context
Part of Accenture Security

Attacking and Defending WPA Enterprise Networks

contextis.com
E info@contextis.com
T +44 (0)207 537 7515

Contents

Contents

contextis.com
E info@contextis.com
T +44 (0)207 537 7515

Introduction

In today's age, wireless networks pose as a highly valuable and accessible attack vector against an organisation for attackers. This is due to many factors, including, but not limited to, corporate wireless networks generally being configured with access to internal networks, authentication occurring using corporate user accounts and a lack of hardening on connecting devices. These traits of typical corporate networks mean that malicious actors have ample attack avenues to target not only network configurations but also an organisation's users. Furthermore, pursuing these attack avenues can be very fruitful for malicious actors, potentially taking an unauthenticated actor straight into an organisation's internal network as an authenticated user.

Wireless networks can be configured with the following two security modes:

- **Personal** This mode uses a Pre-Shared Key (PSK), similar to a passphrase for authenticating clients to the network. All clients use the same PSK to connect to the network.
- **Enterprise** This mode uses a Remote Authentication Dial in User Service (RADIUS) server to handle authentication to the network. This allows each user to have separate credentials.

Organisations will typically utilise enterprise mode security for corporate networks, and this security mode will be the focus of this paper.

About this White Paper

The aim of this White Paper is to help organisations increase their wireless Wi-Fi Protected Access (WPA) Enterprise security posture. This is achieved by demonstrating various attacks against WPA Enterprise networks and giving recommendations on how to mitigate them.

The paper will provide readers with a foundational understanding of the framework which underpins the authentication process within WPA Enterprise networks (Section 1); detail how to identify the different authentication methods supported by a WPA Enterprise network (Section 2); outline practical attacks and mitigations against popular authentication methods (Section 3); and introduce a holistic approach to securing a wireless environment (Section 4).

A basic understanding of WPA Enterprise security concepts is assumed. Please visit the Context website[1] to read our introductory blog post 'A crash course into WPA Enterprise security and deployment'[2] for further information.

1 https://www.contextis.com/

2 https://www.contextis.com/en/blog/a-crash-course-into-wpa-enterprise-security-and-deployment

E info@contextis.com
T +44 (0)207 537 7515

1 The Extensible Authentication Protocol (EAP) authentication framework

This section at a glance

The EAP framework is a crucial element in Enterprise wireless security. All authentication is based on this framework and its many implementations.

Section 1 of this White Paper will give an overview of the EAP framework and the steps involved in order to perform authentication, giving details on currently utilised EAP methods, including general method details, key features, and supplicant and authentication server support.

This section is heavily theory based, however, it contains important information to ensure a sound understanding of the EAP framework, and popular EAP methods. This will help offensive teams to understand why certain attacks work, and will allow defensive teams to gain an understanding of why certain EAP methods are weak.

The Extensible Authentication Protocol (EAP) is a framework which defines message formats and common functionality, not a specific authentication implementation itself. This allows easy development of additional authentication mechanisms which can be implemented as EAP methods for easy distribution.

1.1 How are EAP messages transmitted?

As EAP is only an authentication framework (not an actual protocol), all protocols which implement EAP must have a way to transmit EAP messages within their own protocol, i.e. encapsulate EAP messages.

So which protocols do encapsulate EAP? Networking protocols IEEE 802.1X and RADIUS, and the Protected Extensible Authentication Protocol (known as Protected EAP or PEAP) all support EAP encapsulation. While these are not the only protocols which support EAP encapsulation, these three protocols are relevant to the focus of this paper. A brief summary of each is as follows:

- **IEEE 802.1X** — 802.1X is an IEEE standard which defines a group of networking protocols related to port-based Network Access Control. Within this group of protocols, the standard also defines a method to encapsulate 'EAP over LAN' networks, known as EAPOL.
- **RADIUS** — RADIUS is a networking protocol which provides centralised Authentication, Authorisation, and Accounting. This is known as an AAA protocol. The RADIUS protocol is utilised by RADIUS servers which provide authentication. In order for 802.1X authentication to utilise a RADIUS authentication server, the EAP messages encapsulated within 802.1X must be forwarded to the RADIUS server. A Network Access Device, such as a Wireless Access Point or switch is able to encapsulate EAP messages within the RADIUS protocol to be forwarded to the RADIUS server.

contextis.com
E info@contextis.com
T +44 (0)207 537 7515

— **PEAP** PEAP is an EAP method itself (IANA assigned EAP type of 25), which is also designed to encapsulate EAP messages within a TLS tunnel. PEAP utilises EAPOL to setup a TLS tunnel. Once set up, the PEAP protocol then defines the chaining of additional EAP methods which can be used within the encrypted tunnel.

We can see this encapsulation in play when viewing these protocols within Wireshark:

IEEE 802.1X:

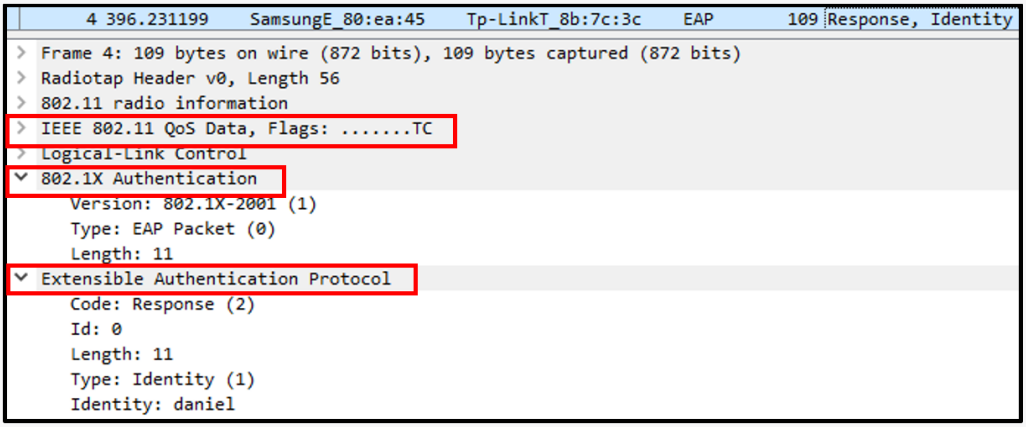

Figure 1 - For 802.1X, the EAP layer is encapsulated within the IEEE 802.11 wireless LAN layer

RADIUS:

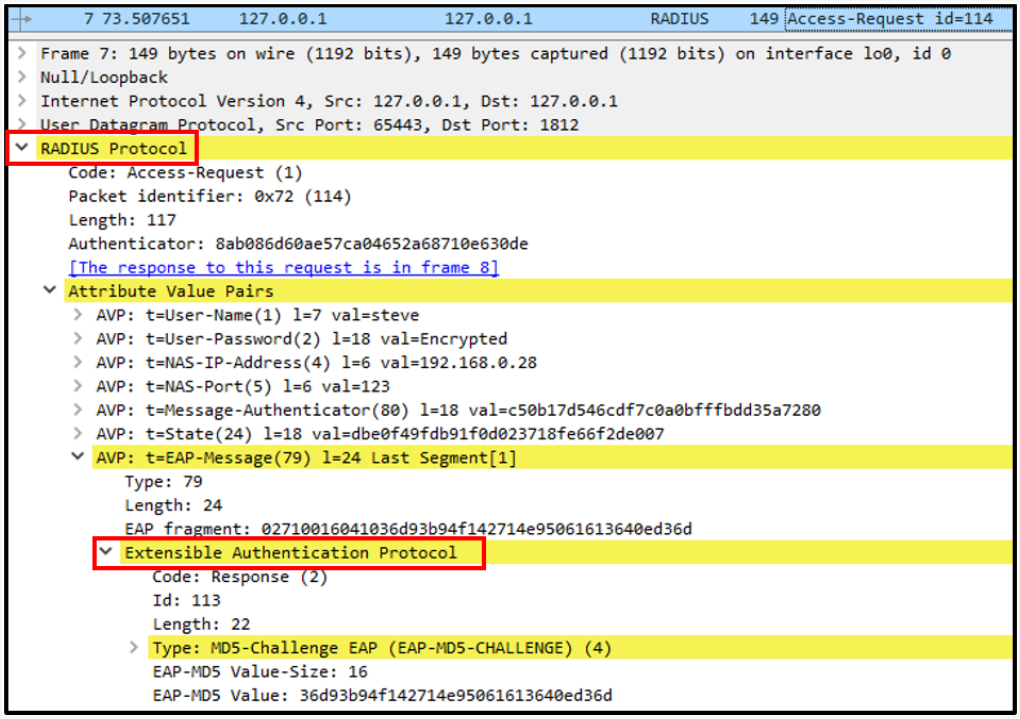

Figure 2 - The RADIUS protocol does not have an encapsulated EAP Layer, instead the EAP data is encapsulated within the RADIUS protocol itself

PEAP:

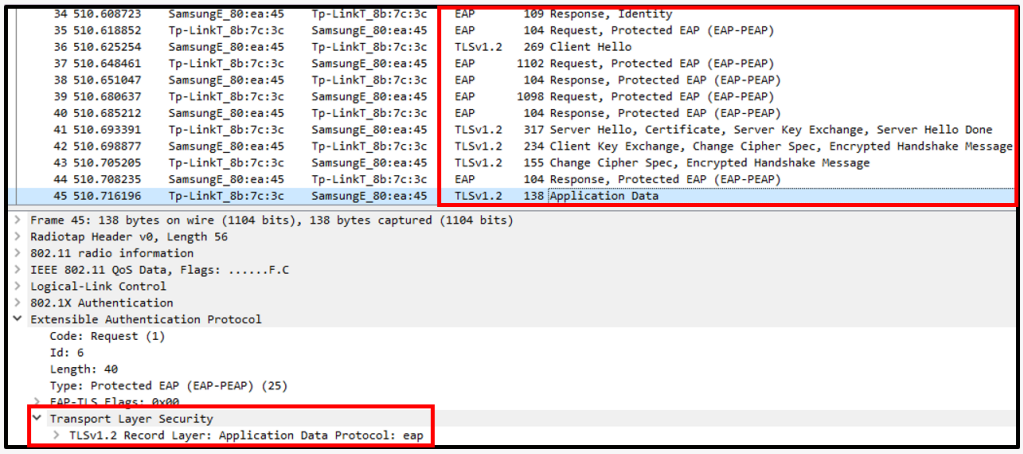

Figure 3 - The TLS tunnel created by the PEAP authentication communicates EAP messages

1.2 EAP authentication types

Multiple EAP methods exist (over 40 have been defined by the Internet Assigned Numbers Authority (IANA)[3], each implementing their own authentication mechanism. These methods can be broken up into two distinct types, native and tunnelled EAP methods.

— **Native EAP Types** Native EAP methods simply implement EAP messages and are not protected by any sort of encryption (outer tunnel). As such, these methods are transmitted in clear-text over the network and can be sniffed and analysed by attackers.

— **Tunnelled EAP Types** Tunnelled EAP methods are protected by an encrypted layer (outer tunnel), mitigating an inherent weakness within the EAP framework, whereby a protected communication channel for EAP messages was assumed.

When a native EAP method is used, authentication is simply referred to by its EAP method name, e.g. EAP-MD5 authentication. When a tunnelled EAP method is utilised, the outer encrypted layer is known as the outer tunnel/authentication, while the EAP method authentication occurring within the outer tunnel is known as inner authentication. In a tunnelled EAP method, authentication is referred to as <outer_authentication>/<inner_authentication>, e.g. EAP-TTLS/EAP-MSCHAPv2.

Also note that the term outer tunnel and outer authentication is used loosely interchangeably. The term outer tunnel refers to the TLS tunnel created by the authentication method, while outer authentication refers to the authentication of the server to the client (and optionally client to server) through X.509 certificates.

Additionally, a side effect of having an outer authentication method, is that all tunnelled EAP methods ensure mutual authentication. Authentication of the server to the client is achieved via outer authentication (when configured appropriately), and once the outer tunnel is set up, a secondary authentication occurs, typically an EAP method, whereby the client provides credentials to the server, i.e. the client authenticates to the server. This is helpful in cases where the inner EAP method does not support mutual authentication.

3 https://www.iana.org/assignments/eap-numbers/eap-numbers.xhtml#eap-numbers-4

1.3 Overview of EAP messages

1.3.1 Native EAP authentication

The EAP authentication process consists of a series of message exchanges. For native EAP methods these EAP messages are depicted in the following diagram:

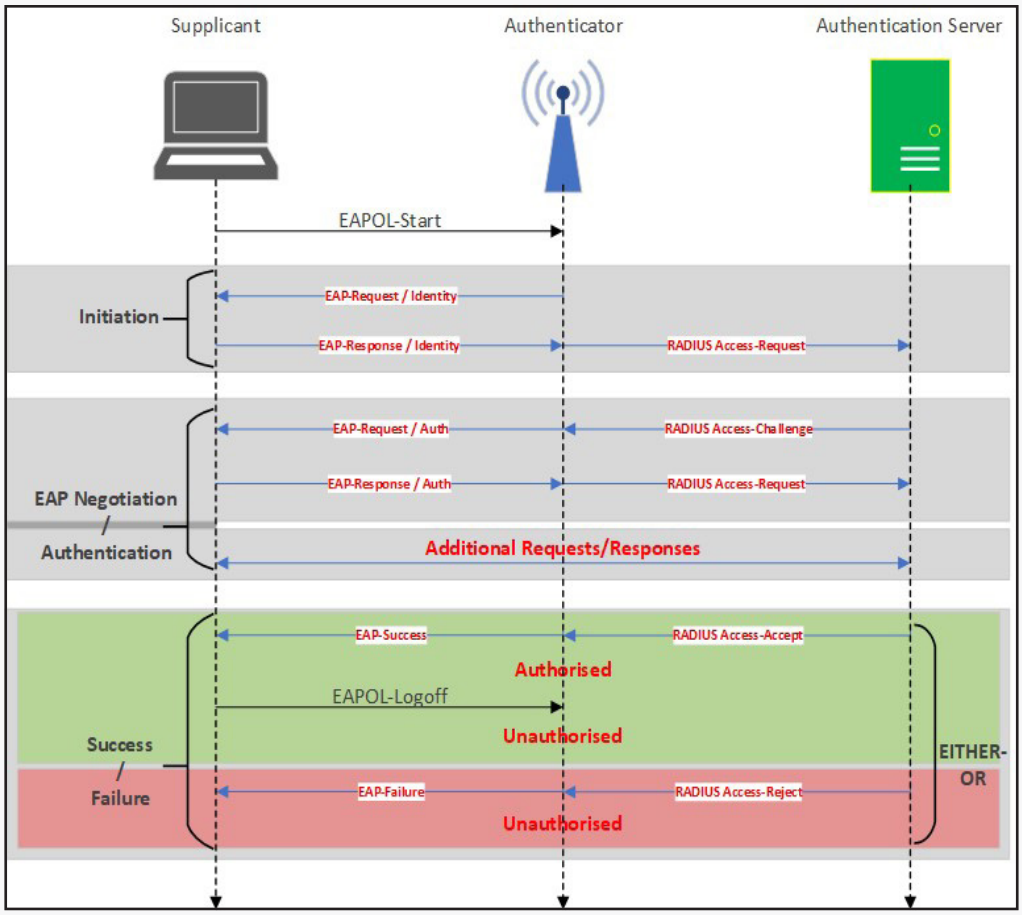

Figure 4 - EAP messages sent during the authentication process of a native EAP method

The EAP message exchanges can be distilled into the following three overarching steps:

- **Initiation** — EAP Identity Request/Response message. Where an 'identity' used for routing information is provided.
- **EAP negotiation / authentication**
  - EAP Auth Request/Response message. Where <Auth> is substituted with an EAP Method, e.g. PEAP or EAP-TTLS.
  - If the supplicant declines the EAP method proposed by the authentication server, Auth EAP-Request/Responses are continually exchanged until a valid EAP method is chosen by both parties.
  - If the supplicant accepts the proposed EAP method, then authentication is typically started automatically and sent in the EAP response. If EAP negotiation was performed, or the authentication type requires additional EAP-Request/ Responses, then further requests/responses are made.
- **Success / failure** — EAP Success or Failure message indicating if authentication was successful or not.

Viewing EAP messages within Wireshark helps to consolidate understanding of EAP messages and gives a somewhat visual picture of how EAP messages are actually sent over the network.

