Exploring the Security Issues of Trusted CA Certificate Management

Yanduo Fu[1,2,3], Qiongxiao Wang[1,2(✉)], Jingqiang Lin[4,5], Aozhuo Sun[1,2], and Linli Lu[1,3]

[1] State Key Laboratory of Information Security, Institute of Information Engineering, CAS,Beijing 100089, China
wangqiongxiao@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100089, China
[3] Data Assurance and Communication Security Research Center, CAS, Beijing, China
[4] School of Cyber Security, University of Science and Technology of China, Hefei 230027, Anhui, China
[5] Beijing Institute, University of Science and Technology of China, Beijing, China

Abstract. Public Key Infrastructure (PKI) is widely used in security protocols, and the root certification authority (CA) plays a role as the trust anchor of PKI. However, as researches show, not all root CAs are trustworthy and malicious CAs might issue fraudulent certificates, which can cause Man-in-the-Middle attacks and eavesdropping attacks. Besides, massive CAs and CA certificates make it hard for users to manage the CA certificates by themselves. Though PKI applications generally provide the implementation of trusted CA certificate management (called CA manager in this paper) to store, manage, and verify CA certificates, security incidents still exist, and a malicious CA certificate can damage the entire security. This work explores the security issues of CA managers for three popular operating systems and eight applications installed on them. We make a systematic analysis of the CA managers, such as the modification of the certificate trust list, the source of trust, and the security check of the CA certificates, and propose the functionalities that a CA manager should have. Our work shows that all CA managers we analyzed have security issues, e.g., silent addition of CA certificates, inefficient validation on CA certificates, which will result in insecure CA certificates being falsely trusted. We also make some suggestions on the security enhancement for CA managers.

Keywords: Certification authority · Public key infrastructure · CA certificate management

1 Introduction

Public key infrastructure (PKI) plays a critical role in secure networking, offering security functionalities such as confidentiality, data integrity, and authen-

D. Gao et al. (Eds.): ICICS 2021, LNCS 12918, pp. 384–401, 2021.
https://doi.org/10.1007/978-3-030-86890-1_22

tication. Well-known applications based on PKI include but are not limited to TLS, HTTPS, code signing, eSign documents, S/MIME, and OpenID Connect. Research [15] shows that as of March 2020, more than 60% of the top one million websites in *Alexa* have used HTTPS, and the number of PKI deployments is still growing.

The security of a PKI system is based on a secure and trustworthy root certification authority (CA). Authoritative organizations such as WebTrust [30] will audit public CAs to ensure their legitimacy and security. However, not all CAs (including public CAs and unaudited self-built CAs) are trustworthy. Recent researches and events have shown that due to malicious attacks or misbehaviors, CAs might issue fraudulent certificates [10,12], which may cause Man-in-the-Middle (MITM) attacks or phishing attacks to the users. The common way to realize a PKI system is that users need to choose which CA can be trusted. A user can trust and accept an end-entity certificate based on the trust of the root CA. While if a root CA is not trusted by the verifier, any certificate issued by this CA cannot be trusted.

Since there are a large number of CA companies and the quantity of root *CA certificates* is much larger, users even with professional knowledge cannot identify the trustiness of each CA certificate by themselves [6,7,9]. According to statistics from *Censys* [21], there have been more than 68 million self-signed CA certificates, and 88% are unexpired. Many applications, such as Windows, macOS, Firefox, and Acrobat, have integrated with *trusted CA certificate management* to help the users with certificate verification and storage. Some of them also offer preset Certificate Trust List (CTL), which usually contains the globally trusted root CA certificates and some platform/application specified root CA certificates. When a root CA certificate is added to the *local CTL*, all the certificates issued by it can be accepted.

Previous researches have disclosed the security incidents caused by the improperly *preset CTL*. *Lenovo* shipped some laptops with a pre-installed traffic scanning software called *Superfish* [3] in 2014, which installed a CA certificate and then actually injected advertisements to even encrypted web pages through a MITM attack. The same private key across laptops made things worse as a third-party entity could interpret or modify encrypted traffic without triggering any security warnings. Besides the preset CTL, if a trusted CA certificate can be added to local CTL arbitrarily, the security problem mentioned above still exists [1,5,8,32].

In this paper, we focus on the implementation of trusted CA certificate management integrated with operating systems (OSs) and applications to explore the security issues of trusted CA certificate management. For simplicity, we refer to the implementation of trusted CA certificate management as a *"CA manager,"* which focuses on the local management of CA certificates such as the modification of the CTL, the storage of local certificate files, and the security check of CA certificates. Our work is accomplished by exploring the CA managers on three OSs (i.e., Windows, macOS, and Linux) and eight applications (i.e., seven Web Browsers and Adobe Acrobat) installed on these OSs, which are carefully chosen

according to the market share and security features. We find that all the CA managers we studied have some security issues. For example, we find that there is no requirement for explicit participation of the user (e.g., inputting a password) while importing a CA certificate to the local CTL of some CA managers (e.g., Windows), which means malicious CA certificates can be installed silently and the attackers can even accomplish the addition by replacing the CTL files with a file including malicious CA certificates. Besides, some CA managers have a weak verification on the certificate security and key usage (or certificate purpose) of the CA certificate, and some CA managers (e.g., Firefox) do not verify the complete certificate chain when trusting an intermediate CA certificate.

