2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) | 978-1-6654-9425-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/TrustCom56396.2022.00045

You Cannot Fully Trust Your Device: An Empirical Study of Client-Side Certificate Validation in WPA2-Enterprise Networks

Li Song*†,Qiongxiao Wang* †, Shijie Jia*†, Jingqiang Lin‡§,Linli Lu*†, Yanduo Fu*†
*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
†School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
‡School of Cyber Security, University of Science and Technology of China, Hefei, China
§Beijing Institute of USTC, Beijing, China
{songli,wangqiongxiao,jiashijie,lulinli,fuyanduo}@iie.ac.cn, linjq@ustc.edu.cn

***Abstract*—WPA2-Enterprise networks offer access to the Internet widely for multifarious client devices. Certificate-based authentication is adopted on the client-side to authenticate the server during network connection. Due to a lack of professional knowledge, client users commonly fully trust the devices, which may result in insecure network connection and user credential leakage. Previous works commonly focus on the security vulnerabilities due to the design weaknesses of the user interfaces from mainstream operating systems, while the built-in certificate validation implementations, which act as a block box for users to validate the received certificates, are not taken into consideration.**

In this paper, we design a series of comprehensive testings to evaluate the built-in certificate validation implementations of mainstream client devices for the first time. Moreover, we investigate the configuration options provided by the devices from different vendors, which may downgrade the security of the certificate validation. We select both Windows and Android (from vendors with the largest five market share) devices as our empirical study target. The results show that more than one security vulnerability exists in the built-in certificate validation implementations of the selected devices, and all the selected devices provide a certain option which may downgrade the security of certificate validation. We also conduct a real Evil Twin attack, which reveals that the user credentials can be cracked due to the discovered security vulnerabilities. Our findings have been responsibly disclosed to the relevant device vendors, and we received an assortment of responses, meanwhile many vendors (e.g., Huawei) have already positively acknowledged our findings.

***Index Terms*—WPA2-Enterprise, Certificate validation, Evil Twin attack**

I. Introduction

Wi-Fi is deployed ubiquitously to provide wireless network services. Peculiarly, the Wi-Fi connection mode defined in the IEEE 802.1X standard [18] (i.e., WPA2-Enterprise mode) has been widely utilized in enterprises, universities and educational institutions. The most well-known WPA2-Enterprise mode application scenario is Eduroam, which is available in thousands of hotspots across over 100 countries worldwide [6].

This work is supported by National Key R&D Program of China (No. 2020YFB1005800). Corresponding author: Qiongxiao Wang and Shijie Jia.

Typical WPA2-Enterprise authentication is carried out in two essential phases (i.e., a handshake phase and a data phase). Specifically, in the handshake phase, the client (e.g., mobile phones, laptops and tablets) utilizes certificate validation to authenticate the authentication server (i.e., authServer) in the TLS negotiation, aiming to establish a secure connection between the client and the target authServer. In the data phase, the authServer validates the client credentials (e.g., username and password) by an inner authentication protocol (e.g., PAP, MSCHAPV2), which is executed inside the TLS tunnel. The validity of certificate validation in the handshake phase and the establishment of TLS tunnel will determine the whole security of the WPA2-Enterprise networks [17].

Certificate-based authentication has been widely adopted, while numerous security vulnerabilities on certificates validation have been proposed in many scenarios, such as HTTP browsers [14], [22], SSL/TLS library APIs [16], SSL/TLS libraries applied in IoT devices [12] and SSL/TLS usage in Android apps [15]. Significantly, the situation is even worse in the case of WPA2-Enterprise networks, this is because if an insecure certificate is accepted by the client, which will open the door for connecting to rogue access points (e.g., Evil Twin attack) [9], [21] and downgrade the security of the entire network. Moreover, the credentials transmitted in the data phase would be obtained by the malicious adversary and the network traffic would be intercepted by rogue access points claiming to be legal. Worse still, users commonly reuse the same credentials for other services (e.g., email) through an existing single sign-on (SSO) service [9].

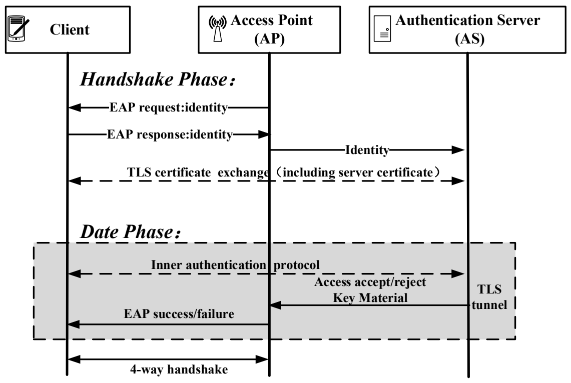
Several works have been proposed to evaluate the security of certificate-based authentication in the WPA2-Enterprise networks, while they commonly focus on discovering and exploiting the design weaknesses of the connection configurations supported by the user interfaces (UIs) on the client-side. Brenza et al. [11] exploited the fact that many Eduroam users have a missing or incorrect CA certificate configuration on the client devices, thus being able to manipulate the network

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on July 11,2024 at 08:25:28 UTC from IEEE Xplore. Restrictions apply.

traffic of the victims by capturing authentication data with an Evil Twin access point. Bartoli et al. [9] pointed out a reality that an option of "do not validate the certificate" is available in almost all operating systems (OSs) to skip certificate validation, and then they focused on discovering leaking credentials based on the above reality. Hue et al. [17] first presented a framework to evaluate the configuration UIs of mainstream OSs based on their achievable configurations on the client-side, then evaluated the TLS parameters used by authServers and discovered perilous practices (e.g., the use of expired certificates, deprecated versions of TLS, weak signature algorithms) on the server side.

