Chains of Distrust: Towards Understanding Certificates Used for Signing Malicious Applications

Omar Alrawi
QCRI - HBKU
oalrawi@qf.org.qa

Aziz Mohaisen
SUNY Buffalo
mohaisen@buffalo.edu

ABSTRACT

Digital certificates are key component of trust used by many operating systems. Modern operating systems implement a form of digital signature verification for various applications, including kernel driver installation, software execution, etc. Digital signatures rely on digital certificates that authenticate the signature, which then verify the validity of a given signature for a signed binary.

Malware attempts to subvert the chain of trust through several techniques to achieve execution, evasion, and persistence. In this paper, we examine a large corpus of malware (3.3 million samples) to extract digital signatures and their corresponding certificates. We examine several characteristics of the digital certificates to study features in the process of malware authorship that will potentially be used for characterizing and classifying malware. We look at many features including the certificate's chain length, the issue and expiration year, the validity duration of a certificate, the issuing country, validity, top issuing certificate authorities (CAs), and others, highlighting potentially discriminatory features.

Keywords

Malware; Certificates; Authenticode

1. INTRODUCTION

A digital signature is a method to validate digital content like software, data, and messages. They rely on digital certificates which are virtual documents that bind the ownership of a signing key to a real world entity. This virtual document is commonly used in public key infrastructure to establish a base of trust. There are several well known root certificate authorities, or CAs, that are used as the trusted entities where all publishers relay their trustworthiness back to one of the root CAs. Root CAs verify a publisher's real entity and issue a certificate stating that a given publisher is trustworthy based on the criteria set by the CA. Once a publisher has a certificate linking them back to a trusted root CA, the publisher can then sign digital content using digital signatures and attach their corresponding certificate to the signature for the consumer to verify [11].

The main purpose of this system is to establish accountability for digital content published to consumers. The consumer, before executing the software, consuming data, or reading a message, will check whether the publisher is indeed the publisher of the digital content before consuming it. The certificate attached to the signed digital content serves as means for the consumer to verify the trustworthiness of a certificate all the way up to a trusted root CA. There can be more than one intermediate CA between the publisher and the root CA. This certificate verification method is known as the chain of trust. Each certificate is chained to a trusted certificate ending up at the root CA or a common trusted CA between the publisher and the consumer [8].

Digital signatures are widely used by software publishers for signing software [9]. In this work, we focus specifically on Windows binaries, and how certificates are used for them. Windows operating systems implement a form of digital signature verification known as Authenticode [16]. The purpose of this technology is twofolds: to know the origin of the code to be executed and to verify that the code has not been modified. The implementation of digital signatures can have vulnerabilities that some malware authors take advantage of to bypass protection within an operating systems. One such vulnerability is MS13-098 [15], in which attackers can exploit signed software to run malicious code via stack space after the embedded signature. This allows the attacker to sign a non malicious software and then attach malicious functionalities to a later version via an update. The binary would still remain valid since the function WinVerifyTrust, a Microsoft Windows API function used to verify the signed content with a digital certificate, will only consider content before the signature. This vulnerability has been fixed and no longer works on patched systems. Other types of attacks include certificates compromise, where attackers steal a valid certificate and use in high-profile attacks such as Flame [21].

In this work we examine a large corpus of malware and their corresponding digital certificates. We then examined several features in the certificates to identify trends within malicious signed code. Our study took one months of data, with about 3.3 million samples, and extracted certificates from signed malware, of about 800k samples. In each certificate, we examined features like the size of the trust chain, the distribution of the issuing year, the validity duration of the certificate, country code of the subject certificate, the common name used, and the number of CAs within a chain, and highlight various interesting insights. In this work, our contribution is the following. First, we process a large corpus of malware and their certificates and give access to the metadata via an online portal [6]. Second, we give high-level statistics for the features found in the digital signatures. Third, we share our cleaned and processed malware corpus with the community for future empirical studies

Organization. In the rest of this section we describe our our data and system. In section 2 we describe the features captured from our processing phase and high-light the importance of each feature.

WWW 2016 Companion, April 11-15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4143-1/16/04.
http://dx.doi.org/10.1145/2872518.2888610.