The main contributions are the followings:

1) We investigate the most popular desktop OSs and PKI applications (e.g., web browsers) for studying the implementation and the use of CA managers. We propose the functionalities a CA manager should have.
2) We conduct a comprehensive test on different CA managers from the perspective of the source of trust, modification to the local CTL, control of the certificate purpose, and security check of CA certificates. Several security issues are disclosed and described in this paper.
3) Based on the security issues we found, we make some suggestions to enhance the security of the CA managers.

The rest of this paper is organized as follows. Related work is introduced in Sect. 2. In Sect. 3 we explore the CA managers in the wild, including the source of trust and the functionalities. Section 4 specifies our method for analyzing the CA managers and reports the security issues we found. We give our suggestions in Sect. 5 and offer the conclusions of the paper in Sect. 6.

2 Related Work

In recent years, CA managers have been analyzed extensively. Some researches have exposed many security issues of the CA managers, and several organizations also develop a *root store* (or *root program*), which contains the preset CTL. In the meantime, there have been some schemes proposed for specific situations.

Schemes for CA Manager. Many well-known organizations (such as Microsoft [25], Mozilla [27], Apple [16], and Adobe [14]) maintain their CTLs and root stores for global users. A CA intending to be included in these lists must comply with the baseline requirements of the CA/Browser Forum [19,20]. Besides, it should be audited by European Telecommunications Standards Institute (ETSI) [22] or WebTrust [30] to ensure its security and legality. These CA certificates will be installed by default in the OSs and applications and are considered secure and trustworthy enough to be used to provide users with security services.

Besides, due to the arbitrary addition of CA certificates and the lack of security knowledge of users, some researchers have introduced some new technologies

for improvements based on the current CA managers. Li et al. [11] propose a locally-centralized CA certificate management solution named *vCertGuard* in private cloud environments combined with desktop virtualization technology characteristics that realize centralized CA management by a professional administrator based on the granularity of trust. CA-TMS [4] proposes a CA reputation evaluation system based on a computational trust model, which determines the number of trusted CA certificates and decision-making rules by learning the browser's certificate-related parameters and the users' behavior habits.

Terrible Situations of CA Manager. Some researches have revealed the terrible situations of the current CA managers. Perl et al. [13] surveyed the local CTL of eleven OSs or applications, finding that 34% CAs in the root store were not used to sign HTTPS server certificates and can be removed to reduce the attack interface without triggering any security warnings in browsers. Vallina-Rodriguez et al. [32] analyzed the CTLs of thousands of Android devices installed by hardware vendors, mobile operators, and Android OS, finding that the Android CAs have no distinction between trust levels and no restrictions on the purpose of the certificate and the rooted applications might install malicious CA certificates without any barriers. Besides, it was found that some manufacturers [2,3] would pre-install some insecure CA certificates, which caused severe security problems. Malicious CA certificates can also be imported by audio drivers [31], antivirus and parental-control software [5], and programming [1].

These works have made a huge contribution to the security of CA managers. However, there is no study analyzing the specific implementation of current CA managers, and we focus on this part, exploring the security issues of CA managers.

3 CA Manager in the Wild

This section specifies the studied target objects and then elaborates their actual situation in the wild from two perspectives: source of trust and functionality of the CA manager.

3.1 Target Object

Our research is based on the most mainstream desktop OSs, including Windows, Linux, and macOS. The specific version is Windows 10 Professional, Ubuntu 16.04 with Linux kernel 4.15.0, and macOS High Sierra 10.13.6.

OS. The OS generally maintains a *system-level* local CTL shared by all accounts in the same machine, and each account maintains a *user-level* local CTL, which will not affect other users in this machine. We treat them as two different CTLs for analysis in the following. Note that two system-level local CTL exists in macOS. One CTL stores the system-level CA certificates which can be modified at any time by users, and the other CTL stores the original CA certificates maintained by Apple, which is referred to as *"root-level" CTL* in this

paper. For Ubuntu, it stores the system-level CTL in the system directory (i.e., */usr/share/ca-certificates/*), and the CTL, which is stored in the *.pki* folder under the user directory, is considered as user-level CTL in this paper.

Application. We investigate several PKI applications containing seven desktop browsers and one PDF reader named Adobe Acrobat DC (Acrobat). The seven browsers consist of the top six desktop browsers (i.e., Chrome, Safari, Edge, Firefox, Opera, and IE), which are ranked by market share in *StatCounter* [29] during the last 12 months, and the Tor browser (Tor), which is known for its open-source, anonymity and privacy technologies. Table 1 shows the specific version of applications. It is worth noting that two different installation methods in Ubuntu need to be considered: (*a*) the *apt*, which gets the applications from the Ubuntu repositories; and (*b*) the *snap*, which is cross-platform, dependency-free, and more secure than *apt*.

We also conduct experiments on Windows 7 Ultimate and CentOS 8.3 with Linux kernel 4.18.0. Windows 7 has similar experiment phenomena with Windows 10. CentOS is different from Ubuntu in the addition and storage location of the system-level CTL. CentOS stores the CTLs in */usr/share/pki/ca-trust-source* and */etc./pki/ca-trust/source*. Many browsers including Firefox will trust the certificates in the two folders, while the Tor browser and the applications installed by *snap* maintain their own CTLs. Overall, CentOS is similar to Ubuntu in terms of addition, storage security, and verification of certificate security. The following mainly introduces Windows 10, macOS 10.13.6, and Ubuntu 16.04.

