GDS
RESEARCH · LABS
A DIVISION OF GOTHAM DIGITAL SCIENCE

The Black Art of Wireless Post-Exploitation: Bypassing Port-Based Access Controls Using Indirect Wireless Pivots

Gabriel Ryan (@s0lst1c3)

August 2017

Contents

Abstract

Most forms of WPA2-EAP have been broken for nearly a decade. EAP-TTLS and EAP-PEAP have long been susceptible to rogue access point attacks, yet most enterprise organizations still rely on these technologies to secure their wireless infrastructure. The reason for this is that the secure alternative, EAP-TLS, is notoriously arduous to implement. To compensate for the weak perimeter security provided by EAP-TTLS and EAP-PEAP, many organizations use port based NAC appliances to prevent attackers from pivoting further into the network after the wireless has been breached. This solution is thought to provide an acceptable balance between security and accessibility.

The problem with this approach is that it assumes that EAP is exclusively a perimeter defense mechanism. In a wireless network, EAP plays a subtle and far more important role. WPA2-EAP is the means through which the integrity of a wireless network's physical layer is protected. Port-based access control mechanisms rely on the assumption that the physical layer can be trusted. Just as NACs can be bypassed on a wired network if the attacker has physical access to the switch, they can also be bypassed in a wireless environment if the attacker can control the physical layer using rogue access point attacks.

In this paper, we will apply this concept by introducing a novel technique for bypassing port-based access control mechanisms in wireless networks. We will also introduce a technique that leverages similar principles to steal Active Directory credentials without direct network access. In doing so, we will challenge the assumption that reactive approaches to wireless security are an acceptable alternative to strong physical layer protections such as WPA2-EAP using EAP-TLS.

The research contained in this whitepaper was first introduced at DEF CON 25, BSides Las Vegas, and 44con in similarly named presentation. The slides can be found at:

- https://media.defcon.org/DEF%20CON%2025/DEF%20CON%2025%20presentations/DEFCON-25-Gabriel-Ryan-The-Black-Art-of-Wireless-Post-Exploitation-UPDATED.pdf

I. Introduction

Wireless networks have been plagued with security problems since their inception. Even with modern wireless security mechanisms, perimeter defense can be difficult to do effectively. Many companies recognize this, and focus their resources on minimizing the impact of wireless breaches rather than preventing them outright. During red team engagements, the wireless perimeter is cracked within the opening days of the assessment, or it isn't cracked at all. From an attacker's perspective, the real challenge lies in moving laterally out of the isolated sandbox in which network administrators typically place their wireless networks. Enterprise network teams are typically aware of this fact, and many will attempt to justify weak wireless perimeter security by pointing out how difficult it is to pivot from the WLAN into production.

However, preventing an attacker from doing so is only easy when the network in question is used exclusively for basic functions such as providing Internet connectivity to employees. When wireless networks are used to provide access to sensitive internal infrastructure, the issue of access control gets significantly messier. A door must be provided through which authorized entities can freely traverse. As with cryptographic backdoors, a door that requires a key is a door no less.

In this paper, we will provide a brief history of rogue access point attacks and their use as a means of compromising WPA2-EAP networks. We will then discuss why companies often focus on breach containment rather than breach containment when attempting to secure wireless networks, in spite of the existence of technologies such as EAP-TLS. We will then demonstrate how the problem of breach containment can be just as complex as the problem of breach prevention by discussing some common attempts at effective network access control. Finally, we will demonstrate how port-based access control mechanisms can be completely evaded on WPA2-EAP networks that use weak forms of EAP. In the process of doing this, we will introduce two new wireless attacks: the *Hostile Portal Attack* and the *Indirect Wireless Pivot.*

II. Background

Evil twin attacks first surfaced in the early 2000s, when they were first documented by C.W. Klaus in his "Wireless LAN Security FAQ" in 2002 [1]. Shortly afterwards, the more sophisticated Karma attack was discovered and documented by Dino Dai Zovi and Shane Macauly in 2004 [3]. By 2008, Joshua Wright and Brad Antoniewitz had adapted the rogue access point attack into a means of breaching WPA-EAP networks with their release of Freeradius-wpe [4]. By 2014, Domanic White and Ian de Villiers had introduced several improvements to the Karma attack that could be used to counteract the numerous improvements in supplicant design that had diminished the effectiveness of Karma up until that point [5]. White and de Villiers also introduced multiple new developments in attacking EAP [5]. Finally, in 2017 the Lure10 Attack was developed by George Chatzisofroniou as a means of exploiting Window 10's WiFi-Sense capabilities [30].

It is clearly evident that rogue access point attacks have not only existed for quite some time, but have evolved considerably in the decade and a half since they were first introduced. Throughout this time period, rogue AP attacks have primarily been used to fill one of two roles: stealing credentials or breaching protected wireless networks. In this paper we will explore something a bit different: the use of rogue AP attacks as a means of lateral movement.

II.A EAP Overview

Before we begin, we must take an obligatory look at WPA2-EAP implemented with EAP-PEAP or EAP-TTLS , along with the evil twin attacks to which it can be susceptible. From a simplified, high level perspective, EAP authentication occurs between a supplicant (the wireless client) and an authentication server [6][7][8].