Certificate verification is a complicated and multifaceted process, which requires to check all the contents of the corresponding certificate. In the case of WPA2-Enterprise network connection, most verification efforts are taken by the built-in implementations of the client devices as a black box, and only a few connection options are provided by configuration UIs. However, previous works only consider the security vulnerabilities due to the design weaknesses of the UIs from mainstream OSs, while leaving the void of discovering security vulnerabilities from built-in certificate verification implementations of different client devices. Moreover, due to a lack of professional knowledge, client device users commonly fully trust the built-in device implementation and the provided configuration options during WPA2-Enterprise network connection. To the best of our knowledge, neither the security of the client-side built-in opaque certificate validation from different client devices, nor the differences between the multifarious customized configuration options provided by the numerous device vendors are evaluated by the previous work.

In this paper, we tackle the aforementioned limitations and aim to answer the following two questions: 1) *Q1: How the mainstream client devices execute built-in certificate validation under secure configurations?* 2) *Q2: May the configuration options provided by the devices from different vendors downgrade the security of the certificate validation?* To answer the above two questions, we implement an empirical study to evaluate the client-side certificate validation security on different client devices with mainstream OSs from different vendors during WPA2-Enterprise network connection. In more details, we select various Windows devices and Android devices (from five vendors with the largest market share [2]) as the target of our study. Then we first generate a series of secure and insecure certificates and propose a black box testing technique to discover the security vulnerabilities during built-in certificate validation of mainstream client devices with secure configurations. Then we manually configure all the options provided by the device UIs from different vendors to find out all the combinations which may turn the insecure certificates from being rejected to being accepted. The evaluation results show that numerous security vulnerabilities are introduced during the customized certificate validation implementations of all the selected devices, including security vulnerabilities of built-in certificate validation (e.g., ignore weak key pair and weak hash algorithm, insecure trust anchor addition) and security downgrade due to insecure configurations (e.g., do not check certificate by default and bypass hostname check).



Client
Access Point (AP)
Authentication Server (AS)
Handshake Phase:
EAP request:identity
EAP response:identity
Identity
TLS certificate exchange (including server certificate)
Date Phase:
Inner authentication protocol
Access accept/reject Key Material
TLS tunnel
EAP success/failure
4-way handshake

Fig. 1: Authentication in WPA2-Enterprise network on EAP.

Our contributions are summarized in the following:

- We perform the first empirical study to evaluate the built-in certificate validation implementations of numerous client devices with mainstream OSs during WPA2-Enterprise networks connection. We find that all the selected client devices suffer from more than one security vulnerability during built-in certificate validation.
- We investigate the configuration options provided by the devices from different vendors and show that they all provide a certain option which may downgrade the security of certificate validation.
- We implement a real Evil Twin attack by taking advantage of the discovered security vulnerabilities during WPA2-Enterprise networks connection. Proving that user credentials can be cracked by attackers. We propose multiple built-in certificate validation and secure client configuration countermeasures to enhance the security of the certificate validation during WPA2-Enterprise networks connection.

II. Preliminaries

A. Authentication in WPA2-Enterprise network on EAP

The authentication in WPA2-Enterprise networks is carried out following the IEEE 802.1X standard. As shown in Fig. 1, the main components in the WPA2-Enterprise protocol involve a *Client*, an *Access Point* (AP) and an *Authentication Server* (AS) [18]. *Client* is a device (e.g., mobile phone, laptop) that wishes to connect to the network. *AP* is a network device, which communicates with AS and client. *AS* is a server using RADIUS or Diameter protocols, which is responsible for authenticating users by checking received credentials.

The communications among the above three main components make use of Extensible Authentication Protocol (EAP) methods [8], which is a generic authentication framework that does not dictate a particular way of authenticating users. The most widely deployed EAP methods in the wild mainly include EAP-TLS, EAP-TTLS and EAP-PEAP, which all take advantage of certificate validation to authenticate AS during the

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on July 11,2024 at 08:25:28 UTC from IEEE Xplore. Restrictions apply.

TLS negotiation. The tunneled authentication in the WPA2-Enterprise is carried out in two phases. Specifically, the handshake phase executes an outer authentication, where either the AS is authenticated by the client through certificate validation or both the AS and the client are mutual authenticated (e.g., EAP-TLS) following the standard TLS procedures. If the outer authentication succeeds, a cryptographically secure tunnel will be created between the AS and the client for the subsequent data phase. In the data phase, an inner authentication is executed and encapsulated within the tunnel between the client and the AS, using the specific credentials of the users. If the inner authentication succeeds, the AS will send a key material to the AP, which is utilized in the 4-way handshake to generate a session key. The corresponding session key will be used to protect the incoming network traffic.

