Short Paper: Exploiting WPA2-Enterprise Vendor Implementation Weaknesses through Challenge Response Oracles

Pieter Robyns, Bram Bonné, Peter Quax, Wim Lamotte
iMinds/tUL/UHasselt
Wetenschapspark 2
3590 Diepenbeek, Belgium
{pieter.robyns, bram.bonne, peter.quax, wim.lamotte}@uhasselt.be

ABSTRACT

Many of today's enterprise-scale wireless networks are protected by the WPA2-Enterprise Protected Extensible Authentication Protocol (PEAP). In this paper it is demonstrated how an attacker can steal a user's credentials and gain unauthorized access to such networks, by utilizing a class of vulnerable devices as MSCHAPv2 challenge response oracles. More specifically this paper explains how on these devices, Lightweight EAP (LEAP) MSCHAPv1 credentials can be captured and converted to PEAP MSCHAPv2 credentials by using a rogue Access Point. This man-in-the-middle vulnerability was found to be present in all current versions of Apple's iOS and OS X operating systems, and may impact other devices as well. A proof-of-concept implementation is available that shows how Authentication Server certificate validation and certificate pinning mechanisms may be bypassed. Mitigation strategies for the attack and protective actions which can be undertaken by end-users are also described in this paper.

Keywords

Network security; WPA2-Enterprise; PEAP; LEAP

Categories and Subject Descriptors

C.2.0 [**Computer-communication networks**]: Security and protection.

General Terms

Experimentation, Security

1. INTRODUCTION

Since its inception, wireless networking has become increasingly popular. More and more users desire access to network resources or the internet without having to struggle with network cables. As anyone with a wireless network card can eavesdrop on data sent wirelessly, it is self-evident that data security and user privacy are crucial aspects. This is especially true for enterprises, where confidential company data may be transmitted over the air. Fortunately, this data can be encrypted using a secure communication protocol.

For the average home user, the procotol that is considered most secure for wireless communication is WPA2-PSK. Here, the user configures a single password that is used for authentication. This password is shared with all users that require access to the network. For enterprises, this approach is infeasible: different users may require different access rights on the network, access may need to be revoked to former employees or the password may unintentionally leak to unauthorized parties. Therefore, the most popular choice for enterprises is WPA2-Enterprise. When this protocol is used, each user has their own login and password.

Though WPA2-Enterprise is considered secure in general, many attacks exist that are based on the Man-In-The-Middle (MITM) principle. Here, a victim user is tricked into connecting to a rogue Access Point (AP) that has the same SSID as the enterprise network. To add to the problem, many devices on the market automatically join a wireless network in their Preferred Network List (PNL) by default. This is convenient for the user, but it also introduces the risk of joining a network under control of an attacker [15].

To solve these MITM issues, the authenticity of the APs themselves can be verified by the device. This verification happens in the background, so the user fully relies on the used network protocol for its security. In context of WPA2-Enterprise networks, the IEEE 802.1X Standard specifies that the EAP protocol should be used for this purpose. EAP is an extensible authentication protocol that implements a wide variety of authentication procedures, called EAP methods.

Though EAP methods are well-defined and thoroughly examined for flaws by security experts, a correct protocol implementation is the responsibility of the device vendor. Unsurprisingly, there are subtle differences between various vendor implementations. Some of these may contribute to significant security vulnerabilities, such as those described in this paper.

For our research we focused mainly on the PEAP method, because it is popular, widely supported and considered secure. We tested the PEAP implementation of some of the most popular operating systems used today: Windows, Mac

WiSec'14, July 23–25, 2014, Oxford, UK.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2972-9/14/07 ...$15.00.
http://dx.doi.org/10.1145/2627393.2627411.

OS X, Android and iOS [9]. A practical attack for which Apple devices are particularly vulnerable resulted from our findings. The vulnerability has been reported to Apple prior to the release of this paper, on February 5, 2014.