Table 1. The version number of applications studied

App \ OS	**Windows**	**macOS**	**Linux**	
			Via ***snap***	Via ***apt***
Chrome	87.0.4280.88	87.0.4280.88	⊘	87.0.4280.88
Safari	⊘	13.1.2	⊘	⊘
Edge	87.0.664.55	87.0.664.60	⊘	89.0.723.0
Firefox	83	83	83	83
Opera	76.0.4017.123	76.0.4017.123	76.0.4017.123	76.0.4017.123
IE	11.630.19041.0	⊘	⊘	⊘
Tor	10.0.15	10.0.15	⊘	10.0.15
Acrobat	2019.021.20061	2019.021.20058	2020.013.20064	⊘

⊘: It indicates that the OS does not support the installation of the application.

Challenge. (***i***) There is no unified standard and specification for the CA manager. Different OS platforms have various policies and methods for managing CTLs, which makes it necessary to employ several unique processing and observation methods. For example, Windows OS manages the local CTL using Group Policy, and we can access and modify the CA certificates in the CTL through the

user interface; while in Ubuntu, each CA certificate is stored as a separate file, and we can only modify the CTL by moving the file. (***ii***) The internal implementation logic of the applications' local CA managers is different and opaque. In order to explore the vulnerabilities of the CA manager, it is necessary to design numerous black box testings. For example, to study the security check of CA certificate and the use of CA certificate purpose, we create a website and a document, generate many CA certificates with insecure fields and special certificate purpose, and conduct various tests on each browser and Acrobat separately. (***iii***) Different CA managers may have various forms of unpredictable phenomena for the same test case. Therefore, many parts of the experiment require manual intervention, making it difficult to automate. For example, importing a CA certificate through the user interface to the user-level CTL requires no password but shows the information about the certificate on Windows, while the system-level CTL requires the administrator's permission without a password and shows no notification about the addition. We have to perform experiments manually on each surveyed object and record the results.

3.2 Exploring the Source of Trust

In a PKI system, OSs and applications generally configure a preset CTL as their source of trust. The preset CTLs of our target objects mainly originate from the four mainstream platforms, namely Microsoft, Mozilla, Apple, and Adobe. Among them, Microsoft and Apple maintain the local CTL by themselves, Mozilla root store is a part of the Network Security Services (NSS) cryptographic library [28], and Adobe manages the Adobe Approved Trust List (AATL) by itself and regards European Union Trusted Lists (EUTL) as a third-party source. The number of CA certificates in each CTL is shown in Table 2, which is obtained on April 2021.

Table 2. The widely acknowledged CTLs

Platform	Quantity	CTL
Microsoft	417	Microsoft Included CA Certificate List [24]
Mozilla	142	Mozilla Included CA Certificate List [26]
Apple	217	Apple Included CA Certificate [17]
Adobe	246	AATL [14] & EUTL [23]

To figure out the local default source of each OS and application, we carry out a systematic exploration of the CA managers. Then, we classify the target objects' management mode of their local CTL into two categories: *Global Level (global-level)* and *Application Level*. Table 3 shows the source of trust and the category of each manager.

Table 3. The source of trust and the category of each CA manager

OS	CA manager		Source of trust				Category
			Microsoft	Apple	Mozilla	Adobe	
Windows	System-level/User-level		√	–	–	–	⋆
	Chrome		√	–	–	–	∓
	Edge		√	–	–	–	∓
	Opera		√	–	–	–	∓
	IE		√	–	–	–	∓
	Firefox		–	–	√	–	◇
	Tor		–	–	√	–	◇
	Acrobat		–	–	–	√	◇
macOS	System-level/User-level		–	√	–	–	⋆
	Chrome		–	√	–	–	∓
	Edge		–	√	–	–	∓
	Safari		–	√	–	–	∓
	Opera		–	√	–	–	∓
	Firefox		–	–	√	–	◇
	Tor		–	–	√	–	◇
	Acrobat		–	–	–	√	◇
Linux	System-level/User-level		–	–	√	–	⋆
	Install via *snap*	Firefox	–	–	√	–	◇
		Opera	–	–	√	–	◇
		Acrobat	–	–	–	√	◇
	Install via *apt*	Chrome	–	–	√	–	∓
		Firefox	–	–	√	–	◇
		Opera	–	–	√	–	∓
		Edge	–	–	√	–	∓
		Tor	–	–	√	–	◇

⋆: Global level. ◇: Application level. ∓: Use global-level CA manager from OS.

Global Level. At the Global Level, CA managers maintain a global-level CTL, including user-level CTL and system-level CTL, and can be modified by any authorized entities. The modification to the global-level CTL will affect all the entities trusting it. Specifically, OSs maintain a local CTL on the computer as a system-level CTL that can be trusted directly by some applications (see Table 3 for details). Besides, some browsers installed via *apt* package in Ubuntu jointly trust a user-level CTL from Mozilla located in the *.pki* folder under the user directory. Note that the applications that trust the global-level CTL actually do not have an independent CA manager, and they use the third-party CA managers directly, which are provided by the user-level CTL of the OS by default, according to our observation.