The following images demonstrate the authentication related EAP messages discussed above within Wireshark (note that EAP messages can be filtered within Wireshark by simply using the display filter 'eap'):

EAP Identity

Request: This is where an authentication server will request the identity of the connecting peer from the supplicant:

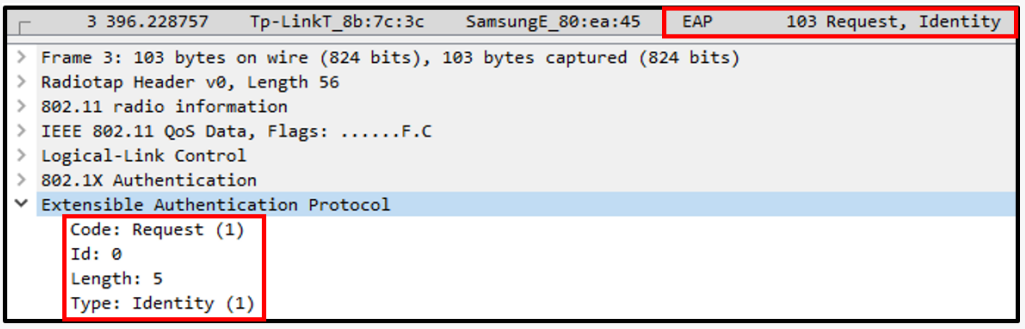

Figure 5 - EAP Identity Request

Response: The supplicant responds to the authentication server's identity request with a response that contains an 'identity' value. Note that the purpose of this identity is not for authentication, but for request routing and EAP negotiation purposes:

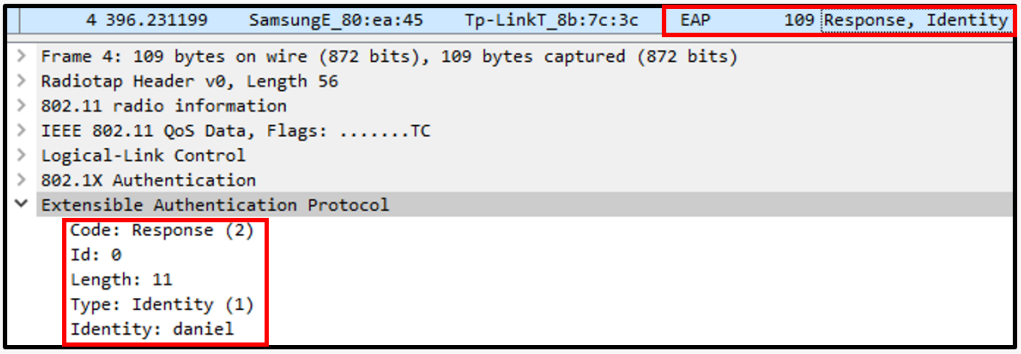

Figure 6 - EAP Identity Response

EAP Auth

Request: This is where an authentication server will suggest an EAP method to use for authentication by the supplicant:

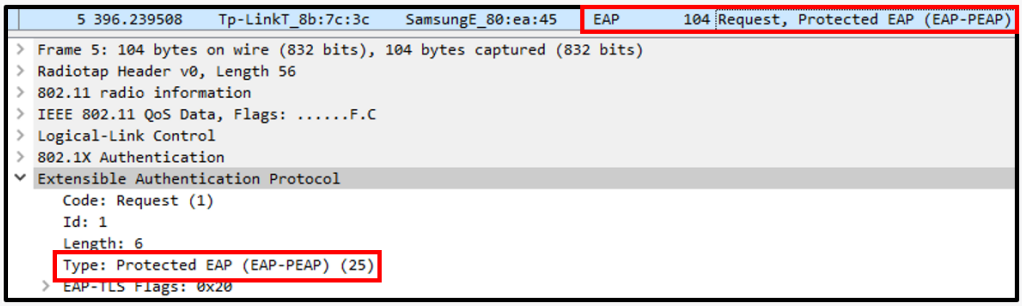

Figure 7 - EAP Auth Request

contextis.com
E info@contextis.com
T +44 (0)207 537 7515

Response: If the supplicant accepts the EAP method requested by the authentication server, then the supplicant will instantly start the appropriate EAP method process, i.e. no 'accept' message is returned:

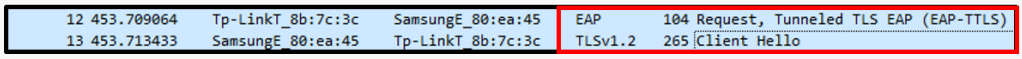

Figure 8 - Supplicants will automatically start the EAP method process if the requested EAP method is accepted, in this case EAP-TTLS was accepted

If the supplicant does not like the proposed EAP method, then the supplicant can send back an EAP Legacy Nak Response. This response contains a desired EAP method which the supplicant actually wants to use (see below):

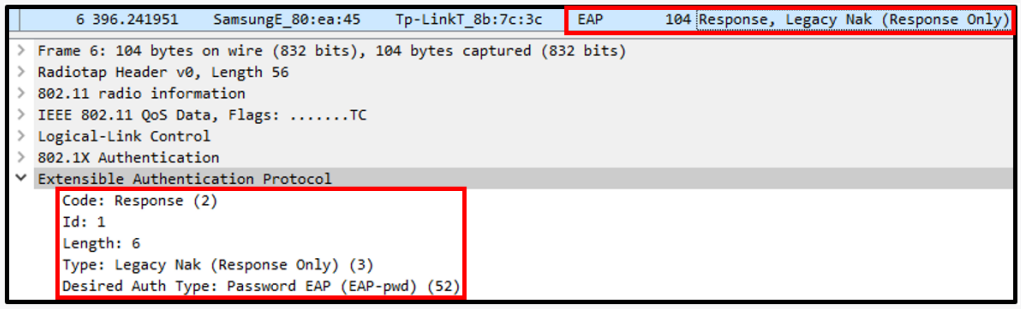

Figure 9 - EAP Legacy Nak Response is sent when the supplicant wants to request a specific EAP method for authentication

If the authentication server supports the EAP method requested within the EAP Legacy Nak response, then the authentication server will response with an EAP Auth request for that specific EAP method and the supplicant will start that EAP methods process (see figure 7).

EAP Failure

If the authentication server does not support the EAP method requested by the supplicant within the EAP Legacy Nak response, then an EAP Failure message is returned:

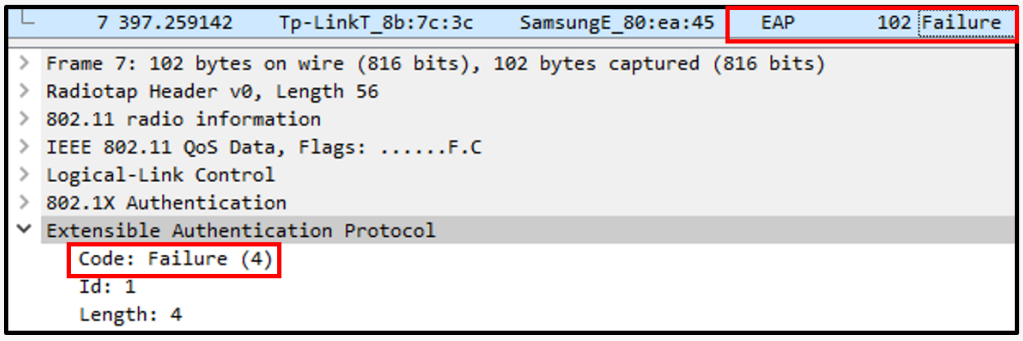

Figure 10 - EAP Failure Message

EAP Success

If the authentication server and supplicant have agreed on an EAP method for authentication, the authentication process is started (packets 5 and 6 below). The EAP method authentication process is played out (packets 6 to 20 below), and if authentication is successful, then an EAP Success message is returned; if not an EAP Failure message is returned:

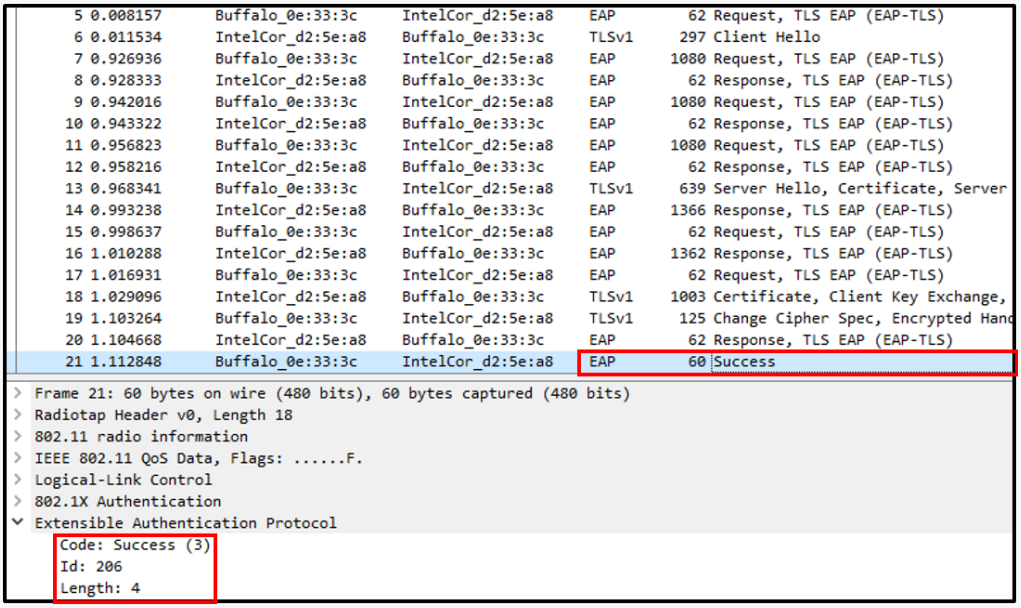

Figure 11 - EAP Success Message

1.3.2 Tunnelled EAP authentication

The authentication process for tunnelled EAP methods is simply an extension of native EAP authentication.

Tunnelled EAP authentication consists of two phases. Phase one (also known as outer authentication) consists of negotiating which initial tunnelled EAP method should be utilised, and ends with an encrypted TLS tunnel being generated. Phase two (also known as inner authentication) utilises the encrypted tunnel setup in phase one in order to perform typical EAP authentication, similar to how authentication occurs in a native EAP method.

EAP messages are not directly sent within the inner tunnel, instead, EAP messages are encapsulated within another protocol, e.g. within the encrypted tunnel created by EAP-PEAP, EAP messages are encapsulated within EAP-TLV messages. Assessing these encapsulations is out of the scope for this whitepaper, but is still a good note to understand.

E info@contextis.com
T +44 (0)207 537 7515

The following diagram depicts the typical authentication process for tunnelled EAP methods:

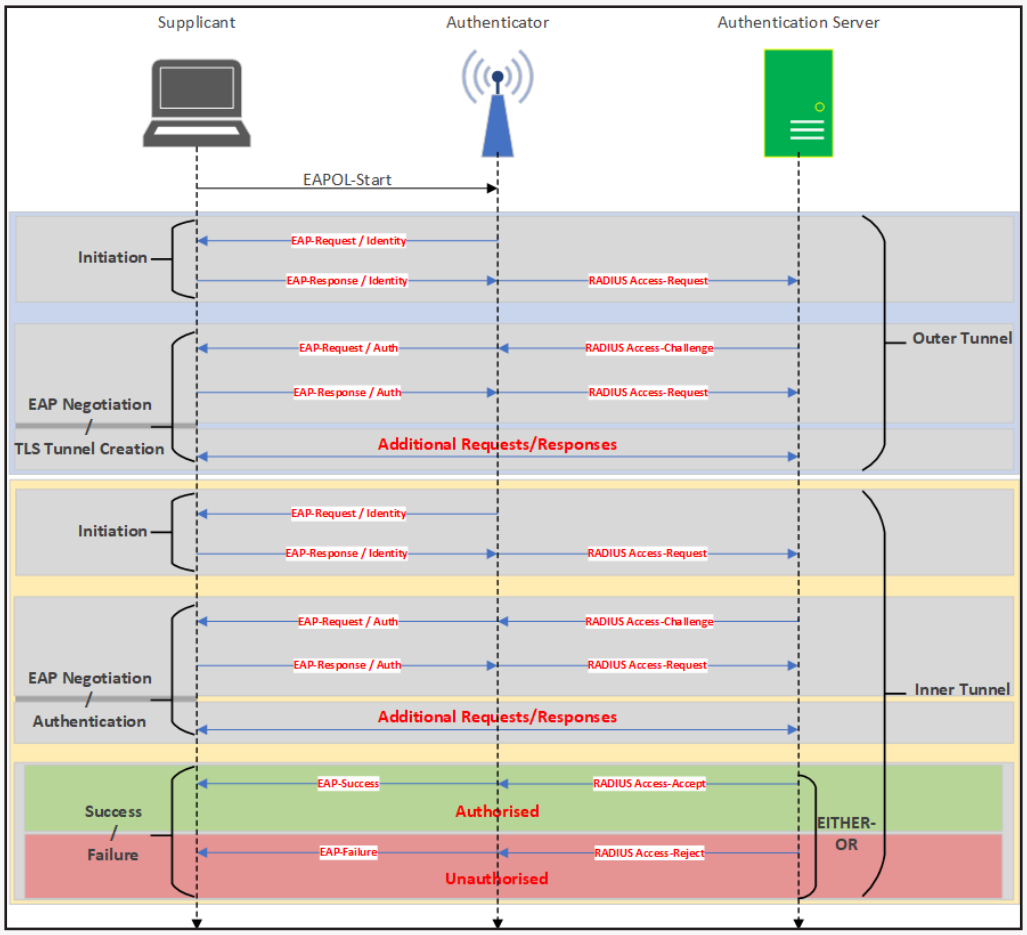

Figure 12 - Messages sent during authentication of a tunnelled EAP method

1.4 Popular EAP methods

For this paper, we have chosen to analyse and discuss only EAP methods which are supported by both Microsoft Network Policy Server (NPS) and pfSense FreeRADIUS. NPS is available in Microsoft Windows Server 2008 and above and can act as a RADIUS server. pfSense is an open source firewall and router with extensive support for network management features and more. Through pfSense's plugin support, a FreeRADIUS server can be hosted on the system. Covering the EAP methods supported by these two servers will give readers a solid foundation for the most likely authentication methods to be encountered within organisations.

The following are the EAP methods that we will focus on within this paper.

1.4.1 Tunnelled EAP methods

Protected Extensible Authentication Protocol (PEAP)

This EAP method provides a TLS tunnel for additional EAP methods to be used as an inner authentication method. The chaining of EAP mechanisms is defined, not the chaining of a specific EAP method, allowing almost all EAP methods to be utilised for inner authentication.

Although PEAP only specifies the chaining of multiple EAP types, there are two specific versions of PEAP with defined inner authentication methods. These are:

— IPEAPv0/EAP-MSCHAPv2
— IPEAPv0/EAP-TLS
— IPEAPv1/EAP-GTC

Version 0 and 1 were implemented by Microsoft and Cisco respectively. In today's age however, RADIUS servers and supplicants have advanced to the point where a large number of inner EAP authentication methods are supported and usable in either PEAP version.

EAP Tunnelled Transport Layer Security (EAP-TTLS)
This EAP method is very similar to PEAP in that an encrypted tunnel is setup in order to perform authentication within the tunnel (inner authentication). EAP-TTLS differs from PEAP in the supported inner authentication methods. PEAP can only tunnel additional EAP methods, whereas EAP-TTLS can tunnel not only EAP methods, but additional legacy authentication protocols such as PAP, CHAP, MS-CHAP, MS-CHAP-V2, etc.

EAP Flexible Authentication via Secure Tunnelling (EAP-FAST)
This EAP method is a Cisco proprietary protocol, and was introduced to improve upon weaknesses within LEAP. EAP-FAST specifically mitigates against passive sniffing attacks by creating an encrypted TLS tunnel, set up utilising a Protected Access Credential (PAC) file. The outer EAP-FAST tunnel can then be used to establish an inner tunnel for further EAP method authentication.

1.4.2 Native EAP methods

EAP Transport Layer Security (EAP-TLS)
This EAP method is considered the most secure EAP method. This is due to the requirement that both a server and client-side X.509 certificate must be in place. This allows a client to authenticate a server via its certificate, and vice versa for the server to authenticate the client. Additionally, the use of digital certificates means that a compromise of a user's password, will not result in the compromise of the network. EAP-TLS utilises X.509 digital certificates to perform authentication.

EAP Microsoft Challenge-Handshake Authentication Protocol (EAP-MSCHAPv2)
This authentication method is an implementation of Microsoft's Challenge-Handshake Authentication Protocol within EAP. MSCHAPv2 improves upon MSCHAPv1 by introducing mutual authentication, dynamic cryptographic key generation, and disables support for some weak LAN Manager encoded responses. EAP-MSCHAPv2 utilises credentials to perform authentication.

EAP Generic Token Card (EAP-GTC)
This authentication method was created by Cisco as an alternative to Microsoft's EAP-MSCHAPv2 authentication method. EAP-GTC can utilise credentials and One-Time-Pins (OTPs) to perform authentication.

EAP Message-Digest-5 (EAP-MD5)
This authentication method utilises a three-way handshake to authenticate a client to the server. This handshake involves creating a hash of the users' credentials, which can be transferred to the authentication server to be compared with the server's locally hashed copy. EAP-MD5 utilises credentials to perform authentication.

Lightweight Extensible Authentication Protocol (LEAP)
This EAP method (also known as EAP-LEAP) is a Cisco proprietary protocol. LEAP performs authentication through a modified version of the MSCHAPv1 authentication protocol. LEAP utilises credentials to perform authentication.

E info@contextis.com
T +44 (0)207 537 7515

1.5 EAP method attributes

The following table outlines some key attributes of the identified tunnelled EAP methods:

EAP Method	Outer TLS Tunnel Authentication	Inner Authentication	Client Certificate Required	Server Certificate Required
PEAP	Authentication server authenticated and optionally client depending on client certificate enforcement	EAP- MSCHAPv2 EAP-TLS EAP-MD5 EAP-GTC	Optional	Yes
EAP-TTLS		Supports the use of EAP methods, but also supports other legacy methods such as PAP, CHAP, MS-CHAP, etc.	Optional	Yes
EAP-FAST	Both authentication server and client mutually authenticated in 'Server-Authenticated Provisioning Mode'. Only client authenticated in 'Server-Unauthenticated Provisioning Mode'	EAP-MSCHAPv2 EAP-GTC EAP-TLS	No (PAC file utilised)	No (PAC file utilised)

Furthermore, the following table outlines some key attributes of the identified native EAP methods:

EAP Method	Client Certificate Required	Server Certificate Required	Authentication	Per-Session / Dynamic Keys Generated*
EAP-TLS	Yes	Yes	Mutual	Yes
EAP-MSCHAPv2	No	No	Mutual	Yes
EAP-GTC	No	No	One way - client authenticated to server through credentials	N/A
EAP-MD5	No	No	One way - client authenticated to server through credentials	N/A
LEAP	No	No	Mutual	Yes

**Rows which have a value of 'N/A' indicate that the given EAP method doesn't generate per-session/dynamic keys and also doesn't generate any keying material. This means that these EAP methods cannot be used for wireless authentication themselves and must be used within a tunnelled EAP method which can itself generate keys for wireless network encryption.*

E info@contextis.com
T +44 (0)207 537 7515

1.6 EAP authentication compatibility

For reference, the following table highlights popular tunnelled EAP authentication methods and their compatibility with popular devices and authentication servers:

	Authentication Server		Supplicant				
EAP Method	NPS	Free RADIUS	Windows	Linux	MacOS	Android	iOS
PEAP / EAP-MSCHAPv2	Yes	Yes	Yes	Yes	Yes	Yes	Yes
PEAP / EAP-TLS	Yes	Yes	Yes	No	No	No	No
PEAP / EAP-MD5	No	Yes	No	Yes	No	No	No
PEAP / EAP-GTC	No	Yes	No	Yes	Yes	Yes	Yes
EAP-TTLS / EAP-TLS	No	Yes	Yes	No	No	No	No
EAP-TTLS / EAP-GTC	No	Yes	No	Yes	No	Yes	No
EAP-TTLS / EAP-MSCHAPv2	Yes	Yes	Yes	Yes	Yes	Yes	Yes
EAP-FAST*	No	Yes	No	Yes	Yes	Yes	Yes

Similarly, the following table highlights popular native EAP authentication methods and their device and authentication server compatibility:

	Authentication Server		Supplicant				
EAP Method	NPS	Free RADIUS	Windows	Linux	MacOS	Android	iOS
EAP-TLS	Yes	Yes	Yes	Yes	Yes	Yes	Yes
EAP-MSCHAPv2	Yes	Yes	No	No	No	No	No
EAP-GTC	No	Yes	No	No	No	No	No
EAP-MD5	No	Yes	No	No	Yes	No	No
LEAP	No	Yes	No	Yes	Yes	No	No

* *Inner EAP methods for EAP-FAST have not specifically been assessed in this whitepaper*

** *Only 'default' native client support has been assessed. Devices (such as Windows) can typically extend their supported EAP methods by installing extension packs, utilising different supplicants, etc*

*** *The above table is based off the following systems:*
- *NPS Server - Microsoft Windows Server 2012 R2*
- *FreeRADIUS Server - pfSense version 0.15.6_17, Freeradius3 BSD version 3.0.20*
- *Windows Client – Windows 10 Build 18362*
- *Linux Client – wpasupplicant version 2:2.6-15ubuntu2.4*
- *MacOS Client – MacOS Mojave version 10.14.6*
- *Android Client – Android version 6*
- *iOS Client – iOS 13.1*

2 Fingerprinting Wireless Enterprise Networks

This section at a glance

In order to attack or defend a network effectively, teams must understand the configuration of the network. As such, this section will cover popular methods to gain a quick understanding of the EAP methods supported by an authentication server.

