PKI, the common name is usually an organization's name such as Google Inc. and Microsoft Corporation. We observe 330 common names in 455 Korean certificates; on average, a company has 1.4 different code-signing certificates to sign malicious samples. The top 10 publishers are enumerated in Table 2. cloudweb Inc has the largest signed malicious samples in our data set. The publisher had three different certificates to sign 1,040 malicious samples.

Furthermore, we could find some reputable Korean companies within the certificates misused to sign malicious samples. We believe that their private keys associated with the certificates were likely stolen and used to sign malicious samples. For example, the certificate of a Korean software company that develops not only software tools but also an AV product was misused to sign malware, called "plugx." The malware is a kind of Remote Access Trojan (RAT). Fortunately, the certificate was explicitly revoked, and the malware is no longer valid. Moreover, an English education company located in the Gangnam district

released a program with a Trojan downloader malware payload. The malware was distributed at a legitimate website. Since it is a reputable and legitimate company, we believe that the development infrastructures were compromised, and the payload was injected into the legitimate program.

Next, we take a look at where these publishers are located. We use the Province filed (OID: 2.5.4.8) to locate the regions of the publishers. According to the minimum requirements [7], the Province field is required to be specified when the Locality Name field (OID: 2.5.4.7) is absent. However, we observe that 20 certificates issued by YesSign do not include any information in both the Province field and the Locality Name field; YesSign does not obey the requirement. Specifically, YesSign specifies their CA name on the Organization Name filed (OID: 2.5.4.10) rather than the publisher's organization name. Most malicious publishers (58.7%) are located in Seoul as depicted in Table 2. We also manually investigate certificates located in a small, rural, agricultural area where IT companies are less likely to exist. We observe that two certificates located in a small agricultural area are issued to non-existing IT companies. The same name of the IT companies exists, but they are located in Seoul, not the small rural area. Moreover, the two certificates were issued on the same day, and the certificates were misused to sign the same malicious sample families; onescan, kraddare, and jaik. Therefore, we believe that the two publishers are related to each other, even though they use different publisher names. This goes along with our findings from analyzing the malware families.

4.2 Issuer

In this section, we answer the questions: *Q2: Who issued the certificate?* and *Q3: Are the certificates issued with safe cryptographic guarantees?*.

Certificate Authority (CA). CAs issue code-signing certificates to software publishers (e.g., software developers). In the certificates, CAs specify their information such as the country, address, name of the issuer CAs. Similar to the Subject field, the issuer information is located in the Issuer field.

We observe only seven CAs, and certificates issued by Thawte are the majority (73.8%), which contradicts the finding [13] that VeriSign dominates the code signing certificate market share. We believe it is because Thawte is distributed by one of the accredited CAs in Korea, as we described in Subsect. 2.4. Also, Thawte allows publisher names with Korean alphabets[6], which may have boosted the market share. In addition, we find "YesSign[7]" in our data set, a CA which is hardly observed in prior works [13,14]. YesSign is one of the largest Korean CAs, and is operated by Korea Telecommunications and Clearings Institute (KFTC). The CA no longer issues code-signing certificates, but it still provides the OCSP and TSA service. However, as they stopped the business, there is a chance the revocation checking services may shut down in the future, which may make users vulnerable.

[6] Provided by crosscert: https://www.crosscert.com/symantec/02_1_04.jsp.
[7] https://www.yessign.or.kr.

Table 3. Signature and public key algorithm breakdown.