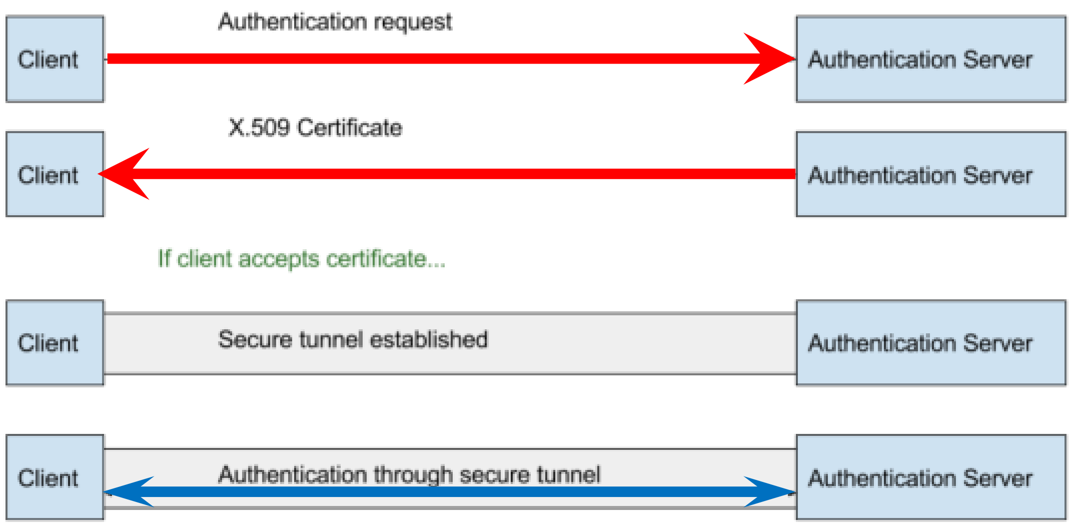

Figure 1 – A Simplified EAP Workflow

As shown in the diagram above, the supplicant first makes an authentication request to the authentication server. The authentication server then attempts to prove its identity to the supplicant by responding with an x.509 certificate. The supplicant must then make the decision to trust or distrust the authentication server by accepting or rejecting the validity of the x509 certificate. This initial transaction between the supplicant and the authentication server is referred to as the Outer Authentication process. If the supplicant accepts the server's x509 certificate as valid, the Outer Authentication process succeeds and a secure tunnel is established between the supplicant and the authentication server. The supplicant and authentication server then transition to the Inner Authentication process, which takes place through the secure tunnel [6][7][8].

Remember that the encryption provided by WPA2 does not become active until after the authentication process is complete. This means that the entire EAP authentication process occurs over open wireless. Without the secure tunnel, the Inner Authentication process can be sniffed by an attacker. In fact, legacy implementations of EAP such as EAP-MD5 were highly susceptible to this issue [13].

Now that we have provided a high-level overview of how EAP-PEAP and EAP-TTLS work, let's look at the protocol in more detail. So far, we have left out a critical component of the EAP authentication process: the authenticator.

The authenticator serves as an intermediary between the supplicant and the authentication server. In the case of WPA2-EAP, the access point serves as the authenticator, and forwards information back and forth between the wireless client and the authentication server. To do this, the authenticator communicates with the authentication server over Layer 7 using the RADIUS protocol. The authenticator also communicates with the supplicant, but at Layer 2 in the OSI stack [6][7][8].

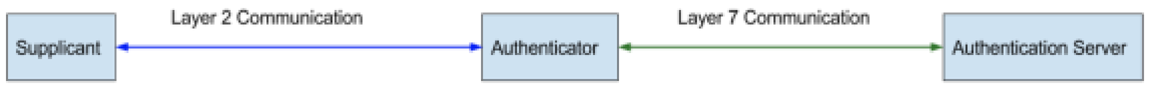

Figure 2 – In WPA2-EAP, the Authenticator (a.k.a. access point) facilitates communication between the Supplicant (a.k.a. wireless client) and the Authentication Server.

Therefore, our complete EAP-TTLS and EAP-PEAP diagram looks like the one shown below. Notice that it is nearly identical to the first diagram we used to describe the authentication process, but now shows Authenticator forwarding communication between the Supplicant and the Authentication Server [6][7][8].

