Configure Wireshark and FreeRADIUS in order to decrypt 802.11 WPA2-Enterprise/EAP/dot1x over-the-air Wireless Sniffer

Contents

Introduction
Prerequisites
Requirements
Components Used
Background Information
Procedure
Step 1. Decrypt PMK(s) from Access-accept Packet.
Step 2. Extract PMK(s).
Step 3. Decrypt the OTA Sniffer.
Example of a Decrypted 802.11 Packet
Example of an Encrypted 802.11 Packet
Related Information

Introduction

This document describes a how-to of decrypting Wi-Fi Protected Access 2 - Enterprise (WPA2-Enterprise) or 802.1x (dot1x) encrypted wireless over-the-air (OTA) sniffer, with any Extensible Authentication Protocol (EAP) methods.

It is relatively easy to decrypt PSK based/WPA2-personal 802.11 OTA capture as long as the full four-way EAP over LAN (EAPoL) handshakes are captured. However, Pre-shared Key (PSK) is not always recommended from a security perspective. Cracking a hard-coded password is just a matter of time.

Hence, many enterprises choose dot1x with Remote Authentication Dial-In User Service (RADIUS) as a better security solution for their wireless network.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- FreeRADIUS with **radsniff** installed
- Wireshark/Omnipeek or any software that is capable of decrypting 802.11 wireless traffic
- Privilege to obtain the shared secret between network access server (NAS) and

Authenticator
- Ability to capture radius packet capture between NAS and authenticator from the first access-request (from NAS to Authenticator) to the last access-accept (from Authenticator to NAS) throughout the EAP session
- Ability to perform Over-the-Air (OTA) capture containing four-way EAPoL handshakes

Components Used

The information in this document is based on these software and hardware versions:

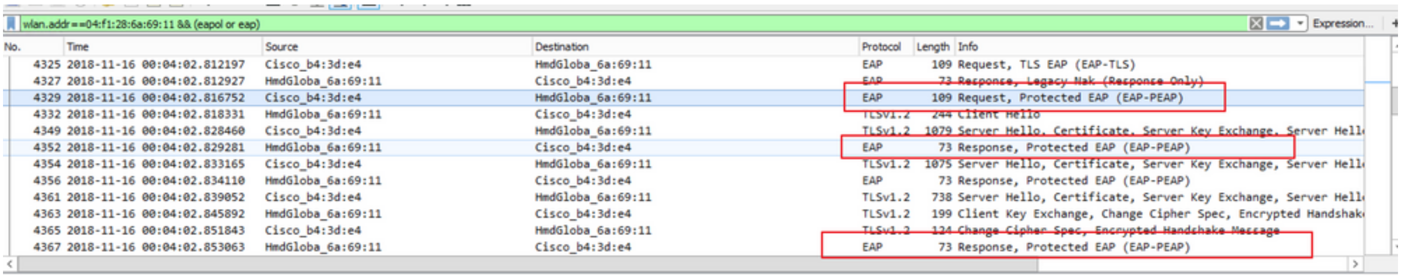
- Radius server (FreeRADIUS or ISE)
- Over-the-Air capture device
- Apple macOS/OS X or Linux device

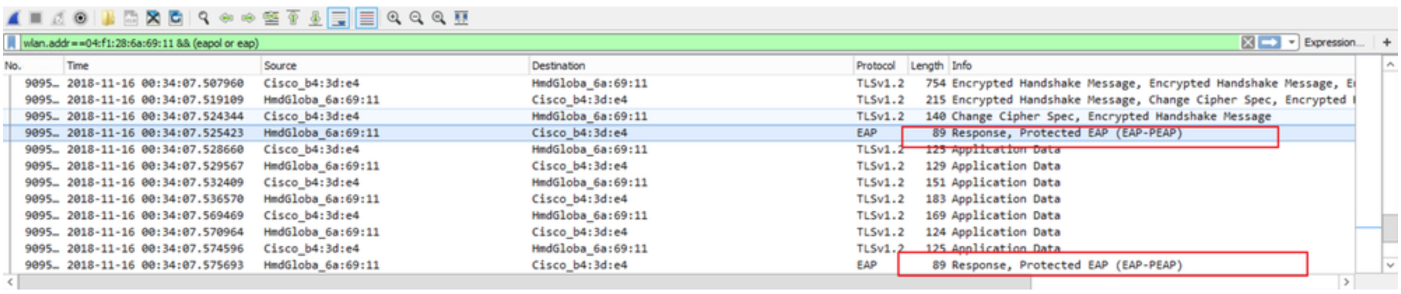
The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