Performing reconnaissance against a network is a key step in identifying which EAP methods a network supports. This is important for offensive teams who may be looking for weaknesses within the network, and also defensive teams who are monitoring configuration changes and performing network analysis.

2.1 Tools

A number of tools have been created to fingerprint EAP types within enterprise wireless networks. The following are two well-known tools to perform this:

EAPeak[4]
EAPeak is a tool written in Python and developed by Spencer McIntyre (@zeroSteiner). The EAPeak project comes with two tools, eapeak and eapscan.

- The eapeak tool provides analysis functionality for EAP wireless networks, including identification of a networks EAP type, and information about connecting clients (stations). Additionally, eapeak has functionality to export X.509 certificates captured over the air, aiding in creating legitimate looking rogue access points.
- The eapscan tool deviates from being a passive sniffing tool like eapeak and works in an active fashion. The tool will attempt to determine the EAP types supported by an access point by performing active EAP negotiation with the authentication server.

4 https://github.com/rsmus-llp/eapeak

E info@contextis.com
T +44 (0)207 537 7515

Fingerprinting Wireless Enterprise Networks

```
9/06/20|10:04:22  eapeak git:(master) ✗  $ eapeak -l -i wlan0mon -s EnterpriseDemo

 _________
< eapeak >
 ---------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||

Welcome To EAPeak
Version: 0.1.8


Begining Live Capture...
^Cckets: 7030 Wireless Networks: 11
Stopping Live Capture...

**************************************
* EAPeak Summary of Wireless Networks *
* Found 1 Network(s)                  *
**************************************

SSID: EnterpriseDemo
    BSSIDs:
        34:e8:94:8b:7c:3c
    EAP Types:
        PEAP
        EAP-TTLS
    Client Data:
        Client #1
        MAC: ac:d1:b8:e1:c1:1d
        Associated BSSID: 34:e8:94:8b:7c:3c
        Identities:
            ASuperFakeIdentity
        EAP Types:
            EAP-TTLS
```

Figure 13 - eapeak being utilised to passively fingerprint a network

```
11/06/20|3:42:35  eapeak git:(master) ✗  $ sudo ./eapeak -s EnterpriseDemo -b 34:E8:94:8B:7C:3C -l -i wlan1mon --expor
t-x509

 _________
< eapeak >
 ---------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||

Welcome To EAPeak
Version: 0.1.8


Begining Live Capture...
^Cckets: 2909 Wireless Networks: 1
Stopping Live Capture...

Available Certificates:
Certificate #1 SSID: UNKNOWN_SSID
    Expiration Date: May 31 05:59:44 2030 GMT
    Issuer:
        CN: freeradius-temp-ca
    Subject:
        CN: freeradius-temp-server

Certificate #2 SSID: UNKNOWN_SSID
    Expiration Date: May 31 05:59:44 2030 GMT
    Issuer:
        CN: freeradius-temp-ca
    Subject:
        CN: freeradius-temp-ca

Certificate To Export (1 - 2): 1
Certificate Successfully Saved To: UNKNOWN_SSID_cert.pem
Now Exiting...
```

Figure 14 - eapeak being utilised to export a RADIUS server's X.509 certificate

```
11/06/20|4:31:41  eapeak git:(master) ✗  $ sudo ./eapscan -e EnterpriseDemo -b 34:E8:94:8B:7C:3C -i wlan1mon -c 6 --id
entity testing
[*] Checking Connection To AP OK!
[+] EAP Type: MD5 Supported
[*] EAP Type: ONE TIME PASSWORD not Supported
[+] EAP Type: GENERIC TOKEN CARD Supported
[+] EAP Type: EAP-TLS Supported
[+] EAP Type: LEAP Supported
[+] EAP Type: EAP-TTLS Supported
[+] EAP Type: PEAP Supported
[*] EAP Type: EAP-FAST not Supported
[*] EAP Type: EXPANDED EAP not Supported
```

Figure 15 - eapscan being used to perform active fingerprinting against an Access Point
**Note, an identity value, any value, must be specified for the tool to work appropriately*

contextis.com
E info@contextis.com
T +44 (0)207 537 7515

crEAP[5]

crEAP is a python script which can determine the EAP type used by a wireless network through passive sniffing. crEAP performs well and is able to display results in real time, however only displays information about access points, not clients. Additionally, crEAP is only able to identify the following EAP types: 'EAP-MD5', 'EAP-PEAP' and 'EAP-TLS' (e.g. does not identify 'EAP-TTLS').

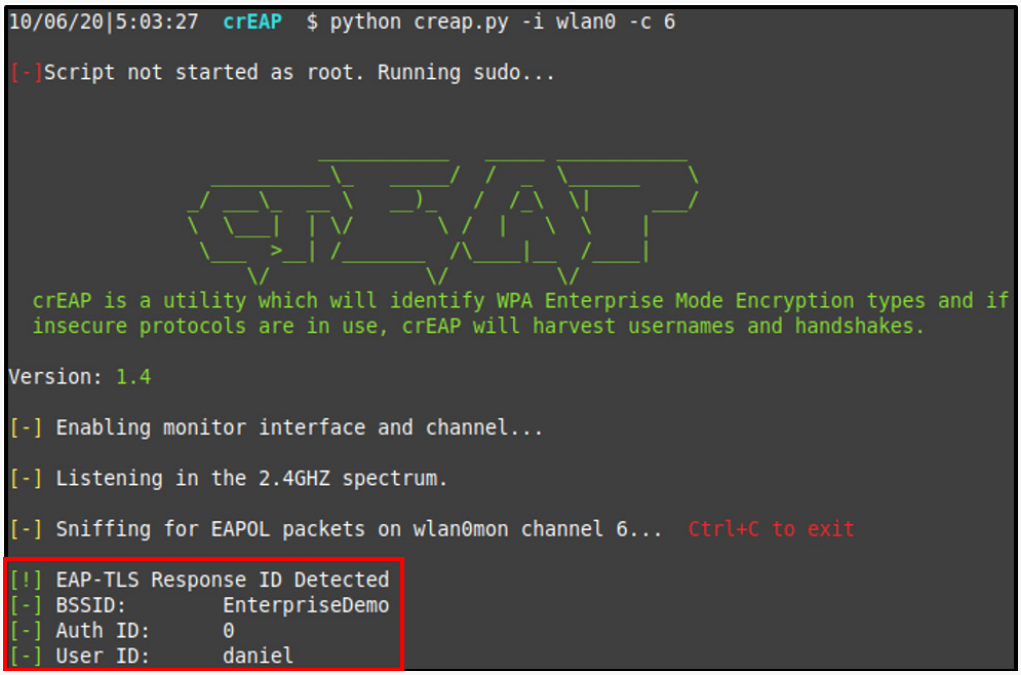

Figure 16 - crEAP being utilised to passively fingerprint a network

2.2 Wireshark

While tools can help in gaining a quick initial understanding of an environment, quality reconnaissance can also be performed easily in Wireshark. Utilising Wireshark also means that analysis can be performed live in real time, and additionally, any potential EAP methods which are not identified by fingerprinting tools can easily be manually identified.

The following display filter can be utilised to narrow down the Wireshark interface to the packets we are interested in:

```
((eap) && (eap.code == 1) && !(eap.type == 1) && !(ssl)) || eap.code == 3 || eap.code == 4
```

Where we would only like to display packets from: Group 1 or Group 2 or Group 3

- **Group 1**
  - eap - Display only EAP messages
  - eap.code == 1 - Display only EAP request messages
  - !(eap.type == 1) – Do not display any EAP identity messages
  - !(ssl) - With tunnelled EAP types, the tunnel which is created will appear within Wireshark as EAP requests. We are not interested in these specific packets and thus can choose to not display them, i.e. Do not display packets which have an SSL layer.
- **Group 2** eap.code == 3 – Display EAP success messages
- **Group 3** eap.code == 4 – Display EAP failure messages

5 https://github.com/Snizz/crEAP/blob/master/crEAP.py

E info@contextis.com
T +44 (0)207 537 7515

Utilising this display filter will show packets similar to the following:

No.	Time	Source	Destination	Protocol	Length	Info
26434	243.118878086	Tp-LinkT_8b:7c:3c	SamsungE_80:ea:45	EAP	78	Request, Cisco Wireless EAP / Lightweight EAP (EAP-LEAP)
26436	243.142642912	Tp-LinkT_8b:7c:3c	SamsungE_80:ea:45	EAP	62	Request, Tunneled TLS EAP (EAP-TTLS)
26471	243.320214579	Tp-LinkT_8b:7c:3c	SamsungE_80:ea:45	EAP	60	Success
41241	385.156950769	Tp-LinkT_8b:7c:3c	IntelCor_2c:4e:ee	EAP	78	Request, Cisco Wireless EAP / Lightweight EAP (EAP-LEAP)
41245	385.163955556	Tp-LinkT_8b:7c:3c	IntelCor_2c:4e:ee	EAP	62	Request, Protected EAP (EAP-PEAP)
41292	385.277391069	Tp-LinkT_8b:7c:3c	IntelCor_2c:4e:ee	EAP	60	Success
2418…	2583.4859647…	Tp-LinkT_8b:7c:3c	SamsungE_80:ea:45	EAP	78	Request, Cisco Wireless EAP / Lightweight EAP (EAP-LEAP)
2419…	2584.5228651…	Tp-LinkT_8b:7c:3c	SamsungE_80:ea:45	EAP	60	Failure

Figure 17 - EAP packets filtered within Wireshark

Supported EAP methods can be identified in the above as the authentication server will 'Request' supplicants for methods that it supports. When a 'Success' message is returned it can be assumed that the previous EAP type was accepted by the supplicant. When a 'Failure' message is returned it can be assumed that the supplicant did not accept any of the EAP types proposed by the authentication server during that specific negotiation.

contextis.com
E info@contextis.com
T +44 (0)207 537 7515

3 Attacking and Defending Enterprise Networks

This section at a glance

This section will focus on various weaknesses within different enterprise wireless configurations. We demonstrate various practical attacks being exploited for the offensive teams, but also specifically call out how to mitigate each weakness for the defensive teams.

Weaknesses discussed within this section are categorised based on the following:

Tunnelled vs Native EAP method weaknesses

— Lack of encryption due to use of native EAP authentication methods

— Weak anonymous identity configuration

Inherent weaknesses within the EAP method in use

— EAP-MSCHAPv2 hash cracking attack

— EAP-GTC clear-text credential disclosure

— EAP-MD5 hash cracking attack

— LEAP dictionary based password brute force attack

Weak client configurations

— Supplicant performs no server validation (rogue access point)

— Supplicant trusts external root certificate authority (rogue access point)

— EAP-FAST phase 0 access point impersonation attack (rogue access point)

— Supplicant allows permissive EAP negotiation (EAP downgrade attack)

Now that we know how to identify the EAP types which are supported by an Enterprise network, we can begin to start attacking and defending enterprise networks.

When attacking Enterprise networks, attacks can be performed based on a number of factors. The following categories have been created to group related attack vectors together in a logical manner.

— Tunnelled vs Native EAP Method Weaknesses – This section categorises issues which relate to attributes inherent to the EAP type chosen

— Inherent Weaknesses Within The EAP Method In Use – This section categorises issues which relate to weaknesses within a EAP method itself. An interesting note is that only native EAP methods appear within this section. This is due to the fact that weaknesses within tunnelled types take form as weak client configurations

— Weak Client Configurations – This section categorises issues which relate to the weak configuration of supplicants which connect to an enterprise network

E info@contextis.com
T +44 (0)207 537 7515

3.1 Tunnelled vs Native EAP method weaknesses

3.1.1 Lack of encryption due to use of Native EAP authentication methods

All native EAP methods contain an inherent weakness as the EAP framework assumes that messages are sent over a protected communication channel. Due to this, native EAP types are transmitted in the clear over the network, and communication can be passively sniffed by an attacker.

Furthermore, out of the five native EAP methods assessed in this paper, four of those methods are affected by vulnerabilities which can result in the recovery of user credentials (EAP-TLS is the only EAP native method which is not vulnerable to any known issues when configured appropriately). Thus, if a network does support the use of native EAP methods, an attacker could trivially attack a network through passive attacks/analysis, and also active rogue access point attacks against clients.

Mitigation

Ensure that Native EAP methods are not supported by the authentication server, allowing only tunnelled EAP methods within a network will ensure that authentication details are transferred within the encrypted "outer tunnel" and cannot be passively sniffed by malicious actors.

Currently, RADIUS servers can support native EAP types. pfSense FreeRADIUS in particular supports the following native methods:

- EAP-MD5
- EAP-GTC
- LEAP

While both pfSense FreeRADIUS and Windows NPS support 'EAP-MSCHAPv2'.

Note that although these EAP methods are supported and potentially enabled on RADIUS servers, there is little chance of running into a supplicant which supports the use of these native methods in the current wireless landscape.

3.1.2 Weak anonymous identity configuration

EAP Identity messages (detailed in the section titled 'Overview of EAP Messages', p.7) have the potential to disclose a user's identity (username in most cases), if the authentication method is not appropriately chosen and configured.

The purpose of the EAP Identity request/response is to facilitate request routing and EAP method negotiation. This is detailed by the following quote within IETF RFC 3748[6]:

> "It is RECOMMENDED that the Identity Response be used primarily for routing purposes and selecting which EAP method to use."

In practice however, this is generally not the case. Native EAP methods will typically only ask a user for their credentials for simplicities sake and ease of use. Both native and tunnelled EAP methods must respond to the authentication server's EAP identity request message however, so in almost all cases, supplicants will respond with the user's username within the identity response. This essentially means that all native EAP methods will disclose a user's username within Identity responses.

6 https://tools.ietf.org/html/rfc3748#section-5.1

E info@contextis.com
T +44 (0)207 537 7515

In the case of tunnelled EAP methods, although authentication occurs inside the outer tunnel where the identity value is not disclosed over the network, the outer tunnel must initially respond to an EAP Identity request. For the outer tunnel, supplicants will typically request an 'anonymous identity' value from the user which is then used to respond to the initial outer Identity request.

Demo

Tools specified within Section 2 'Fingerprinting Wireless Enterprise Networks' (p.15) are able to report on identities disclosed within EAP Identity responses. Additionally, this information can be looked up manually in Wireshark.

The following display filter can be used to narrow Wireshark packets to only EAP Identity responses:

```
(eap) && (eap.code == 2) && (eap.type == 1) && !(ssl)
```

Where:

- eap - Display only EAP messages
- eap.code == 2 - Display only EAP response messages
- eap.type == 1 – Display EAP identity messages
- !(ssl) - With tunnelled EAP types, the tunnel which is created will appear within Wireshark as EAP requests. We are not interested in these specific packets and thus can choose to not display them, i.e. Do not display packets which have an SSL layer.