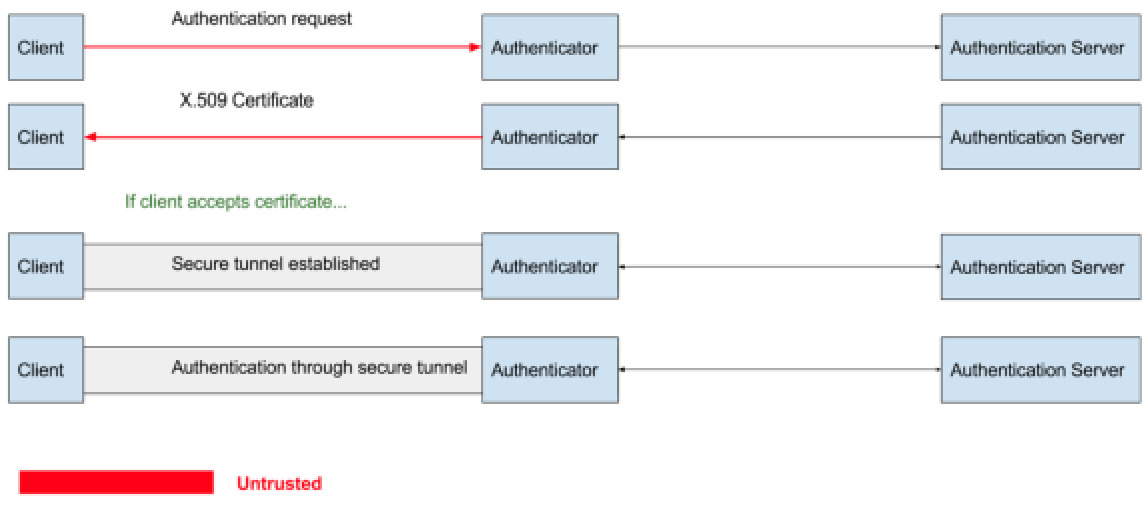
Client
Authentication request
Authenticator
Authentication Server
X.509 Certificate
If client accepts certificate...
Secure tunnel established
Authentication through secure tunnel
Untrusted

Figure 3 – Communication between the wireless client and Authenticator (a.k.a. access point) is untrusted and unencrypted until the secure tunnel is established.

II.B Classic Attacks Against WPA2-EAP Networks

The problem with this system lies in the fact that the Authenticator is acting as an open access point until the authentication process is complete. This means that just like an open access point, it is vulnerable to rogue access point attacks. From the supplicant's perspective, there is no way to accurately verify the identity of the authenticator.

Additionally, the x509 certificate serves as the only mechanism through which the supplicant can verify the identity of the authentication server. Assuming that the supplicant is not configured to automatically reject invalid certificates, which is not always the case, the onus is placed on the user to reject invalid certificates when prompted [4][5]. As you can imagine, this becomes dangerous within the context of large enterprise organizations in which a single attack can be carried out against hundreds of users with varying levels of situational awareness.

This leads us to the classic attack against WPA using EAP-PEAP or EAP-TTLS described in 2008 by Brad Antoniewicz and Joshua Wright during their seminal presentation at Shmoocon 2008 [4]. The attacker first uses an evil twin attack to force an initial association, then coerces the victim into commencing the authentication process with his or her own rogue RADIUS server. The attacker's RADIUS server responds to client's authentication request by issuing a self-signed certificate [4].

If the supplicant is properly configured, the supplicant will not automatically accept the certificate [4]. Instead, the user will be prompted to either accept or reject the untrusted certificate [4]. The degree of warning that the user receives varies greatly depending on the operating system and version in use.

If the user accepts the untrusted certificate, the attacker and victim establish a secure tunnel through which the Inner Authentication is completed. The result is that the attacker obtains both the EAP challenge and response, which can be cracked to obtain the user's password or NT hash offline [4].

II.C Cracking MS-CHAPv2

Further increasing the severity of this issue is the fact that the strongest Inner Authentication protocol available for use with EAP-PEAP and EAP-TTLS is MS-CHAPv2 [31][14]. In 2012, Moxie Marlinspike and David Hulton found that MS-CHAPv2 uses the same 56-bit DES encryption as NTLMv1 [31][14], and that its security can be reduced to that of a *single* DES encryption [31][14]. Due to these issues, the a captured EAP challenge and response can be converted to a password equivalent NT hash in less than 24 hours with a 100% success rate when using an FPGA based cracking rig such as Crack.sh (previously CloudCracker) [28]. This means that the EAP challenge and response are not only easily obtainable by an attacker performing a rogue access point attack, but are trivially crackable as well, regardless of the complexity of the victim's password.

II.D Solution: EAP-TLS

EAP-TLS was introduced in 2008 by RFC 5216, at least in large part, as a proposed mitigation to the severe security issues that affect EAP-PEAP and EAP-TTLS [10]. The strength of EAP-TLS lies in its ability to stop rogue access point attacks in their tracks through the use of mutual authentication using x.509 certificates during the outer authentication process [10].

Despite its ability to mitigate the issues that affect EAP-PEAP and EAP-TTLS, EAP-TLS remains a wildly unpopular option for network administrators [11]. The reason for this is that use of mutual certificate based authentication makes EAP-TLS considerably more difficult to integrate into existing network architectures. Not only is it daunting to provision each device that requires access to the wireless network with a certificate, but many devices may not even support certificate based authentication. The potential for incompatibility of this nature is less of an issue in office environments than it is industrial or medical settings, but exists none-the-less.

This introduces a classic security vs. convenience scenario, in which network administrators are forced to choose between authentication mechanisms with known weaknesses or an authentication mechanism that is highly secure yet seemingly impractical. As a result, a market gap is created for products that can be used to compensate for the security issues found in EAP-PEAP and EAP-TTLS but are still easy to use. The current trend is to focus on breach containment, rather than breach prevention. The goal of this paper is to explore the question of whether or not this approach actually works within a wireless environment.

II.E Common Approaches To Wireless Breach Containment

Network Access Control (NAC) appliances are by far the most commonly used tool used to contain wireless breaches that we see on pentesting engagements. NAC appliances grant or deny access to network resources by distinguishing between authorized or unauthorized endpoints [12]. When a new endpoint is added to the wireless network, the NAC identifies whether or not the endpoint is an authorized or unauthorized device. If the new endpoint is recognized as an unauthorized device, the NAC places a restriction on the device, such as placing the device on a quarantine VLAN or blocking the device's port entirely.

