

# The Devil is in the Details: Hidden Problems of Client-Side Enterprise Wi-Fi Configurators

Ka Lok Wu <sup>1</sup>   Man Hong Hue <sup>1,2</sup>   Ka Fun Tang <sup>1</sup>   Sze Yiu Chau <sup>1</sup>

<sup>1</sup>Department of Information Engineering, The Chinese University of Hong Kong

<sup>2</sup>Georgia Institute of Technology

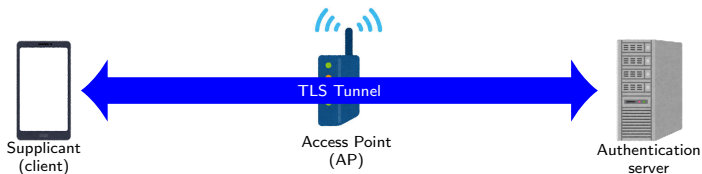
May 31, 2023



香港中文大學  
The Chinese University of Hong Kong



- Enterprise Wi-Fi uses Extensible Authentication Protocol (EAP) for mutual authentication
- PEAP and TTLS are popular EAP methods (phase 1) that uses Transport Layer Security (TLS) to authenticate the server and establish a secure tunnel
- Inside the tunnel, the client may be authenticated by password-based methods (phase 2 authentication)
- For server authentication in TLS using digital certificates, we need
  - 1 Chain of trust from some trust anchor, and
  - 2 End-entity certificate with a correct name in the subject or subject alternative name (SAN) field



# Configuring Enterprise Wi-Fi is difficult



- Credential theft through “Evil-Twin” (ET) attack
- The attacker can spoof the SSID of the target AP and obtain the user’s credentials
- The attacker setup can be as simple as a Raspberry Pi
- Organizations reuse single sign-on (SSO) credentials for Wi-Fi access  
⇒ initial access



Supplicant  
(client)







Fake AP +  
Authentication Server  
(Evil Twin)







- Relying on human users to manually configure server configuration leads to insecure outcomes
- Wi-Fi configurators make enterprise Wi-Fi configuration easier by redesigning the UI or automate the configuration process with bootstrapping pre-configured profiles
- Automatically checking the authenticity of the server
- To test these Wi-Fi configurators, we used an ET setup in a controlled environment



- ① TOFU on Android 
- ② Open Source Wi-Fi Configurators 
- ③ ChromeOS Built-in Configurator 
- ④ A Proprietary Wi-Fi Configurator 



- ① TOFU on Android 
- ② Open Source Wi-Fi Configurators 
- ③ ChromeOS Built-in Configurator 
- ④ A Proprietary Wi-Fi Configurator 



- Proposed in RFC 7435
- Assume an unauthenticated public key obtained during the first connection is secure and can be retained for future connections (pinning)
- More commonly, requires the **user to verify** the server's identity for the **first connection**

```
The authenticity of host [redacted] can't be established.  
ED25519 key fingerprint is SHA256:[redacted].  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? █
```

Figure: TOFU prompt in OpenSSH

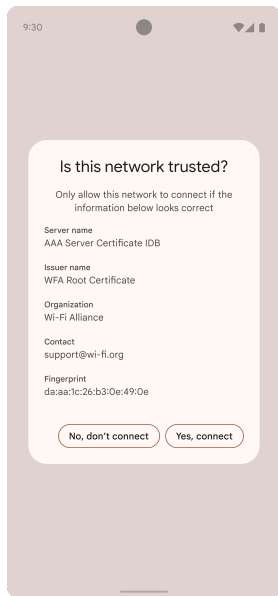


- a Impersonating the server during the first use, thus achieving credential theft
- b User can still connect to the legit server after the attack (stealthy attack)





- Introduced since Android 12
- Relies on the user to verify the server's identity for the first connection instead of prior configuration
- Before connection, a prompt is shown to the user with some attributes of the certificate, and option to accept or reject the certificate
- If the user accepts the certificate, certain attributes of the certificate are remembered for future connections





## 1 Send First, Verify Later

- Credentials are sent **before** the user is even prompted  $\implies$  **a** trivially succeeds

### Is this network trusted?

Only allow this network to connect if the information below looks correct.

Server Name:  
Sectigo RSA Domain Validation Secure  
Server CA

Issuer Name:  
USERTrust RSA Certification Authority

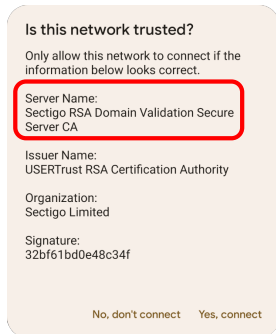
Organization:  
Sectigo Limited

Signature:  
32bf61bd0e48c34f

No, don't connect    Yes, connect

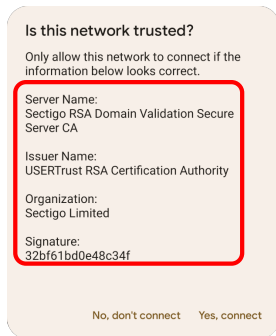


- 1 Send First, Verify Later
  - Credentials are sent **before** the user is even prompted  $\implies$  **a** trivially succeeds
- 2 Attributes of the **highest-level** certificate of the chain is shown instead of that of the end entity
  - Different from the documentation
  - Even if **1** is fixed, **a** can still succeed