2. ATTACK DESCRIPTION

Our attack exploits a combination of two vulnerabilities. A first vulnerability is the fact that some devices accept the older LEAP method for authentication. This EAP method is Cisco proprietary and uses the MSCHAPv1 algorithm to authenticate users. Past research has already proven that both MSCHAPv1 and MSCHAPv2 are insecure for various reasons when used without the protection of a TLS tunnel [19]. Since the LEAP method does not establish a TLS tunnel from client (or "Supplicant") to Authentication Server (AS) prior to exchanging credentials, it is vulnerable to a rogue AS MITM attack [6].

The second vulnerability is that when the user configures or joins a PEAP network, some devices reuse the supplied credentials for all supported EAP methods. Hence, LEAP credentials do not have to be entered explicitly by the user. Existing MITM attacks try to capture these LEAP credentials using a rogue AS, and then crack them with dictionary attack tools like `asleap`[1]. In our attack, we will use the credentials for a different purpose.

Before we discuss a practical implementation of our attack, let us first examine how credentials are exchanged in LEAP. The three entities participating in the authentication are the Supplicant, the Authenticator, and the AS. For simplicity, assume that Authenticator and AS reside on the same machine. The LEAP authentication procedure is performed as follows [6]:

1. The Supplicant associates with the AP and exchanges its identity with the AS. This step is identical for all EAP methods.

2. The AS sends an 8-byte challenge C_s, where $C_s = Random8(seed)$, to the Supplicant.

3. The Supplicant generates a 24-byte challenge response R_p, where $R_p = ChallengeResponse(C_s, H)$, $H = MD4(Unicode(PW))$ and PW is the password of the user. R_p is then sent to the AS.

4. The AS calculates $R_{check} = ChallengeResponse(C_s, H)$. The exchange is successful if R_p and R_{check} match.

5. In case of success, an EAP-Success message is sent from Authenticator to the Supplicant. Then, AS and Supplicant switch roles and repeat steps 2 to 4. This time we denote the challenge sent by the Supplicant as C_p, and the response by the AS as R_s.

6. The AS derives the Session Key (SK) as

$$SK = MD5(MD4(Unicode(H)) \\ ||C_s||R_p||C_p||R_s) \quad (1)$$

where "$||$" is the concatenation operator. The AS encrypts this value with the RADIUS secret and sends it to the Authenticator. The Supplicant also derives the SK, so this key can be used for WEP encrypted unicast communication. Finally, a random broadcast key is generated by the Authenticator and sent encrypted with the unicast key to the Supplicant.

[1]This tool can be downloaded from the following URL: `http://www.willhackforsushi.com/?page_id=41`

Note that a LEAP exchange is practically identical to performing two MSCHAPv1 authentications (steps 2 to 4): one from AS to Supplicant ($C_s \rightarrow R_p$) and one from Supplicant to AS ($C_p \rightarrow R_s$). [25].

Next, let us examine PEAP. This authentication method is significantly more complex, and among other features supports MSCHAPv2 mutual authentication to protect against MITM attacks [16, 17]. Assuming cryptographic binding is not used (see Section 5.3), PEAP authentication is performed as follows:

1. The Supplicant associates with the AP and exchanges its identity with the AS.

2. In Phase 1, the Supplicant and AS set up a TLS tunnel similar to the procedure described in RFC 5246 [7]. From the TLS master secret, a Master Session Key (MSK) is derived via a one-way function. This key serves a comparable purpose to the Session Key from LEAP.

3. Phase 2 is performed inside the TLS tunnel and implies usage of an EAP inner authentication method. MSCHAPv2 is frequently used for this purpose. Assuming MSCHAPv2 is used, the AS starts by generating a 16-byte random server challenge $C_s = Random16(seed)$ and sends it to the Supplicant.