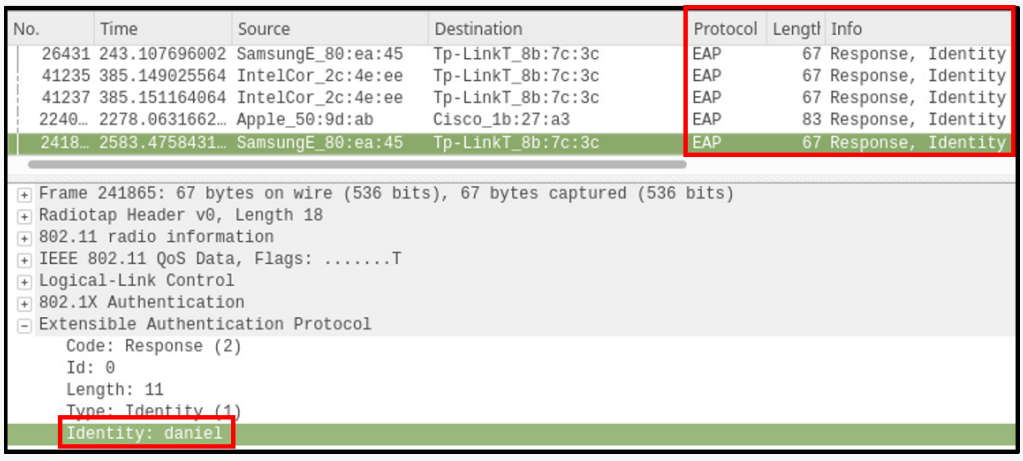

Figure 18 - EAP Identity Response messages disclose the configured Identity value

Tunnelled EAP methods allow for an anonymous identity to be sent within the outer tunnel, while the user's real identity can be sent within the encrypted inner tunnel. This can be seen in Figure 12, in the section titled 'Overview of EAP Messages' (p.11).

contextis.com
E info@contextis.com
T +44 (0)207 537 7515

The following image demonstrates this process in Wireshark:

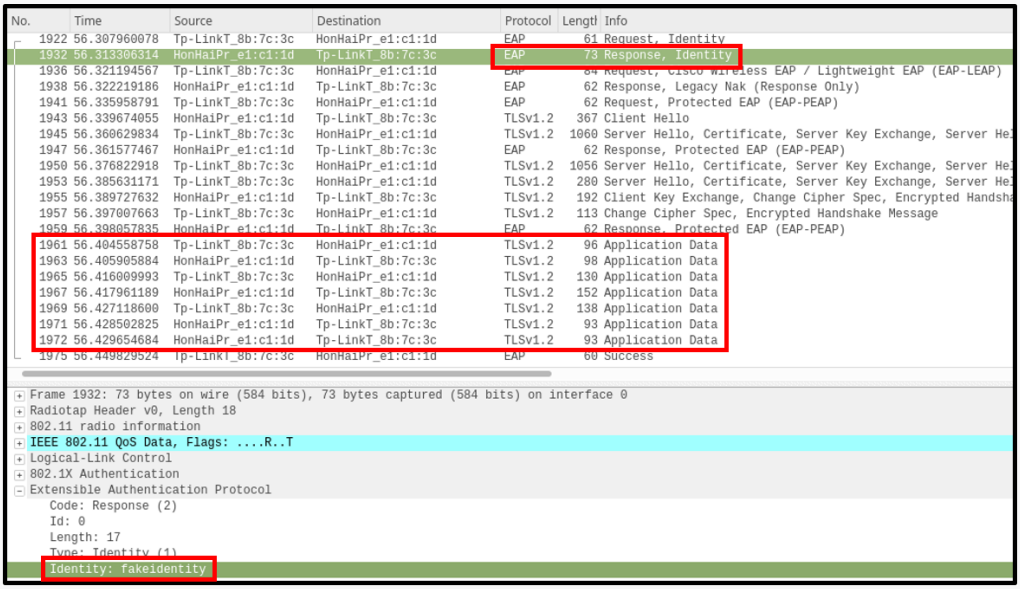

Figure 19 - When tunnelled authentication is used, the anonymous identity is disclosed in the outer tunnel (packet 1932), while the inner tunnel identity cannot be retrieved as it is encrypted (packets 1961 to 1972)

Mitigation

The following steps should be taken to ensure usernames are not leaked through identity responses:

1. Ensure only tunnelled EAP methods are used, preventing automatic disclosure through native EAP methods
2. Ensure that tunnelled EAP methods are configured to send an anonymous identity, preventing disclosure within the outer EAP authentication

Additionally, where possible, any device which connects to a corporate network should have managed wireless profiles which automate the network profile configuration. Not only will this prevent users from using weak configurations unintentionally, but managed profiles also bring additional options, e.g. iOS devices will not allow outer identities to be configured within the User Interface (UI), however this can be configured through a wireless profile.

Where wireless profiles cannot be managed, such as a Bring Your Own Device (BYOD), ensure users are given extremely clear steps to follow in order to configure their own network settings, as unaware users may accidentally disclose their own username within the anonymous identity.

3.2 Inherent weaknesses within the EAP method in use

Section notes:

- All demonstrations within this section assume EAP method identification through network fingerprinting have been performed. Fingerprinting techniques are detailed in Section 2 'Fingerprinting Wireless Enterprise Networks' (p.15)
- In a real world scenario, the process of de-authentication a client in order to force the client to associate to a rogue access point can be performed through de-authentication attacks. In the demonstrations within this section however, de-authentication attacks have been simulated through simply turning the clients network on and off

— This section will utilise rogue access points in its demonstrations, but will gloss over the setup and mitigations for these. Further details about rogue access attacks and tools are detailed in the next section 'Weak client configurations' (p.32). Furthermore, when rogue access points are setup, network fingerprinting was pre-performed in order to get the legitimate networks channel, and BSSID for imitating

3.2.1 EAP-MSCHAPv2 hash cracking attack

This authentication method is an implementation of Microsoft's Challenge-Handshake Authentication Protocol within EAP. Similar to the brute force attack identified against MSCHAPv1 within LEAP authentication, MSCHAPv2 is also vulnerable to a brute force attack against the methods challenge-handshake.

EAP-MSCHAPv2 can be used both as a native and tunnelled EAP method, thus can be sniffed by attackers passively, or captured through a rogue access point attack respectively.

Demo

Although EAP-MSCHAPv2 can be used as a native authentication method, most current day supplicants do not support native use, and will only support this method within a tunnelled EAP method. If native EAP is used, then an attacker could passively sniff the MSCHAPv2 challenge-handshake and perform an offline brute force attack to recover the password (similar to the demonstration within Section 3.2.4 'LEAP dictionary based password brute force attack', p.29). For this demo however, we will utilise a rogue access point to capture the challenge-handshake.

The following image demonstrates a WPA-Enterprise network being created using the 'EAPHammer' tool:

Figure 20 - Rogue Access Point is created using EAPHammer

E info@contextis.com
T +44 (0)207 537 7515

After the client has been de-authenticated from its legitimate PEAP/EAP-MSCHAPv2 network and connected to our rogue access point, a PEAP authentication attempt is captured in Wireshark. Within the network capture, we can see a TLS tunnel has been created where the inner EAP method is performed:

No.	Time	Source	Destination	Protocol	Lengtl	Info
2988	48.585678876	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	EAP	101	Request, Identity
2992	48.589928909	HonHaiPr_e1:c1:1d	Tp-LinkT_8b:7c:3c	EAP	109	Response, Identity
2996	48.591701552	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	EAP	102	Request, Protected EAP (EAP-PEAP)
3000	48.596093876	HonHaiPr_e1:c1:1d	Tp-LinkT_8b:7c:3c	TLSv1.2	405	Client Hello
3002	48.611331734	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	TLSv1.2	1499	Server Hello
3084	51.586744820	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	TLSv1.2	153	Server Hello
3085	51.587758194	HonHaiPr_e1:c1:1d	Tp-LinkT_8b:7c:3c	EAP	104	Response, Protected EAP (EAP-PEAP)
3086	51.589559186	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	TLSv1.2	136	Application Data
3087	51.590868557	HonHaiPr_e1:c1:1d	Tp-LinkT_8b:7c:3c	TLSv1.2	144	Application Data
3088	51.592582888	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	TLSv1.2	144	Application Data
3089	51.595856180	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	TLSv1.2	164	Application Data
3090	51.597824634	HonHaiPr_e1:c1:1d	Tp-LinkT_8b:7c:3c	TLSv1.2	198	Application Data
3091	51.603683985	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	TLSv1.2	135	Application Data
3092	51.606233095	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	EAP	100	Failure

Figure 21 - Inner EAP method (EAP-MSCHAPv2) is performed in the TLS tunnel in packets 3086 to 3991

Looking back at the output of our rogue access point, the following details are disclosed:

```
wlan0: STA ac:d1:b8:e1:c1:1d IEEE 802.11: authenticated
wlan0: STA ac:d1:b8:e1:c1:1d IEEE 802.11: associated (aid 1)
wlan0: CTRL-EVENT-EAP-STARTED ac:d1:b8:e1:c1:1d
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=25
wlan0: CTRL-EVENT-EAP-RETRANSMIT ac:d1:b8:e1:c1:1d


mschapv2: Thu Jun 11 02:24:55 2020
         domain\username:                daniel
         username:                       daniel
         challenge:                      2a:23:73:39:27:88:4c:a8
         response:                       36:b0:67:e1:39:7e:5b:6e:fd:cf:6e:24:bd:0b:cf:a3:
28:e1:35:16:83:14:16:49

         jtr NETNTLM:                    daniel:$NETNTLM$2a23733927884ca8$36b067e1397e5b6
efdcf6e24bd0bcfa328e1351683141649

         hashcat NETNTLM:                daniel::::36b067e1397e5b6efdcf6e24bd0bcfa328e135
1683141649:2a23733927884ca8
```

Figure 22 - Rogue access point has captured a EAP-MSCHAPv2 authentication attempt

The MSCHAPv2 challenge and response disclosed above are the required bits of information to perform a brute force attack and recover the clear-text password. These details are also provided in properly formatted strings to be passed to password cracking tools (John The Ripper 'jtr' and Hashcat 'hashcat' above).

Also of note when talking about hash cracking are online cracking services such as 'crack.sh'. These services are able to provide guarantees on key recovery, and in the case of MSCHAPv2 is able to search the entire key space to recover the key. This highlights the real risk if these challenge-response hashes are disclosed, and the need to implement appropriate mitigations.

The following example demonstrates the Hashcat tool being used to crack the captured challenge-response hash. The hash has been formatted with the following format for EAP-MSCHAPv2 authentication:

<username>::::<response>:<challenge>

The following image shows the Hashcat tool being used to perform the password cracking:

```
13/06/20|11:55:06  hashes  $ hashcat -m 5500 -a 0 'daniel::::36b067e1397e5b6efdcf6e24bd0bcfa32
8e1351683141649:2a23733927884ca8' rockyou.txt --show
daniel::::36b067e1397e5b6efdcf6e24bd0bcfa328e1351683141649:2a23733927884ca8 password
```

Figure 23 - The password 'password' was successfully recovered

Mitigation

MSCHAPv2 can be feasibly cracked within a day by utilising cloud cracking services. Thus a stronger EAP method which is not susceptible to credential recovery attacks should be utilised e.g. EAP-TLS within a tunnelled EAP method.

Additionally, utilising EAP-MSCHAPv2 within an encrypted TLS tunnel, i.e. within a tunnelled EAP method, is also not a suitable mitigation. Tunnelled authentication methods can still be targeted through rogue access point attacks, leading clients to be affected by the same hash cracking attack.

Furthermore, although not mentioned within this whitepaper, it is also possible to perform relay attacks against the inner MSCHAPv2 authentication, whereby attackers can relay EAP-MSCHAPv2 authentication between a client and an authentication server in order to authenticate without cracking any hashes ("PEAP Relay Attacks with wpa_sycophant" Michael Kruger, entry posted January 19, 2013, accessed August 18, 2020[7]).

3.2.2 EAP-GTC clear-text credential disclosure

This authentication method was created by Cisco as an alternative to Microsoft's EAP-MSCHAPv2 authentication method. EAP-GTC authentication can integrate with multiple identity servers, and also supports many credential types, such as:

Protocol	Clear-text	NT hash	MD5 hash	Salted MD5 hash	SHA1 hash	Slated SHA1 hash	UNIX Crypt
EAP-GTC	✓	✓	✓	✓	✓	✓	✓

Figure 24 - Credential types supported by EAP-GTC authentication[8]

This EAP method becomes insecure when configured to authenticate using clear-text credentials. Although this method is useful for One-Time-Pin (OTP) tokens, many implementations will have users utilise usernames and passwords to authentication, resulting in these potentially being disclosed in cleartext.

Demo

Due to a lack of support for native EAP-GTC authentication in current day supplicants, we have chosen to demonstrate EAP-GTC as an inner authentication method to the PEAP method. This means that the EAP-GTC authentication details will be captured through a rogue access point, however if native EAP-GTC authentication is used then the same values will be disclosed in clear-text over the air.

7 https://sensepost.com/blog/2019/peap-relay-attacks-with-wpa_sycophant/

8 Source: https://documentation.meraki.com/zGeneral_Administration/Other_Topics/PEAPv1%2F%2FEAP-GTC_support_on_a_Windows_client

The following image demonstrates a WPA-Enterprise network being created using the 'EAPHammer' tool:

```
[11-Jun-20 00:52:59] 192.168.0.23 eaphammer > ./eaphammer -i wlan0 -e EnterpriseDemo -b
34:E8:94:8B:7C:3C -c 1 --auth wpa-eap --creds --negotiate gtc-downgrade

                    .__
 ____ _____  ______ |  |__ _____  ______ ____ _______
/ __ \\__  \ \____ \|  |  \\__  \ /  ___// __ \\_  __ \
\  ___/ / __ \|  |_> >   Y  \/ __ \|  \___\  ___/ |  | \/
 \___  >____  /   __/|___|  (____  /____  >\___  >|__|
     \/     \/|__|        \/     \/     \/     \/

                      Rogue AP attacks for operators.

                           Version:  1.12.1
                          Codename:  Power Overwhelming
                            Author:  @s0lst1c3
                           Contact:  gabriel@specterops.io


[?] Am I root?
[*] Checking for rootness...
[*] I AM ROOOOOOOOOOOT
[*] Root privs confirmed! 8D
[*] Saving current iptables configuration...
[*] Reticulating radio frequency splines...

[*] Using nmcli to tell NetworkManager not to manage wlan0...

100%|                                                                                | 1/1 [00:01<00:00,  1.00s/it]

[*] Success: wlan0 no longer controlled by NetworkManager.
[*] WPA handshakes will be saved to /opt/eaphammer/loot/wpa_handshake_capture-2020-06-11
-00-53-06-iPhsMJHLisJxDamcWupy1UhsQgzPUYxc.hccapx
Configuration file: /opt/eaphammer/tmp/hostapd-2020-06-11-00-53-06-NUu2Ewl6kimpSebypYzhV
Sa8DMLUFBrS.conf

[hostapd] AP starting...

wlan0: interface state UNINITIALIZED->COUNTRY_UPDATE
Using interface wlan0 with hwaddr 34:e8:94:8b:7c:3c and ssid "EnterpriseDemo"
wlan0: interface state COUNTRY_UPDATE->ENABLED
wlan0: AP-ENABLED


Press enter to quit...
```

Figure 25 - Rogue Access Point is created using EAPHammer

After the client has been de-authenticated from its legitimate PEAP/EAP-GTC network and connected to our rogue access point, a PEAP authentication attempt is captured in Wireshark. Within the network capture, we can see a TLS tunnel has been created where the inner EAP method is performed:

287	8.391429437	SamsungE_80:ea:45	Tp-LinkT_8b:7c:3c	EAP	109 Response, Identity
289	8.393192434	Tp-LinkT_8b:7c:3c	SamsungE_80:ea:45	EAP	102 Request, Protected EAP (EAP-PEAP)
292	8.398782314	SamsungE_80:ea:45	Tp-LinkT_8b:7c:3c	TLSv1.2	265 Client Hello
295	8.414128221	Tp-LinkT_8b:7c:3c	SamsungE_80:ea:45	TLSv1.2	1499 Server Hello, Certificate, Server Key Exchange,
300	8.419324358	SamsungE_80:ea:45	Tp-LinkT_8b:7c:3c	EAP	104 Response, Protected EAP (EAP-PEAP)
302	8.429327403	Tp-LinkT_8b:7c:3c	SamsungE_80:ea:45	TLSv1.2	1001 Server Hello, Certificate, Server Key Exchange,
310	8.439904242	SamsungE_80:ea:45	Tp-LinkT_8b:7c:3c	TLSv1.2	197 Client Key Exchange, Change Cipher Spec, Encryp
312	8.441993777	Tp-LinkT_8b:7c:3c	SamsungE_80:ea:45	TLSv1.2	153 Change Cipher Spec, Encrypted Handshake Message
314	8.447548511	SamsungE_80:ea:45	Tp-LinkT_8b:7c:3c	EAP	104 Response, Protected EAP (EAP-PEAP)
316	8.449473415	Tp-LinkT_8b:7c:3c	SamsungE_80:ea:45	TLSv1.2	136 Application Data
318	8.454816852	SamsungE_80:ea:45	Tp-LinkT_8b:7c:3c	TLSv1.2	144 Application Data
320	8.456360271	Tp-LinkT_8b:7c:3c	SamsungE_80:ea:45	TLSv1.2	144 Application Data
322	8.460093908	SamsungE_80:ea:45	Tp-LinkT_8b:7c:3c	TLSv1.2	146 Application Data
324	8.461838336	Tp-LinkT_8b:7c:3c	SamsungE_80:ea:45	TLSv1.2	135 Application Data
326	8.465917866	SamsungE_80:ea:45	Tp-LinkT_8b:7c:3c	TLSv1.2	137 Application Data

Figure 26 - EAP-GTC negotiation occurs within the encrypted TLS tunnel starting at packet 316

E info@contextis.com
T +44 (0)207 537 7515

Once the client has connected to the rogue access point, the following details are disclosed:

```
wlan0: STA 6c:c7:ec:80:ea:45 IEEE 802.11: authenticated
wlan0: STA 6c:c7:ec:80:ea:45 IEEE 802.11: associated (aid 1)
wlan0: CTRL-EVENT-EAP-STARTED 6c:c7:ec:80:ea:45
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
wlan0: CTRL-EVENT-EAP-RETRANSMIT 6c:c7:ec:80:ea:45
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=25


GTC: Thu Jun 11 00:53:48 2020
         username:      daniel
         password:      password
wlan0: CTRL-EVENT-EAP-FAILURE 6c:c7:ec:80:ea:45
```

Figure 27 - EAP-GTC authentication is captured with plain-text credentials disclosed

Although EAP-GTC supports multiple credential types. By default, all devices tested disclosed their credentials in plain-text, as the devices were not pre-configured/ setup to support differing EAP-GTC credential types. This is the default configuration however, and can also be utilised to perform plain-text credential downgrade attacks (this attack is detailed in Section 3.3.4 'Supplicant allows permissive EAP negotiation (EAP downgrade attack)', p.44).

Mitigation

This EAP method should only be used with specific OTP implementations, not authentication requiring credentials. Many of the hashed credential types supported by EAP-GTC are also weak however. As such, a stronger EAP method should be utilised for authentication, e.g. EAP-TLS within a tunnelled EAP method.