In section 3 we discuss the high-level statistics collected from the feature and elaborate on our findings. In section 4, we highlight related work and in section 5 we provide a discussion. In section 6, we provide concluding remarks.

1.1 Data

For our data, we collected one months worth of malware samples from commercial feed, which is captured from two main regions, the Americas and Europe/Asia. The capturing process for the malware samples is proprietary and for our experiment we used both regions for the malware samples. We collected the malware samples over one months period (July 2015) and processed it through our system. The total Windows binaries collected after the initial filtering was roughly 3.3 million malware samples.

The 3.3 million samples were fed into our processing scripts to extract and decode the authenticode signature format. Our processing program is written in Python and it utilizes libraries like `pefile` and `asn1`. We first use the pefile module to load the binary. If we encounter any errors we discard the sample and move on to the next. We then look for the PKCS#7 [10] signature in the "IMAGE_DIRECTORY_ ENTRY_SECURITY" found under the PE directories entry, if it exists. If nothing was found, we again discard the sample and move on. Then we parse the X509 [7] content to extract the certificate chain along with all the content of each certificate in the chain. The resulting information is represented as a JSON document that we write to file. After running this process on our data we had about 800K samples with digital certificates, which are summarized in Table 1.

Table 1: Dataset breakdown

Category	Count	Percentage
All Samples	3,331,161	100%
w/ certificates	804,834	24.16%
w/ certificates & AV labels	794,753	23.86 %

The resulting JSON document is then augmented with VirusTotal anti-virus detections. For each sample containing digital signature we query VirusTotal for the anti-virus scan results. If results are found from VirusTotal, we augment the scans to the certificates' JSON file. We filter further based on VirusTotal's results to only samples that have 5 or more anti-virus detections. This filtering is required to exclude false positive files that might not be malware but have been captured by our feed provider. The resulting dataset after applying the enrichment phase and further filtering contains about 790K samples.

1.2 System Overview

In this section we will describe the system built for this work and give an overview of the architecture along with the subcomponents of our system. We describe the three major components of the system, which are backend processing, signature verification tools, and front-end web interface. Within each components there are several subcomponents that supports the overall functionality. Our system is comprised of two virtual machines running Ubuntu 14.04 and Windows 7. The Ubuntu server houses the front-end and the back-end components of the system, while the Windows 7 virtual machine houses the signature verification tools. We used a third machine as network file system (NFS) storage between the Ubuntu and Windows 7 machine to pass malware samples. We did not consider it in our system, since we used a generic NFS storage, which is trivial to mount and use. We depict our system in Figure 1.

Our back-end component is comprised of the malware feed downloader/filterer, signature/certificate extractor, data enrichment (via VirusTotal), and the NoSQL database storage. The order of operation is as follows. (1) Malware samples are downloaded from our feed provider. (2) The feed file is extracted and filtered for Windows binaries. (3) The resulting samples are fed into the certificate extractor. (4) The certificate extractor queries the signtool.exe[17] API on the Windows 7 machine to check the validity of signature. (5) The certificate extractor queries the VirusTotal database, if the above two steps are successful. (6) The results from the above process is persisted to a NoSQL database. (7) A nightly batch process is run to collect basic counts about the extracted data and the results are persisted to another database document collection.

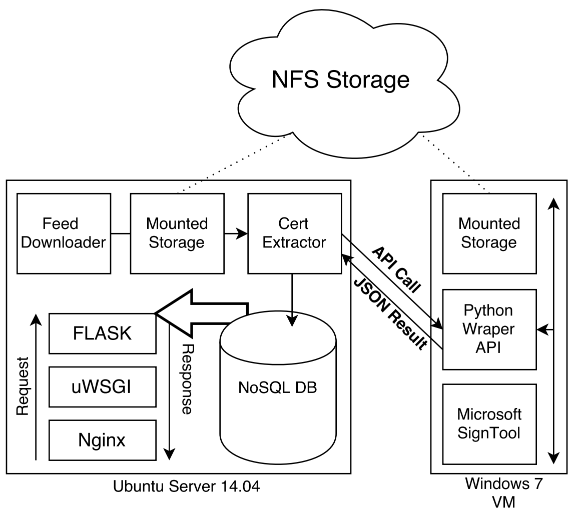

Figure 1: System architecture.