In this example, two Pairwise Master Keys (PMKs) are derived from Radius packets captured from ISE 2.3, as the session timeout on this SSID is 1800 secs, and the capture given here is 34 mins (2040 secs) long.

As shown in the image, EAP-PEAP is used as an example, but this can be applied to any dot1x based wireless authentication.





Procedure

Step 1. Decrypt PMK(s) from Access-accept Packet.

Run the **radsniff** against radius capture between NAS and Authenticator in order to extract PMK. The reason why two access-accept packets are extracted during the capture is that the session timeout timer is set to 30 mins on this particular SSID and the capture is 34 mins long.

Authentication is performed twice.

```
FRLU-M-51X5:pcaps frlu$ radsniff -I /Users/frlu/Downloads/radius_novlan_merged.pcapng -
s <shared-secret between NAS and Authenticator> -x

<snip>

2018-11-16 11:39:01.230000 (24) Access-Accept Id 172
/Users/frlu/Downloads/radius_novlan_merged.pcapng:10.66.79.42:32771 <- 10.66.79.36:1812 +0.000
+0.000

User-Name = "frlu_2"

State = 0x52656175746853657373696f6e3a30613432346632613030303030356537356265653039373 2

Class =
0x434143533a30613432346632613030303030356537356265653039373 23a4953452d322d332f333238323731323338
2f33303432

EAP-Message = 0x03c50004

Message-Authenticator = 0x38c67b9ba349842c9624889a45cabdfb

MS-MPPE-Send-Key = 0xa464cc15c0df8f09edc249c28711eb13a6db2d1a176f1196edcc707579fd6793

MS-MPPE-Recv-Key =
0xddb0b09a7d6980515825950b5929d02f236799f3e8a87f163c8ca41a066d8b3b<<<<<<<<<<<<<<<<<<PMK

Authenticator-Field = 0x6cd33b4d4dde05c07d9923e17ad6c218

<snip>

2018-11-16 11:39:01.470000 (48) Access-Accept Id 183
/Users/frlu/Downloads/radius_novlan_merged.pcapng:10.66.79.42:32771 <- 10.66.79.36:1812 +0.000
+0.000

User-Name = "frlu_2"

State = 0x52656175746853657373696f6e3a30613432346632613030303030356537356265653039373 2

Class =
0x434143533a30613432346632613030303030356537356265653039373 23a4953452d322d332f333238323731323338
2f33303434

EAP-Message = 0x03910004

Message-Authenticator = 0x81c572651679e15e54a900f3360c0aa9

MS-MPPE-Send-Key = 0xeae42cf7c6cd26371eee29856c51824fbb5bbb29887412592847011 4d009b5fb

MS-MPPE-Recv-Key =
0x7cce47eb82f48d8c0a91089ef7168a9b45f3d798448816a3793c5a4dfb1cfb0e<<<<<<<<<<<<<<<<<PMK

Authenticator-Field = 0xa523dd9ec2ce93d19fe4fc2e21537a5d
```

Note: Please remove any virtual LAN (VLAN) tag of the Radius packet capture, otherwise, **radsniff** does not recognise the input pcap file. In order to remove any VLAN tag, for example, [editcap](https://www.wireshark.org/docs/man-pages/editcap.html) can be used.

Tip: Generally, the runtime of **radsniff** command against a RADIUS pcap file can be counted as a scale of seconds. However, if the **radsniff** is stuck in this state shown in the log, please cascade this packet capture (A) with another longer packet capture (B) between the same NAS and Authenticator. Then, run the radsniff command against the cascaded packet (A+B). The only requirement of packet capture (B) is that you are able to run the radsniff command against it and see verbose result.