4. The Supplicant also generates a random 16-byte peer challenge C_p. Then the challenge response is calculated as $R_p = ChallengeResponse(Challenge(C_s), H)$, where $Challenge(C_s) = SHA1(C_p||C_s||U)[0:7]$, U is the username of the user, $H = MD4(Unicode(PW))$, PW is the password of the user and $[0:7]$ means the first eight bytes of the data. This challenge response is transmitted back to the AS, along with C_p and U.

5. The AS calculates R_{check} analogous to R_p in step 4. R_{check} and R_p must match, or the authentication will fail.

6. The AS calculates a peer challenge response

$$R_s = PeerResponse(MD4(Unicode(H)), \\ M_1, R_p, Challenge(C_p), M_2) \quad (2)$$

where M_1 is the string "Magic server to client signing constant" and M_2 is the string "Pad to make it do more than one iteration". This result is SHA1-hashed and sent to the Supplicant.

7. The Supplicant authenticates the server, completing the MSCHAPv2 inner authentication.

8. An EAP-Result-TLV exchange is performed between AS and Supplicant to indicate the result of the PEAP authentication. Then an EAP-Success message is sent.

9. The MSK is used to derive the WPA2 Pairwise Master Key (PMK) and subsequent keys. Secure transmission of data can begin when the 802.11i four-way handshake [11] is completed.

When comparing the core differences between MSCHAPv1 and MSCHAPv2 credentials from RFCs 2433 and 2759, we can see that they are in fact very minor. Table 1 shows a comparison between the two methods [24, 25, 19].

Though RFC 2759 states that MSCHAPv2 is incompatible with MSCHAPv1 [24], the insignificance of the aforementioned differences led us to the conclusion that some MSCHAPv1 messages can be converted to MSCHAPv2 messages and vice versa.

We will now show that C_s from MSCHAPv1 is identical to $Challenge(C_s)$ from the MSCHAPv2 AS and that R_p from the MSCHAPv1 peer is identical to R_{check} at the MSCHAPv2 server. This way we can be sure that all messages converted from MSCHAPv1 to MSCHAPv2 or vice versa will be accepted by the destination host. For the challenges we derive:

$$Challenge(C_s) = SHA1(C_p||C_s||U)[0:7] \tag{3}$$
$$= SHA1(x)[0:7] \ (C_p \text{ and } C_s \text{ are random}) \tag{4}$$
$$= Random8(seed), \quad \text{if x = random} \tag{5}$$
$$= C_s \tag{6}$$

Given that the $ChallengeResponse$ function is the same in MSCHAPv1 and MSCHAPv2, we derive for the challenge responses:

$$R_s = ChallengeResponse(C_s, H) \tag{7}$$
$$= ChallengeResponse(Challenge(C_s), H) \quad \text{(Eq. 6)} \tag{8}$$
$$= R_{check} \tag{9}$$

With the knowledge that the challenge we get from the PEAP MSCHAPv2 AS can be converted to an MSCHAPv1 challenge (Equation 6), and that the challenge response we get from our LEAP MSCHAPv1 victim can be converted to an MSCHAPv2 challenge response that matches R_{check} on the AS (Equation 9), we devised a relay attack that uses a vulnerable device as an MSCHAPv2 challenge response oracle in order to gain unauthorized access to PEAP networks. Figure 1 shows a schematic representation of our attack.

3. PRACTICAL LEAP RELAY ATTACK

In this section we will show how the MSCHAPv1 to MSCHAPv2 conversion can be exploited in practice. First we will discuss the preconditions for the attack. Then, a practical implementation for attacking Apple devices will be demonstrated.

3.1 Preconditions

A device connecting to a PEAP network is considered vulnerable to our attack when all of the following preconditions are met:

- The device supports the LEAP method.
- The device connects automatically to the PEAP network. This is the default behavior.
- The Authenticator does not require and validate client certificates. Server certificate validation and certificate pinning may be enabled on the client.
- The MSCHAPv2 or MSCHAPv1 inner authentication EAP method is supported and allowed on the AS.

