Troubleshooting IPSK

Stefan Kronawithleitner
CWNE Essay

I. Introduction

As our wireless network is increasingly used by devices that could be filed under the "Internet of Things", the desire for special SSIDs to accommodate these devices grew. These devices often cannot authenticate to 802.1X SSIDs or at a captive portal on our campus networks. However, giving each device type its own SSID is difficult to manage and introduces a lot of beacon overhead.

The answer to this problem was to enable a new SSID with per-device-PSKs, and I planned and implemented our IPSK solution - storing MAC address and PSK pairs and giving devices their own PSK that way. WPA2-PSK is widely supported, and the way these solutions work is entirely transparent to the client. I created our application to manage the passphrases, and while I was still developing and polishing it, I already had many devices using the solution, as this functionality was desperately needed. This way, the solution was beta-tested by dozens of devices, and once the application was finished, I gave the persons who controlled the IOT devices access to it.

II. The problem

The rollout went well, but I did receive a few comments that there were troubles joining the SSID. However, after entering the MAC&SSID again in the application, I was told it worked. The problem description sounded like a user error, and as this was not widespread, I decided to leave it at that. However, a few weeks later, I got these comments again from people I trusted would correctly enter addresses and passphrases.

As I have never encountered these problems before, I asked to get access to the device to investigate when this happened the next time.

III. Troubleshooting and solution

In this case, it was a tablet used to control lights. The first thing I checked was the device's MAC address, and I looked it up in the backend database, where it was entered correctly. The whole entry in the database looked good and matched how other devices looked at first glance. A check in the RADIUS logs revealed that the entry was found as expected, and ACCESS-ACCEPT was sent back, which is used to give the "ok" to the controller and send the PSK for the matched device in RADIUS attributes. I reconfigured the passphrase on the device, ensuring it matches the DB entry. Everything matched, but the client still failed to connect.

So, I got a better look at what point this failed and did a packet capture between the client and AP. The packet capture revealed normal 802.11 authentication and association, confirming that the client was allowed successfully. However, the 4-way handshake was never completed. After M1 from the AP, the client sends M2, but I never saw M3 from the AP. The AP resends M1, the client answers with M2, and the cycle repeats.

```
EAPOL          171 Key (Message 1 of 4)
EAPOL          193 Key (Message 2 of 4)
EAPOL          171 Key (Message 1 of 4)
EAPOL          193 Key (Message 2 of 4)
EAPOL          171 Key (Message 1 of 4)
EAPOL          193 Key (Message 2 of 4)
EAPOL          171 Key (Message 1 of 4)
EAPOL          193 Key (Message 2 of 4)
```

M1/M2 loop in Wireshark

I learned from my studies that this is evidence of a PSK mismatch, as the message integrity check in M2 would fail. But as I checked the passphrase multiple times, there had to be another problem leading to this.

I enabled all debug messages on the wireless controller and tried authenticating again. Here, I got the crucial piece during the RADIUS authentication: `(ERR): RADIUS/DECODE: parse response op decode; FAIL`. This told me there had to be an issue decoding what the RADIUS server sent to the controller. So, my subsequent packet capture was between the controller and RADIUS. Looking into the attributes of the ACCESS-ACCEPT, I spotted the issue: in the vendor-specific attribute Cisco-AVPair, which is supposed to read "`psk=thisisapassphrase`" was just: "`thisisapassphrase`". The formatting of this attribute was wrong, and the controller did not know what to do with this simple word, as it got no indication that this was supposed to be a passphrase. Therefore, it used the default

passphrase for the SSID, which did not match the client's.

```
AVP: t=Vendor-Specific(26) l=25 vnd=ciscoSystems(9)
   Type: 26
   Length: 25
   Vendor ID: ciscoSystems (9)
   VSA: t=Cisco-AVPair(1) l=19 val=thisisapassphrase
      Type: 1
      Length: 19
      Cisco-AVPair: thisisapassphrase
```

RADIUS packet decode in Wireshark

As most of the clients - at this time, more than a hundred - were working fine, I now had to find out why this was sent wrong for this specific client.

Checking my RADIUS configuration shows that the attribute is just taken as is from LDAP, where it should be stored precisely in this format. After rechecking the LDAP database, I now saw what I overlooked: the "psk=" is missing from this entry. As the LDAP attribute has a name, this was easy to miss.

Attribute Description	Value
objectClass	***inetOrgPerson (structural)***
cn	**76b8e2a7cadb**
jkuSysIpsk	thisisapassphrase

Attribute Description	Value
objectClass	***inetOrgPerson (structural)***
cn	**76b8e2a7cadb**
jkuSysIpsk	psk=thisisapassphrase

Wrong and correct LDAP entry

Now, to find out why this happens, I just had to read the code of the administration application, where I found an error in the method responsible for editing an entry. As only some entries are edited, most worked without issues. Only when changing an existing device is the PSK written without the "psk=" to LDAP, resulting in an unusable entry until deleting and re-adding.

I fixed the code, edited the affected entry, and rechecked the tablet while running a wireless capture. This time, the 4-way handshake was completed without issues, demonstrating that the problem had been fixed successfully.

IV. Conclusion

Many times in our industry, rarely occurring issues that seemingly fix themselves or with obscure methods are thought of as user errors - because they usually are, followed by driver issues and vendor bugs. However, here I saw that there can be hard-to-reproduce problems that need a whole barrage of troubleshooting, including wireless and wired packet captures and debug logs, to track down a small error when programming the editing function of an administrative application. CWAP studies have helped tremendously in interpreting captured data, to not only see what is happening but also why this is happening.