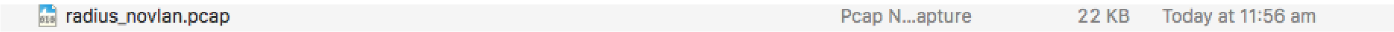
```
FRLU-M-51X5:pcaps frlu$ radsniff -I /Users/frlu/Downloads/radius_novlan.pcap -s Cisco123 -x

Logging all events

Sniffing on (/Users/frlu/Downloads/radius_novlan.pcap)
```

In this example, the Wireless Lan Controller (WLC) control plane logging (A) that is captured via [WLC packet logging] feature, is cascaded with a longer capture from ISE's TCPdump (B). WLC packet logging is used as an example because it is usually very small in size.

WLC packet logging (A)



ISE Tcpdump (B)

Merged (A+B)

Then run the **radsniff** against the merged pcap (A+B) and you will be able to see the verbose output.

```
FRLU-M-51X5:pcaps frlu$ radsniff -I /Users/frlu/Downloads/radius_novlan_merged.pcapng -s
<shared-secret between NAS and Authenticator> -x

<snip>

2018-11-16 11:39:01.230000 (24) Access-Accept Id 172
/Users/frlu/Downloads/radius_novlan_merged.pcapng:10.66.79.42:32771 <- 10.66.79.36:1812 +0.000
+0.000

<snip>
```

Step 2. Extract PMK(s).

Delete of 0x field in each **MS-MPPE-Recv-Key** from the verbose output and the PMKs that is needed for the wireless traffic decode is then presented.

MS-MPPE-Recv-Key =
0xddb0b09a7d6980515825950b5929d02f236799f3e8a87f163c8ca41a066d8b3b

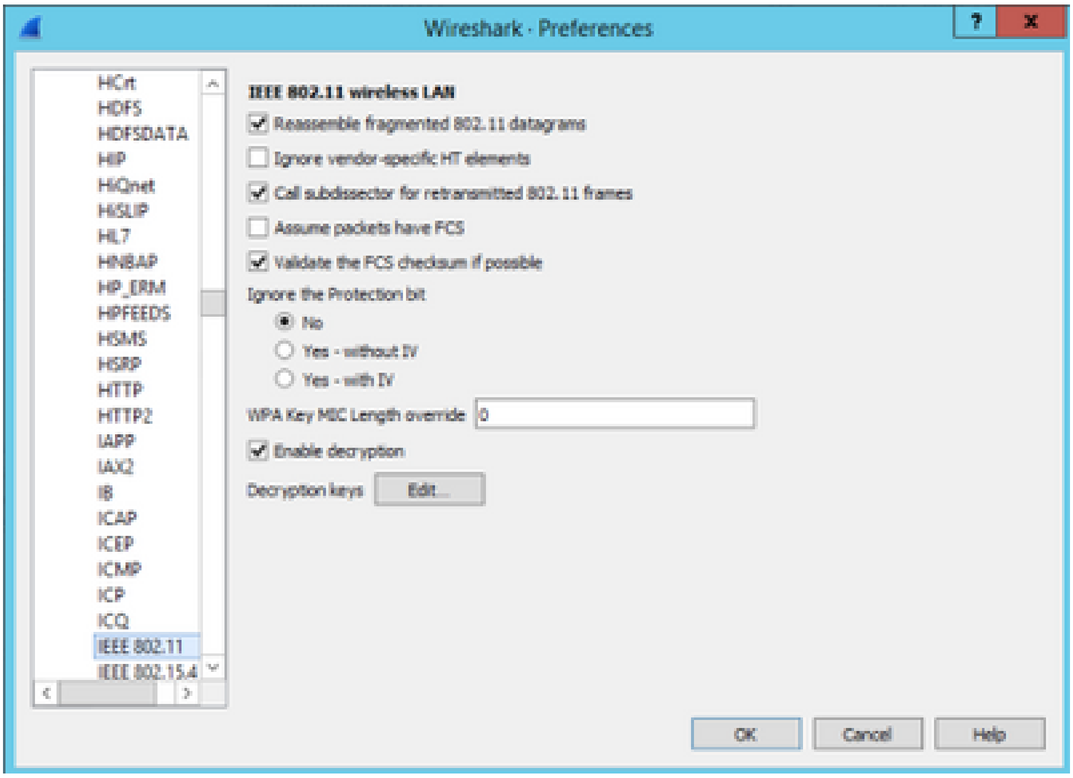
```
PMK:
ddb0b09a7d6980515825950b5929d02f236799f3e8a87f163c8ca41a066d8b3b
```