Application Level. At the Application Level, each CA manager maintains its own CTL, which can only be used and modified by itself. The preset CTL can be customized or originate from mainstream platforms. For instance, every browser installed by *snap* in Ubuntu will occupy a separate folder to store the CTL deriving from Mozilla. Tor, Firefox, and Acrobat manage their CTLs on their own, and they can decide whether to put extra trust in the OS's CTL or not by *preferences*. But the extra trust in Ubuntu is not valid according to our experiments.

3.3 Exploring the Functionalities of CA Manager

Once the CA manager obtains the preset CTL from the source of trust, the CTL can be modified by entities (e.g., users, system, application, etc.) for some usage, as shown in Fig. 1. According to the requirements of the audit agency, such as WebTrust and ETSI, baseline requirements of CA/Browser Forum and our operation, and our observation of each CA manager, we conclude that the CA manager should possess the functionalities including but not limited to the followings.

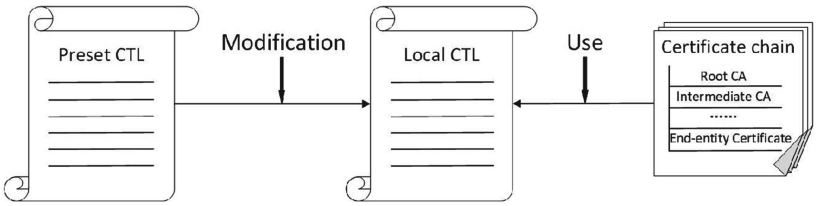

Fig. 1. Overview of the CA manager

Storage Protection. CA manager should store the local CTL and the corresponding trust relationship of these CA certificates in the form of a file locally. The storage of the local CTL should be sufficiently secure, and it should be protected by a security mechanism (such as digital signature and authority management) and not be modified and moved arbitrarily. Moreover, even the file is replaced, the trust relationship of the maliciously replaced CA certificate will not change with the replacement.

Modifying CTL. The CTL can be modified through the user interface and the command line, and users can customize their local CTL. The allowed operations include adding, deleting, and blocking the CA certificate. In particular, the adding operation should require the explicit participation of the user for security, such as inputting the user's password.

Restricting CA Certificate Usage. The usage of CA certificates should be restricted. By default, an added CA certificate cannot be used for any purpose, and users can specify the required CA certificate purpose to be available, which cannot exceed the purposes specified by the key usage and extended key usage (EKU).

Security Audit. Whenever a CA certificate is added or used, it should be checked for security, such as validity period, revocation status, certificate purpose, key size, supported cipher suites, and hash algorithms used. For insecure CA certificates, the CA managers should display a security warning on the user interface.

Isolation. Different accounts (or applications) on the same computer should maintain their local CTLs without affecting each other. However, the OSs maintain a system-level CTL shared by all accounts, which is contrary to isolation. In this case, the system-level CTL should not be modified arbitrarily.

Updating CTL. The CA manager should provide the functionality of updating the local CTL from the source of trust automatically or manually. In this way, the CA manager can obtain the latest CTL timely when the source of trust adds or blocks some CA certificates. This functionality can exist independently or as a part of an overall update of the OS or the applications.

4 Security Analysis on CA Manager

In this section, we conduct a comprehensive security analysis on the CA managers with a systematic and customized experiment method. Our work for the local CA managers mainly concerns the modifications of the local CTL, security checks of CA certificate, restrictions and inspections of CA certificate purposes, and some problems are found during the processes. We elaborate on the disclosed problems and make a detailed analysis separately in the end.

For simplicity, we hide the applications that do not have a CA manager in the following tables. Their behaviors are the same as the user-level CA manager.

4.1 Methodology

Given the diversity of experimental scenarios, our methodology is mainly categorized into two parts, *the black box testing* and *manual alteration of CTL*.

Black Box Testing. It is applied for three purposes, including security checks of CA certificates, inspections of CA certificate purpose, and the verification policy of the certificate chain. To prepare for our experiments, we utilize *OpenSSL 1.1.1* to generate various self-signed root CA certificates with different fields, and create the corresponding three-tier certificate chains. Besides, we build a

website using *Apache* and sign some PDF documents to check the CA managers of browsers and Acrobat.

Among these self-signed root CA certificates with security risks which are used for checking the security of CA certificate, some employ weak hash algorithms such as MD5 and SHA-1, or weak key pairs such as RSA-512, RSA-1024, and ECC-192, and others have disparate certificate purposes (especially the EKU) for the inspections of the key usage. Moreover, expired CAs are also considered. For the sake of eliminating interference factors from other irrelevant aspects, the other certificates in those three-tier certificate chains adopt identical and secure algorithms, specifically, RSA-4096 and SHA-256.

Manual Alteration of CTL. This method places emphasis on the operations of importing new CA certificates to the existing CTLs and replacing the *CTL file* with another file. To achieve our goals, we import our root CA certificates which are secure enough to the local CTL of each CA manager using the user interface or the command line and make them trusted. Meanwhile, we record the required authorities and prompt messages in the process. Besides, we try to replace the CTL file with another file that has the same format and can be identified by the CA manager, observe and record the behaviors of the local CA managers.