Signature algorithm	Count	Public key algorithm	Count
MD5 With RSA	6 (1.31%)	RSA	455 (100%)
SHA1 With RSA	413 (90.77%)	DSA	0 (0%)
SHA256 With RSA	36 (7.91%)	ECDSA	0 (0%)
Total	455 (100%)	Total	455 (100%)

Cryptography Algorithm. It is important to use strong cryptography algorithms for the certificates. Certificates with a weak algorithm may be utilized for collision attacks. It is critical in code-signing as an attacker could perform collision attacks on time-stamped binary samples with weak algorithms. MD5 and SHA1 are weak hash algorithms, vulnerable to collision attacks. We have observed a severe security threat where Flame malware exploited an unknown chosen prefix collision attack against the MD5 hash algorithm [31]. Google and CWI Amsterdam demonstrated that two different files could have the same SHA1 hash [12]. Although the SHA1 collision attack against certificates is not yet reported, it could be exploited to create fake certificates in the near future. Therefore, Microsoft deprecates MD5 and SHA1 hash algorithms in 2013 and 2015, respectively [20,26]. Still, CAs should be aware of this fact and move on to SHA256.

We examine what cryptography algorithms are used for signature and public keys in Korean certificates. As depicted in Table 3, all certificates in our data set use RSA for public key generation. For the signature algorithm, the majority (around 91%) use SHA1. We can also see the use of MD5 in a few certificates (6, 1.31%). It implies that weak algorithms are still prevalent in Korea, which has the potential to lead to serious security problems.

4.3 Certificate Life-Cycle

A life cycle of a certificate starts from its issue date and ends at its expiration date. In case it is compromised, a revocation is conducted to invalidate the certificate. However, we know that some signed binaries may survive even after their expiration and revocation due to the trusted timestamp. To answer the next research question *Q4: How long do the abusive certificates survive in Korea?*, we start the examination from the validity period of the Korean certificates. Then we check how prevalent trust timestamping is among the signed malicious binaries. In the end, we investigate if the revocation is performed effectively for those certificates, invalidating all the signed malware.

Validity Period. Each certificate has two fields, notBefore and notAfter for validity period; notBefore is an issue date and notAfter is an expiration date. In other words, a certificate is only valid between notBefore and notAfter, inclusive. As shown in Fig. 2, most certificates (69.43%) were issued between 2009 and 2012. It does not indicate that the signed malware was collected between the

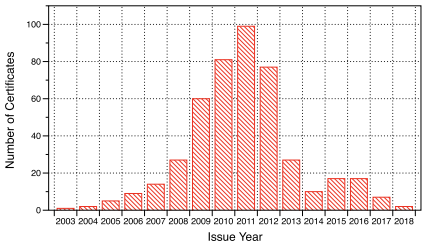

Fig. 2. Issue year. Around 70% of certificates in our data set were issued between 2009 and 2012.

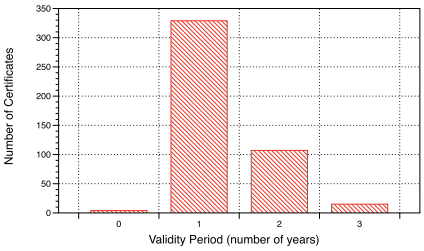

Fig. 3. Validity year. The majority is one-year-valid certificates since CAs usually issue one-year-valid certificates.

periods because the signed samples are still being valid even seven or eight years have passed due to the trust timestamping.

Figure 3 shows that most certificates (70.57%) are only valid for one year as expected since CAs typically issue one-year-valid code signing certificates. However, interestingly, the validity period of four certificates is less than one year. Two certificates issued by Thawte are valid only for three and nine months; one certificate done by VeriSign is valid only for 11 months, and a certificate issued by YesSign is valid for four months. Unlike the TLS certificates, because the code signing PKI has the trust timestamping, signed binary samples can be valid even after their certificate expiration date as long as the samples are trust timestamped. Therefore, expiration dates do not count as much as the Web's PKI. We do not observe that certificates are valid for more than three years.

Trust Timestamp (Signing Date). The distinct difference between the Web's PKI and the code signing PKI is the trust timestamping mechanism (c.f., Sect. 2.2). We measure how many Korean malicious samples are trust-timestamped. Of 8,815 Korean malicious signed PE samples, we observe that 6,190 samples (70.2%) are trust-timestamped. Only when we consider the valid malicious samples ($c_{mal} \geq 10$), 4,625 samples (67.7%, out of 6,835) contain the signing date (trusted timestamps). It means that most malicious samples use trust timestamping to extend their validation period beyond their certificate's expiration date. We also examine when the malicious samples are signed; we utilize issue dates and expiration dates. More than 50% of malicious samples are signed about 200 days before their expiration dates, as shown in Fig. 4. It indicates that most malicious samples are consistently signed with compromised certificates during the validity periods.