MS-MPPE-Recv-Key =
0x7cce47eb82f48d8c0a91089ef7168a9b45f3d798448816a3793c5a4dfb1cfb0e

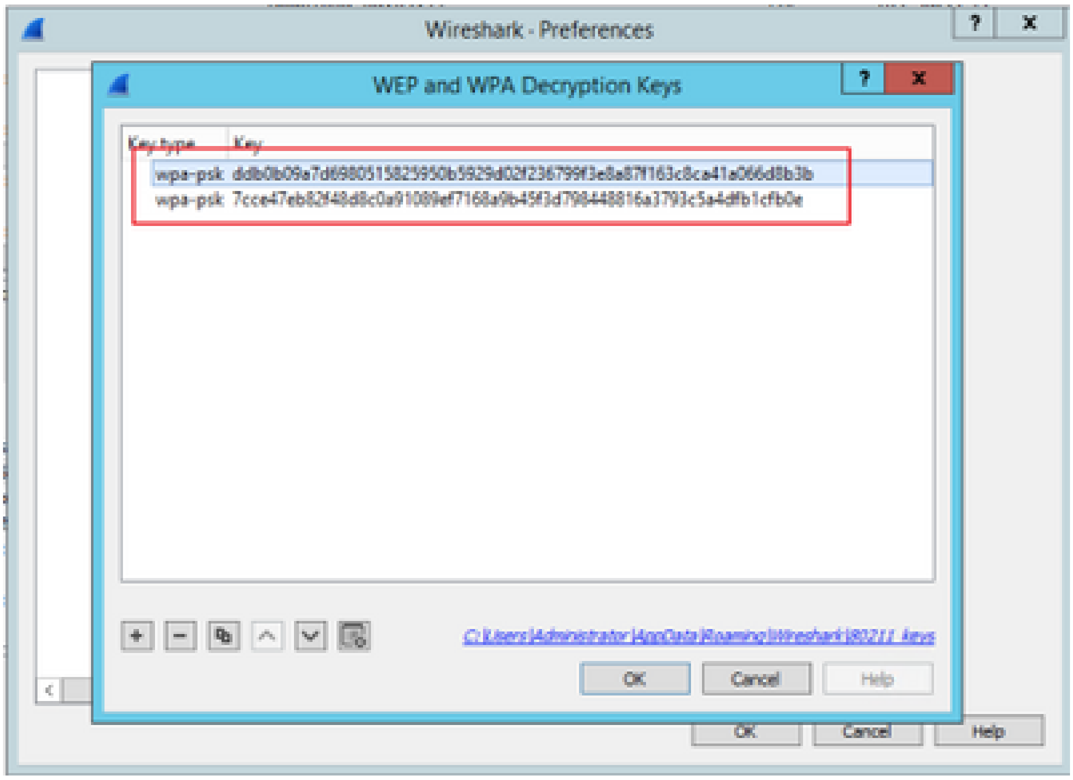
```
PMK:
7cce47eb82f48d8c0a91089ef7168a9b45f3d798448816a3793c5a4dfb1cfb0e
```

Step 3. Decrypt the OTA Sniffer.