4.2 Silent Addition of CA Certificate

Generally, importing a new CA certificate to the local CTL requires the user's knowledge or/and involvement, such as system prompt message or/and password authentication. However, an attacker can maliciously revise the CTL without the user's awareness, such as bypassing the prompt message and adding a CA certificate or directly replacing unprotected files through a malicious program when asking users to install software or drivers [1,5,8]. In our experiment, we adopt the method of manually altering the CTL and find there are several cases demonstrating that a local CTL can be modified silently.

No Password Required for Addition. When adding a new CA certificate to the local CTL and making it trusted, we figure out that the CA managers of Windows, all browsers except Tor, and Acrobat do not require password authentication. Worse still, the system prompt messages can also be bypassed by programs, which has already been disclosed in related research [1].

These issues make it easy for attackers to add malicious or insecure CA certificates to the local CTL arbitrarily in silence. Once a malicious CA is trusted, all certificates issued by it will be used with complete trust, which may result in severe results, such as launching man-in-the-middle attacks, monitoring the behavior of the user's computer, injecting advertisements into the user's computer, or legally installing malicious software. Besides, when we install Alipay's security controls, we also find that the application installed several CA certificates in our Windows without any prompt of certificate information and adding quantity, which may leave the user in a monitored state.

Table 4. Silent addition of CA certificate

OS	CA manager	Vulnerability¶		
		Authority	Password	Storage
Windows	User-level	•	•	⊙
	System-level	○	•	⊙
	Firefox	•	•	⊗
	Tor	∗	∗	⊙
	Acrobat	•	•	⊗
macOS	User-level	○	○	⊙
	System-level	○	○	⊙
	Root-level	∗	∗	⊙
	Firefox	•	•	⊗
	Tor	∗	∗	⊙
	Acrobat	•	•	⊗
Linux	Ca-certificates (system-level)	○	○	⊙
	.pki (user-level)	•	•	⊗
	Firefox (via *snap/apt*)	•	•	⊗
	Opera (via *snap*)	•	•	⊗
	Tor (via *apt*)	∗	∗	⊙
	Acrobat (via *snap*)	•	•	⊗

¶: *Authority* means whether the administrator's authority is required; *Password* means the administrator's password is required; *Storage* means whether the CTL file is protected.
○ means this feature is required, while • means this feature is not required, and ∗ means this CA manager does not support the functionality of addition.
⊙ means the trust relationship or the file itself cannot be replaced, while ⊗ means this trust relationship can be changed by replacing the file.

Malicious Replacement of CTL File. Since the CTL file generally stores many CA certificates and the corresponding trust relationship, it is significantly important to maintain its security. Nevertheless, our experiment reveals that Acrobat and browsers that trust raw CTL of Mozilla, excluding Tor, can replace the existing CTL file with another file coming from a disparate machine, and it only requires the current account's authorities and shows no security warning. Meanwhile, the trust relationship in the new CTL file is taken too, which means the trust relationship in the old CTL file is replaced. Consequently, attackers can use this physical method (which can also be achieved by programming) to add many malicious CA certificates to the local CTL without the user's knowledge, which may pose a more severe threat than adding CA certificates randomly.

Besides, implementations of some current CA managers may cause the consequences of these issues to be more serious, which are the followings:

***a*) Security Issues of Shared CTL.** There are local CTLs shared by several applications on each OS. A malicious CA certificate added to the global-level CTL will be trusted by these applications, which increases the influence scope of risk. Besides, due to the dependence of trust, the applications may not perform any certificate verification but directly trust the global-level CTL, which shows no support for **isolation**.

***b*) Possible Failure in Deleting and Blocking the CA certificates.** The CA certificates in the CA managers of Windows and Acrobat can restore automatically with the update after deletion. Since every CA manager can update its preset CTL, it is difficult for us to delete these CA certificates thoroughly in some CA managers. There are also some managers that do not support the blacklist. For example, Tor does not allow users to make any modifications to the CA manager for security. The system-level CTL of Ubuntu is allowed to cancel the trust of the CA certificates, but this certificate can be added again and trusted. The user may want to cancel the trust of malicious CA certificates, but the automatic recovery mechanism and ineffective blocking make it difficult.

We try to add secure CA certificates and replace files in various OSs and applications, and experiment results are shown in Table 4, including the required authorities during addition and the protection of the CTL file. In summary, we can spot that Acrobat, most browsers, and Windows OS lack a good security mechanism for modifications to CTL. The management methods of Ubuntu and macOS are worth learning.

4.3 Non-strict Security Check of CA Certificate

As required by CA/Browser Forum [19], MD5, SHA-1, RSA with a key size fewer than 2048 bits, and ECC with a key size fewer than 256 bits are viewed as insecure or not recommended. CA managers are expected to check these fields. To observe the security check of each CA manager, we employ the black box testing and use a three-tier certificate chain to do the experiments. We import the insecure root CA certificates to the local CTL and attempt to visit the website using browsers or verify the document signatures using Acrobat, whose end-entity certificate is issued by one of the insecure CAs. We record the behaviors of different CA managers, including warning messages on the user interface and application verification results. Besides, as mentioned in Sect. 4.1, the intermediate CA certificates and end-entity certificates are secure and valid.

To determine the certificate verification policies of these CA managers, only the intermediate CA certificates are added to the CTL and trusted, and we check whether the CA manager verifies the complete certificate chain or not.