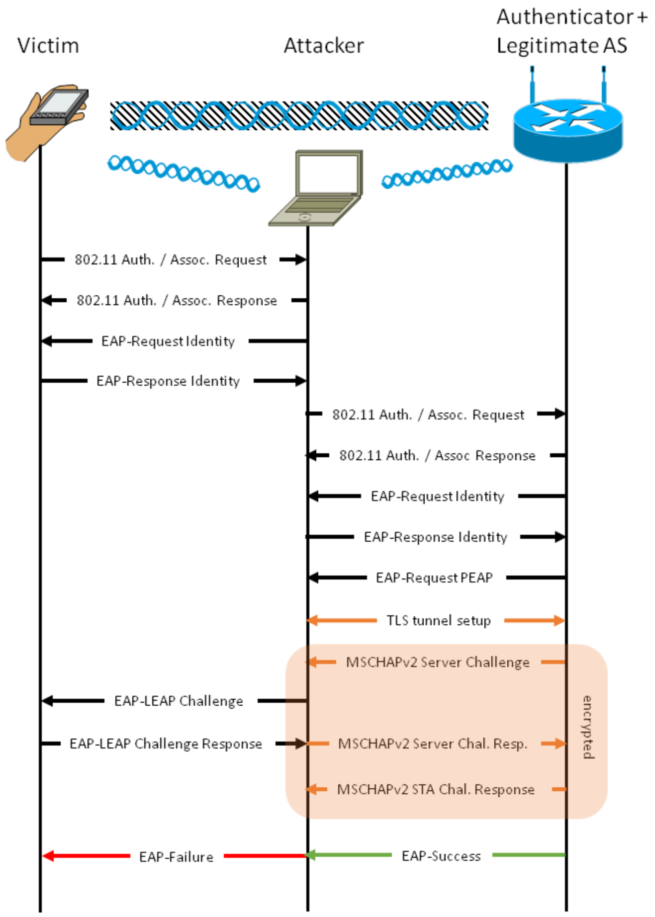
Victim
Attacker
Authenticator + Legitimate AS
802.11 Auth. / Assoc. Request
802.11 Auth. / Assoc. Response
EAP-Request Identity
EAP-Response Identity
802.11 Auth. / Assoc. Request
802.11 Auth. / Assoc Response
EAP-Request Identity
EAP-Response Identity
EAP-Request PEAP
TLS tunnel setup
MSCHAPv2 Server Challenge
EAP-LEAP Challenge
EAP-LEAP Challenge Response
MSCHAPv2 Server Chal. Resp.
MSCHAPv2 STA Chal. Response
encrypted
EAP-Failure
EAP-Success

Figure 1: Schematic representation of the Apple LEAP attack

Note that most of the preconditions listed here are commonly fulfilled by default in enterprise network setups.

3.2 Case study: Apple devices

We will now demonstrate how the exploit can be practically applied to Apple devices (see Figure 1). Our proof-of-concept implementation uses a simple state machine to perform the attack (Figure 2). After successful execution, an attacker gains unauthorized access to the target network by impersonating a legitimate user.

3.2.1 State 1: Association

Before wireless clients can begin the exchange of EAP packets to secured networks, they require association with a wireless AP. We exploit the default auto-join behavior to have clients associate to an AP under our control. In order to accomplish this, we set up a fake wireless AP with the same SSID as the target network. This fake AP broadcasts beacon packets and replies to Probe Requests from clients.