3.2.3 EAP-MD5 hash cracking attack

EAP-MD5 utilises a simple challenge response process in order to authenticate a client to the server. The authentication process involves the supplicant concatenating a randomly chosen id value, password, and a random challenge string. A MD5 hash of this value is then created:

R = MD5(id | P | C)
*where '|' indicates that the values are concatenated together

The authentication server is able to compute this value itself to compare with the value given by the supplicant, meaning the server will have a copy of the user's plaintext password. If the values match, then the user is authenticated.

A good overview of the EAP-MD5 authentication process along with how to break it is given in the paper 'How to Break EAP-MD5' (Liu, F., & Xie, T. (2017). How to Break EAP-MD5. 6th International Workshop on Information Security Theory and Practice, 49-57. doi:10.1007/978-3-642- 30955-7_6)[9].

The 'eapmd5pass' tool can be utilised to crack EAP-MD5 authentication. This tool was developed by Joshua Wright (the same author as 'asleap' for LEAP cracking).

Demo

Due to a lack of support for native EAP-MD5 authentication in current day supplicants, we have chosen to demonstrate EAP-MD5 as a PEAP inner authentication method. The method to capture the required MD5 details will be performed through a rogue access point, however if native EAP-MD5 is used then the same values will be disclosed in clear-text over the network.

9 https://hal.inria.fr/hal-01534313/document

E info@contextis.com
T +44 (0)207 537 7515

The following image demonstrates a WPA-Enterprise network being created using the 'EAPHammer' tool:

Figure 28 - Rogue Access Point is created using EAPHammer

After the client has been de-authenticated from its legitimate PEAP/EAP-MD5 network and connected to our rogue access point, a PEAP authentication attempt is captured in Wireshark.

Within the network capture, we can see a TLS tunnel has been created where the inner EAP method is performed:

No.	Time	Source	Destination	Protocol	Length	Info
377	0.573201522	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	EAP	101	Request, Identity
378	0.575469676	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	EAP	102	Request, Protected EAP (EAP-PEAP)
379	0.594561187	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	TLSv1.2	1499	Server Hello
381	0.611997047	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	TLSv1.2	153	Server Hello
382	0.614634719	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	TLSv1.2	136	Application Data
383	0.618578284	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	TLSv1.2	144	Application Data
384	0.621693628	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	TLSv1.2	153	Application Data
386	0.625814528	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	TLSv1.2	135	Application Data

Figure 29 - EAP-MD5 negotiation occurs within the encrypted TLS tunnel starting at packet 382

E info@contextis.com
T +44 (0)207 537 7515

Once the client has connected to the rogue access point, the following details are disclosed:

```
wlan0: STA ac:d1:b8:e1:c1:1d IEEE 802.11: associated (aid 1)
wlan0: CTRL-EVENT-EAP-STARTED ac:d1:b8:e1:c1:1d
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=25


eap-md5: Thu Jun 11 02:47:54 2020
         domain\username:              daniel
         username:                     daniel
         challenge:                    9c:a1:ff:a4:1e:22:16:e8:54:e7:3e:8d:fe:87:49:1d
         response:                     d4:ae:c5:78:f4:3a:21:61:08:7e:22:60:2d:3d:40:1f

         eap_id:                       110

         jtr NETNTLM:                  daniel:$chap$110*d4aec578f43a2161087e22602d3d401
f*9ca1ffa41e2216e854e73e8dfe87491d


         hashcat NETNTLM:              d4aec578f43a2161087e22602d3d401f:9ca1ffa41e2216e
854e73e8dfe87491d:6e
```

Figure 30 - Rogue access point has captured a EAP-MD5 authentication attempt

The MD5 challenge and response disclosed above are the required bits of information to perform a brute force attack and recover the password. These details are also provided in properly formatted strings to be passed to password cracking tools (John The Ripper 'jtr' and Hashcat 'hashcat' above).

The Hashcat hash has been formatted with the following format for EAP-MD5 authentication:

<response>:<challenge>:<eap_id-in-hex>

The following image shows the Hashcat tool being used to perform the password cracking:

```
12/06/20|4:50:01  hashes  $ hashcat -m 4800 'd4aec578f43a2161087e22602d3d401f:9ca1ffa41e2216e8
54e73e8dfe87491d:6e' rockyou.txt --show
d4aec578f43a2161087e22602d3d401f:9ca1ffa41e2216e854e73e8dfe87491d:6e:password
```

Figure 31- The password 'password' was successfully recovered

Mitigation

EAP-MD5 should not be utilised in any organisations wireless networks. Native authentication would allow an attacker to passively sniff the MD5 challenge response, while tunnelled methods could still be exploited through rogue access points, and weak client configurations. Once an MD5 challenge-response hash is recovered, passwords can be trivially recovered in a short time due to the simplicity of the protocol. Instead organisations should opt for a stronger EAP method such as PEAP/EAP-TLS and EAP-TTLS/EAP-TLS.

3.2.4 LEAP dictionary based password brute force attack

Lightweight Extensible Authentication Protocol (LEAP) is a Cisco-proprietary protocol which utilises a modified version of MSCHAPv1 to perform authentication. MSCHAPv1 was discovered to be vulnerable to a dictionary brute force attack, meaning if an attacker is able to capture the challenge-handshake messages of a MSCHAPv1 authentication attempt, those packets could be taken offline and feasibly cracked (crackable within less than a day by current computing standards).

LEAP is also a native EAP method, this means that the messages transmitted by this EAP method are not transmitted within an encrypted tunnel, and thus can be passively sniffed by attackers.

A practical tool to perform brute force attacks against LEAP was released by Joshua Wright at DEFCON 11 (2003), dubbed 'asleap'. Cisco has since released a security advisory response to the public disclosure of the tool[10].

Demo

For this demo, we will be attacking an Enterprise wireless network setup to accept LEAP authentication.

The following airodump-ng output displays our target:

```
11/06/20|3:10:31  leap  $ sudo airodump-ng --essid EnterpriseDemo -c 6 -w LEAP wlan1mon
03:10:33  Created capture file "LEAP-01.cap".


 CH  6 ][ Elapsed: 48 s ][ 2020-06-11 03:11

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 34:E8:94:8B:7C:3C  -23 100      476       35    0   6  130   WPA  CCMP   MGT  EnterpriseDemo
```

Figure 32 - Sniffing packets related to the 'EnterpriseDemo' network and saving those packets to the file LEAP.cap

LEAP is a native authentication method so no active attacks are required, simply sniffing for a LEAP authentication is sufficient.

After the client has been de-authenticated from its legitimate LEAP network and connected to our rogue access point, a series of LEAP authentication packets are captured:

No.	Time	Source	Destination	Protocol	Lengtl	Info
9448	400.033432973	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	EAP	103	Request, Identity
9454	400.036511443	HonHaiPr_e1:c1:1d	Tp-LinkT_8b:7c:3c	EAP	109	Response, Identity
9460	400.039945923	HonHaiPr_e1:c1:1d	Tp-LinkT_8b:7c:3c	EAP	109	Response, Identity
9464	400.053769782	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	EAP	120	Request, Cisco Wireless EAP / Lightweight EAP (EAP-LEAP)
9466	400.055007859	HonHaiPr_e1:c1:1d	Tp-LinkT_8b:7c:3c	EAP	136	Response, Cisco Wireless EAP / Lightweight EAP (EAP-LEAP)
9468	400.066537606	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	EAP	102	Success
9470	400.067651022	HonHaiPr_e1:c1:1d	Tp-LinkT_8b:7c:3c	EAP	120	Request, Cisco Wireless EAP / Lightweight EAP (EAP-LEAP)
9471	400.073897024	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	EAP	136	Response, Cisco Wireless EAP / Lightweight EAP (EAP-LEAP)
9473	400.075002838	Tp-LinkT_8b:7c:3c	HonHaiPr_e1:c1:1d	EAP	102	Success

Figure 33 - LEAP authentication packets captured within Wireshark

The only details required in order to perform a brute force attack against LEAP authentication is the challenge/response values disclosed during the authentications challenge-handshake process.

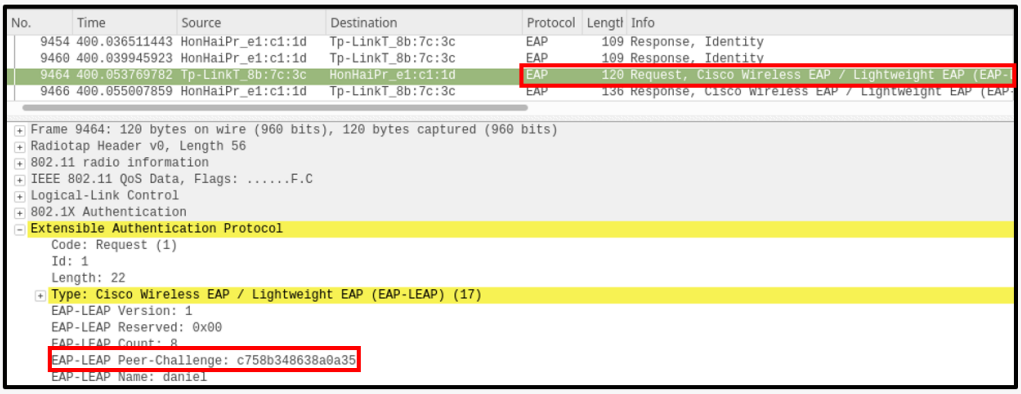

Figure 34 - LEAP challenge is disclosed within the initial EAP LEAP Request (packet 9464)

10 https://www.cisco.com/en/US/tech/tk722/tk809/technologies_security_notice09186a00801aa80f.html

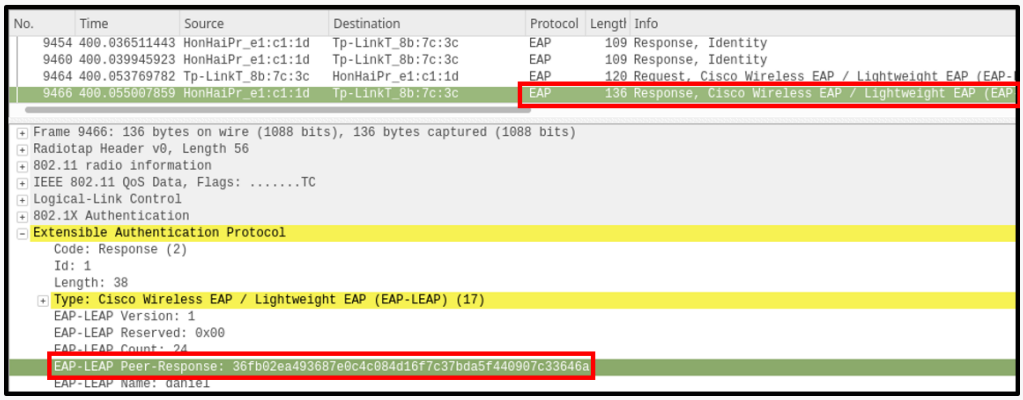

Figure 35 - LEAP response is disclosed within the initial EAP LEAP Response (packet 9466)

Once these values have been recovered, the 'asleap' tool can be utilised to crack the LEAP handshake and recover the original credentials. The cracking process is as follows:

1. Using a wordlist, generate a file with passwords and hashes, along with an index file for the hash file. asleap comes with a tool 'genkeys' which can perform this with the following syntax:

   ./genkeys -r <wordlist>.txt -f <wordlist>.hash -n <wordlist>.index

2. Run the asleap tool with the following parameters:

   ./asleap -r <airodump-ng_cap> -f <wordlist>.hash -n <wordlist>.index -v -C <LEAP-challenge> -R <LEAP-response>

3. The end result will display the NT hash and plaintext password if the password was within the dictionary supplied

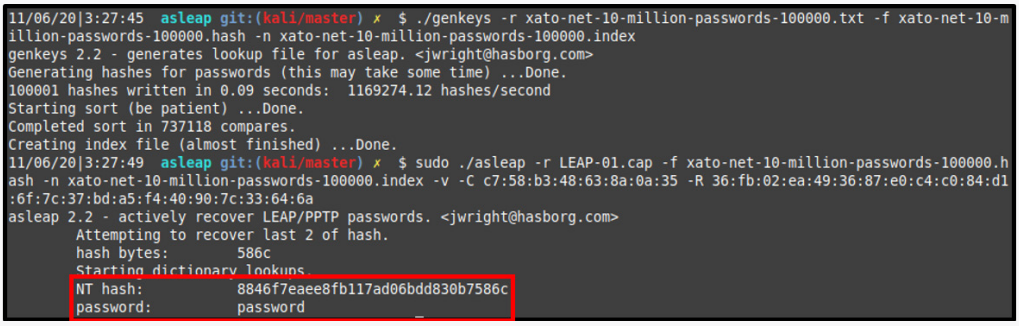

Figure 36 - Cracking process performed using the 'asleap' tool

Mitigation

The only mitigation for the inherent weakness within LEAP is to not use LEAP authentication, instead opt for a stronger EAP authentication method such as PEAP/EAP-TLS and EAP-TTLS/EAP-TLS.

Additionally, ensure that a tunnelled EAP method is utilised. LEAP authentication cannot be used as an inner authentication method by a tunnelled EAP method, thus weaknesses within LEAP cannot be hidden within an encrypted tunnel.

E info@contextis.com
T +44 (0)207 537 7515

3.3 Weak client configurations

Section notes and rogue access point overview:

- This section will focus on weak client configurations, exploited through the use of rogue access points
- The tool 'hostapd-wpe' is a popular tool to create a rogue access point. An excellent wrapper for 'hostapd-wpe' is the tool 'EAPHammer'[11], created by Gabriel Ryan (@ s0lst1c3). This tool allows for easy rogue access point creation without the need for creating hostapd-wpe configuration files manually, and is used primarily in this section
- Although not specifically demonstrated within this whitepaper, other great rogue access point tools exist, e.g. 'Bearte_AP'[12], a tool developed by Michael Kruger (@_cablethief) which can be combined with Kruger's additional tool 'wpa_ sycophant' in order to perform MSCHAPv2 replay attacks through rogue access points

In order to create a rogue access point, an attacker controlled access point is created which mimics a target network. The following information must be mimicked by the rogue access point:

- **ESSID** The name of the network.
- **BSSID** The Basic Service Set Identifier of the target network
- **Channel** The current channel that the target network is on

When a client attempts to connect or re-connect to a network, the client will look at the above information when deciding whether it will attempt to authenticate to a specific access point. Clients will do this for ease of use purposes, e.g. when you enter your home or walk into an office. If you have previously connected to a network the client will attempt to search for that network and automatically connect again.

However, if the above values in a rogue access point match a target network, and the signal strength transmitting from the rogue access point is stronger than the real network, clients will automatically attempt to associate (connect) to the rogue access point.

Demonstrations showing a rogue access point being used to capture and recover credentials can be found in the following attacks covered previously within Section 3.2 'Inherent weaknesses within the EAP method in use' (p.22):

- EAP-MSCHAPv2 Hash Cracking Attack
- EAP-GTC Clear-Text Credential Disclosure
- EAP-MD5 Hash Cracking Attack

3.3.1 Supplicant performs no server validation (rogue access point)

Tunnelled EAP methods create an encrypted TLS tunnel in order to securely transmit a secondary EAP method within the tunnel to perform authentication. This tunnel is created through the use of X.509 digital certificates, with the authentication server presenting their certificate, and the client optionally presenting theirs. When an authentication server presents their certificate to a client, the client is able to verify the certificate presented in order to ensure that the supplicant is connecting to the intended server.

11 https://github.com/s0l-st1c3/eaphammer

12 https://github.com/sense-post/berate_ap

However, supplicants utilised across the many Wi-Fi enabled devices allow users to configure if server certificate validation should be enabled or disabled. When disabled, the supplicant will connect to any access point without validating the identity of the authentication server. This essentially nullifies the benefits of the secure encrypted tunnel, and opens the supplicant up to all attacks against its supported native EAP methods (the inner authentication).

Demo

Just because a client attempts to associate with a rogue access point, does not mean that the client will willingly attempt to automatically authenticate and disclose their credentials.

The following table outlines the behaviour of different systems when encountered with a rogue access point. These results have been obtained from systems which have initially authenticated to a network via an insecure default profile, typical of non-managed devices which have been configured through a User Interface (default configurations for each device are explained further in the scenario details below the table below):

System	Insecure Default Setup	Behaviour Upon Entering Rogue Access Point
Andriod Device	No certificate validation	1. Automatic disconnect from the legitimate network 2. Automatic reconnect to rogue access point 3. Credentials automatically disclosed on reconnect
Linux System	No certificate validation	1. No automatic disconnect from the legitimate network* 2. Upon forced deauthentication, automatic reconnect will connect to the rogue access point 3. Credentials automatically disclosed on reconnect.
iOS Device	Certificate validation performed. SSID & certificate are correlated together	1. Automatic disconnect from legitimate network 2. Automatic reconnect to rogue access point, however certificate validation will fail, resulting in no successful authentication. The device will attempt to retry authentication multiple times until further attempts are stopped 3. User interaction is required to connect to network again, at which point a prompt to trust the rogue access point's certificate will display. 4. Credentials are disclosed if the malicious certificate is trusted

E info@contextis.com
T +44 (0)207 537 7515

System	Insecure Default Setup	Behaviour Upon Entering Rogue Access Point
Windows System	Certificate validation performed. SSID & certificate are correlated together	1. No automatic disconnect from the legitimate network* 2. Upon forced deauthentication, automatic reconnect will connect to the rogue access point 3. User is prompted to trust that the rogue access point is the legitimate access point 4. Credentials are disclosed if the malicious certificate is trusted
MacOS System	Certificate validation performed. SSID & certificate are correlated together	1. Automatic disconnect from legitimate network 2. Automatic reconnect to rogue access point, however certificate validation will fail 3. MacOS will automatically attempt to re-validate the new rogue access point by displaying a prompt to the user to trust the malicious authentication servers certificate 4. The system will automatically authenticate if the malicious certificate is trusted, resulting in credentials being disclosed

* The fact that the client did not automatically disconnect from the legitimate network is also dependant on the signal strength of the rogue access point. During testing, these findings were inconsistent based on the signal strength of the rogue access point, with devices automatically de-authenticating/re-authenticating if the signal strength was strong enough. A safe assumption has been made however, it the most occurring test case being documented.