The importance of certificate validation in ensuring the security of WPA2-Enterprise network cannot be overstated. The well-documented weakness of inner authentication has been proposed. Take MSCHAPV2 [3] as an example, some tools (e.g., Asleap [4]) can be used to decrypt MSCHAPV2 protocol. If the the TLS connection is not secure during the handshake phase, the insecure internal protocols can be attacked by the rogue AS. Worse still, the encrypted network traffic can be intercepted and decrypted by the rouge AP when the victim client searches the Internet [11].

B. Certificate Validation

In the handshake phase, the client authenticates the authentication server (i.e., AS, authServer) by validating its certificates and signatures. There is a common consensus among the baseline requirements for the issuance and management of publicly-trusted certificates [1], which mainly includes the following contents.

Weak Key Pair and Weak Hash Algorithm. CA signs a certificate with a key pair algorithm (e.g., RSA-2048) and a hash algorithm (e.g., SHA-256), which binds the public key of a key pair with the certificate holder. As both MD5 and SHA-1 have been compromised, the security strength of RSA with a key size fewer than 2,048 bits (e.g., RSA-512, RSA-1024) is no longer considered adequate, and attacks against these insecure algorithms have already been demonstrated by previous works [10], [19], thus these weak key pair and weak hash algorithms are not recommended [1].

Certificate Chain. AuthServer sends a certificate chain to client devices for validation during the handshake phase. In a multi-tier certificate chain, it not only includes the authServer's own certificate, but also includes the CA certificates of each tier. A client device should check the chain starting from the authServer's certificate at the bottom all the way to a root CA certificate (i.e., trust anchor). Each-tier certificate in the chain must be signed by the CA directly above it and the root CA must be one of the client's trusted CAs. Note that in the WPA2-Enterprise networks, each client device should be configured with trusted CAs (e.g., a Certificate Trust List (CTL) in the Windows client devices), which come from the system built-in trust store or user self-configured trust store.

TABLE I: The indispensable extensions of certificates

Cert category	Basic Constrains	Key Usage	Extended Key Usage
CA	CA:true	keyCertSign	\
Server	CA:false	keyEncipherment	serverAuth

Certificate Validity. When a certificate is issued, it will be configured with a predefined validity period. However, expiration or revocation may make a certificate invalid. A certificate should be revoked due to a certain security reason (e.g, private key leakage), which commonly occurs before the scheduled expiration date. The client should verify the validity of each certificate and reject the certificate chain when any one in the certificate chain has expired or has been revoked.

Certificate Extensions. The following certificate extensions, *Basic Constrains*, *Key Usage* and *Extended Key Usage* [13] may be set as security-critical, which are helpful to identify both the attribute and the purpose of the certificates. *Basic Constrains* is used to distinguish whether a certificate is a CA certificate or not. Note that only CA certificate (i.e., "Basic Constrains= CA:true") can sign other certificates. *Key Usage* defines the purpose of the key pair bound in the certificate. *Extended Key Usage* further refines *Key Usage* extensions. If a certificate contains both a *Key Usage* extension and an *Extended Key Usage* extension, then the certificate must only be used for a purpose consistent with both two extensions.

Table. I shows the indispensable values of the above three extensions in the case of CA certificate and server certificate. A CA certificate must include the *keyCertSign* usage to indicate that its public key is used for signing. A server certificate must include the *keyEncipherment* usage, as its public key will be used for key transport during TLS negotiation. Meanwhile, if the *Extended Key Usage* extension of a server certificate is enabled, the *serverAuth* usage is required.

Certificate Hostname. A certificate may be associated with one or more hostnames, which can be set by two different attributes, i.e.,*CommonName* (CN) field or *Subject Alternative Name* (SAN) extension. Note that a certificate is valid only if the request hostname matches the hostname value of the connected authServer. Therefore, client should check whether the certificate subject matches the connected authServer through the CN field or the SAN extension. RFC2818 [20] recommends using SAN as the main source of server identifiers and supporting CN for backward compatibility.

III. Goals and Methodologies

During WPA2-Enterprise networks connection, most certificate validation logics are implemented inside the client devices by built-in implementations. Moreover, several certificate validation options can be configured by the UIs, which can be viewed as assigning parameters to the APIs of the built-in certificate validation. Client device users can choose and fill in the options manually with secure configurations, thus making the built-in opaque implementation logics validate the contents of the certificate as many as possible. Due to the multifarious customized implementations of client devices from different

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on July 11,2024 at 08:25:28 UTC from IEEE Xplore. Restrictions apply.

vendors, the goal of our work is to provide insights into *(Q1) how the mainstream client devices execute built-in certificate validation under secure configurations?* and *(Q2) may the configuration options provided by the devices from different vendors downgrade the security of the certificate validation?*

To this end, we leverage the following methodologies. First, we set all the UI options manually with the most secure configurations. Then a black box testing technique is leveraged to discover potential security vulnerabilities during built-in certificate validation. Second, we manually configure the options provided by the devices from different vendors to find out all the combinations which may downgrade the security of the certificate validation. Once a client device accepts an insecure certificate or rejects a secure certificate, we consider that there is a security vulnerability in the client-side certificate validation.