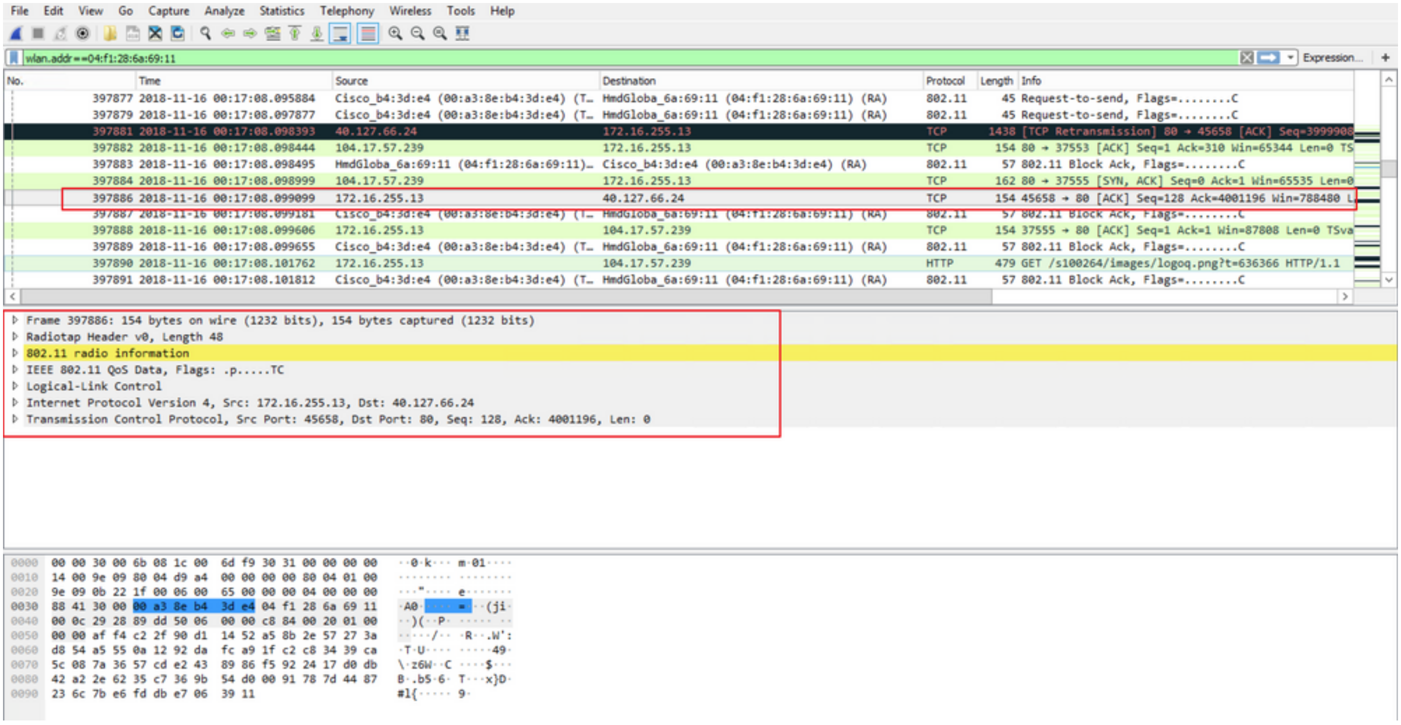
Navigate to **Wireshark > Preferences > Protocols > IEEE 802.11.** Then tick on **Enable Decryption** and click on the **Edit** button next to **Decryption Keys**, as shown in the image.



Next, please select **wpa-psk** as the Key type, and put the PMKs derived in the **Key** field, and then click on **OK**. After this is completed, the OTA capture should be decrypted and you are able to see higher layer (3+) information.