The Windows 7 virtual machine (VM) is a single purpose serving to validate Windows binaries using the signtool.exe found in the Windows Software Development Kit (SDK). There are several approaches to validate Windows binaries, including using the WINE [5] framework to validate on a Linux machine or using OpenSSL [4] with several modifications. The major shortcoming of those techniques is that they do not support 64-bit binaries. We built a Python program to wrap the signtool.exe functionality and provide a web API for our certificate extractor application.

The front-end of our system is a simple python web application built using FLASK Framework [2]. The purpose of the front-end is to provide a capability to explore and query the data stored in our database. We are planning to add more search and analytical features to our online portal to allow researchers to explore, analyze, and visualize the data. The python web application uses a generic stack with nginx as the webserver, uWSGI as the middleware gateway interface for python, and Flask as the web application development framework.

2. FEATURES

In this section we describe the features captured from our processing phase and highlight the importance of each discriminatory feature. We limit our feature study to the certificate chain and its content. The ultimate goal of this exploration of features is to highlight their trends, and show their relevance as discriminatory features for identifying the malware samples. Future work will include static and dynamic features from binary files that can be further used in attribution. In addition to the features found directly in the chain, like the country code and common name of the subject, we included few features that are derived from explicit fields in the certificate's content. For example, the validity duration of a certificate is derived from the issue and expiry dates found in each certificate.

Table 2: Classification of large, medium, and small CAs

Size Catagory	Certificate Authority Names
Large CAs	comodo/utn-userfirst, globalsign, go daddy, microsoft, symantec/verisign/thawte
Medium CAs	certum, digicert, startcom, entrust
Small CAs	addtrust, trustasia, starfield, wosign

2.1 Valid Signature

We consider the validity of the signature in each binary as an important feature. Having a self-signed binary may pass the technical validity, but it is considered invalid since the chain of signers does not extend to one of the well-known trusted CAs, shown in Table 2. By default, operating systems come with a list of trusted CAs mostly in the large and medium size CA categories. If the application is signed by a publisher whose certificate is issued by a trusted CA, then the operating system will consider it valid, assuming it is not revoked. As mentioned earlier, we use a standalone tool found in the Windows Software Development Kit to check the validity of each file. This feature is noted in the final JSON document.

2.2 Issue, Expiry, and Validity Duration

The issue date is the date the CA or the publisher—if self-signed—issued the certificate. The expiry date is the date set by the issuer of the certificate to indicate that the certificate is after which is no longer valid. Signed applications by an expired certificate will be validated if the application is timestamped by a CA indicating that the application indeed was signed during the validity period. The validity duration is the time between the issue date and the expiry date. This feature provides us with the average life of a certificate for a given publisher. The duration can be extended using a timestamp signature extension. The duration granularity is in years.

2.3 Chain Length, CA Count, and TSA Count

The chain length refers to how many certificates are in the certificate store found in the digital signature. For example, if a binary is signed by publisher whose certificate is issued by Symantec, then we will have two certificates in the X509 store, one for the publisher and one for Symantec, where the publisher's certificate chains up to the Symantec CA since they issued or signed the publisher's certificate. The CA count is done by counting the number of CA certificates found in the certificate chain. Many CAs issue intermediary certificate to distribute and manage their signature and issuing process. We generate a list of known CAs from across the web and use their name as a keyword to filter if the subject of the certificate is one of the CA keywords found in our list. If the subject common name or organization name is on the list of CA we count it as a CA for that particular binary.

For timestamp authority (TSA) we follow a similar process. We generate a list of known TSAs from across the web and we check if the common name or the organization name of the subject field found in a certificate belongs to the entities in our list, then we count it as a TSA for that particular binary. These fields can give us insight into how malware authors run a signature heist to get a valid signature on their malicious binary.