Revocation Status. All certificates we have identified in the paper are misused to sign malicious binary samples. Therefore, they should be revoked. We check whether or not the certificates in our data set are revoked using CRLs. We observed three security threats that let signed malware alive. First, only 31 (6.81%) of 455 Korean certificates are explicitly revoked. It implies that mal-

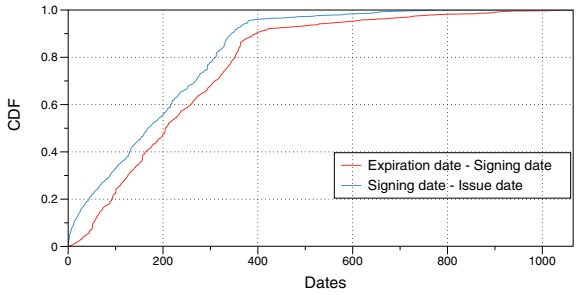

Fig. 4. The difference in days between signing dates and issue dates, and between expiration dates and signing dates.

ware signed with not-revoked certificates may remain valid even after the certificate's expiration date, due to the trust timestamping mechanism (c.f., Sect. 2.2).

Second, we encountered CRLs that are unreachable. Two reasons interfered with accessing and fetching the CRLs; (1) the CRL domain was taken by a domain re-seller, and (2) the CRL file was moved/removed, returning a 404 error. Those findings are in line with prior work [14]. Clients exposed to these certificates become vulnerable as they cannot check the revocation status of these certificates.

Lastly, several certificates were not effectively revoked. Signed malware can continue to be valid, although its certificate is revoked if the revocation date is set erroneously (c.f., Sect. 2.3). We measure if Korean signed malicious samples are still valid as CAs erroneously set the revocation dates after the samples' signing dates. We find that 321 malicious samples are still valid, and eight Korean certificates are used to sign the samples. The average difference between the signing date and the revocation date is 6,013.01 h (250.51 days); the shortest difference is 11.04 h (0.46 days), and the longest one is 25,389.82 h (1,057.91 days, 2.9 years).

Although the certificates are mostly issued to be valid for a year, several signed malware remain a threat for an extended period due to time-stamping and the clumsy set of revocation dates.

5 Related Work

Compared to the Web's PKI, little research has been conducted on the code signing PKI. The first attempt [27] was done in 2010 by *F-Secure*. In the attempt, they introduced the ways of abusing Microsoft Authenticode [25]. However, the work was presentation slides focusing on introducing new threat models rather than a research paper. In 2015, Kotzias et al. [17] examined 356,000 digitally signed samples collected between 2006 and 2015. They observed that most of the collected signed samples were Potentially Unwanted Programs (PUP), while signed malware was relatively uncommon in their corpus. Kim et al. presented new threat models that highlight the breaches of trust in the code signing PKI. Kim et al. also identified the security problems of the revocation mechanisms cur-

rently deployed in the wild. However, those studies are conducted from a global perspective while we measure Korean compromised certificates' characteristics.

6 Conclusion

We investigate the characteristics of code-signing abuse in South Korea. We design a system that extracts abusive Korean code-signing certificates with a simple but effective method. A couple of findings were related to its unique web environment fostered by regulations. South Korea has its own government-accredited CAs, and these CAs affect the certificate landscape. We observe Thawte, re-selled by one of the accredited CAs, dominating the population. Another accredited CA even acted as a code-signing CA. However, the CA is no longer in business, which is a potential threat as they might stop the revocation service. We observed that code-signing abuse is quite prevalent in Korea, and it might be due to the exposure of mandatory installation for using the web. Besides, we also found a common vulnerability reported in prior works. Only 6.8% certificates have been revoked, and eight certificates of them have erroneous effective revocation dates, which extends the validity of signed malicious samples.