A. Black Box Testing

In order to explore the security vulnerabilities of the build-in client-side certificate validation during WPA2-Enterprise networks connection, we first build a real WPA2-Enterprise network, whose certificate structure is a three-tier certificate chain (i.e., the certificates of a self-signed root CA, an intermediate CA and an authServer). The authServer is set up by *Freeradius*, and all the certificates are generated by taking advantage of *OpenSSL*. Second, we generate a series of secure and insecure certificates for comparison. To eliminate the mutual interference between the various contents of the certificates in each tier, we only trigger one insecure or error setting in one of the certificates in a specific three-tier certificate chain. Third, we utilize the generated insecure certificates to conduct numerous black box testings on client devices. We set secure configurations in the devices, guaranteeing that the built-in implementations of certificate validation can be conducted as many as possible. The details of insecure certificates generation are as follows.

Weak Key Pair and Weak Hash Algorithm. To verify this case for CA and server certificate respectively, we designate six certificate chains, and each chain includes three certificates. In each chain, we set one certificate with weak key pairs (e.g., RSA-512 or RSA-1024) or with weak hash algorithms (e.g., MD5 or SHA-1), and the other two certificates are set with secure algorithms (e.g, RSA-2048 and SHA-256). Then we utilize the six certificate chains respectively to evaluate the validation of the client devices.

Invalid or Defective Certificate Chain. We verify the completeness of the certificate chain by modifying one of the certificate from the three-tier certificate chain. Moreover, as the local trust anchor (i.e., the root CA certificate) comes from the client device's trusted CAs, which include the system built-in trust store and user self-configured trust store, thus we also judge the validation by adding or deleting the certificate from the local trust anchor. Meanwhile, we observe whether additional security requirements (e.g., password, face recognition) are needed when a certificate is added to the local trust anchor.

Expired and Revoked Certificate. To carry out the validity check for CA and server certificates respectively, we also designate six certificate chains in this case, and each chain includes three certificates. In each chain, we first select one certificate, and then set it with a time that has expired, or revoke it through certificate revocation list (i.e., CRL). The other two certificates are in normal usage.

Unreasonable Certificate Extensions. By specifying the values of the server certificate extensions, i.e., *Basic Constrains*, *Key Usage* and *Extended Key Usage*, we construct CA and server certificates with various combinations of extensions, some of which do not match the extension value requirements of the corresponding certificate category (e.g., set the *Basic Constrains* of a server certificate with "CA:true").

Mismatched Hostname. Hostname check is only relevant to server certificates. In a generated authServer certificate, we set two different hostname values in its *CommonName* (CN) field and its *Subject Alternative Name* (SAN) extension. Then we design various hostname values to fill in the hostname input box on the devices to find out their certificate hostname matching policy.

B. Manual Configuration of Certificate Validation

The configuration options provided by the client devices commonly include *EAP method*, *Data phase authentication method*, *Certificate authority* and *Hostname*. The first three options commonly adopt a way of dropdown menu. Specifically, the *EAP method* options commonly contain *EAP-PEAP, EAP-TLS and EAP-TTLS*. The *Data phase authentication method* options commonly contain *PAP, CHAP and MSCHAPV2*. The *Certificate authority* options commonly contain *Do not check, System certificate and Specific CA certificate*. Note that *Do not check* disables the certificate validation completely, and the *System certificate* means trusting the certificates in the system built-in trust store, and the *Specific CA certificate* means trusting the specific certificate in the user self-defined trust store. Moreover, an input box is provided for the user to type in the hostname of the target authServer.

We manually configure all the above options provided by the selected client devices from different vendors to find out all the combinations which may turn the insecure certificates from being rejected to being accepted, thus downgrading the security of the certificate validation. Correspondingly, we generate insecure certificates with the above insecure combinations. Similarly, we use black box testing to verify how the client devices handle the generated insecure certificates.

IV. Empirical Study

We conduct an empirical study of client-side certificate validation during WPA2-Enterprise networks connection with black box testing (see Sec. III-A) and manual configurations of certificate validation (see Sec. III-B). This section describes the setup of our practical study and its results.

A. Setup

Commonly speaking, all the mainstream client devices from different vendors customize their own Wi-Fi module based

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on July 11,2024 at 08:25:28 UTC from IEEE Xplore. Restrictions apply.

on WPA-supplicant, which is a free software implementation of an IEEE 802.11i supplicant component with support for wireless networks access [5], thus most client devices have native support for IEEE 802.1X. In this work, we focus on the mainstream client devices in the wild. In the case of Android, to get a more comprehensive picture of the entire landscape, we select five vendors of Android devices with the largest market share (i.e., Samsung (Galaxy S9), Xiaomi (FindX3 Pro), OPPO (Reno7), Vivo (Z1i) and Huawei (P20)) and one native Android phone vendor (i.e., Google (Pixel 3aXL)), which accounts for about 76% of the Android phone market in the real world totally [2]. In the case of Windows, we select Windows 10 (Thinkpad X1), which is the most popular Windows desktop OS for now [7]. Note that in the case of iOS, they commonly judge all the received certificates untrusted and leave the burden of verifying the validity of certificate to the client device users. Therefore, we exclude iOS devices in our study. For comparison, we also involve the latest native WPA-supplicant open source code (Version 2.9) to locate the source of the potential security vulnerabilities.