2.4 Issuing CAs, Country, and Common Name

The issuing CA is the CA that issued the publisher's certificate which the latter used to sign the software, binding the publisher's identity to the software. We count this feature for the malware dataset to understand the ranking of the most abused CA systems. For this count, we only considered the subset of samples that were validated with a valid signature using the Window signtool.exe.

The country code is a field found in each certificate. When extracting this field we discard the TS and CA certificates and focused only on the publisher's certificate. The aggregate numbers are found in Table 5. We note that this field can be factious leaving the verification burden on the certificate issuer; in this case the CA. This field is important because it gives an indicator on where the authors might be located and if the field is populated by an incorrect country code, then we can use this information to further understand what makes a particular country code more attractive.

The common name field is found in each certificate and can be set to an empty string. Although most CAs would consider it as a red flag if left unpopulated. In our study we did encounter several empty string common names but they were insignificant. The field existed in the certificate but was empty and the binary did not have a valid signature. This field can link different publishers distributing various kinds of malware and help in attribution.

There are other fields found in the certificate like the email, physical address, organization name, individual name, locality name, domain name qualifier, and others. We did not consider those fields because they are not found in the majority of our dataset and we leave it for future work to explore their trends.

3. RESULTS

In this section we discuss the high-level counts collected from the features discussed in the pervious section and elaborate on our findings. We did a simple count of all the explicit features found in the certificate and the derived features through several processes. Derived features are issuing CA count, signature and certificate chain validity, TS and CA counts, etc.

3.1 Chain Depth Characteristics

In Figure 2 we have several histograms showing the count for some of the explicit and derived features. Starting with the top figure, we plotted a histogram of the number of TS certificates found in each binary. As depicted in this figure, the majority of our samples do not have a TS certificate associated with them. The ones that do, have two on average which represents a signing TS certificate and the root TS certificate. The root TS certificate is usually the trusted authority or the TSA.

The next histogram shows a plot of the number of CA certificates found in each binary. Note that we plotted the histogram for all the dataset, including the binaries with invalid signatures. We see that 18% of our samples seem to be self-signed as they have no CA certificate. The majority of our samples have at least one CA certificate. We had few binary files with four and five CA certificates totaling 53 binaries. All of the binaries with five CA certificates were invalid where the binaries with four CA certificates had 11 binaries with valid signatures. The 11 binaries with valid signatures had anti-virus labels for adware and potential unwanted programs (PUP). The countries for those files were mostly China and the US.

The histogram plot for the chain length is in line with the number of CAs found in the earlier figure. We observe just over 150k binary samples having zero CA certificates and a depth of one, indicating they are self-signed. The mean number of the binary samples have a depth of two or three again aligning well with the figures observed for the number of CAs. Also, just under 50% of the samples have four or more chain depth, which aligns with the number of TS and CA certificate counts found earlier. We expect one certificate for the publisher, one or two for the CA, and one or two for the TSA summing up to four or more certificates in the chain. We see a few outlier programs that have 9 certificates in their chain attributing their CA to VeriSign Inc., but the signatures on these binaries are