Acknowledgements. We thank the anonymous referees for their constructive feedback. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2021R1F 1A1049822). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsor.

A Appendix

Table 4. SignTool error code & message.

Validation	Error code	Message
Valid	N/A	Successfully verified
	0x800B0101	Expired certificates
	0x800B010C	Revoked
Invalid	0x800B010A	Not a trusted root CA
	0x80096010	Signature does not match the file
	Terminated in a root cert	Not trusted by the trust provider
	No signature found	No signature found

Table 5. Breakdowns. Error code of Korean malicious PE files (left), PE files and certificates (right).

Validation	Error code	KR malware
Valid	Successfully Verified	5,714
	0x800B0101	558
	0x800B010C	1,821
	Valid total number	8,093
Invalid	0x800B010A	405
	0x80096010	94
	Terminated in a root cert.	24
	No signature found	199
	Invalid total number	722
Total number		8,815

Type	PE & Cert.	Number
Total	All malicious sample	5,934,399
	PE file	3,240,176
	Signed PE file	525,071
	PKCS #7	495,124
Korean	Malicious signed PE	8,815
	Malicious cert.	844
	Valid malicious signed PE	8,093
	Valid malicious cert.	783
	Valid malicious signed PE ($c_{mal} \geq 10$)	6,835
	Valid malicious cert. ($c_{mal} \geq 10$)	455

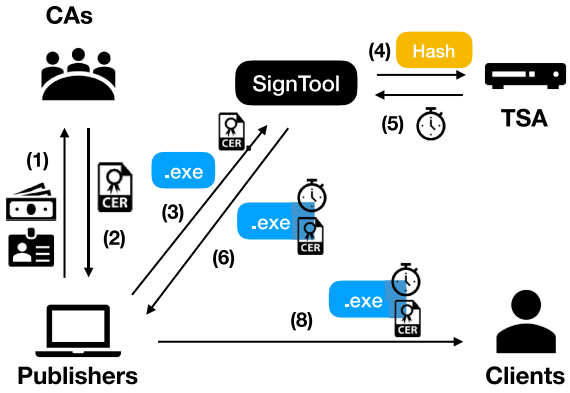

Fig. 5. Code-signing process. (1) A publisher applies for a code-signing certificate to a code-signing CA with her/his identifications such as government-issued photo IDs, (2) After vetting, the CA issues a code-signing certificate to the publisher, (3) Using the `SignTool` (a signing tool provided by Microsoft), the software publisher signs a binary sample with the certificate, (4) when a TimeStamp Authority (TSA) is specified for timestamping (c.f., Sect. 2.2), the signing tool sends the hash value of the binary sample to the TSA server, (5) The TSA server issues the timestamp and signs the timestamp with the TSA's private key, and send them back to the signing tool, (6) The signing tool finally embeds the code-signing and the TSA certificate chain, the digital signature, and the timestamp into the binary sample, and (7) Finally, the publisher distributes the signed binary sample in the wild.

References

1. What should i do with the annoying ads? (in Korean) https://www.donga.com/news/Economy/article/all/20140914/66399483/1. Accessed 03 Sept 2020
2. N. Korea fakes 'code signing' to spread spyware. KBS world radio. http://world.kbs.co.kr/service/news_view.htm?lang=e&Seq_Code=119375. Accessed 30 Aug 2020