The client will associate or reassociate to our fake network AP when it is closer to the target network AP, because better signal strength is preferred [8]. Alternatively, we can force the client to connect to our fake AP by performing an attack similar to [3]. Since we do not want to receive requests from devices which are not vulnerable, our implementation uses the MAC Organizationally Unique Identifier (OUI) to

	MSCHAPv1	**MSCHAPv2**
C_s	$C_s = Random8(seed)$	$C_s = Random16(seed)$
C_p	$C_p = Random8(seed)$	$C_p = Random16(seed)$
R_s	$R_s = ChallengeResponse(C_p, H)$	$R_s = PeerResponse(MD4(Unicode(H)), M_1, R_p, Challenge(C_p), M_2)$
R_p	$R_p = ChallengeResponse(C_s, H)$	$R_p = ChallengeResponse(Challenge(C_s), H)$

Table 1: Differences between MSCHAPv1 and MSCHAPv2 exchanges

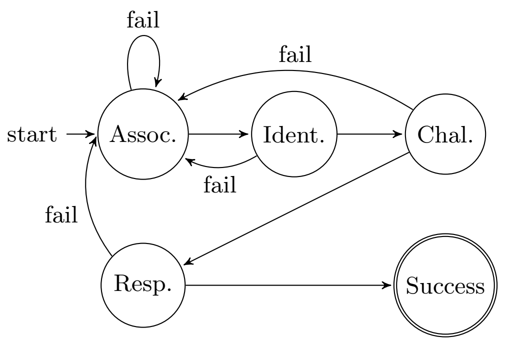

Figure 2: State machine of our attack

identify the device vendor. We can filter out all non-Apple devices this way.

3.2.2 State 2: Identification

The first step after association in WPA2-Enterprise networks is identification. The AS has to know which user wants to authenticate in order to match corresponding credentials. We can learn the identity of the vulnerable device by sending an EAP Identity Request. The device will then reply with an EAP Identity Response which contains the username of our victim.

At this point, data sent over the air is still not encrypted. Hence, some PEAP implementations use anonymous identities. In this case the real username is only disclosed when a TLS tunnel has been established between the Supplicant and the AS. Nonetheless, we can still get the real username in a later phase of our attack.

Our next goal is to get the challenge value from the target AP. We created a modified version of the `wpa_supplicant`[2] tool for this purpose. At the end of this state, the binary executable of this modified version is called from our implementation.

3.2.3 State 3: Challenge

In State 3, we wait for the `wpa_supplicant` tool to establish a TLS tunnel with the target AS and extract an MSCHAPv2 challenge from the inner authentication. We can now see why usage of client certificates would mitigate the attack, as the client certificate validation would not be successful in this case.

When the MSCHAPv2 challenge is retrieved, we pass it on to our tool. Upon receipt, the tool will periodically send LEAP Request messages (containing the extracted challenge) to the Apple device in order to keep the session alive.

[2]`wpa_supplicant` is an open source 802.1X Supplicant implementation by Jouni Malinen.

Device	**Vulnerable**
iPod Touch (iOS 6.1.6)	Yes
iPhone 4 (iOS 7.1)	Yes
iPhone 4S (iOS 7.1)	Yes
Mac OS X 10.8.2 (Mountain Lion)	Yes
Samsung GT-S5570 (Android 2.3.4)	No
Google Nexus 7 (Android 4.4.2)	No
Samsung GT-I8750 (Windows Phone 8.0)	No
Windows 7 Desktop	No

Table 2: Devices vulnerable to the LEAP relay attack

3.2.4 State 4: Response

After receiving the LEAP Request, our victim will reply with a LEAP Response which contains an MSCHAPv1 challenge response to our MSCHAPv2 challenge. Should the target PEAP network enforce anonymous identities, the real or inner identity of the victim will also be revealed to the attacker through this LEAP Response. Next, our implementation will forward the received MSCHAPv1 challenge response as an MSCHAPv2 challenge response to the modified `wpa_supplicant` tool, which will in turn forward the challenge response to the legitimate PEAP network AS.

3.2.5 State 5: Success

When the AS receives our modified challenge response, authentication proceeds as usual, which means the AS has to authenticate to our Supplicant. However, since we are not in possession of the NT password secret, we cannot derive H. Hence, when receiving the peer challenge response from the AS, we are forced to accept any sent value.