Historically, NAC appliances have fallen into one of two categories: agent-based and agentless. Agent-based NACs work through software components that are installed on each authorized device on the network [12]. These software components are referred to as agents, and perform deep internal checks on authorized devices while communicating with the "brain" of the NAC [12]. This capability to perform internal checks of network endpoints makes agent-based NACs highly effective. However, their reliance on software that must be installed on all authorized network endpoints makes them nearly as impractical to deploy as EAP-TLS.

On the other side of the spectrum we have agentless NACs. These devices use a combination of passive fingerprinting and active scanning to identify unauthorized devices [12]. Since they do not require the use of an agent to operate effectively, they are much easier to deploy than agent-based NACs [12]. However, traditional agentless NACs are also unable to examine the internals of network endpoints. This means that to bypass an agentless NAC, an attacker must merely masquerade as an authorized device on the network [12].

II.E.1 Recurring Dilemma: Insecurity vs. Impracticality

Once again we run into our recurring dilemma: insecurity vs impracticality. Administrators who wish to use a NAC appliance to contain wireless breaches must choose between secure agent-based NACs that can be as difficult to deploy as EAP-TLS, or agentless NACs that are highly ineffective as security mechanisms. This creates a high demand for products that offer the deep interrogation capabilities of agent-based NACs, but without the additional overhead [13]. This creates yet another market gap, and numerous so-called “next generation” NAC vendors have risen to the occasion to fill it.

II.E.2 Next-generation NACs: the Best of Both Worlds?

One approach commonly used by next-generation NACs is to use WMI to interrogate new devices as they are added to the network. This allows the NAC to perform internal checks without the use of an agent. Devices that use this technique are usually marketed as easily deployable replacements for agent-based NACs.

The problem with this idea is that it requires the NAC to use a service account to authenticate over SMB with every device that is added to the network. Not only must this service account have login rights across the entire domain, but it must possess enough administrative power to perform deep device interrogations. This creates a single point of failure that introduces two additional threat scenarios to the network’s overall security posture.

In the worst-case scenario, these NACs are deployed on networks in which SMB signing has not been enabled. This creates a situation in which an attacker can leverage the NAC’s authentication attempts using SMB Relay attacks to authenticate with every authorized device on the network. In the best-case scenario, in which SMB signing has been enabled at the Group Policy level [15], you still have a scenario in which password hashes for a privileged account are being sent to every device that connects to the network.

The point here is not to call out particular NAC vendors for their shortcomings, but rather to reinforce the idea that “security with convenience” is often a paradox. Many well thought-out attempts have been made to reconcile this paradox throughout the security industry’s relatively short history. However, ignoring this principle often leads to unforeseen complications, such as the ones we just discussed and the ones we are about to explore throughout the rest of this paper.

III. A Typical Containment Scenario

So far, the question we have been attempting to answer is whether or not deploying a NAC is an effective strategy in stopping an attacker from directly attacking sensitive network resources after a wireless breach has occurred. However, we should consider the possibility that this line of discussion may in fact be missing the point. Any conclusion we derive from this debate is moot if we cannot depend on NACs to prevent indirect attacks against authorized devices.

Consider a scenario in which we have breached the perimeter of a wireless network that is used to provide access to sensitive internal resources such as PCI or HIPAA data. The network is protected by a NAC appliance that has placed us into a quarantine VLAN. The sensitive network resources are located on a restricted VLAN that cannot be accessed from the sandboxed VLAN on which we are currently located. An authorized device is connected to the wireless network as well, but is located on the restricted VLAN. Presumably we already have the RADIUS credentials used by this device, as we would have had to have stolen these already in order to access the network in the first place. However, in the IV.C Overcoming MS-CHAPv2 section of this document, we will demonstrate that the attacks we are about to describe can succeed even if we do not have access to the device's credentials.

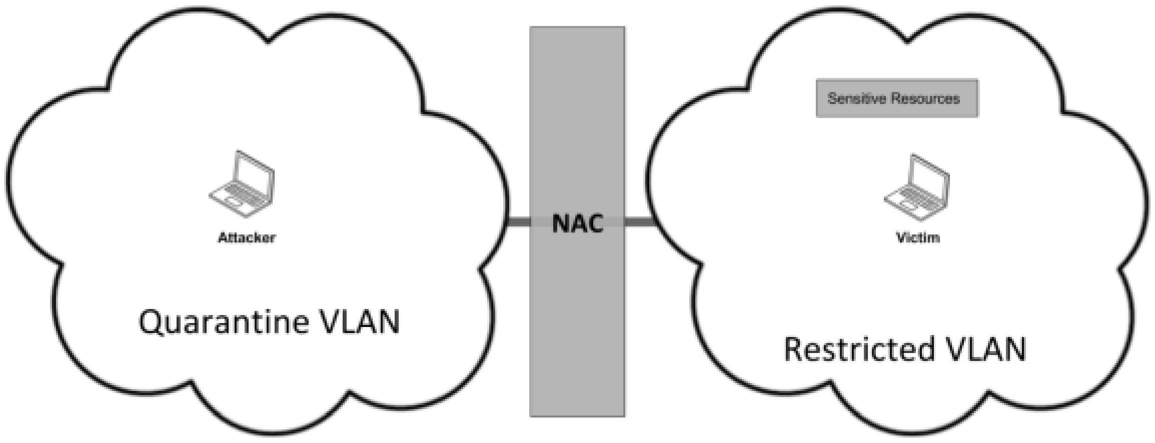

Figure 4 – A typical scenario encountered when attacking a WPA2-EAP network that uses EAP-PEAP. After gaining access to the wireless network by using a rogue AP attack to steal RADIUS credentials, the attacker is immediately placed on a quarantine VLAN and prevented from accessing production data.