For global-level CTLs, we observe the behaviors through the applications trusting their CTL, such as Safari in macOS. The system-level CTL of Ubuntu stores the certificates and trust status but does not provide verification functions. Many command lines such as *wget* and *curl* will trust and use the certificates, and the verification policy mainly depends on the cryptographic libraries such as OpenSSL and NSS. We only display the results of *wget* with OpenSSL. Tor and the root-level CTL of macOS are not considered in this section.

False Trust in CA Certificate with Insecure Fields. During our experiment, it is indicated that only the CA managers of the macOS consider MD5 to be insecure, and ECC-192 is regarded as secure except for the browsers that trust Mozilla's CTL. In addition, SHA-1 and RSA-1024 are thought as secure in all CA managers. Acrobat can verify the signature of any end-entity certificates issued by insecure CA, as long as the CA certificate is imported and trusted. All of these situations should not happen and need to be warned by the CA managers. But only the CA managers of Windows and macOS show the security warnings for the insecure CA certificates in the user interface. Such CA certificates have great potential to be exploited by attackers. For instance, if a CA certificate uses MD5 as the hash algorithm, the attacker already has the ability to forge it with the same hash value. And if a CA certificate encrypts a message with a public key whose length is no longer considered secure, it will be under the risk that the information can be cracked in a limited time. Any of these can pose a great threat to the security of applications, OSs, and users. We reported to Mozilla the issue of not checking the MD5 algorithm in a trusted CA certificate and got a response. They stated that if the certificate was trusted by the user, the security of the digest algorithm would not be verified.

Table 5. Trust status of every CA manager for insecure root CA certificate*

OS	CA manager	SHA-1		MD5		RSA-512		RSA-1024		ECC-192		Expired	
		Inter	Root	Inter	Root	Inter	Root	Inter	Root	Inter	Root	Inter	Root
Windows	User-level[1,2]	⊕	△	⊕	△	⊕	⊕[3]	⊕	△	⊕	△	⊕	⊕[3]
	System-level[1,2]	⊕	△	⊕	△	⊕	⊕[3]	⊕	△	⊕	△	⊕	⊕[3]
	Firefox	△	△	△	△	⊕	⊕	△	△	△	⊕	△	⊕
	Acrobat	△	△	△	△	△	△	△	△	△	△	△	△
macOS	User-level[1,2]	⊕	△	⊕[3]	⊕[3]	⊕	⊕[3]	⊕	△	⊕	△	⊕	⊕[3]
	System-level[1,2]	⊕	△	⊕[3]	⊕[3]	⊕	⊕[3]	⊕	△	⊕	△	⊕	⊕[3]
	Firefox	△	△	△	△	⊕	⊕	△	△	△	⊕	△	⊕
	Acrobat	△	△	△	△	△	△	△	△	△	△	△	△
Linux	Ca-certificates (system-level)[2]	⊕	△	⊕	△	⊕	⊕	⊕	△	⊕	△	⊕	⊕
	.pki (user-level)	△	△	△	△	⊕	⊕	△	△	△	⊕	△	△
	Firefox via *snap/apt*	△	△	△	△	⊕	⊕	△	△	△	⊕	△	⊕
	Opera via *snap*	△	△	△	△	⊕	⊕	△	△	△	⊕	△	△
	Acrobat via *snap*	△	△	△	△	△	△	△	△	△	△	△	△

△: (Intermediate) CA certificate is trusted. ⊕: (Intermediate) CA certificate is not trusted.
∗: **Inter** indicates that only the intermediate CA certificate with RSA-1096 and SHA-256 is added to the CTL and set as trusted. **Root** indicates that the root CA certificate is added to the CTL and form a complete trusted certificate chain.
1: This CA manager has a user interface to show the security warning.
2: This CA manager verifies the complete certificate chain.
3: The user interface shows this root CA certificate is insecure.

Incomplete Verification of Certificate Chain. Due to the different policies of certification path validation between CA managers, some CA managers will not verify the complete certificate chain and stop verifying the rest certificates of the chain when they encounter a trusted CA certificate (not necessarily a root CA certificate). With only the intermediate CA certificates added into the CTL, we retry the above tests and find that CA managers of Acrobat, Firefox, and other applications (except Tor) that trust Mozilla's CTL in Ubuntu do not

verify the complete certificate chain. For example, when we trust an intermediate CA certificate issued by a root CA using ECC-192 or an expired CA (the intermediate CA certificate is issued during the validity of the root CA and is secure enough), the website can still be accessed successfully in Firefox. Since the incomplete verification of certificate chains can cause some insecure CA certificates to bypass the verification, the risk of the non-strict security check is much larger.

The results are displayed in Table 5. It shows the behaviors of various CA managers for different insecure fields in the certificate. In particular, CA managers with user interfaces also show us the trust status of the CA manager. The results of different verification policies are also displayed. In general, we can see that many managers make no warnings about the insecure fields. Insecure CA certificates with weak algorithms weaken the security and may bring undetectable attacks to users, and the risks may be larger if the certificate chain is incompletely validated.