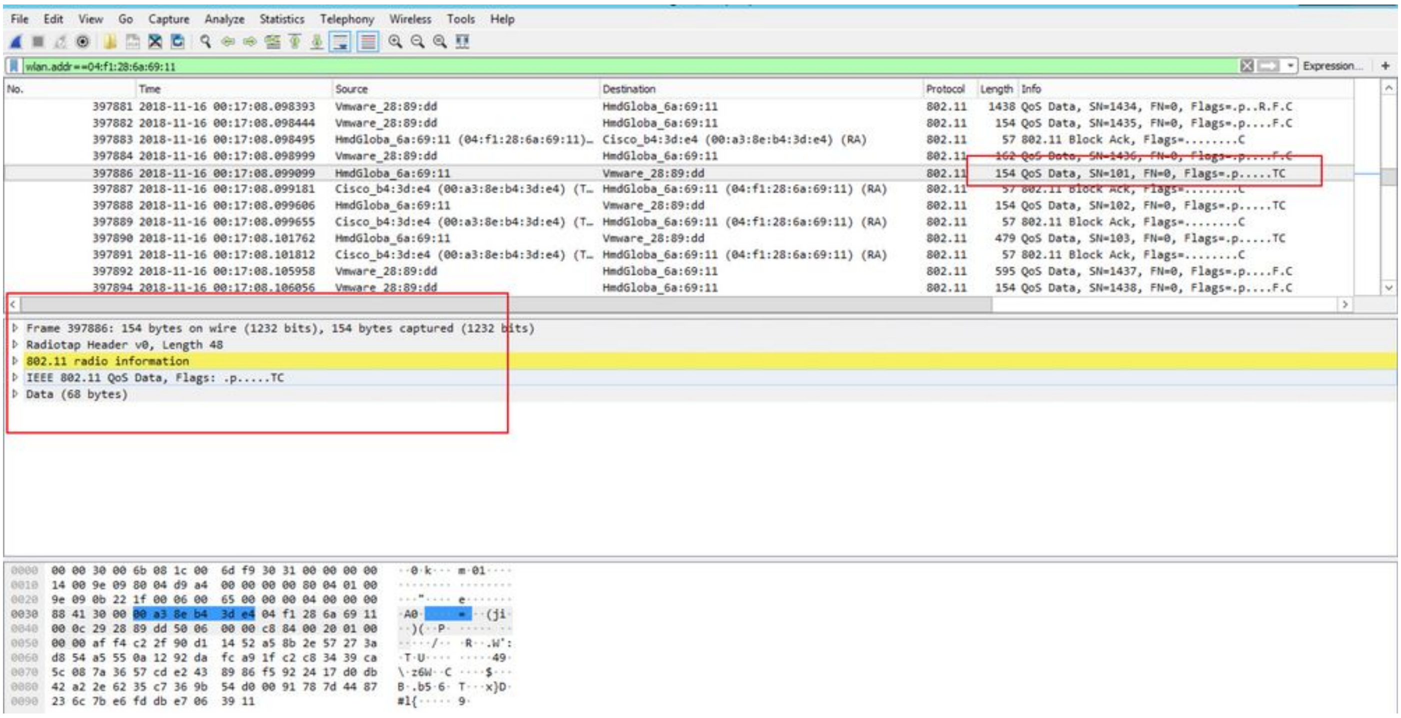


Example of a Decrypted 802.11 Packet



If you compare the second result where the PMK is not included, with the first result, where the PMK is included, packet 397886 is decrypted as 802.11 QoS data.

Example of an Encrypted 802.11 Packet



Caution: You may encounter issue with Wireshark on decryption, and in that case, even if the right PMK is provided, (or if PSK is used, both SSID and PSK are provided), Wireshark does not decrypt the OTA capture. The workaround is to turn Wireshark off and on a few times until higher layer information can be obtained and 802.11 packets are no longer shown as QoS data, or to use another PC/Mac where Wireshark is installed.

Tip: A C++ code called pmkXtract is attached in the first post in Related Information. Attempts to compiled were successfully and an executable file is obtained, but the executable program does not seem to perform the decryption properly for some uknown reasons. In addition, a Python script that attempts to extract PMK is posted in the comment area on the first post, which can be further explored if readers are interested.

Related Information

- **Tweaking EAP’s weak link – sucking WiFi PMKs out of RADIUS with pmkXtract**
- **How to Decode Radius MS-MPPE-Recv-Key**
- **Technical Support & Documentation - Cisco Systems**