In the next two sections, we will introduce two new attacks that can be used to pivot indirectly from the quarantine VLAN to the restricted VLAN by performing a rogue access point attack against the authorized device. The first attack is a *Hostile Portal Attack*, which can be used to steal Active Directory credentials without direct network access. The second attack is an *Indirect Wireless Pivot,* which is a kill chain that leverages the Hostile Portal attack as a means of bypassing port-based network access control mechanisms.

IV. Hostile Portal Attacks

Hostile Portal attacks operate very similarly to the captive portals typically used to protect open wireless networks in coffee shops and hotels. Captive Portals use a combination of DNS spoofing, DNS redirection, and HTTP redirection to force wireless clients to visit a login page. This is accomplished by using a DHCP Option to instruct wireless devices to use a specialized DNS server once they have connected to the network. The DNS server then resolves all DNS requests to the captive portal login page.

Since client devices can always specify a DNS server manually, many captive portals also redirect all DNS traffic to the specialized DNS server. Similarly, the destination IP of any HTTP packets may also be altered to prevent the client device from accessing HTTP resources via IP address rather than by hostname.

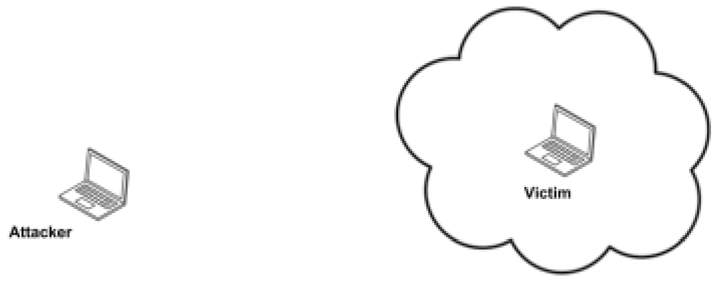

Figure 5 – The victim lies on a separate network as the attacker.

A Hostile Portal operates very similarly. The attacker first uses a rogue access point attack to force the client to associate with the Hostile Portal, as shown in the diagram below.

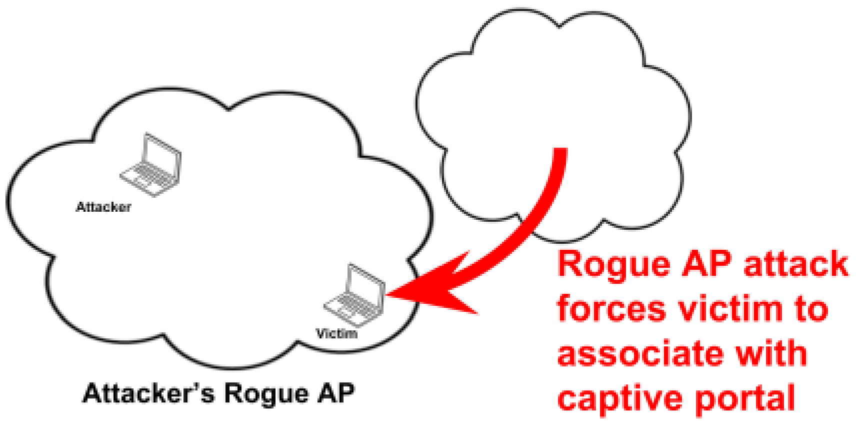
Attacker
Victim
Attacker's Rogue AP
Rogue AP attack forces victim to associate with captive portal

Figure 6 – The attacker uses a rogue AP attack to force the victim to associate with a captive portal.

However, instead of redirecting traffic to a login page, as in a captive portal, the hostile portal redirects traffic to a rogue

SMB server running on the attacker's machine. The result is that any time the victim makes an HTTP request, it is forced to use NTLM to authenticate with the attacker's rogue SMB server.

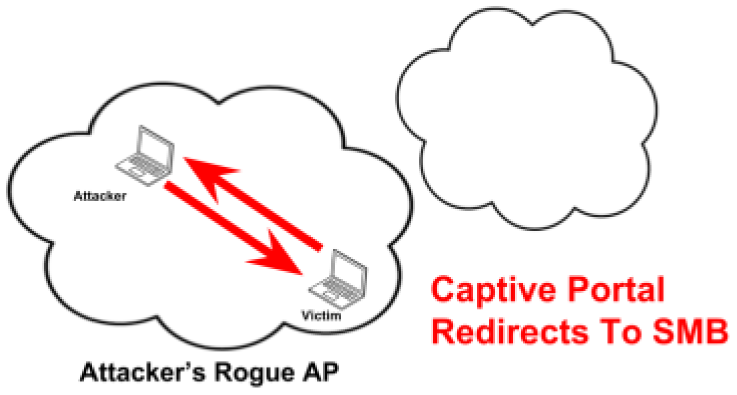

Figure 7 – The attacker's captive portal redirects HTTP traffic to the attacker's rogue SMB server, forcing the client to authenticate with the attacker using NTLM.