B. Security Vulnerabilities of Built-in Certificate Validation

We first set all the selected client devices with secure certificate validation configurations, and then utilize the generated certificates to conduct a black box testing. The detailed results are shown in Table II.

Ignore Weak Key Pair and Weak Hash Algorithm. We find that the native WPA-supplicant rejects all the insecure certificates with weak key pair (i.e., RSA-512 and RSA-1024) or weak hash algorithms (i.e., MD5 and SHA-1). However, both the selected Android and Windows client devices accept the insecure certificates (including CA certificate and server certificate). Fake certificates can be forged if a certificate use MD5 or SHA-1 due to the collision attacks [10], [19], thus a rogue authServer can use the fake certificate with the same hash as the legitimate one to pass the certificate validation.

Insecure Trust Anchor Addition. All the selected client devices verify the completeness of the certificate chain, and if the root CA of the certificate chain is not contained in the local trust anchor, all the selected devices will reject the received server certificate. As the specified local trust anchor is selected from pre-installed trusted CAs. Generally speaking, importing a new CA certificate to the local trusted CAs requires the knowledge or/and the involvement from the users. However, we find that in the native WPA-supplicant, Windows and partial Android (i.e., Google and OPPO) devices, no authentication is required when a new trust anchor is added. The rest Android devices (i.e., Samsung, Xiaomi, Vivo, Huawei) require users authentication (e.g., inputting device unlock password, face recognition) to add a new trust anchor.

Note that the missing of users authentication of trust anchor addition make it easy for attackers to add malicious root CA certificates in silence (e.g., by malware). Moreover, we find that all the selected devices would replace an old certificate when a new certificate (with the same certificate name as the old one) is added. Once a malicious CA certificate is trusted, all the certificates issued by it will be trusted by the victim client device. Then the rogue authServer can use the certificates issued by the malicious CA certificate to disguise the legal server certificate to pass certificate validation.

Lack of Revoked Certificate Check. If the received certificate has expired, all the selected client devices could verify its invalidity. However, we find that all the selected client devices would accept insecure CA/server certificates which have been revoked through CRL. Note that the CRL is always provided by a publicly accessible service (e.g., URL).

In order to confirm whether the selected devices just omit the revoke check from CRL by default or skip revoke check due to lacking of network connection to CRL, we keep all the 5G cellular network of the selected devices available during network connection. Disappointingly, the revoked certificates can still always be accepted by all the selected devices. In addition, we find that there is no pre-downloaded CRL file on the selected devices. Taking advantage of this issue, a rogue server can first collect the revoked server certificates, and then utilize them to pass certificate validation.

Loose Restriction on Certificate Extensions. In the case of CA certificate, if the *Basic Constrains* is set as “CA:false” or the *Key Usage* does not include “keyCertSign”, the corresponding CA certificate can not sign a server certificate, thus the extensions of the CA certificate are commonly set properly. Therefore, we focus on the restriction on server certificate extensions. In a server certificate, the *Basic Constrains* should be set as “CA:false”, the*Key Usage* should include “keyEncipherment”, and the *Extended Key Usage* should include “serverAuth”. We design many combinations among the above three extensions including both correct and incorrect settings. We find that all the selected devices only make the wrong decision when the *Basic Constrains* is incorrect, simultaneously both the *Key Usage* and the *Extended Key Usage* are correct. Therefore, we can conclude that all the selected devices conduct a loose restriction on the *Basic Constraints* extension of server certificate. Taking advantage of this issue, once a server is attacked, an attacker could use the legal server certificate to further issue illegal lower-level server certificates, and then utilize these certificates to pass certificate validation.

No-recommended Certificate Hostname Check Policy. We find that all the Android devices adopt recommended hostname check policy. However, Windows devices use CN rather than SAN as the main source of server identifiers. In the case of WPA-supplicant, they adopt a substring matching hostname check policy in the CN field. For example, if the user types in “Example” in the user interface, but receives a server certificate with “CN=Example Server” in the CN field, the WPA-supplicant will accept this server certificate.

Based on the above results of our empirical study, we can answer our first proposed question: even though secure configurations are set by the UIs, the mainstream client devices execute built-in certificate validation with a certain security vulnerability. More specifically, during the customized implementations from the native WPA-Supplicant, both Android

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on July 11,2024 at 08:25:28 UTC from IEEE Xplore. Restrictions apply.