After this, the MSCHAPv2 inner authentication will complete successfully and the port will be authenticated. The AS and our Supplicant will derive the MSK, and from this we can derive the PMK. We now have all components required to access resources on the internal network.

4. TEST RESULTS

We tested our attack on devices from multiple vendors. Table 2 shows on which devices the LEAP relay attack was successfully performed.

Assuming that the same network protocol stack is used on all Apple operating systems, we concluded from these results that all Apple devices are vulnerable. The attack was executed analogously on each device for multiple APs, using different AS implementations. These included a TP-Link WN422G using `hostapd` and the latest `freeradius` implementation on the same machine, a Linksys WRT54G AP using the latest `freeradius` implementation on a dedicated machine, and a Ubiquity UniFi AC 3.x AP using Windows RADIUS server on a dedicated machine.

5. MITIGATION

The attack we described in this paper can be mitigated in various ways. We will discuss five methods in this section.

5.1 Client certificates

In State 3 of our attack, a TLS tunnel has to be established between the attacker and the target network AS. When using client certificates, each client's certificate must be provided in the "Client Hello" phase of the TLS tunnel setup. When this verification fails, the TLS setup will be aborted and hence, our attack will fail because the MSK cannot be derived from the TLS master secret.

This countermeasure is very effective and by far the most secure. However, it would require a lot of administration effort for enterprises. Especially in enterprises with a Bring Your Own Device (BYOD) policy, because a signed certificate for every device allowed on the network must be installed on the AS.

5.2 iPhone Configuration Utility

iPhone configuration profiles allow the network administrator to choose which EAP methods clients must use. They are the only way in which LEAP can be disabled on Apple devices. If this method is chosen to mitigate the attack, care must be taken in BYOD environments: if one user does not install the network profile, the attack can nevertheless be executed. Furthermore, network profiles can be accidentally removed by the user. For these reasons, security is put in the hands of the end user and therefore this method is not as secure as using client certificates.

5.3 Cryptobinding

An optional feature described in the PEAP version 2 internet draft is cryptographic binding [17]. This feature introduces the use of a new Type-Length-Value (TLV), the CryptoBinding TLV, to address MITM attacks. A two-way handshake containing a Compound MAC Key (CMK) proves that the two authentications terminate at the same PEAP peer and PEAP server [14].

To calculate the CMK, the Supplicant is required to use keying material from both Phase 1 and Phase 2 of the PEAP exchange. In practical terms this involves the calculation of the Tunnel Key (TK) and the Inner Session Key (ISK). These keys are combined in the cryptobinding algorithm to form the CMK.

The TK is calculated similarly to the MSK from the TLS master secret, and would be available to an attacker. The ISK however, is calculated at the Supplicant as $ISK = InnerMPPESendKey||InnerMPPERecvKey$. The $InnerMPPESendKey$ and $InnerMPPERecvKey$ are both derived from the inner MSCHAPv2 Master Key (MK), which is derived as

$$MK = GetMasterKey(MD4(Unicode(H))[0:15], R_s) \quad (10)$$

Since H is unknown to the attacker, the ISK cannot be derived and authentication will fail.

If all consumer devices would support cryptobinding, this method would probably be the best way to mitigate our attack. However, from our experiments we concluded that Apple devices do not support cryptographic binding at this time.

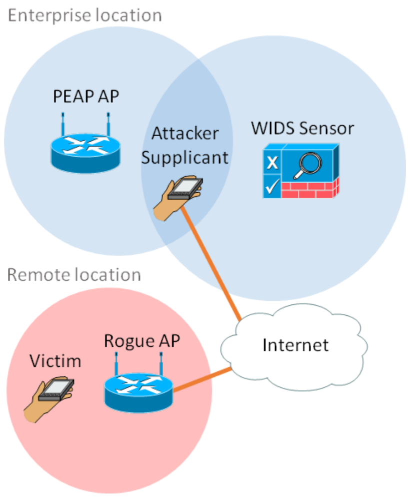

Figure 3: Remote LEAP relay attack

5.4 Intrusion detection

A signature based WIDS might be able to detect our attack by passively scanning for LEAP requests. Since these packets will never be sent by a legitimate AP, IDS sensor nodes have a clear indication that the network is under attack. Analytic approaches to detect our attack may include station counts, association counts, OS fingerprinting and RSSI value analysis, though these methods often lead to false positives [4, 12].