In addition to performing this forced Redirect to SMB [24], the attacker also poisons all LLMNR/NBT-NS lookups made by the victim. Since the victim has been removed from its usual habitat and has been forced to connect to the attacker's network, it can be guaranteed that any attempts at NetBIOS name resolution made by the victim will be resolved only by the attacker.

IV.A What this gets you: lots and lots of NTLM Hashes

The Hostile Portal Attack is a method for quickly and efficiently harvesting large numbers of NTLM hashes within an enterprise wireless environment. The results are similar to what you would get using LLMNR/NBT-NS poisoning [23], with some distinct advantages:

- **Stealthy:** No direct network access is required
- **Large Area of Effect:** Works across multiple subnets – you get everything that is connected to the wireless network
- **Efficient:** This is an active attack that forces clients to authenticate with you. The attacker does not have to wait for a network event to occur, such as with LLMNR/NBT-NS poisoning.

IV.B Hostile Portal Attacks Against WPA2-EAP Networks

The attack is fairly straightforward to perform as long as we are attacking an open network or we have prior knowledge of the victim's password or NT password hash. The attacker merely has to use a rogue access point attack to force the victim to associate with his or her hostile portal. However, the introduction of WPA2-EAP complicates matters significantly. Of course, attacking implementations that use EAP-TLS is out of the question. However, even weaker forms of EAP can pose a challenge if MS-CHAPv2 is used as the Inner Authentication protocol. The reason for this is that MS-CHAPv2 requires mutual authentication between the supplicant and the authentication server [29]. To understand how we can overcome this limitation, let's take an in-depth look at the MS-CHAPv2 protocol.

The MS-CHAPv2 protocol begins with an authentication request sent from the client to authentication server [29].

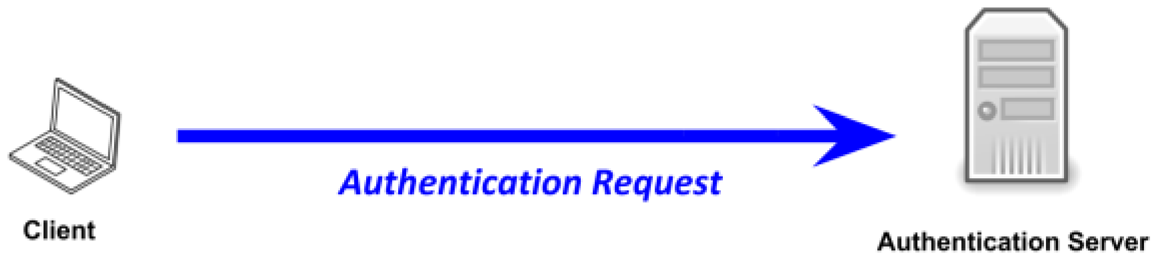

Figure 8 – The client sends an authentication request to the authentication server.

The authentication server then responds with an MS-CHAPv2 challenge, which is a plaintext string of characters [29].

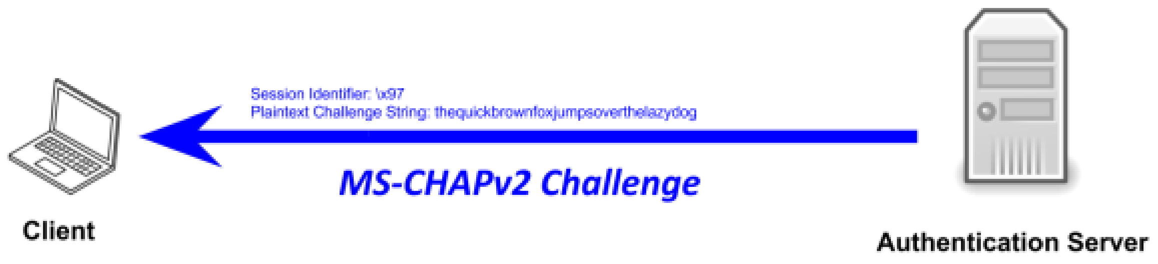

Figure 9 – The authentication server responds with an MS-CHAPv2 challenge.

The client must then prove knowledge of its own password by using its password construct the MS-CHAPv2 response, which is sent back to the server [29].

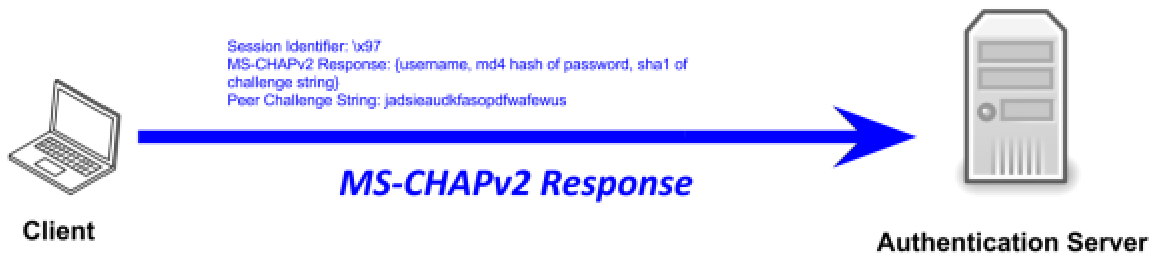
Session Identifier: \x97
MS-CHAPv2 Response: (username, md4 hash of password, sha1 of challenge string)
Peer Challenge String: jadsieaudkfasopdfwafewus
MS-CHAPv2 Response
Client
Authentication Server

Figure 10 – The client's password is used to construct the MS-CHAPv2 response, which is sent to the authentication server.