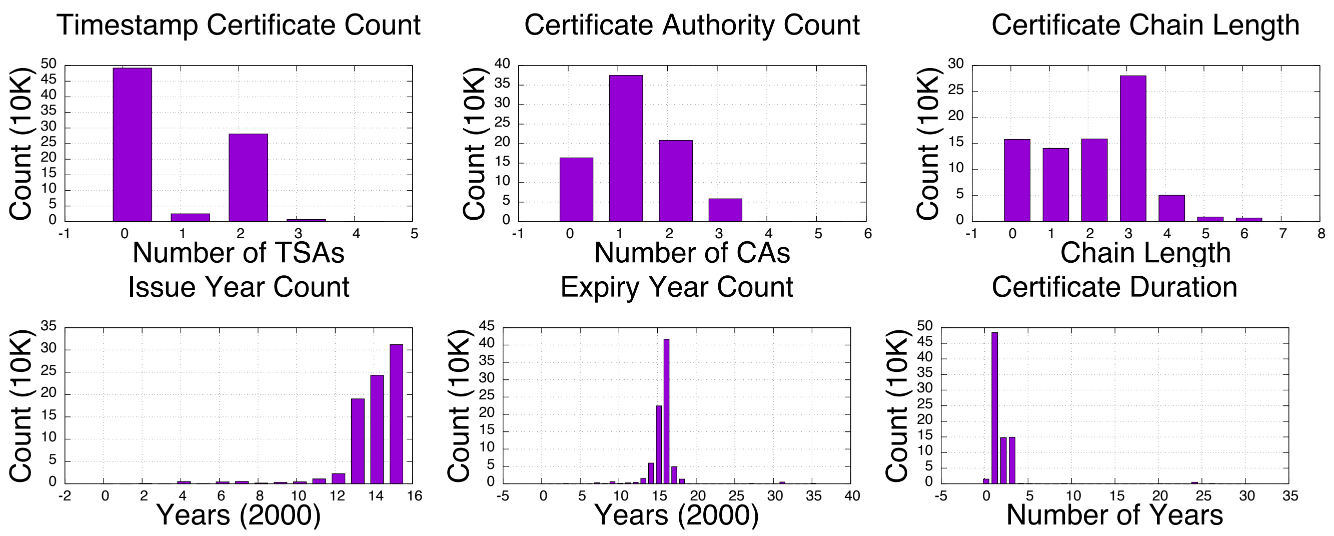

Figure 2: High-level counts for certificate features.

corrupt which causes the signtool.exe to consider the files to have no signatures.

3.2 Certificate dates

The next aspect we analyzed was the issue year, expiry year, and the duration of the validity of the certificate. We observed, as shown in the histogram, that the issue year clusters around 2013, 2014, and 2015 and the expiry dates cluster around 2015 and 2016. This observation aligns correctly with the certificate duration plot, which shows the majority of the certificates having one year validity period. We see that there are a few outliers where the expiry year is between 2028 and 2040. We observe the same outliers in the duration plot having 30 or more years of validity.

Table 3: The output of signtool.exe for all of the 800K malware samples used in this study.

Count	%	Error Code	Error Message
1	0.0%	0x80096002	The certificate for the signer of the message is invalid or not found.
2	0.0%		The signing certificate is not valid for the requested usage.
50	0.0%	0x80096005	The timestamp signature and/or certificate could not be verified or is malformed.
55	0.0%	0x80096003	One of the counter signatures was invalid.
189	0.0%	0x80096004	The signature of the certificate cannot be verified.
1,405	0.18%		A certificate chain processed, but terminated in a root
2,153	0.27%	0x800B010A	A certificate chain could not be built to a trusted root authority.
3,767	0.47%	0x80096010	The digital signature of the object did not verify.
42,548	5.36%		No signature found.
145,273	18.32%	0x800B0101	A required certificate is not within its validity period when verifying against the current system clock or the timestamp in the signed file.
206,322	26.02%	0x800B010C	A certificate was explicitly revoked by its issuer.
403,069	50.82%		valid

3.3 Signature and Certificate Validity

In Table 3 we showcase the results of the signtool with the error message mapping. Just over 50% of our dataset has a valid signature, which is interesting since we expected the validity of the signatures to be lower for malicious software. Nonetheless, our results aligns with pervious work [12] with slight skewness due to the difference in data sources and period of observation. From the total binaries in dataset, 26% of them were revoked by the issuer due to several reasons like compromise, abuse, or violation of terms of the issuer. The second largest category of invalid signatures is related to the date validity of the certificate. Our Windows 7 VM was set to the current date and had access to the Internet syncing the system clock with Microsoft's time server. Our timezone was set to Eastern Standard Time, which leaves us to conclude that this category of certificates are expired

The third largest category for invalid signatures processed by the signtool reports no signature found. The tool did not find a signature on the binary but our tool parsed out certificates store found in the binaries. The majority of these binaries are Trojans with fake certificate appended to the PE file. When observed under a Windows system, one can clearly see there are certificates attached to the binary, but they have common names of large and trusted software publishers like Adobe, Mozilla, Microsoft, 360.cn, Oracle, Realtek, Intel, Nvidia, and Apple, among others. This type of category is the most interesting to study since the majority of the files are malware files containing fake certificates to be used as a decoy for an inexperienced system user.