3. To bypass code-signing checks, malware gang steals lots of certificates. ars technica. https://arstechnica.com/information-technology/2016/03/to-bypass-code-signing-checks-malware-gang-steals-lots-of-certificates/. Accessed 30 Aug 2020
4. Adobe. Electronic Signature Laws and Regulations - South Korea (2020). https://helpx.adobe.com/sign/using/legality-south-korea.html
5. Alrawi, O., Mohaisen, A.: Chains of distrust: towards understanding certificates used for signing malicious applications. In: WWW 2016, Republic and Canton of Geneva, Switzerland (2016)
6. Chai, S.-W., Min, K.-S., Lee, J.-H.: A study of issues about accredited certification methods in Korea. Int. J. Secur. Appl. **9**(3), 77–84 (2015)
7. Code Signing Working Group. Minimum requirements for the issuance and management of publicly-trusted code signing certificates. Technical report (2016)
8. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., 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
9. Durumeric, Z., Wustrow, E., Halderman, J.A.: ZMap: fast internet-wide scanning and its security applications. In: Proceedings of the 22Nd USENIX Conference on Security, SEC 2013, Berkeley, CA, USA, pp. 605–620. USENIX Association (2013)
10. Falliere, N., O'Murchu, L., Chien, E.: W32.Stuxnet dossier. Symantec Whitepaper (February 2011)
11. Geater, J.: How to remove Kraddare. https://www.solvusoft.com/en/malware/potentially-unwanted-application/kraddare/
12. Google: Announcing the first SHA1 collision (February 2017)
13. Kim, D., Kwon, B. J., Dumitras, 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, CCS 2017 (2017)
14. Kim, D., Kwon, B.J., Kozák, K., Gates, C., Dumitras, T.: The broken shield: measuring revocation effectiveness in the windows code-signing PKI. In: 27th USENIX Security Symposium, USENIX Security 2018. USENIX Association (2018)
15. KLRI: Digital Signature Act, 2017. https://elaw.klri.re.kr/eng_service/lawView.do?hseq=42625&lang=ENG
16. Kotzias, P., Bilge, L., Caballero, J.: Measuring PUP prevalence and pup distribution through pay-per-install services. In: Proceedings of the USENIX Security Symposium (2016)
17. Kotzias, P., Matic, S., Rivera, R., Caballero, J.: Certified PUP: abuse in authenticode code signing. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS 2015. ACM, New York (2015)
18. Kozák, K., Kwon, B.J., Kim, D., Gates, C., 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)
19. Kwon, B.J., Srinivas, V., Deshpande, A., 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 (2017)
20. Microsoft: Microsoft security advisory: update for deprecation of MD5 hashing algorithm for Microsoft root certificate program, 13 August 2013
21. Microsoft: Trojan:win32/delf. https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Trojan:Win32/Delf
22. Microsoft: Trojan:win32/kraddare. https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Trojan:Win32/Kraddare

23. Microsoft: Win32/onescan. https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?name=win32%2Fonescan
24. Microsoft: Erroneous VeriSign-issued Digital Certificates Pose Spoofing Hazard (2001)
25. Microsoft: Windows Authenticode portable executable signature format (March 2008). http://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/Authenticode_PE.docx
26. Morowczynski, M.: SHA-1 deprecation and changing the root CA's hash algorithm (2018)
27. Niemela, J.: It's Signed, therefore it's Clean, right? (2010)
28. NLIC: Electronic Financial Transaction Act, 2017. http://www.law.go.kr/eng/engLsSc.do?menuId=1&query=electronic+financial+transactions+act&x=0&y=0#liBgcolor0
29. Park, H.M.: The web accessibility crisis of the Korea's electronic government: fatal consequences of the digital signature law and public key certificate. In: 2012 45th Hawaii International Conference on System Sciences, pp. 2319–2328. IEEE (2012)
30. Sebastián, M., Rivera, R., Kotzias, P., Caballero, J.: AVclass: a tool for massive malware labeling. In: Monrose, F., Dacier, M., Blanc, G., Garcia-Alfaro, J. (eds.) RAID 2016. LNCS, vol. 9854, pp. 230–253. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45719-2_11
31. Swiat: Flame malware collision attack explained (June 2012)
32. Wood, M.: Want my autograph? The use and abuse of digital signatures by malware. In: Virus Bulletin Conference, September 2010, pp. 1–8 (September 2010)
33. Yilek, S., Rescorla, E., Shacham, H., Enright, B., 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, IMC 2009. ACM (2009)