The above table was populated by running a standard rogue access point attack against multiple devices. These scenarios have been detailed below for reference:

Samsung Galaxy Note9 (Android 9)
An Android mobile device is authenticated to a WPA Enterprise network 'EnterpriseDemo', and was manually configured to not perform certificate validation of the authentication servers certificate (as shown in Appendix – Certificate Validation Configuration Options – Android Devices).

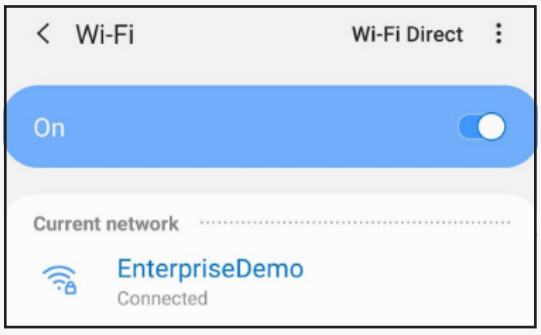

Figure 37 - Initially the device is connected to the legitimate access point

```
[*] Success: wlan0 no longer controlled by NetworkManager.
[*] WPA handshakes will be saved to /opt/eaphammer/loot/wpa_handshake_capture-2020-08-09-20-30-51-wtAFYE
w.hccapx

Configuration file: /opt/eaphammer/tmp/hostapd-2020-08-09-20-30-51-M5DtUrUFYBqq08VaQzHFZzlm3MuPUEuf.conf
[hostapd] AP starting...

wlan0: interface state UNINITIALIZED->COUNTRY_UPDATE
Using interface wlan0 with hwaddr 34:e8:94:8b:7c:3c and ssid "EnterpriseDemo"
wlan0: interface state COUNTRY_UPDATE->ENABLED
wlan0: AP-ENABLED
```

Figure 38 - A malicious rogue access point is setup utilising the EAPHammer tool

Figure 39 - The Android device automatically disconnects when detecting a stronger signal from the same network

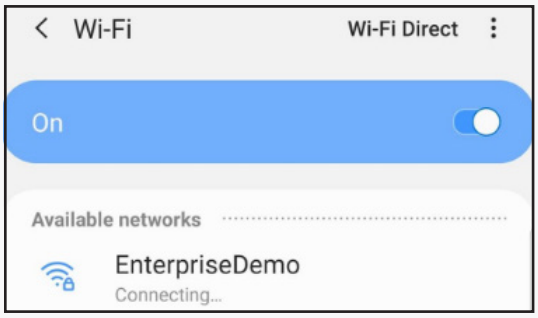

Figure 40 - The Android device attempts to reconnect to the network, i.e. to the rogue access point

```
wlan0: CTRL-EVENT-EAP-FAILURE 6c:c7:ec:80:ea:45
wlan0: STA 6c:c7:ec:80:ea:45 IEEE 802.1X: authentication failed - EAP type: 0 (unknown)
wlan0: STA 6c:c7:ec:80:ea:45 IEEE 802.1X: Supplicant used different EAP type: 25 (PEAP)
wlan0: STA 6c:c7:ec:80:ea:45 IEEE 802.11: authenticated
wlan0: STA 6c:c7:ec:80:ea:45 IEEE 802.11: associated (aid 1)
wlan0: CTRL-EVENT-EAP-STARTED 6c:c7:ec:80:ea:45
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=25

mschapv2: Sun Aug  9 20:31:23 2020
         domain\username:            daniel
         username:                   daniel
         challenge:                  06:be:34:5a:96:58:39:68
         response:                   39:67:30:fd:28:a7:5f:1d:6b:60:23:1a:38:23:08:04:0c:55:ba:44:85:bf:54:1d

         jtr NETNTLM:                daniel:$NETNTLM$06be345a96583968$396730fd28a75f1d6b60231a382308040c55ba4485bf541d

         hashcat NETNTLM:            daniel:::396730fd28a75f1d6b60231a382308040c55ba4485bf541d:06be345a96583968
```

Figure 41 - The users hashed credential is disclosed to the rogue access point

E info@contextis.com
T +44 (0)207 537 7515

iPhone 6S (iOS 13.1 iOS)
An iOS mobile device is authenticated to a WPA Enterprise network 'EnterpriseDemo', with the user trusting the legitimate authentication servers certificate (as shown in the 'iOS devices' section of the Appendix, p.50).

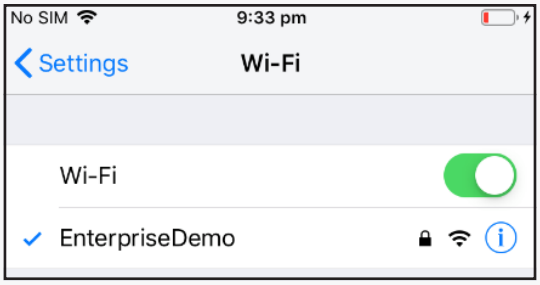

Figure 42 - Initially the device is connected to the legitimate access point

A rogue access point was once again created, similarly to the Android rogue access point scenario above.

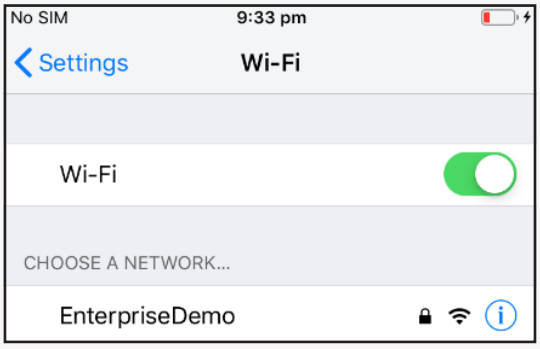

Figure 43 - The iOS device detects an access point for the network with a stronger signal and disconnects from the current access point

Once the iOS device has disconnected from the original access point, the device will attempt to connect to the rogue access point. This connection will fail however, as the device has not trusted the SSID & server certificate combination. The user will have to manually reconnect to the network, at which point the rogue access points certificate will be shown.

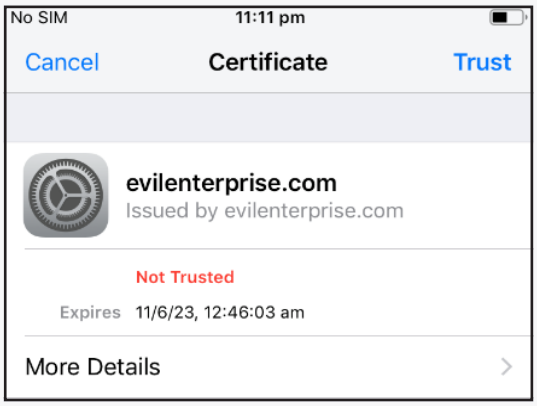

Figure 44 - The device will ask the user to trust the certificate of the rogue access point

```
wlan0: STA e4:e4:ab:56:f6:68 IEEE 802.11: authenticated
wlan0: STA e4:e4:ab:56:f6:68 IEEE 802.11: authenticated
wlan0: STA e4:e4:ab:56:f6:68 IEEE 802.11: associated (aid 1)
wlan0: CTRL-EVENT-EAP-STARTED e4:e4:ab:56:f6:68
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=25

GTC: Wed Aug 12 00:40:26 2020
        username:       daniel
        password:       password
```

Figure 45 - The users credentials are disclosed to the rogue access point

Linux Desktop (wpasupplicant version 2:2.6-15ubuntu2.4)
A Linux system is authenticated to a WPA Enterprise network 'EnterpriseDemo', and was manually configured to not perform certificate validation of the authentication servers certificate (as shown in the 'Linux systems' section of the Appendix at the end of the paper, p.51).

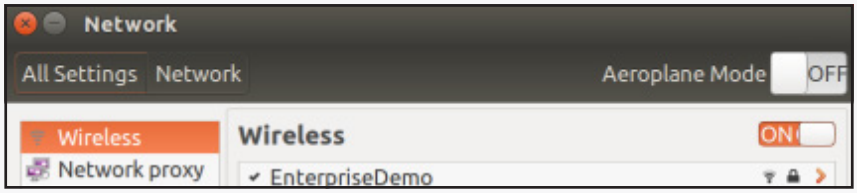

Figure 46 - Initially the host is connected to the legitimate access point

A rogue access point was once again created, similarly to the Android rogue access point scenario above. The Linux system did not automatically disconnect from the legitimate access point upon the creation of the rogue access point. To simulate a de-authentication attack, the systems network manager was restarted.

```
wlan0: STA ac:d1:b8:e1:c1:1d IEEE 802.11: authenticated
wlan0: STA ac:d1:b8:e1:c1:1d IEEE 802.11: associated (aid 1)
wlan0: CTRL-EVENT-EAP-STARTED ac:d1:b8:e1:c1:1d
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=25

mschapv2: Sun Aug  9 20:47:35 2020
        domain\username:                daniel
        username:                       daniel
        challenge:                      c2:a2:f9:f6:d3:47:cd:1e
        response:                       a7:af:f1:93:65:a1:c0:25:a3:a6:b0:63:98:3a:3b:64:cb:fa:71:67:1b:06:af:bd

        jtr NETNTLM:                    daniel:$NETNTLM$c2a2f9f6d347cd1e$a7aff19365a1c025a3a6b063983a3b64cbfa71671b06afbd

        hashcat NETNTLM:                daniel::::a7aff19365a1c025a3a6b063983a3b64cbfa71671b06afbd:c2a2f9f6d347cd1e
```

Figure 47 - Upon reconnecting to the network, the Linux system does not validate the rogue access point resulting in credential hashes being disclosed

Windows Desktop (Windows 10)
A Windows host is authenticated to a WPA Enterprise network 'EnterpriseDemo', with the user trusting the legitimate authentication servers certificate by default (as shown in the 'Windows systems' section of the Appendix, p.52).

A rogue access point was once again created, similarly to the Android rogue access point scenario above. Similar to the Linux desktop scenario, the Windows host did not automatically disconnect from the legitimate access point and reconnect to the rogue access point.

After simulating a de-authentication attack by restarting the systems network adaptor, the system did automatically connect to the rogue access point.

Upon connecting to the rogue access point, the following warning is displayed to the user:

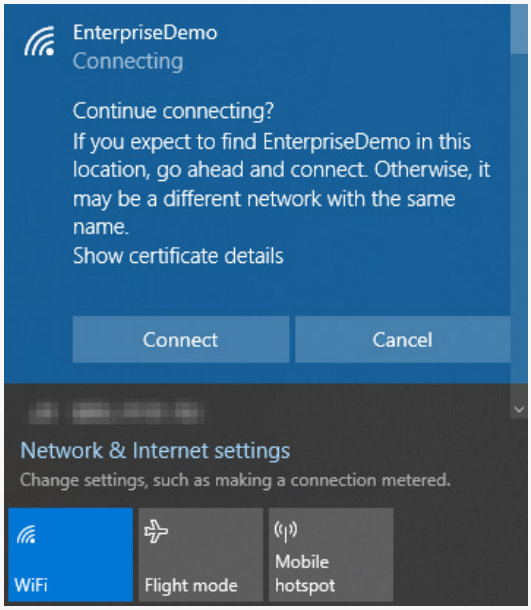

Figure 48 - The device will ask the user to connect to the rogue access point

The above warning acts as a certificate trust validation prompt, where the device will expect users to view and validate the certificate details.

Upon accepting the above prompt and connecting to the rogue access point, the user's credentials are disclosed:

```
wlan0: interface state UNINITIALIZED->COUNTRY_UPDATE
Using interface wlan0 with hwaddr 34:e8:94:8b:7c:3c and ssid "EnterpriseDemo"
wlan0: interface state COUNTRY_UPDATE->ENABLED
wlan0: AP-ENABLED
wlan0: STA 00:c0:ca:97:b3:c2 IEEE 802.11: authenticated
wlan0: STA 00:c0:ca:97:b3:c2 IEEE 802.11: associated (aid 1)
wlan0: CTRL-EVENT-EAP-STARTED 00:c0:ca:97:b3:c2
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=25

mschapv2: Wed Aug 12 03:43:56 2020
         username:      daniel
         challenge:     09:44:c0:05:fd:be:56:62
         response:      95:65:ca:5e:17:a1:5f:3c:01:0b:6e:04:42:08:9a:f7:40:df:aa:9e:31:54:dc:cd
         jtr NETNTLM:   daniel:$NETNTLM$0944c005fdbe5662$9565ca5e17a15f3c010b6e0442089af740dfaa9e3154dccd
wlan0: STA 00:c0:ca:97:b3:c2 IEEE 802.11: disassociated
```

Figure 49 - Once authenticated, the users hashed credentials are disclosed to the rogue access point

MacOS Laptop (MacOS Mojave version 10.14.6)
An MacOS system is authenticated to a WPA Enterprise network 'EnterpriseDemo', with the user trusting the legitimate authentication servers certificate (as shown in the 'MacOS systems' section of the Appendix, p.54).

E info@contextis.com
T +44 (0)207 537 7515

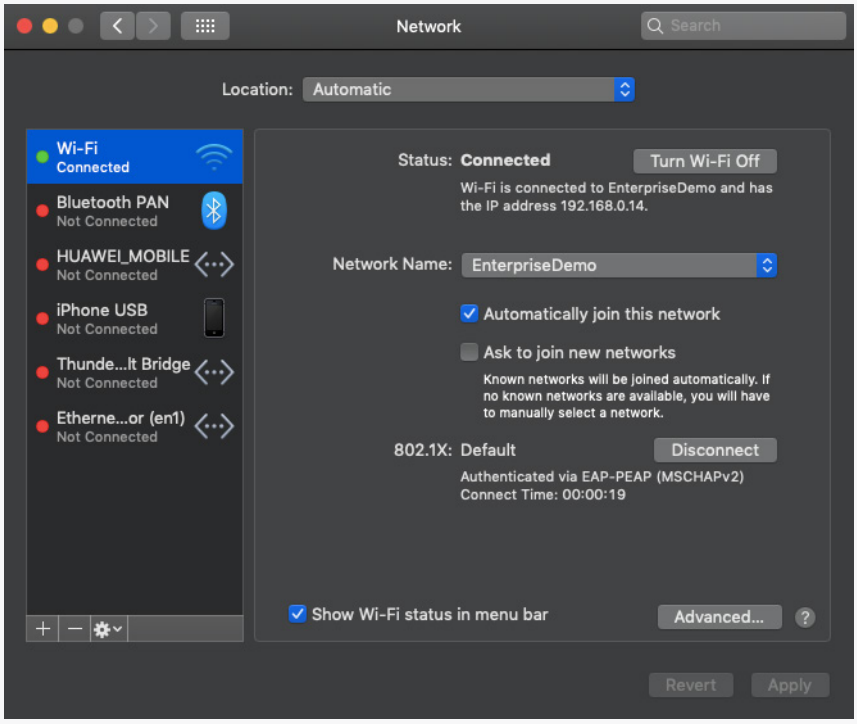

Figure 50 - Initially the device is connected to the legitimate access point

A rogue access point was once again created, similarly to the Android rogue access point scenario above.

Once the MacOS system has disconnected from the original access point, the system will attempt to automatically connect to the rogue access point. This connection will fail though, as the host has not trusted the SSID & server certificate combination. However, the system will then automatically prompt the user to 'trust' the certificate of the rogue authentication server, at which point the rogue access points certificate will be shown.

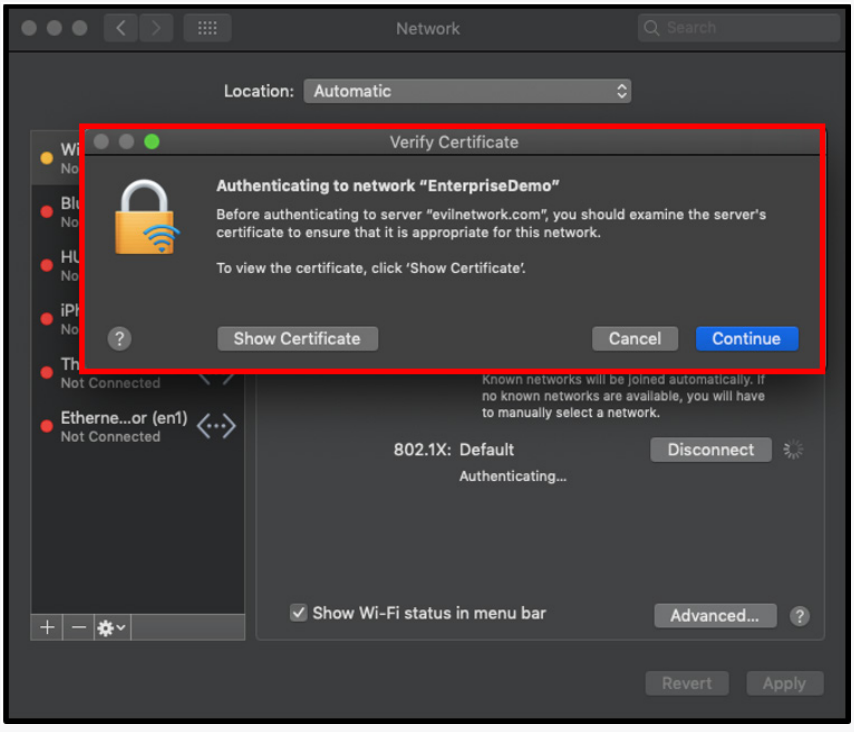

Figure 51 - The system will ask the user to trust the certificate of the rogue access point

contextis.com
E info@contextis.com
T +44 (0)207 537 7515

If the above prompt is accepted, the device will connect to the rogue access point and the user's credentials are disclosed:

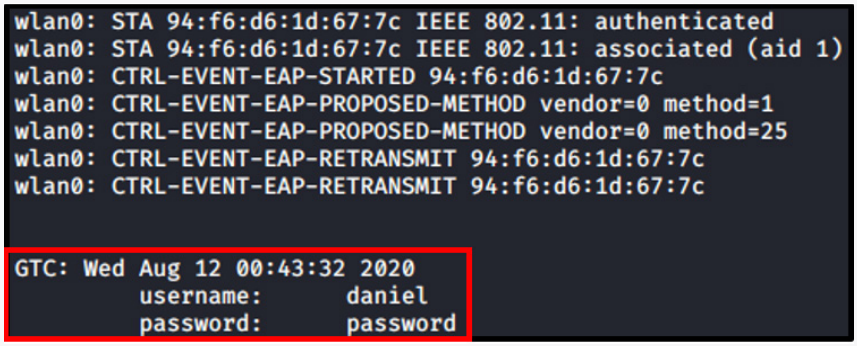

Figure 52 - The users credentials are disclosed to the rogue access point

Mitigation

Ensure that server certificate validation is enabled on all connecting clients when using a tunnelled EAP method.