The authentication server then evaluates the response. What happens next is where things get interesting. If the response is invalid, the authentication attempt fails and a failure message is sent back to the client. However, if the response is valid, the authentication server must then authenticate itself with the client. It does this by constructing an Authentication Response using the peer challenge string and MS-CHAPv2 response sent by the client, as well as the NT hash of the user's password. The Authentication Response is sent to the client with the Authentication Success message [29].

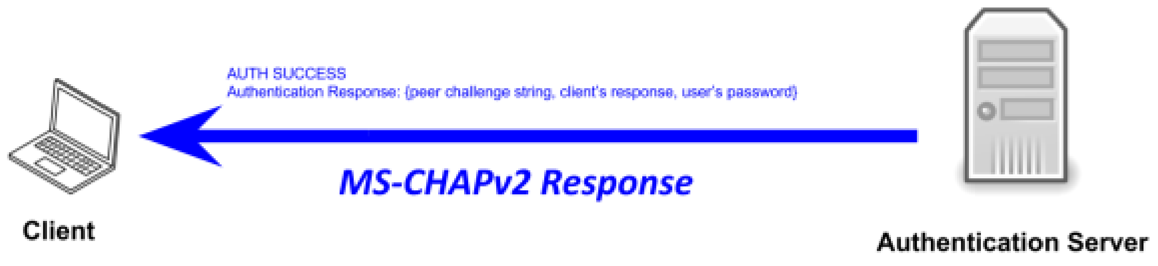
AUTH SUCCESS
Authentication Response: (peer challenge string, client's response, user's password)
MS-CHAPv2 Response
Client
Authentication Server

Figure 11 – If the MS-CHAPv2 response is valid, the authentication server must then authenticate itself with the client using an Authentication Response constructed from the client's password.

This is where traditional rogue access point attacks against WPA2-EAP networks begin to fall apart. Although the attacker's authentication server can complete the first three steps of the authentication process, which is enough to harvest RADIUS credentials, the attacker presumably does not have knowledge of the victim's password. Therefore, the attacker's authentication server will fail this final step of the authentication process.

IV.C Overcoming MS-CHAPv2

In order to use our Hostile Portal attack against WPA2-EAP networks, we need to find a way of overcoming this problem. This is true for any rogue access point attack that requires the client to fully associate with the attacker, including captive phishing portals and wireless MITM attacks. Fortunately, we have a least a couple of options at our disposal. For weak passwords, we can use the autocrack 'n add technique introduced by Dominic White and Ian de Villiers in 2014 [5]. For stronger passwords, we can simply leverage the cryptographic flaws in MS-CHAPv2 to crack the MS-CHAPv2 challenge and response offline before finishing the attack later.

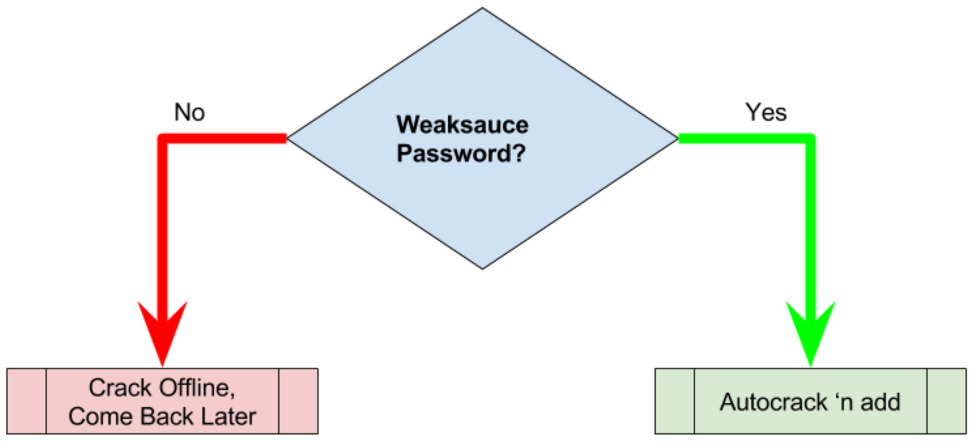

Figure 12 – Depending on the strength of the victim's password, the attacker can use an "autocrack 'n add" attack or can simply crack the victim's password and resume the attack later.

Let's talk about both of these two options in further detail.

IV.C.1 First Solution: Auto Crack 'n Add Technique

If the victim of the attack is using a weak RADIUS password, the best option is to use a technique that was pioneered by Dominic White and Ian de Villiers in 2014 known as "auto crack 'n add" [5]. Hostapd uses a text file known as the eap_user file as a primitive database for storing user credentials.