3.4 Issuer, Origin, and Common Names

Digging deeper into the binaries with valid signatures we categorize the top issuing CAs for our dataset as in Table 4. The top issuing CA is Symantec because they own VeriSign CA [1] and Thawte CA, which both account for approximately 53% of the valid certificates. Trailing Symantec in certificate issuing is Comodo, which covers just over 30% of the valid certificate subset. These two CAs are the biggest market shareholders [23] and as classified by Table 2. Note that this finding is alarming: both of those CAs are popular, and those findings are just a counterexample for reducing by relying on popular CAs.

The country codes found in each certificate are sorted in Table 5. We observe the US, Israel, Spain, Ukraine, Russia, and Ireland account for most of the certificate country codes. As mentioned earlier, these country codes can be incorrect and the issuing CA is responsible to verify this information. We assume the information found in a given certificate is incorrect if the binary does not have a valid signature and certificate chain. However, we note that those findings are in line with recent findings on understanding the sources of malicious attacks [24].

Finally, Table 6 enumerates the top 20 most occurring common

Table 4: Dataset breakdown

CA Name	Count	Percentage
TrustAsia	103	0.03%
WoSign	1040	0.26%
Entrust	1271	0.32%
Go Daddy	3306	0.82%
Starfield	3511	0.87 %
Symantec	6142	1.53%
Microsoft	12805	3.18%
DigiCert	13086	3.24%
Certum	29145	7.7%
GlobalSign	31038	6.46%
AddTrust	49005	12.16%
Thawte	85221	21.14%
VeriSign	120139	29.81%
COMODO	123811	30.72 %
Total	403,069	100%

Table 5: Dataset breakdown

Country Name	Count	Percentage
Seychelles	1437	0.18%
Barbados	1515	0.19%
Germany	3293	0.42%
Panama	3776	0.48%
United Kingdom	5631	0.71%
Belarus	5924	0.75%
Poland	6113	0.77%
Romania	6530	0.82%
France	6896	0,87%
Canada	9929	1.25%
Dominica	10279	1.30%
Cyprus	15585	1.97%
China	23835	3.01%
Ireland	53197	6.71%
Russia	59450	7.50%
Ukraine	107783	13.59%
Spain	125866	15.87%
Israel	150533	18.98%
United States	195512	24.65%
Total	793084	100%

names observed in our study. We notice the majority of the common names found in the dataset belong to individual names or organizations that brand themselves as advertisement companies. Studying the anti-virus labels for the top 20 common name we found they are all classified as adware or potentially unwanted programs (PUP). We did not find any malware labels in the top 20 most common names observed. We found that the majority of these companies have multiple names and operate under various entities, but distribute similar adware or PUP to generate revenue.

4. RELATED WORK

Starting in 2010, the first known malware certificate measurement study was done in [22]. The work is a white paper highlighting several aspects of the authenticode process and how a malicious publisher can abuse it. Another work in this area is in [26], where the author discusses the complexity of code signing and verification outlining several ways a signature scheme can be subverted by a malicious author. The work also provides some statistics about the number of malware's that have exploited those vulnerabilities.

Ladikov [13] looks at threats associated with signed files and how malware can take on such role. He provides an overview of the signing process and shows that malware authors are capable of exploit the weaknesses and trust in the signature process. Finally, concurrent to our effort Kotzias et al. [12] did a thorough analysis of certificate abuse for potentially unwanted programs (PUPs)

Table 6: Top 20 common names

Common Name	Count	Percentage
daniel hareuveni	95,496	12.04 %
pigater system	31,517	3.97%
investena concept	23,137	2.92%
stepan rybin	13,281	1.67%
ivan kostin	10,857	1.37%
astori llc	10,299	1.3%
digital plugin sl	10,073	1.27%
give away software	9,923	1.25%
apps market abc	9,768	1.23%
popeler system	9,748	1.23%
app secure llc	9,537	1.20 %
artur kozak	8,041	1.01%
digit network	7,659	0.97 %
smart secure software s.l.	7,494	0.94 %
global apps roi	7,078	0.89%
kaydar llc	6,983	0.88%
dmitry banak	6,859	0.86%
proekt	6,676	0.84%
air software	6,654	0.84%
normands	6,555	0.83%
total	793,084	100%

and malware based on 365K malware files collected during 2006 – 2015, where the authors look at several features within the certificate chain, the signature, and static artifacts of the malware to classify and cluster the samples.