Many supplicants will provide support for validating if a server certificate is trusted, and for validating the server name within the certificate against a whitelist of allowed Common Names (CN). Microsoft Windows is unique and provides a number of additional validation options, such as root CA validation and restricting authorisation of new servers. More information about these options can be found in the 'Client Security Considerations' section within Context's blog post on WPA Enterprise Security and Deployment[13].

In general, manually allowing users to configure network configurations without clear instructions will result in unintended human errors. It is recommended where possible, to utilise managed configuration deployment software, e.g. Windows systems can use Active Directory, MacOS systems can use Apple Configurator, iOS devices can use Apples' Mobile Device Management, and third-party solutions can be utilised for Linux and Android devices.

3.3.2 Supplicant trusts external root certificate authority (rogue access point)

This configuration weakness affects supplicants which trust external root Certificate Authorities (CA) when performing validation of an authentication servers certificate. As such, this issue is targeted at Windows environments, as Windows systems can be configured with additional certificate trust settings, including the option to trust root CAs (as shown in the 'Windows systems' section of the Appendix, p.52).

When a supplicant is configured to trust an external root CA, malicious actors have the opportunity to purchase a legitimate certificate from the same CA, and utilise that certificate in rogue access point attacks against clients. In this scenario, users will not be prompted to validate any new (potentially malicious) authentication servers which present certificates from the same trusted root CA. This bypasses certificate warnings when performing a rogue access point attack allow malicious actors to perform rogue access point attacks with a higher chance of successful credential capture.

Furthermore, the certificates utilised by an organisations wireless network can be obtained by an attacker (more details about network fingerprinting can be found in the section titled 'Fingerprinting Wireless Enterprise Networks', p.15). If the certificate presented by the network is from an external CA, then attackers can take a guess that the corresponding root CA for that certificate is also trusted by supplicants in the network.

13 https://www.contextis.com/us/blog/a-crash-course-into-wpa-enterprise-security-and-deployment

Demo

To setup this attack, certificates were obtained from the open Certificate Authority 'LetsEncrypt'. Additionally, for simplicities sake during testing, our test network 'EnterpriseDemo' was setup to trust all external root Certificate Authorities:

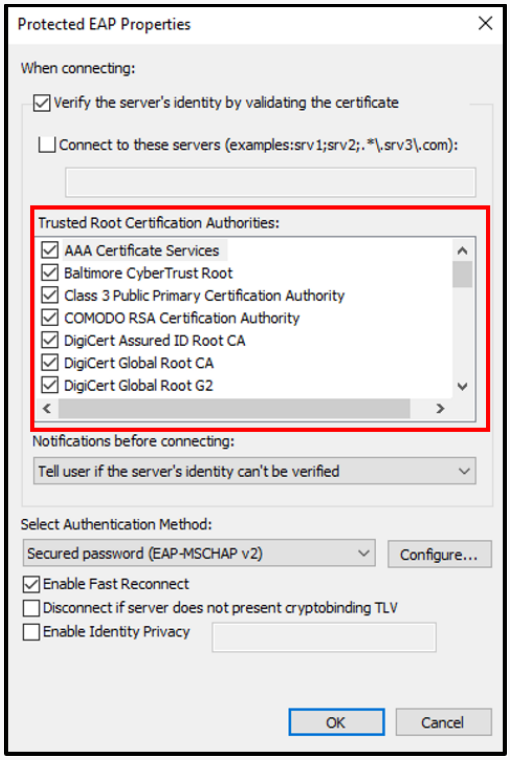

Figure 53 - Weak network configuration with trust configured for all external Root Certificate Authorities

Before launching a rogue access point attack, we will have to import our LetsEncrypt certificate into a format that EAPHammer can understand. The following command can be run to import a LetsEncrypt certificate directly into EAPHammer:

```
./eaphammer --cert-wizard import --server-cert /path/to/fullchain.pem
--private-key /path/to/private_key.pem
```


contextis.com
E info@contextis.com
T +44 (0)207 537 7515

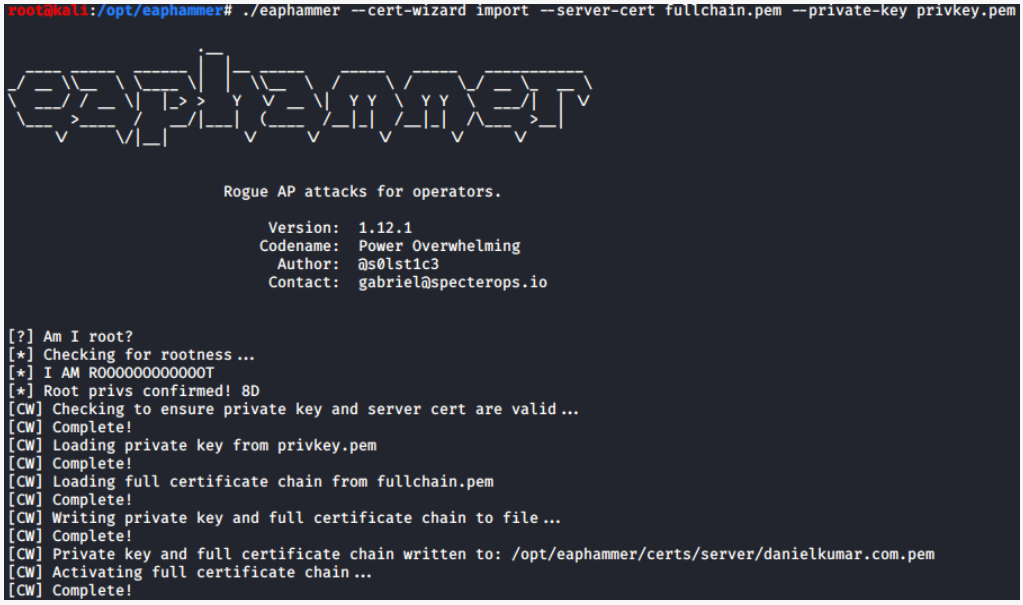

Figure 54 - Utilising EAPHammer to convert a LetsEncrypt certificate into a usable format

The EAPHammer tool will always use the last imported/generated certificate by default. This means we can instantly start a rogue access point mimicking our test network 'EnterpriseDemo':

Figure 55 - Rogue Access Point is created using EAPHammer

A deauthentication attack was simulated by restarting the system's network adapter. Upon a restart, our test Windows client attempted to automatically reconnect and authenticate to the rogue access point.

Unlike the 'Windows Desktop' scenario demonstrated in the section titled 'Supplicant performs no server validation (rogue access point)' (p.32), the client did not prompt the user with a warning message (as previously demonstrated in figure 48).

E info@contextis.com
T +44 (0)207 537 7515

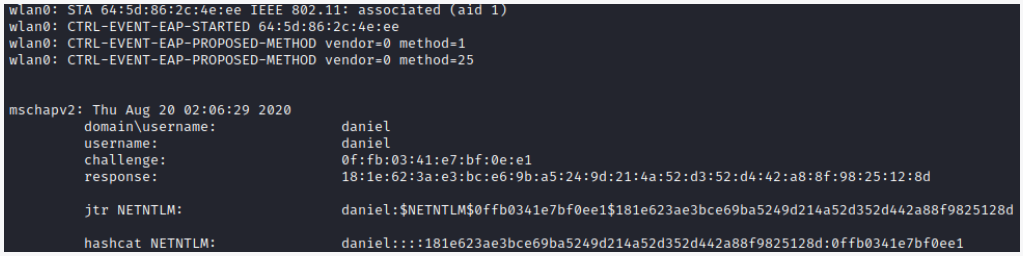

Figure 56 - Rogue access point has captured a EAP-MSCHAPv2 authentication attempt

Mitigation

In a secure Enterprise wireless environment, all authentication servers should present certificates which have been distributed by an internal Certificate Authority (CA). This internal CA should then be trusted by all clients which connect to the network, and applicable supplicants can be configured to automatically trust that internal root CA.

Enforcing this configuration will heavily increase the security of the network, as clients will only connect to authentication servers with certificates issued by the organisations internal CA, thus heavily limiting the potential attack vectors against the network

Furthermore, enforcing this configuration will ensure that certificate warning messages will not be shown to users during typical usage, e.g. when a new authentication server is encountered, such as at a secondary building within an organisation. This will prevent users from becoming accustomed to these warnings, and in turn makes users more alert if they actually are under threat of a legitimate rogue access point.

3.3.3 EAP-FAST phase 0 access point impersonation attack (rogue access point)

This EAP method was created by Cisco to improve upon weaknesses within the LEAP authentication method. EAP-FAST specifically aims to mitigate passive sniffing attacks by creating an encrypted TLS tunnel between clients and authentication servers.

Unlike other tunnelled EAP methods such as PEAP and EAP-TTLS, EAP-FAST does not require the use of certificates on either the authentication server or clients. Instead, EAP-FAST creates an encrypted TLS tunnel through the use of a client side Protected Access Credential (PAC) file (not to be confused with Proxy auto-config (PAC) files).

EAP-FAST consist of three phases, known as phase 0, 1, and 3. Phase 2 and 3 align with phase 1 and 2 of typical tunnelled EAP methods (an overview of the tunnelled EAP authentication process can be seen in Figure 12, in the section titled 'Overview of EAP Messages' (p.11).

Phase 0 however, is a new phase unique to EAP-FAST known as 'Automatic PAC provisioning'. The sole purpose of phase 0 is to automatically deploy a PAC file to a connecting client on the fly, thus not requiring PAC files to be generated and distributed to users via an out-of-band method.

If phase 0 is utilised however, connecting clients are not able to validate that the authentication server providing the PAC file is indeed the legitimate authentication server. This is due to the fact that the authentication server is allowed to authenticate any client on the fly, i.e. impersonate any authentication server.

E info@contextis.com
T +44 (0)207 537 7515

If an attacker were to perform a rogue access point attack against a client with phase 0 support, the attacker could simply supply a PAC file which would authenticate the server to the client and initiate a Man-in-the-Middle (MitM) attack.

Demo

Although the rogue access point attacks performed in this whitepaper have been used utilising the 'EAPHammer' tool, this tool will not work for this attack. EAPHammer leverages a modified version of the 'hostapd-wpe' tool in order to perform rogue access point attacks. The version of the tool included does not contain support for EAP-FAST Phase 0 rogue access point attacks. The original 'hostapd-wpe' tool which was developed by Brad Antoniewichz does contain this functionality however, and can be compiled from source on GitHub[14].

For this demo, our test network was impersonated using 'hostapd-wpe'. An Android device was used to connect to the network, which was caught by our rogue access point. Below is an image showing MSCHAPv2 hashes disclosed from the Android device (EAP-MSCHAPv2 is the only supported inner method for the 'hostapd-wpe' tool when using EAP-FAST as the outer method):

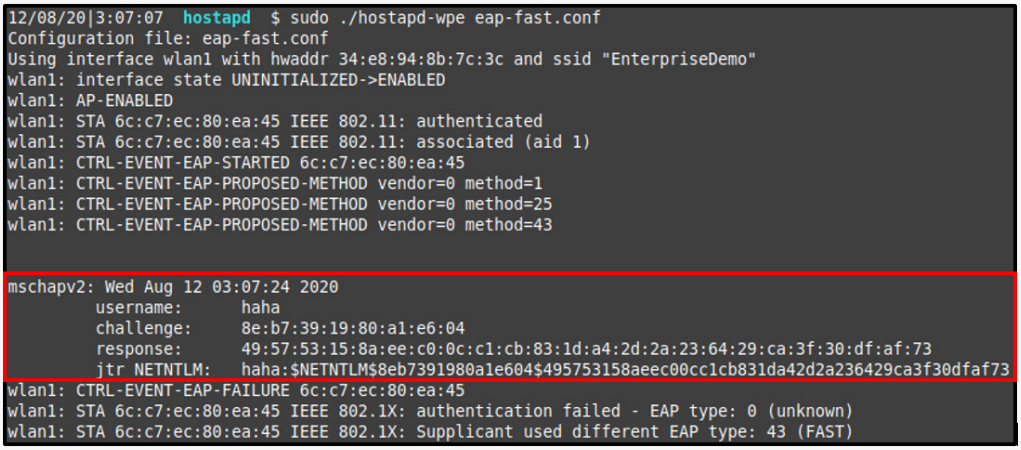

Figure 57 - Rogue access point has captured EAP-MSCHAPv2 credential hashes

Mitigation

If EAP-FAST is utilised, then Phase 0 (Automatic PAC provisioning) should never be used. The use of this phase is not able to provide a guarantee that clients are interacting with intended authentication servers.

Instead, PAC files should be provisioned to clients out-of-band. Doing this will ensure that if a client interacts with a malicious authentication server, the pre-provisioned PAC file will not allow the client to authenticate malicious authentication server.

Furthermore, devices should be configured to only support authenticated PAC provisioning. Any devices which are configured to support automatic PAC provisioning could be targeted by malicious rogue access points.

3.3.4 Supplicant allows permissive EAP negotiation (EAP downgrade attack)

Many supplicants will support multiple EAP methods, and will attempt to negotiate with authentication servers in order to determine the most secure EAP method which is supported by both parties.

14 https://github.com/OpenSecurityResearch/hostapd-wpe

E info@contextis.com
T +44 (0)207 537 7515

Attacking and Defending Enterprise Networks

When connecting to a network, in cases where a specific network configuration has not been manually configured, supplicants will generally opt for the most user-friendly experience possible. This means that when the supplicant attempts to connect to an authentication server and is denied, the supplicant will rotate through all its supported EAP methods until the authentication server accepts one, or all are rejected and authentication is not possible.

In practice, if the authentication server suggests its supported EAP methods from weakest first to strongest, then there is a chance that user friendly clients will connect using their weakest supported EAP method. The most useful EAP method to downgrade a negotiation to is EAP-GTC, which can result in plain-text credentials being automatically disclosed.

The process for EAP negotiation is demonstrated extremely well through this GIF created by Gabriel Ryan (@s0lst1c3)[15].

Additionally, there are two blog posts by Gabriel Ryan that do an excellent job on explaining how EAP downgrade attacks work[16] [17].

Demo

iOS devices are a good example of devices which are generally vulnerable to this attack. Users who connect to a network manually are connected in a very user-friendly manner, with minimal configuration options.

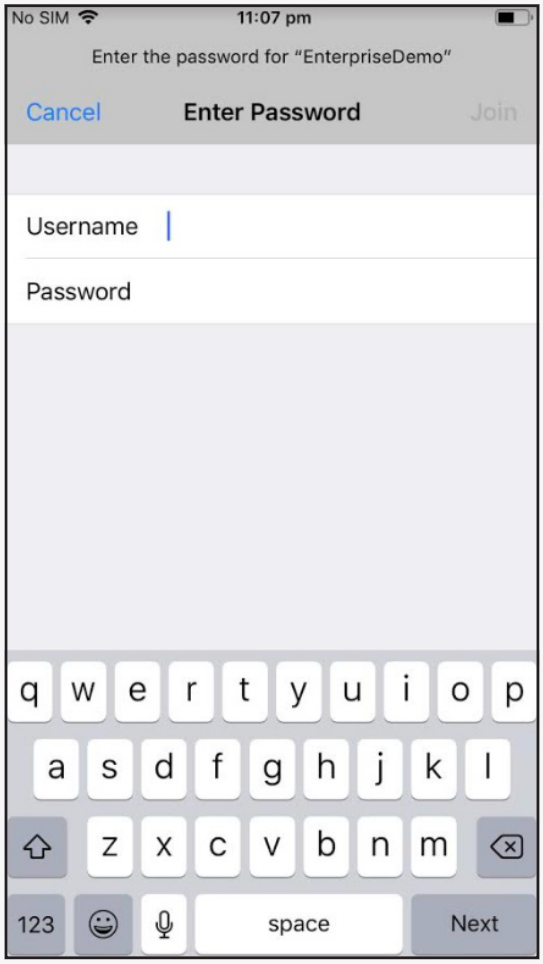

Figure 58 - User Interface shown when connecting to a WPA Enterprise network via an iOS device

15 https://raw.githubusercontent.com/s0lst1c3/s0lst1c3.github.io/master/images/eap-negotiation/eap-user-file-diagram.gif

16 https://solstice.sh/2019/09/10/eap-downgrade-attacks/

17 https://solstice.sh/iii-eap-downgrade-attacks/

The demonstration provided within the 'EAP-GTC Clear-Text Credential Disclosure' attack actually performs a GTC downgrade attack against an iOS device through the EAPHammer tool. When supplying the '--negotiate gtc-downgrade' parameter to the tool, the hostapd-wpe tool will be configured to return EAP-GTC as its first authentication method, instead of a stronger method such as PEAP/EAP-MSCHAPv2.

By reconfiguring the original 'EAP-GTC Clear-Text Credential Disclosure' demo network to utilise PEAP/MS-CHAPv2 and connecting an iOS device. Running the same rogue access point attack (using the –negotiate gtc-downgrade parameter) and reconnecting the client will result in the exact same output, i.e. clear-text credentials being disclosed through the use of EAP-GTC.

This is possible on iOS devices as users are not able to pick a desired EAP method when manually authenticating. This is done in order to display a user friendly interface when connecting. iOS devices are not the only vulnerable devices, however this vulnerability is usually exposed due to configuration weaknesses, e.g. an Android device with an outer authentication method selected but the inner authentication method set to 'auto'.