4.4 Potential Abuse of CA Certificate Purpose

The CA certificate usually contains several fields for specifying the certificate purpose, such as *Key Usage* and *Extended Key Usage*, and the certificate should be used for the purpose for which it is intended. CA managers may have their unique methods to process the certificate purpose. For example, as EKU is not required, a CA manager (e.g., Windows) may consider that a certificate without the EKU field has all the certificate purposes, which include the ones to manage the OS and can be abused to tamper with or attack the OS. Furthermore, CA managers (e.g., Firefox) can also provide some commonly used certificate purpose options on its user interface for authorized users to choose manually, such as verifying websites and emails. We will describe in detail later. These purposes may not exist in the certificate purpose field, resulting in inconsistent certificate purposes. To explore the verification of the certificate purpose, we add the self-created CA certificate to each CA manager and observe their behaviors.

Loose Verification on CA Certificate Purpose. Our findings manifest that different OSs vary greatly in verification on certificate purposes of CAs. The certificate purpose selected by the user on the user interface may be inconsistent with the actual key usage of the CA certificate. For macOS, it behaves as if it does not verify the key usage and EKU of the CA certificate after we import the self-created CA with an EKU *Timestamp* only to its CTL. If the key usage of *SSL* is selected in the user interface, we can still visit the website successfully whose end-entity certificate is issued by the CA normally. This issue can lead to the abuse of CA certificates, which may cause some malicious CA certificates to issue fraudulent certificates and have a great impact on the local users. We reported that macOS had the issue of inconsistent certificate purposes to apple and had not received a response yet.

No Restriction on CA Certificate Purpose. For Windows, when the EKU field of a CA certificate is empty, any purposes in the EKU list are selected

and allowed, including some system's functionalities such as *Windows Update* and *Microsoft Trust List Signing*. In our work, we create a CTL using a CA certificate with the corresponding EKU and successfully add it to the local CTL in Windows. Then all CAs in this CTL are directly trusted. Besides, we find that no additional prompt or authentication is required for importing a CA certificate with such a special purpose. Attackers can inject such a CA certificate and a CTL signed by it through programs, which can add a set of CA certificates to the user's computer at one time. The malicious CA certificates with such high-privileged certificate purposes can always pass the verification, which can cause a significant impact on users.

5 Suggestions for Secure CA Manager

Based on the above problems of the CA managers, we put forward the following suggestions for building a more secure CA manager.

User Participation during Modification. When modifying the local CTL, especially when a new CA certificate is imported, the user's participation is necessary, and the need for a password is a recommended way, which can reduce the risk of being tampered with by malicious entities. For example, the system-level CTL and user-level CTL of macOS and system-level CTL of Ubuntu stored in */usr/share/ca-certificates/* need the administrator's password for addition, which is considered secure. Furthermore, it is recommended that displaying an explicit prompt when new CA certificates are imported to the CTL, which can tell the users about the target local CTL and the specific information and the quantity of the CA certificates. Therefore, users can be aware of what has happened and take action to ensure its security.

Meanwhile, deleting or blocking the insecure CA certificates should also be allowed and supported by CA managers so that users can cancel the trust of some certificates permanently. For example, when a user discovers a vulnerability caused by a CA certificate and that the certificate also exists in his computer, he should be able to delete or block the CA certificate.

Enhanced Security of Local Storage. There should be a certain security mechanism in the storage of the local CTL so that the corresponding files should not be moved, copied, or deleted at will. The non-replicability of the trust relationship is a sign of a secure file, too. The even better proposal is that the file could be signed by the current user or OS to ensure its integrity and security. Additionally, we also advise that each application or OS can maintain the CTL by itself instead of sharing it with others since the isolation of different entities can greatly reduce the possibility of being attacked.

Here are some pretty good cases in our research. Each CA certificate in the system-level CTL of Ubuntu is stored in a separate file, and the file replacement has the same requirement as adding, which demands the password of the

administrator. Tor stores its CTL in a dynamic link library file, which can not be modified at all. Besides, the root-level CTL file in macOS is protected by System Integrity Protection (SIP) [18]. As for Windows, the CTL file is occupied since the user logins in, and we can't perform any operations on the file.

Strict Security Check on CA Certificate. Though the verification policy of some CA managers is based on the trust of the user, users may not be able to determine whether to trust a CA certificate, and a strict security check on the CA certificate when importing it or periodically is recommended. There are some fields requiring special attention, such as the validity period, the security of the cryptographic algorithm, and the certificate purpose/key usage. Besides, it is worth noting that those applications that directly trust the OS's list should also strictly check the security of the list when using it.

More importantly, it is necessary to verify the complete certificate chain for applications. In detail, the validity, the hash algorithm, and key size of all certificates in the certificate chain should be verified. Furthermore, we suggest that if the security strength of the upper-level CA certificate in a certificate chain is not stronger than the lower-level CA certificate, then the latter should not be considered secure. Last but not least, updating the CTL from the trusted source timely can mitigate the risk of trusting in CAs which have been deleted or blocked.

Restrictions on Certificate Purpose. We recommend that CA certificates in the CTL provide users with no certificate purpose by default and users can turn on the certificate purposes by selecting them from common certificate purposes such as SSL and S/MIME. Besides, the CA manager should verify whether the selected key usage is consistent with the purpose declared in the certificate. For the key usage related to some system functionalities, additional authentication or completely disabling is recommended.

6 Conclusion

This work has analyzed and reported the security issues of the trusted CA certificate management in current OSs and PKI applications. We explored three OSs and eight applications installed on each OS, focused on the source of trust, the functionalities of the CA managers, the modification to the local CTL, control of the certificate purpose and security check of the CA certificate, and found several security issues which may bring troubles and risks to users. Furthermore, we propose some suggestions for these security problems.