The result of their system is to generate a blacklist of certificates that are associated with malware and that are still valid. We believe this work is the closest to our work and we differ in the following points. 1) Our goal is to provide high-level statistics of malware certificate along with the release of data for researchers. 2) Our portal contains a continuos feed of data that is based on the most recent captured malware. 3) We provide the malware samples, which can be shared with verified researchers from the community. 4) Finally, we aggregate the high-level counts overtime as more data is fed into our system to understand the change in certificate abuse. We believe this will foster a collaborative environment and provide clean data to researchers who are interested in this area of research.

5. DISCUSSION

In this section we discuss some of our findings and speculate on trends found in our frequency analysis.

5.1 Features

We break down our data into two categories, malware with valid signatures and ones without. The malware that contains a signatures but it is not valid is shown in Table 3. The table highlights the error messages accordingly for each category. We turn the readers attention to the category with error message âĂIJNo signature found,âĂİ which we found to be the most interesting. Although, this category of malware contains no signature, a user observing the certificate store will see a list of certificate that look like they belong to a large trusted authority like Microsoft, Apple, Mozilla, etc.

We speculate this is used to trick users in believing that the software is distributed by a trusted authority. Signature verification is prone to failure due to skewed system clock, old trusted certificates, or corrupt file due to network transfer or external factors. Hence a user is maybe more lenient with the signature verification process believing that an error occurred for the aforementioned reasons.

There are several features observed in Figure 2, which give an overall trend of what the majority of malware samples features look like. For example, the number of TSAs for the majority of samples is around zero, which indicate that malware authors understand the digital signature system and use an opportunistic approach to obtain a valid signature on their malware. TSAs are used to extend the validity of digital good by vouching that the digital artifact was signed during a valid period of the issued certificate. Hence, malware authors do not use TSAs much because their certificates are revoked or blacklisted. This speculation can be verified by the issue, expiry, and duration of a certificate used to sign malware.

5.2 Empirical methods

We used simple frequency analysis to look at the data from a high-level to understand the population and find anomalies that can be analyzed in details. Our approach aims to identify areas of interests where we can do a further inspection on subsets of the data to find insights. For example we identified 45k samples that seem to not have a signatures associated with the binary, but contained certificates.

Our approach also aims to identify techniques and methods used by malware authors to abuse the signature and certificate system. By looking for anomalies that might be apparent in frequency analysis, we can apply a more rigorous and in-depth approach to find open vulnerabilities in the current CA signing process. For example, [12] shows that malware authors can abuse the time-stamping feature in the signature process to keep their malware valid even if revocation takes place.

The advantage of our approach is that it is simple, easy to observe, and gives a representation of the malware population (ab)use of signing process. The disadvantages of our frequency approach is that we miss finer details that are blinded by majority of the population. We believe a hybrid analysis technique can help us find many unique cases within the malware population that are still open to abuse by malware authors.

In [12] the authors use a hybrid approach, but focus on the financial aspect of a malware campaign. They calculate the cost based on the cost of obtaining a signature with wide spread of the campaign and how they related to each other. This approach is an in-depth analysis of the financial dimension to the certificate and signature problem. They also attempt to generate a blacklist based on similar software.

6. CONCLUSION

In this study, we processed over 3 million malware samples to understand their use of certificates. We gave high-level overview for some features found in the certificates used by malware, and finally we built a portal to query the data and provide an interface to explore the dataset. We found more than 50% of our malware samples that had anti-virus detection contained a valid digital signature and the majority were branded as adware or PUPs. We observed a unique set of malware that contains no binary signature but has a certificate attached to the malware to decoy itself as a software from prominent publishers like Apple and Adobe.