Mitigation

Do not allow users to manually connect to a corporate network unless specifically allowed. When manually connecting to a network, especially through iOS devices and older Windows systems, user friendly User Interfaces will prevent users from specifying a suitable EAP method and will attempt all supported methods.

Instead, ensure users are given access to networks through centrally managed configuration software, where specific EAP methods can be configured. On devices where users can manually setup specific configurations such as Android and Windows devices, extensive setup documentation should be provided to users to ensure 'automatic' authentication method selections are not configured.

contextis.com
E info@contextis.com
T +44 (0)207 537 7515

4 Recommendations for a Secure Enterprise Network

This section at a glance

A final conclusion and advise for implementing a secure enterprise network configuration, drawn from the pros and cons of the EAP methods discussed.

Overall, there is a clear hierarchical ranking regarding the strength of an enterprise wireless network, based on the EAP methods supported by the authentication server, and enforcement of secure client configurations.

In the case where a weak authentication server configuration is supported, clients could be configured to utilise weak authentication methods which are susceptible to attacks, e.g. passive sniffing. Whereas in the case that a weak client configuration is enforced, connecting clients could be vulnerable to rogue access point attacks. As such, both the authentication server and client configurations are as equally important in protecting the network.

The following recommendations should be kept in mind when configuring an enterprise network on the authentication servers side:

1. Always use tunnelled authentication methods, never native authentication methods. This will ensure that the inner authentication process cannot be captured by an attacker.
2. Only configure support for stronger authentication methods such as EAP-TLS. In cases where EAP-TLS cannot be utilises, then EAP-MSCHAPv2 could be utilised, e.g. in BYOD environments. Given the option between EAP-TLS and EAP-MSCHAPv2 however, EAP-TLS should be configured where possible. EAP-MSCHAPv2 is vulnerable to serious known credential recovery attacks, and is also affected by replay attacks. This authentication method should only be utilised where EAP-TLS is an absolute non-option

The EAP methods available for us in an organisation will differ, however by today's wireless standards, native EAP methods will never be the only available option and thus point 1 above should always be enforced. Point 2 in the above will be dependent on if an organisation is able to support a robust Public Key Infrastructure (PKI). Note that implementing a PKI is very achievable for organisations which already run managed configuration deployment software such as Active Directory, and thus striving to implement EAP-TLS over EAP-MSCHAPv2 will greatly enhance an organisations security posture.

On the client side, the following recommendations should be kept in mind when configuring access for connecting clients:

1. Ensure clients are configured to always validate the authentication servers' certificate before authentication occurs. This will ensure that clients will only connect to authorised authentication servers.
2. If possible, ensure that clients have configurations pushed to their devices via centrally managed configuration software. These could be software such as the below, and also third party vendor solutions:

E info@contextis.com
T +44 (0)207 537 7515

Recommendations for a Secure Enterprise Network

- Windows systems can use Active Directory
- MacOS systems can use Apple Configurator
- iOS devices can use Apples' Mobile Device Management
- Third-party solutions can be utilised for Linux and Android devices

3. In cases where users are allowed to configure their own network settings, ensure that clear steps are provided to the user which outline the organisations supported devices and configuration.

When organisations take this holistic approach to their wireless networks, and take into account the above points to secure both the wireless infrastructure and client devices, a high level security baseline can be achieved for an organisations wireless environment.

For organisations who have successfully achieved a sufficient security level within their wireless environments and are looking to take further steps, ensuring the physical security of wireless network devices and signal leakage are under control, and looking into performing active rogue access point suppression are perfect next steps.

contextis.com
E info@contextis.com
T +44 (0)207 537 7515

Appendix

Appendix - Certificate Validation Configuration Options

The following images demonstrate where certificate validation settings are specified within different systems during a typical authentication attempt to an Enterprise WPA network. The below only shows where these settings are accessible via the systems User Interface (UI) when manually configuring a wireless network, i.e. automated configuration deployment software such as Active Directory and Mobile Device Management (MDM) software is not shown.

Android devices

When attempting to connect to an enterprise wireless network, a detailed configuration form will be displayed and the user is asked to specify the trusted certificate which the authentication server certificate should be validated against:

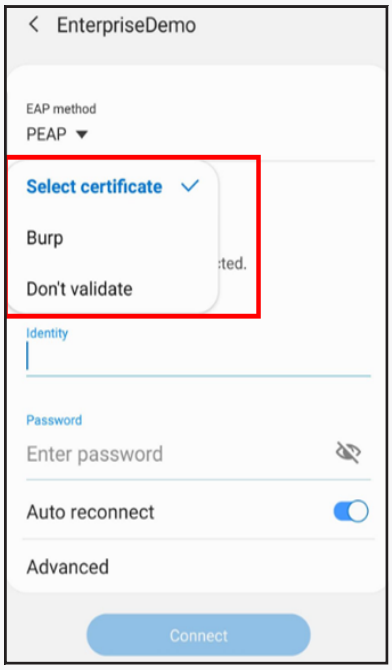

Figure 59 - Certificate validation option for PEAP authentication on a Samsung Galaxy Note9 (Android 9)

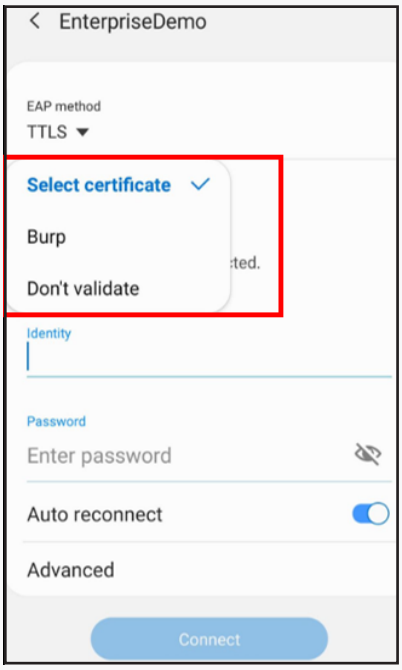

Figure 60 - Certificate validation option for EAP-TTLS authentication on a Samsung Galaxy Note9 (Android 9)

Appendix

iOS devices

When connecting to an enterprise network on an iOS device, the user is not given the option to disable certificate validation, or to choose a certificate to validate the authentication server against. When connecting to the network a simple credential form is displayed:

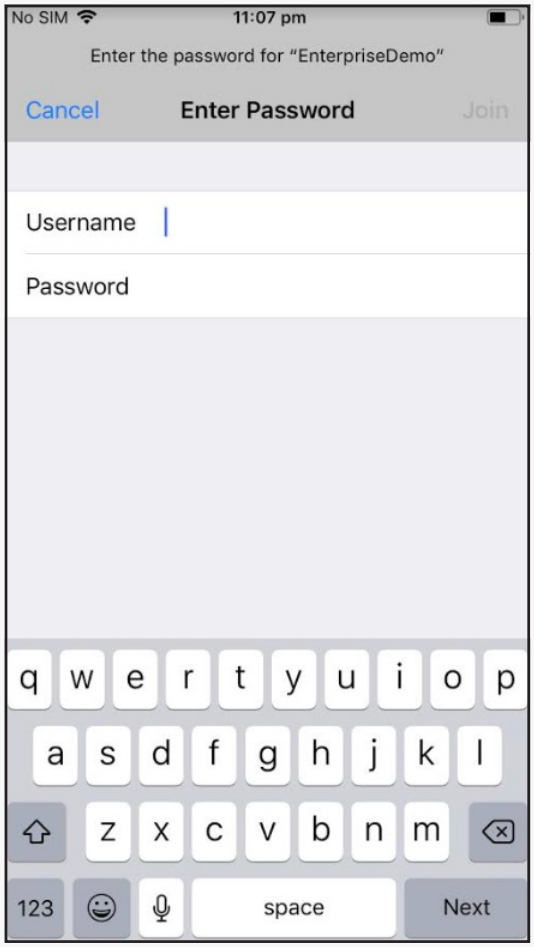

Figure 61 - Initial connection to the network will simply ask for credentials (iOS 13.1 iOS device)

Once credentials are entered, if the iOS supplicant has not trusted the current network combination (SSID, server certificate), the user will be asked to trust the certificate:

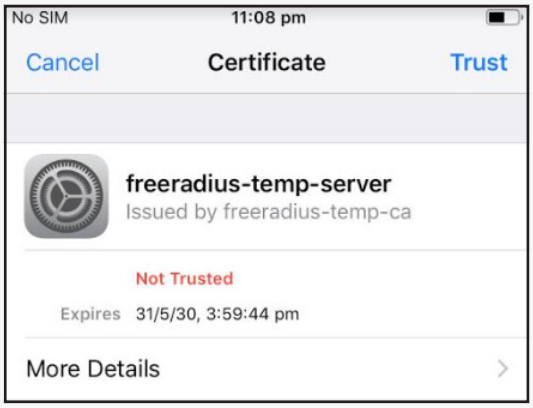

Figure 62 - Before authentication is performed, the user is asked to manually validate and trust the authentication servers certificate (iOS 13.1 iOS device)

Appendix

Linux systems

When attempting to connect to an Enterprise network via the NetworkManager widget, if the server certificate is not already trusted by the systems trusted root store, then the connection will not commence.

Attempting to connect to the network manually will show a dropdown where a specific server certificate can be selected for validation:

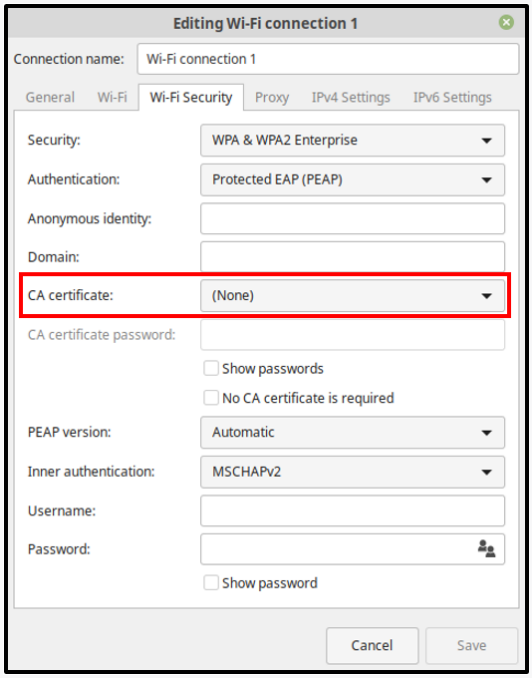

Figure 63 - Certificate validation can be disabled (set to None in Linux systems, specifically wpasupplicant version 2:2.6-15ubuntu2.4)

Appendix

Windows systems

When attempting to initially connect to an Enterprise network from a Windows system, if the server certificate is not trusted by the system, then the user will typically be asked to confirm the identity of the network (by viewing the server certificate).

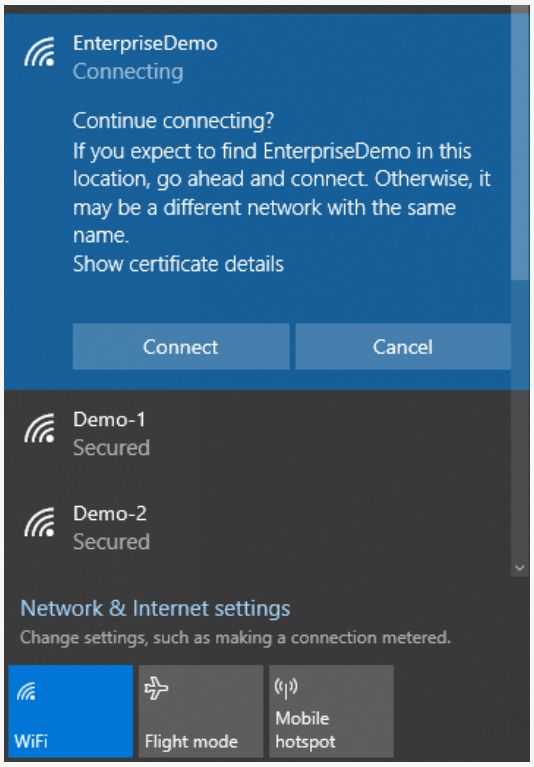

Figure 64 - Before authentication is performed, the user is asked to manuallyvalidate and trust the authentication servers certificate via a non-technical prompt (Windows 10 system)

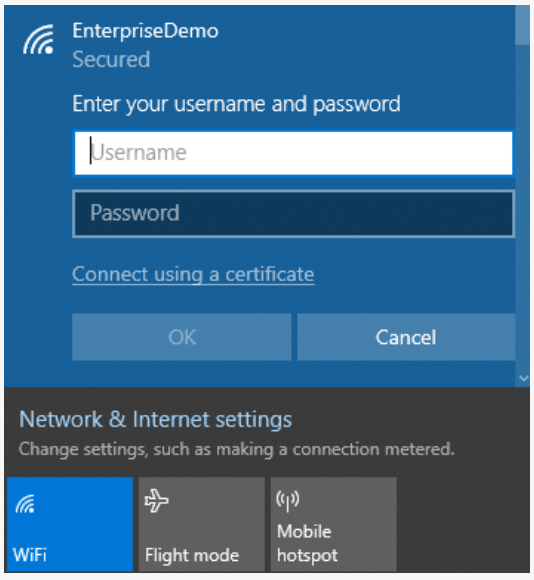

Figure 65 - Initial connection will then ask for credentials (Windows 10 system)

Appendix

Furthermore, within Windows systems, if a network profile is manually created for an enterprise wireless network, the following extensive certificate validation settings can be configured:

- Connect to these servers – Should be configured to specify only the Common Name of the authentication server. This will help prevent rogue servers which are incorrectly configured.
- Trusted Root Certificate Authorities – Should be configured to only trust the single root certificate authority for the authentication server. This should be an internal Certificate Authority (CA), as if an external CA is trusted, a valid rogue access point could be setup utilising a publicly available certificate from the trusted external CA.
- Notifications before connecting – Should be set to 'Don't ask user to authorise new servers or trusted CAs', preventing users from accidently authorising new servers which potentially belong to a rogue access point.

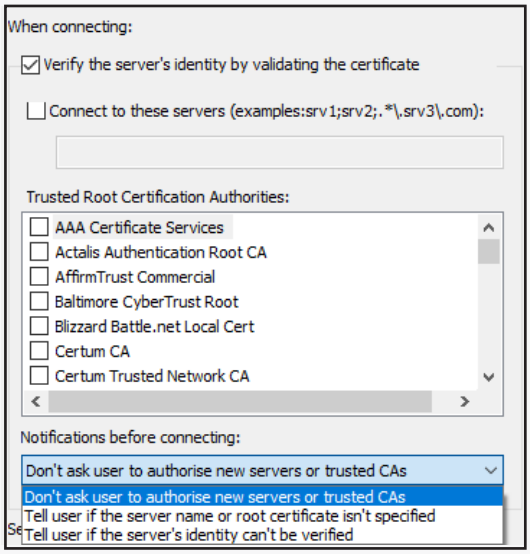

Figure 66 – Windows systems contain additional certificate validation settings available when manually configuring a wireless network

Appendix

MacOS systems

When connecting to an enterprise network on an MacOS system, similar to user friendly iOS devices, users are not given the ability to select a certificate to validate against an authentication server, or to disable certificate validation. Connecting to an enterprise network returns a simple credential form:

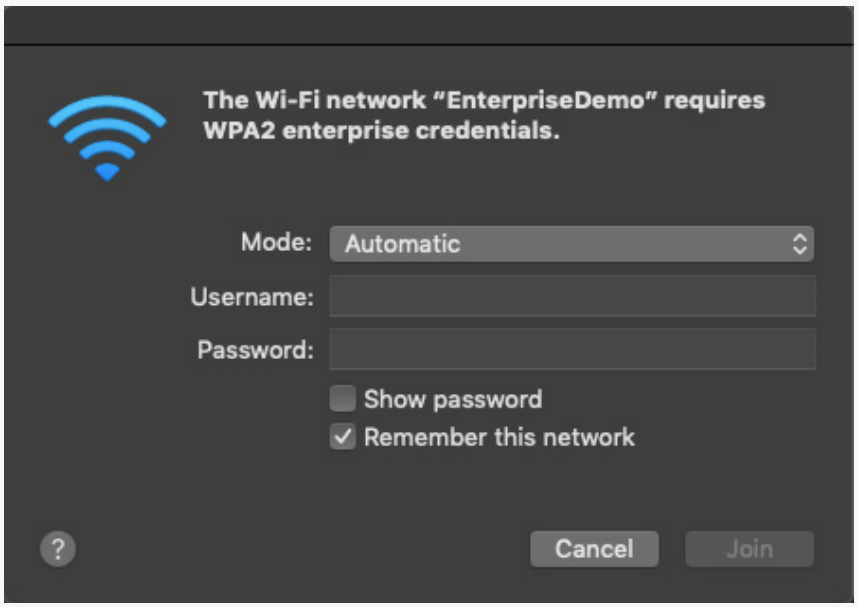

Figure 67 - Initial connection to the network will simply ask for credentials (MacOS Mojave version 10.14.6 system)

Once credentials are entered, if the MacOS system does not contain the network combination (SSID, server certificate), the user will be asked to add the certificate to the systems trust store (keychain):

Figure 68 - Before authentication is performed, the user is asked to add, i.e. 'trust', the authentication servers certificate (MacOS Mojave version 10.14.6 system)

About Context

About Context

Context Information Security is a global cyber security consultancy with a comprehensive portfolio of advisory and advanced technical services, ranging from penetration testing, red teaming and cyber incident response to product and application security testing.

We focus on helping clients avoid potential breaches and to deter, detect and respond to the most sophisticated cyber-attacks.

With over 20 years experience, Context's client base includes some of the world's leading blue chip companies, alongside public sector and government organisations. With Context teams in the UK, USA, Germany and Australia, and now being a part of Accenture Security, Context teams can serve clients worldwide.

About the author

Daniel Kumar is part of Context's Assurance team and is based in our office in Sydney, Australia. He specialises in Infrastructure, Web and Mobile Applications and Wireless assessments.

contextis.com

E info@contextis.com
T +44 (0)207 537 7515

contextis.com
E info@contextis.com
T +44 (0)207 537 7515