TABLE II: The built-in certificate validation results

Certificate Validation Contents				Android 11	Windows 10	WPA-supplicant 2.9
Weak Key Pair and Weak Hash Algorithm	RSA-512			⊗	⊗	●
	RSA-1024			⊗	⊗	●
	MD5			⊗	⊗	●
	SHA-1			⊗	⊗	●
Certificate Chain	Incomplete certificate chain			●	●	●
	Users authentication for trust anchor addition			None for OPPO and Google	None	None
				Needed for Samsung, Vivo, Xiaomi and Huawei		
Certificate Validity	Expired certificate			●	●	●
	Revoked certificate			⊗	⊗	⊗
Server Certificate Extensions	**Basic Constrains**	**Key Usage**	**Extended Key Usage**			
	CA:true	Null or including keyEncipherment	Null or serverAuth	⊗	⊗	⊗
	CA:true	Null or including keyEncipherment	Excluding serverAuth	●	●	●
	CA:true	Excluding keyEncipherment	Any	●	●	●
	CA:false	Null or including keyEncipherment	Null or serverAuth	●	●	●
	CA:false	Null or including keyEncipherment	Excluding serverAuth	●	●	●
	CA:false	Excluding keyEncipherment	Any	●	●	●
Certificate Hostname	Incorrect CN and Correct SAN			●	◇	●
	Correct CN and Incorrect SAN			●	⊗	⊗
	Correct CN and Correct SAN			●	●	●
	Incorrect CN and Incorrect SAN			●	●	●
	Correct CN and Empty SAN			●	●	●

● indicates that the client device accepts a secure certificate or rejects an insecure certificate.
⊗ indicates that the client device accepts an insecure certificate.
◇ indicates that the client device rejects a secure certificate.

and Windows devices introduce new security vulnerability of certificate validation. They commonly both have the problems on lacking the check of some security-critical certificate fields, such as insecure cryptographic algorithms, invalid revoked certificate, and loose restriction on server certificate extensions. Moreover, partial Android devices even require no authentication when a new trust anchor is added.

C. Security Downgrade Due to Configuration Options

Client devices from different vendors may provide different configuration options with specific UI design for certificate validation. Commonly speaking, the configuration options mainly include *EAP method*, *Data phase authentication method*, *Certificate authority* (i.e., trust anchor) and *hostname* (see Sec. III-B). We find that all the selected devices commonly choose EAP-PEAP protocol as the default *EAP method*, and choose MSCHAPV2 as the default *Data phase authentication method* (note that there is no default method in OPPO devices). The discovered insecure configuration options provided by the selected client devices are shown in Table. III.

Do Not Check Certificate by Default. In the case of *Certificate authority*, partial Android devices (i.e., Huawei, OPPO and Xiaomi) choose *"Do not check"* by default, which will disable the certificate validation completely. In addition, we find an interesting and insecure mechanism in the Xiaomi and Vivo devices. When certificate validation fails, Xiaomi and Vivo devices will offer users a "continue connection" option. If the option is selected blindly, the devices will roll back to a low security level of certificate validation mode, where the certificate validation will be disabled. We also find that different devices vendors design different sources of trust anchor. In the Windows, the trust anchor lies in the CTL. In the Android devices, the trust anchor can be system built-in trust store or specific user self-configured trust store, while the OPPO and Vivo devices only support the latter option.

Bypass Hostname Check. We find that the hostname check is not an mandatory option in Windows. Coincidentally, we find that all the selected Android devices allow users not to type in a hostname when the trust anchor lies in user self-defined trust store. However, if the trust anchor lies in system built-in trust store, all the selected Android devices require users to type in a specific hostname. Commonly speaking, both the certificates in the user self-defined trust store of Android devices, and the certificates in the CTL of Windows devices, are generated by commercial CAs or self-built CAs. However, over 75% of self-built CA certificate chains suffer from verification errors [23]. If the hostname check has been bypassed in a certain client device, then a malicious adversary could attack the device by applying for a new certificate chain from the corresponding commercial CA or breaking down the self-built CA system, then the victim device will not identify the illegal certificate during certificate validation.

In order to verify the security vulnerability due to improper configuration options, we generate insecure certificates with incomplete certificate chain and mismatched hostname

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on July 11,2024 at 08:25:28 UTC from IEEE Xplore. Restrictions apply.

TABLE III: Insecure configuration options provided by the selected client devices

Certificate validation configuration		Windows 10	Android 11					
			Samsung	Xiaomi	OPPO	Vivo	Huawei	Google
Certificate authority	Verify certificate by default?	Yes	Yes	No	No	Yes	No	Yes
	Trust anchor source	CTL	Θ Δ	Θ Δ	Δ	Δ	Θ Δ	Θ Δ
Is hostname inputting mandatory?		No	No(Δ)	No(Δ)	No(Δ)	No(Δ)	No(Δ)	No(Δ)
			Yes(Θ)	Yes(Θ)			Yes(Θ)	Yes(Θ)
Can roll back to a low security level of validation?		No	No	Yes	No	Yes	No	No

Θ indicates the trust anchor lies in system built-in trust store. Δ indicates the trust anchor lies in specific user self-configured trust store.

respectively for black box testing. All the generated insecure certificates are accepted by the selected client devices. Based on the above results of our empirical study, we can answer our second proposed question: the configuration options provided by the devices from different vendors indeed can downgrade the security of the certificate validation.

D. Countermeasures

Based on the all the discovered security vulnerabilities in our empirical study, we put forward the following countermeasures to enhance the security of certificate validation in WPA2-Enterprise networks.