- 3 All the attributes in the prompt are unauthenticated
- ET attacker can spoof the prompt with arbitrary attributes  $\implies$  a succeeds
  - it is possible to spoof the prompt and have the legit attributes pinned  $\implies$  b succeeds





- 3 All the attributes in the prompt are unauthenticated
  - ET attacker can spoof the prompt with arbitrary attributes  $\implies$  **a** succeeds
  - it is possible to spoof the prompt and have the legit attributes pinned  $\implies$  **b** succeeds
- 4 Forgetful Pinning
  - The system **forgets** the pin after 4 unsuccessful connection attempts
  - The attacker can **induce first use** again and again, making the former attacks practical

## Is this network trusted?

Only allow this network to connect if the information below looks correct.

Server Name:  
Sectigo RSA Domain Validation Secure Server CA





Issuer Name:  
USERTrust RSA Certification Authority

Organization:  
Sectigo Limited

Signature:  
32bf61bd0e48c34f

No, don't connect    Yes, connect



- ① TOFU on Android 
- ② Open Source Wi-Fi Configurators 
- ③ ChromeOS Built-in Configurator 
- ④ A Proprietary Wi-Fi Configurator 



- eduroam Configuration Assistant Tool (CAT)
- eduroam CAT and geteduroam apps on Android
- Programmatically load pre-configured profiles (policies) uploaded by IT admins
- Server authentication using the policies in the profile
- Security depends on
  - 1 The **specification** of policies
  - 2 The **enforcement** of policies
- Previous work<sup>1</sup> considered the profiles (policies)
- Focus on the **enforcement** of policies
- Test by our test platform and manual code review

---

<sup>1</sup>Hue et al., “All your Credentials are Belong to Us: On Insecure WPA2-Enterprise Configurations” (ACM CCS 2021)



- eduroam CAT uses the `setSubjectMatch` API method in Android to match server name
- Can only handle **one single** server name
- Multiple server names may be needed since an organization can have multiple authentication servers residing in different subdomains
- For multiple server names, eduroam CAT improvises an algorithm to compute a common suffix of a list of server names





- 1 Whenever multiple servers are specified in the profile, the algorithm peels off the leftmost subdomain
- 2 If the remaining suffix is shared by all server names, feed that to `setSubjectMatch`
- 3 else, feed empty string into `setSubjectMatch` (so all domains are accepted)



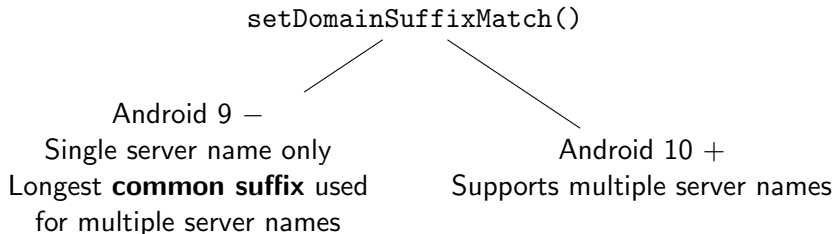
- Specifying [a.b.c, a.d.c] will give empty string as the output
- We targeted a real profile that will output **.at** to `setSubjectMatch` and specifies a commercial CA as the intermediate CA
  - We are purchased a **.at** domain and certificate from that CA
  - ET attack successful on an Android 11 device
- **If the policy specifies a commercial CA as trust anchor and the common suffix is loose, the profile is vulnerable**



- `setSubjectMatch` performs a **substring match** (instead of suffix match)
  - If `setSubjectMatch` outputs `.at`, the attacker can purchase `abc.attacker.com` and the name matching logic will accept it
  - ET attack successful with our test platform with our custom profile
  - **If the policy specifies a commercial CA as trust anchor, then the profile is vulnerable**



- geteduroam is a replacement of eduroam CAT
- in the stable release of version 1.0.21, server validation with `setDomainSuffixMatch()`, behavior depend on Android version





- We targeted a real profile that pins another commercial CA, and specifies two server names, one domain + one ip address, so the longest common suffix is empty string
- We purchased a certificate from that CA and ET attack is successful
- If the policy specifies a commercial CA as trust anchor and the **common suffix** is loose, the profile is vulnerable

# How many are vulnerable?







- Any profile that specifies a commercial CA as trust anchor is vulnerable to the substring matching problem in eduroam CAT
- If the policy specifies a commercial CA as trust anchor and the **common suffix** is loose, the profile is vulnerable to the longest common suffix problem in both apps
- We crawled 3854 profiles from eduroam CAT back in Oct 2022
- 1759 (45.6%) of the profiles are vulnerable to eduroam CAT `setSubjectMatch` substring matching problem
- 52 (1.35%) of the profiles are vulnerable to eduroam CAT common suffix enforcement problem
- 15 (0.39%) of the profiles are vulnerable to geteduroam common suffix enforcement problem