Acknowledgment. We thank all the reviewers and our shepherd for their helpful feedback and advice. This work was partially supported by the National Cyber Security Key Research and Development Program of China (No. 2018YFB0804600).

References

1. Alsaid, A., Mitchell, C.J.: Installing fake root keys in a PC. In: Chadwick, D., Zhao, G. (eds.) EuroPKI 2005. LNCS, vol. 3545, pp. 227–239. Springer, Heidelberg (2005). https://doi.org/10.1007/11533733_16
2. Sloppy Security Software Exposes Dell Laptop to Hackers — Laptop Mag. https://www.laptopmag.com/articles/dell-certificate-security-flaw
3. Superfish - Wikipedia. https://en.wikipedia.org/wiki/Superfish
4. Braun, J., Volk, F., Classen, J., Buchmann, J., Mühlhäuser, M.: CA trust management for the web PKI. J. Comput. Secur. **22**(6), 913–959 (2014)
5. de Carnavalet, X.D.C., Mannan, M.: Killed by proxy: analyzing client-end tls interception software. In: Network and Distributed System Security Symposium (2016)
6. Chung, T., et al.: Measuring and applying invalid ssl certificates: the silent majority. In: IMC (2016)
7. Durumeric, Z., Kasten, J., Bailey, M., Halderman, J.A.: Analysis of the https certificate ecosystem. In: IMC (2013)
8. Durumeric, Z., et al.: The security impact of https interception. In: NDSS (2017)
9. Krombholz, K., Mayer, W., Schmiedecker, M., Weippl, E.: " I have no idea what i'm doing"-on the usability of deploying {HTTPS}. In: 26th {USENIX} Security Symposium ({USENIX} Security 17) (2017)
10. Li, B., et al.: Certificate transparency in the wild: exploring the reliability of monitors. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (2019)
11. Li, B., Lin, J., Wang, Q., Wang, Z., Jing, J.: Locally-centralized certificate validation and its application in desktop virtualization systems. IEEE Trans. Inf. Forensics Secur. **16**, 1380–1395 (2020)
12. Li, B., Wang, W., Meng, L., Lin, J., Liu, X., Wang, C.: Elaphurus: ensemble defense against fraudulent certificates in TLS. In: Liu, Z., Yung, M. (eds.) Inscrypt 2019. LNCS, vol. 12020, pp. 246–259. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-42921-8_14
13. Perl, H., Fahl, S., Smith, M.: You won't be needing these any more: on removing unused certificates from trust stores. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 307–315. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45472-5_20
14. Adobe Approved Trust List members. https://helpx.adobe.com/acrobat/kb/approved-trust-list1.html, Accessed 30 Apr 2021
15. Top 1 Million Analysis - March 2020. https://scotthelme.co.uk/top-1-million-analysis-march-2020/
16. Apple Root Certificate Program. https://www.apple.com/certificateauthority/ca_program.html
17. List of available trusted root certificates in macOS High Sierra. https://support.apple.com/en-us/HT208127, Accessed 11 May 2021
18. About System Integrity Protection on your Mac. https://support.apple.com/en-us/HT204899
19. Certificate Contents for Baseline SSL - CAB Forum. https://cabforum.org/baseline-requirements-certificate-contents/
20. CA/Browser Forum - CAB Forum. https://cabforum.org/
21. Censys. https://censys.io/certificates?q=, Accessed 30 Apr 2021
22. ETSI - Welcome to the World of Standards!. https://www.etsi.org/

23. European Union Trusted Lists. https://helpx.adobe.com/document-cloud/kb/european-union-trust-lists.html, Accessed 30 Apr 2021
24. MICROSOFT Included CA Certificate List. https://ccadb-public.secure.force.com/microsoft/IncludedCACertificateReportForMSFT, Accessed 30 Apr 2021
25. Program Requirements - Microsoft Trusted Root Program. https://docs.microsoft.com/en-us/security/trusted-root/program-requirements
26. MOZILLA Included CA Certificate List. https://ccadb-public.secure.force.com/mozilla/IncludedCACertificateReport, Accessed 30 Apr 2021
27. Mozilla Root Store Policy-Mozilla. https://www.mozilla.org/en-US/about/governance/policies/security-group/certs/policy/, Accessed 1 May 2021
28. Why Does Mozilla Maintain Our Own Root Certificate Store? - Mozilla Security Blog. https://blog.mozilla.org/security/2019/02/14/why-does-mozilla-maintain-our-own-root-certificate-store/
29. StatCounter Global Stats - Browser, OS, Search Engine including Mobile Usage Share. https://gs.statcounter.com/, Accessed 30 Apr 2021
30. Principles and criteria and practitioner guidance. https://www.cpacanada.ca/en/business-and-accounting-resources/audit-and-assurance/overview-of-webtrust-services/principles-and-criteria
31. Root CA Certificate: When you shouldn't trust a trusted root certificate — Malwarebytes Labs. https://blog.malwarebytes.com/security-world/technology/2017/11/when-you-shouldnt-trust-a-trusted-root-certificate/
32. Vallina-Rodriguez, N., Amann, J., Kreibich, C., Weaver, N., Paxson, V.: A tangled mass: the android root certificate stores. In: Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies (2014)