Build-in Certificate Validation. As most certificate verification efforts are taken by built-in implementations of client devices, thus more attention should be paid on built-in certificate verification implementations. First, as attacks against insecure algorithms have already been demonstrated by previous works [10], [19], certificates with weak key pair and weak hash algorithms should not be accepted any more. Second, we recommend the client devices to download the online CRL proactively to local storage when the network is connected successful, then revoked certificate check can be performed by the local CRL. Third, the client devices should verify whether the extensions of a certificate is consistent with the purpose for which it is intended. Last but not least, user authentications (e.g., inputting unlock password , face recognition) should be encouraged when a new trust anchor is added.

Secure Client Configuration. Insecure client configurations mainly include two aspects, i.e., ignoring certificate validation and bypassing the hostname check. First, we suggest the customized UIs provided by different vendors should be prudent to provide the “Do not check" option, meanwhile providing explicit prompt message if users choose this option. Second, the hostname should be input compulsively by the UI to avoid accepting insecure certificates.

V. Evil Twin Attack Implementation

In an Evil Twin attack, as Fig. 2 shows, a rogue AP should impersonate a legitimate AP by assuming the same SSID, and set its signal to be stronger than that of the legitimate AP. Then the rogue AP tricks the client to execute WPA2-Enterprise network connection with the rogue authServer.

To execute an insecure client-side certificate validation in WPA2-Enterprise networks, we first connect the client device with the rogue AP, whose rogue authServer are set as the same

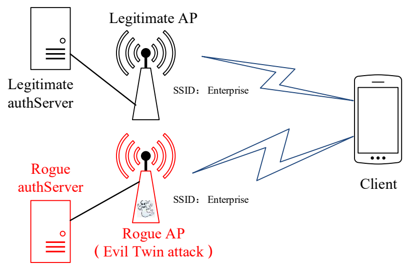


Fig. 2: Evil Twin attack scenario.

configurations (e.g., *EAP method*, *Data phase authentication method*, *Certificate authority* and *Hostname*) with the legitimate authServer. Second, we use the security vulnerability of *Bypass Hostname Check* (see Sec. IV-C) to generate insecure certificates. Specifically, we adopt a self-signed CA to issue two certificate chains for the legitimate authServer and the rogue authServer (using *Freeradius-wep*), respectively. Note that the hostname value of the certificates in the legitimate authServer and the rogue authServer are set differently. Third, we set the *Hostname* configuration as empty to bypass the hostname check. As we predicted, the insecure certificate is accepted by the client device and a TLS tunnel is established in the handshake phase. Last, we install the *Asleap* tool [4] on the rogue authServer to crack the user credentials, we recovered the user name and its password successfully.

VI. Related Work

Certificate-based authentication has been widely adopted in many scenarios and numerous security vulnerabilities on certificates validation have been proposed.

In the case of WPA2-Enterprise networks, Brenza et al. [11] assumed the client does not check the CN string of the offered certificate or does not setup the trusted CA certificate, thus enabled an attacker to authenticate users on the fly without owning the required username and password. They observed that 52% of the client devices turned out to be vulnerable to their attack. Bartoli et al. [9] found that a commonly provided “do not validate the certificate” option on client devices may fall prey of a successful Evil Twin attack virtually. Based on this insecure option, they focused on analyzing the potential security impact of different inner authentication protocols (in the data phase) of different OSs. Hue et al. [17] investigated the security of configurations on

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on July 11,2024 at 08:25:28 UTC from IEEE Xplore. Restrictions apply.

the WPA2-Enterprise network from both the client-side and the server-side. At the client-side, they used a framework to evaluate the configurations supported by the UIs of mainstream OSs to discover potential design weaknesses, and showed 85.7% of the tertiary educational institutions are susceptible to Evil Twin attack. However, their assessment does not take the customisation differences between various device vendors (even with the same OS) into account. At the server-side, they conducted a study on the various parameters related to the trustworthiness of the TLS tunnel and X.509 certificates to evaluate the strength of TLS connections induced by the back-end authServers, which revealed that weak parameters (e.g., expired certificates, deprecated versions of TLS, weak signature algorithms) are widely used. All the above previous works only considered the security vulnerabilities due to the provided options of the UIs configurations, while did not consider the security vulnerabilities from built-in certificate verification implementations of different client devices. In this work, we focus on the both the security vulnerabilities from the built-in certificate verification implementations of different client devices and the security downgrade due to customized UIs design from different device vendors.

Apart from WPA2-Enterprise networks, certificate validation security vulnerabilities on other application scenarios also have been widely evaluated. Wang et al. [22] disclosed browser defects on handling HTTPs errors in terms of certificate verification which result in a forged or revoked certificate will be accepted by PC browser. Debnath et al. [14] showed that many proxy-based mobile browsers downgrade the overall quality of TLS sessions and accept unsatisfactory certificates (e.g., old versions of TLS, weak cryptographic algorithms), thus exposing users to potential security and privacy threats. Besides, some non-browser softwares using TLS/SSL also have certificate validation issues. Geogiev et al. [16] demonstrated SSL certificate validation is completely broken in many security-critical applications and libraries, because of the badly designed APIs of SSL implementations. Chan et al. [12] analyzed the small footprint SSL/TLS libraries applied in IoT systems and exposed 48 instances of noncompliance in X.509 certificate validation implementations from 4 major families of SSL/TLS source base. Fahl et al. [15] presented an investigation of the current state of SSL/TLS usage in Android and revealed that 8.0% of the apps, which contain SSL/TLS codes, are potentially vulnerable to MITM attacks.