As a final note, we would like to indicate that care must be taken when relying on a WIDS for detection of an attack, as we believe that in many cases the IDS may be bypassed. For example, a victim may be in range of the rogue AP, while the latter is out of range from a WIDS sensor. In Figure 3, an example scenario is shown where the relay attack is executed over the internet.

5.5 Rogue AP mitigation

If one can prevent the attacker from setting up a rogue AP, the LEAP relay attack cannot be performed. Several methods have already been developed to mitigate the rogue AP attack [2, 18, 22]. However, we believe not all of these mitigation strategies will work. Context leashing will only work when creating the rogue AP in a different context, EAP-SWAT will have the same problems as PEAP, and other mitigation strategies in these works rely partly on the awareness and expertise of the user. We believe link layer protection mechanisms would be the most effective in this case.

6. FUTURE WORK

Future work could be done by using the same attack principles described in this paper. From our experiments we determined that other devices, for example Android devices, do not employ certificate pinning by default. If the victim user did not configure a server certificate, we believe a more generic MITM attack may be executed as described in RFC 7029 [10]. Note that in this case, the preconditions are stricter: it is required that *server* certificates are not used by the Android device, which was not the case for Apple devices.

Another option for future work would be to implement the attack for EAP-TTLS. This EAP method is similar to PEAP, and we believe the attack may therefore apply to EAP-TTLS secured networks as well.

7. RELATED WORK

A similar, generalized MITM attack on tunneled authentication protocols was demonstrated by N. Asokan et al. in 2002 [1]. Related attacks on PEAP vendor implementations such as the EAP dumb-down attack were introduced by Raul Siles in 2013. This attack exploits the default lack of certificate validation in mobile devices. However, for Apple devices, the dumb-down attack requires user intervention whereas our attack is automatic [20]. Furthermore, a correct configuration of authentication server certificates does not mitigate our attack for Apple devices.

Other related attacks were proposed at numerous security conferences. In 2008, Joshua Wright and Brad Antoniewicz demonstrated how EAP credentials such as MSCHAPv2 exchanges can be collected using `freeradius-wpe`, a rogue AS implementation [21]. By using the `asleap` tool, these credentials can then be cracked with a dictionary attack [5]. More recently, in 2012, Moxie Marlinspike showed how MSCHAPv2 credentials can be cracked in less than 24 hours using cloud-based FPGA nodes [13]. Finally, Josh Yavor indicated the dangers of BYOD and default certificate validation behavior of mobile devices in 2013 [23].

8. CONCLUSIONS

We demonstrated how MSCHAPv1 challenges and challenge responses can be converted to MSCHAPv2 challenges and challenge responses. Then, we indicated how this can be exploited in practice when a Supplicant supports the insecure LEAP method and when credentials are reused between EAP methods.

From our experiments we concluded that all Apple devices are currently vulnerable to our attack. Mitigation is possible in various ways. However, we noted why some mitigation strategies might not be feasible for enterprises. Therefore, users and network managers should take care when their devices satisfy all mentioned vulnerability preconditions.

9. ACKNOWLEDGEMENTS

We would like to thank Arno Barzan from the Expertise Centre for Digital Media for his support and insightful suggestions.

10. REFERENCES

[1] N. Asokan, V. Niemi, and K. Nyberg. Man-in-the-middle in tunnelled authentication protocols. In *Security Protocols*, pages 28–41. Springer, 2005.

[2] K. Bauer, H. Gonzales, and D. McCoy. Mitigating evil twin attacks in 802.11. In *Performance, Computing and Communications Conference, 2008. IPCCC 2008. IEEE International*, pages 513–516, Dec 2008.