We will continue our collection and processing of the malware samples and add more functionality to our online portal to allow users to explore, analyze, and visualize the data. We plan to do in depth study to explore how we can use the certificate features to attribute malware to authors and identify polymorphic malware samples through this process, building on our prior work in [25, 19, 18, 14, 20]. Our data will be published and is available for the research community to further study this area. We want to provide a data platform to enable in-depth empirical research of malware abuse of digital signatures and certificates. With the rise of initiatives like Let's Encrypt [3], we believe there will be interesting trends in the area of malware and digital certificates.

7. REFERENCES

[1] —. VeriSign's Sale of Authentication Services Business to Symantec Closes. symc.ly/1QFHyXb, 2010.

[2] —. Flask: web development, one drop at a time. http://flask.pocoo.org/, 2016.

[3] —. Let's Encrypt Initiative. https://letsencrypt.org/, 2016.

[4] —. OpenSSL. https://www.openssl.org/, 2016.

[5] —. WINE HQ. bit.ly/20o6QRF, 2016.

[6] O. Alrawi and A. Mohaisen. MalCert: Latest Malware Samples with Certificates. http://malcert.localhost.qa/, 2016.

[7] Cooper et al. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, 2008.

[8] J.-w. Jang, H. Kang, J. Woo, A. Mohaisen, and H. K. Kim. Andro-autopsy: Anti-malware system based on similarity matching of malware and malware creator-centric information. *Elsevier Digital Investigation Journal*, 2015.

[9] J.-w. Jang, H. Kang, J. Woo, A. Mohaisen, and H. K. Kim. Andro-dumpsys: anti-malware system based on the similarity of malware creator and malware centric information. *Computers & Security*, 2016.

[10] B. Kaliski. PKCS #7: Cryptographic Message Syntax. RFC 2315, 1998.

[11] H. J. Kang, J.-w. Jang, A. Mohaisen, and H. K. Kim. Androtracker: Creator information based android malware classification system. In *WISA*, 2014.

[12] P. Kotzias, S. Matic, R. Rivera, and J. Caballero. Certified PUP: Abuse in Authenticode Code Signing. In *ACM CCS*, 2015.

[13] A. Ladikov. Why You Shouldn't Completely Trust Files Signed with Digital Certificates. http://bit.ly/1Myc7hh, 2015.

[14] H. Mekky, A. Mohaisen, and Z.-L. Zhang. Separation of benign and malicious network events for accurate malware family classification. In *IEEE CNS*, 2015.

[15] Microsoft Corp. Vulnerability in Windows Could Allow Remote Code Execution. Microsoft Security Bulletin, 2014.

[16] Microsoft Corp. Authenticode (Windows). http://bit.ly/23HT6QU, 2016.

[17] Microsoft Corp. SignTool (Windows). http://bit.ly/1GbEH1o, 2016.

[18] A. Mohaisen. Towards automatic and lightweight detection and classification of malicious web contents. In *IEEE HotWeb*, 2015.

[19] A. Mohaisen and O. Alrawi. Amal: High-fidelity, behavior-based automated malware analysis and classification. *Elsevier Computers & Security*, 2015.

[20] A. Mohaisen, A. G. West, A. Mankin, and O. Alrawi. Chatter: Classifying malware families using system event ordering. In *IEEE Conference on Communications and Network Security, CNS 2014*, pages 283–291, 2014.

[21] MSRC Team. Microsoft releases Security Advisory – 2718704. bit.ly/1NOPReP, 2012.

[22] J. Niemela. It's signed, therefore it's clean, right? CARO Workshop, 2010.

[23] w3techs – Web Technology Surveys. Usage of SSL certificate authorities for websites. bit.ly/1zePzg8, 2016.

[24] A. Wang, A. Mohaisen, W. Chang, and S. Chen. Delving into Internet DDoS Attacks by Botnets: Characterization and Analysis. In *IEEE DSN*, 2015.

[25] A. West and A. Mohaisen. Metadata-driven threat classification of network endpoints appearing in malware. In *DIMVA*, 2014.

[26] M. Wood. 'want my autograph?': the use and abuse of digital signatures by malware. In *Virus Bulletin*, 2010.