VII. Conclusion

In this paper, we implement an empirical study to evaluate the security of client-side certificate validation during WPA2-Enterprise networks connection. We consider both the vulnerabilities due to built-in certificate validation implementations and configuration options of client devices from different vendors. The results show that more than one security vulnerability exists in the built-in certificate validation implementations of the selected devices, and all the selected devices provide a certain option which may downgrade the security of certificate validation. A real Evil Twin attack is conducted to prove the utilizability of our discovered security vulnerabilities.

References

[1] Certificate contents for baseline ssl. https://cabforum.org/baseline-requirements-certificate-contents//

[2] Top manufacturers. https://www.appbrain.com/stats/top-manufacturers, year = 2022

[3] Divide and conquer: Cracking ms-chapv2 with a 100% success rate. https://web.archive.org/web/20160316174007/https://www.cloudcracker.com/blog/2012/07/29/cracking-ms-chap-v2/ (2012)

[4] Cracking wpa2 enterprise wireless networks with freeradius wpe, hostapd and asleap & john the ripper. http://phreaklets.blogspot.com/2013/06/cracking-wireless-networks-protected.html (2013)

[5] wpa_supplicant. https://w1.fi/wpa_supplicant/ (2020)

[6] Eduroam. https://www.eduroam.org/ (2022)

[7] Monthly market share held by windows operating system for desktop pcs worldwide from january 2017 to december 2021, by version. https://www.statista.com/statistics/993868/worldwide-windows-operating-system-market-share// (2022)

[8] Aboba, B.: Extensible authentication protocol (eap) method requirements for wireless lans. heise zeitschriften verlag (2005)

[9] Bartoli, A., Medvet, E., Onesti, F.: Evil twins and wpa2 enterprise: A coming security disaster? computers & security **74**, 1–11 (2018)

[10] Bhargavan, K., Leurent, G.: Transcript collision attacks: Breaking authentication in tls, ike, and ssh. In: Network and Distributed System Security Symposium–NDSS 2016 (2016)

[11] Brenza, S., Pawlowski, A., Pöpper, C.: A practical investigation of identity theft vulnerabilities in eduroam. In: Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks. pp. 1–11 (2015)

[12] Chau, S.Y., Chowdhury, O., Hoque, E., Ge, H., Kate, A., Nita-Rotaru, C., Li, N.: Symcerts: Practical symbolic execution for exposing noncompliance in x. 509 certificate validation implementations. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 503–520. IEEE (2017)

[13] D.Cooper, S.S.: RFC 5280: Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile (2008)

[14] Debnath, J., Chau, S.Y., Chowdhury, O.: When tls meets proxy on mobile. In: International Conference on Applied Cryptography and Network Security. pp. 387–407. Springer (2020)

[15] Fahl, S., Harbach, M., Muders, T., Smith, M., Baumg?Rtner, L., Freisleben, B.: Why eve and mallory love android: An analysis of android ssl (in)security. In: CCS '12: Proceedings of the 2012 ACM conference on Computer and communications security. p. 50 (2012)

[16] Georgiev, M., Iyengar, S., Jana, S., Anubhai, R., Shmatikov, V.: The most dangerous code in the world: validating ssl certificates in non-browser software. In: Acm Conference on Computer & Communications Security (2012)

[17] Hue, M.H., Debnath, J., Leung, K.M., Li, L., Minaei, M., Mazhar, M.H., Xian, K., Hoque, E., Chowdhury, O., Chau, S.Y.: All your credentials are belong to us: On insecure wpa2-enterprise configurations. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 1100–1117 (2021)

[18] IEEE Standards Association: IEEE Std 802.11-2016, IEEE Standard for Local and Metropolitan Area Networks-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (2016)

[19] Leurent, G., Peyrin, T.: From collisions to chosen-prefix collisions application to full sha-1. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 527–555. Springer (2019)

[20] Rescorla, E.: Rfc2818: Http over tls (2000)

[21] Shrivastava, P., Jamal, M.S., Kataoka, K.: Evilscout: Detection and mitigation of evil twin attack in sdn enabled wifi. IEEE Transactions on Network and Service Management **17**(1), 89–102 (2020)

[22] Wang, C., Lin, J., Li, B., Li, Q., Zhang, X.: Analyzing the browser security warnings on https errors. In: ICC 2019 - 2019 IEEE International Conference on Communications (ICC) (2019)

[23] Zhang, Y., Liu, B., Lu, C., Li, Z., Duan, H., Li, J., Zhang, Z.: Rusted anchors: A national client-side view of hidden root cas in the web pki ecosystem (2021)

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on July 11,2024 at 08:25:28 UTC from IEEE Xplore. Restrictions apply.