[3] A. Cassola, W. Robertson, E. Kirda, and G. Noubir. A Practical, Targeted, and Stealthy Attack Against WPA Enterprise Authentication. In *Proceedings of NDSS*, volume 2013, 2013.

[4] M. Ciampa. *CWNA Guide to Wireless LANs*. Cengage Learning, 2012.

[5] Cisco. Dictionary Attack on Cisco LEAP Vulnerability, 2003. `http://www.cisco.com/en/US/tech/tk722/tk809/technologies_security_notice09186a00801aa80f.html`.

[6] A. DeKok and A. Sulmicki. Cisco LEAP protocol description, 2001. `http://freeradius.org/rfc/leap.txt`.

[7] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol. RFC 5246, IETF, August 2008.

[8] M. S. Gast. *802.11 Wireless Networks: The Definitive Guide, Second Edition*. O'Reilly, 2005.

[9] A. Gupta, R. Cozza, and C. Lu. Market Share analysis: Mobile phones, worldwide, 4Q13 and 2013. *Gartner*, 2014.

[10] S. Hartman and M. Wasserman. Extensible Authentication Protocol (EAP) Mutual Cryptographic Binding. RFC 7029, IETF, October 2013.

[11] C. He and J. C. Mitchell. Analysis of the 802.11 i 4-way handshake. In *Proceedings of the 3rd ACM workshop on Wireless security*, pages 43–50. ACM, 2004.

[12] K. Hutchison. Wireless Intrusion Detection Systems. *SANS Institute InfoSec Reading Room*, October 2004.

[13] M. Marlinspike. Divide and Conquer: Cracking MS-CHAPv2, 2012. `https://www.cloudcracker.com/blog/2012/07/29/cracking-ms-chap-v2/`.

[14] Microsoft. Cryptobinding, 2014 (accessed). `http://msdn.microsoft.com/en-us/library/cc238384.aspx`.

[15] L. Nussel. The Evil Twin problem with WPA2-Enterprise. *SUSE Linux Products GmbH*, 2010.

[16] A. Palekar, D. Simon, J. Salowey, H. Zhou, G. Zorn, and S. Josefsson. Protected EAP Protocol (PEAP). Work in Progress 6, IETF, March 2003.

[17] A. Palekar, D. Simon, J. Salowey, H. Zhou, G. Zorn, and S. Josefsson. Protected EAP Protocol (PEAP) Version 2. Work in Progress 10, IETF, October 2004.

[18] V. Roth, W. Polak, E. Rieffel, and T. Turner. Simple and effective defense against evil twin access points. In *Proceedings of the First ACM Conference on Wireless Network Security*, WiSec '08, pages 220–235, New York, NY, USA, 2008. ACM.

[19] B. Schneier, Mudge, and D. Wagner. Cryptanalysis of Microsoft's PPTP Authentication Extensions. *CQRE '99*, October 1999.

[20] R. Siles. EAP dumb-down attack. In *RootedCON 2013*, pages 27–28. DinoSec, 2013.

[21] J. Wright. FreeRADIUS-WPE, 2008. `http://www.willhackforsushi.com/?page_id=37`.

[22] Z. Yang, A. C. Champion, B. Gu, X. Bai, and D. Xuan. Link-layer protection in 802.11i wlans with dummy authentication. In *Proceedings of the Second ACM Conference on Wireless Network Security*, WiSec '09, pages 131–138, New York, NY, USA, 2009. ACM.

[23] J. Yavor. The BYOD PEAP Show. In *DefCon 21*. iSEC Partners, 2013.

[24] G. Zorn. Microsoft PPP CHAP Extensions, Version 2. RFC 2759, IETF, January 2000.

[25] G. Zorn and S. Cobb. Microsoft PPP CHAP Extensions. RFC 2443, IETF, October 1998.