- Android's API design should also take some blame
- Before API level 18 (Android 4), there were simply no ways to setup server name checking
- from API level 18 to 23 (Android 6), only `setSubjectMatch` is available that can only handle **one single** server name
- `setAltSubjectMatch` and `setDomainSuffixMatch` are introduced on API level 23







- ① TOFU on Android 
- ② Open Source Wi-Fi Configurators 
- ③ ChromeOS Built-in Configurator 
- ④ A Proprietary Wi-Fi Configurator 





- ChromeOS has a built-in configurator with their custom profile format
- `SubjectMatch` (old attribute): can only accommodate **one single** server name in the subject
- This explains the loose name matching we observed in previous work
- Two new attributes: `SubjectAlternativeNameMatch` and `DomainSuffixMatch` only checks the subject alternative name (SAN) but not the subject
- Setting both `SubjectMatch` and the SAN attributes will require both the subject and SAN to be matched (logical AND), but the desired behavior by the standard (RFC 5216) is to match either one (logical OR)



- ① TOFU on Android 
- ② Open Source Wi-Fi Configurators 
- ③ ChromeOS Built-in Configurator 
- ④ A Proprietary Wi-Fi Configurator 



- SecureW2 JoinNow
- A commercial Wi-Fi Configurator that provides services to organizations that use enterprise Wi-Fi on multiple platforms
- On Android, Windows and MacOS: Fetching (or bootstrapping) pre-configured proprietary profiles from the SecureW2 server, similar to eduroam CAT and geteduroam
- We used a man-in-the-middle (MitM) proxy to intercept network traffic and understand how it works, and fetched the profiles from the SecureW2 server
- Investigated profiles that use PEAP or TTLS as their EAP method



- A `enableServerValidation` attribute that can disable server validation on Windows 10 and Android 9 –
- For those profiles that disabled server validation, we have tested that ET attack is indeed possible

OS\ Numbers	Total	Vulnerable
Windows	474	11 (2.32%)
Android	471	12 (2.55%)
macOS	475	10 (2.11%)

**Table:** The number of valid (unexpired and non-test/guest profiles) SecureW2 profiles crawled and with disabled server validation check on each mainstream OS.



- Some attributes will allow the application to inject certificates to the trust store of the system or of other applications
- Depending on the circumstances, this may allow the user's TLS traffic to be intercepted
- Could be problematic if the user is not aware of this



- For SecureW2 JoinNow on Windows, if TTLS is specified as the EAP method and PAP for phase 2, only two ciphersuites will be offered in its TLS Client Hello
- TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA and TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA, both are considered weak due to the use of RSA key exchange which lacks perfect forward secrecy (PFS)



- The trivial credential theft problem in Android TOFU is reported to Google in Oct 2022. Fix is planned to release soon.
- CVE-2023-20965



- The common suffix and substring matching problems are reported to the maintainers, and have positively acknowledged our Findings
- The developers note that for Android 8 (API level 26) or above, the `geteduroam` app is recommended instead of the eduroam CAT app
- However, the eduroam CAT app is still available on Google Play Store and are used by schools
- They argued that `setSubjectMatch` substring matching is a known problem, and their official guideline requires the use of a dedicated CA
- However, 45% of the profiles use commercial CAs as trust anchors
- `geteduroam` on Google Play hosts a different (older), non-vulnerable version of the app





- ChromeOS UI problem (where some critical inputs are optional) is fixed on M112



- The weak ciphersuites problem is reported to SecureW2, and we will continue to work with them to fix the problem
- For SecureW2 JoinNow profiles that are vulnerable to ET, we have reported the problem to the 15 respective organizations, received 2 responses stating that the issue is already solved



- We have investigated some enterprise Wi-Fi configurators
- We found several design and implementation flaws in Android TOFU mechanism, two open-source configurators, ChromeOS built-in configurator and a commercial configurator
- Despite the ease of use, there is a gap between the expectation and **enforcement** in enterprise Wi-Fi configurators
- The threat of ET is not over



- 1 For API designers: Design carefully and understand real-world needs. A wrong design decision can haunt for years (e.g. Android and ChromeOS)
- 2 For developers: Adaptively use the best API method available for policy enforcement. New OS versions should not suffer from old problems (e.g. eduroam CAT `setSubjectMatch`)!
- 3 For protocol designers: TOFU/TOOFU + pinning might be more usable and more secure. Wi-Fi server authentication has limited number of authentication servers, and do not need to scale like web.



- TOFU on Android allows the attacker to **induce** first-use (unpin)
- In some other instances of TOFU, where the user gets a vibrant warning message, and the user has to **manually unpin** the certificate
- We probably want this kind of “Trust-only-on-first-use” (TOOFU) instead

```

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@      WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!      @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ED25519 key sent by the remote host is
SHA256:
Please contact your system administrator.
Add correct host key in /etc/ssh/known_hosts to get rid of this messag
e.
Offending ED25519 key in /etc/ssh/known_hosts:
Host key for 192.168.1.1 has changed and you have requested strict checking.
Host key verification failed.

```

Figure: Warning message by OpenSSH.



# Thank you!

Questions?