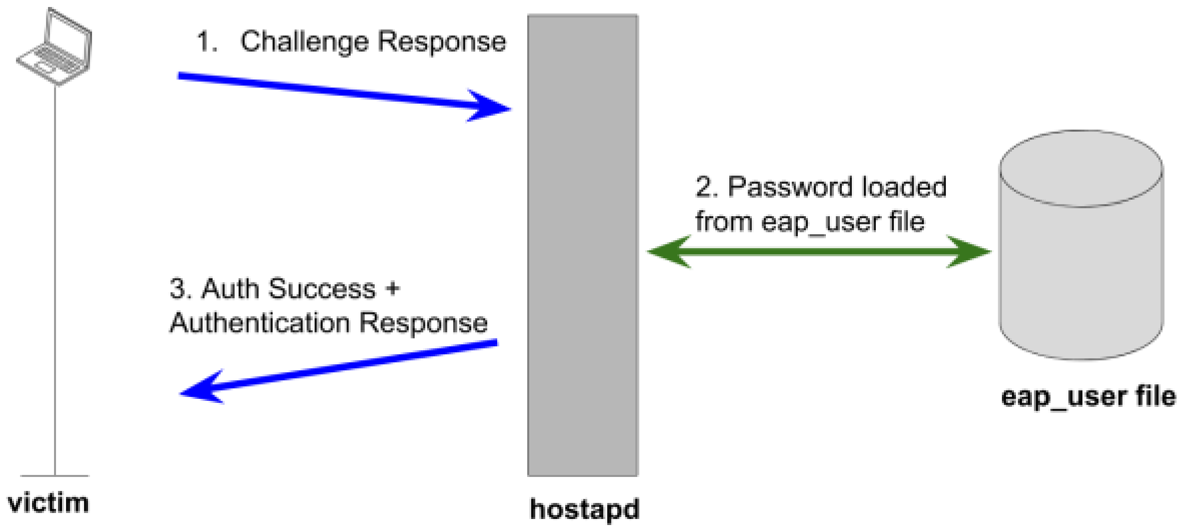
1. Challenge Response
2. Password loaded from eap_user file
3. Auth Success + Authentication Response
eap_user file
victim
hostapd

Figure 13 – Hostapd loads the client's password or NT password hash from the eap_user file immediately before constructing the Authenticate Response to send to the client.

When a user authenticates with hostapd using MS-CHAPv2, the user's password is retrieved from the eap_user file and used to construct the Authentication Response. As previously mentioned, if the user's password is not found in the eap_user file, the authentication attempt fails because hostapd's integrated RADIUS server cannot prove knowledge of the user's password.

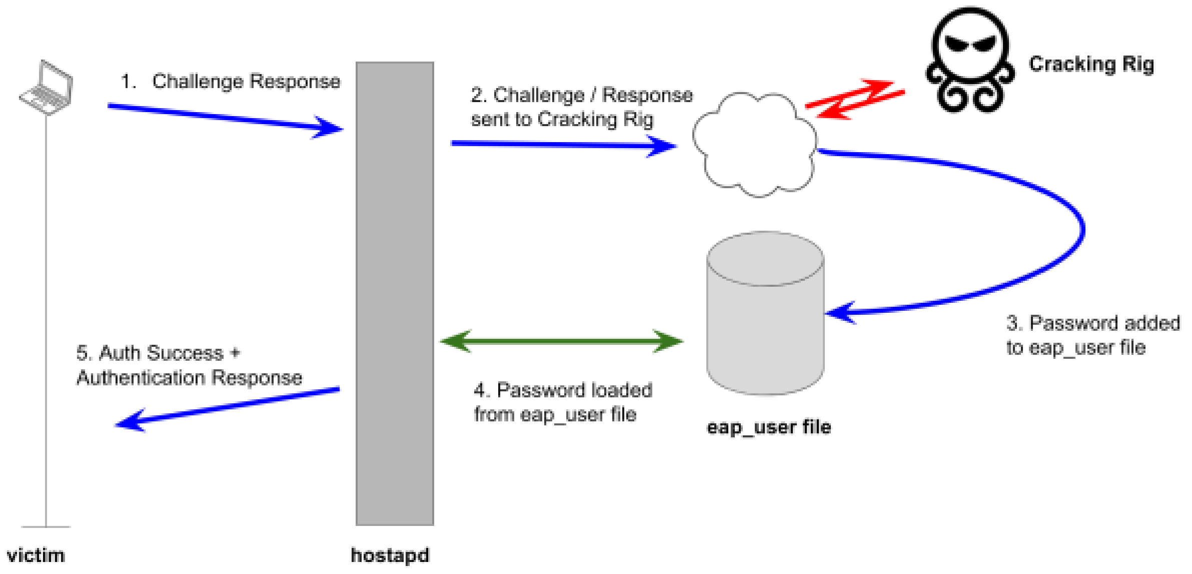
1. Challenge Response
2. Challenge / Response sent to Cracking Rig
Cracking Rig
3. Password added to eap_user file
4. Password loaded from eap_user file
5. Auth Success + Authentication Response
eap_user file
victim
hostapd

Figure 14 – In an autocrack 'n add attack, the eap_user file is updated with a cracked hash immediately before hostapd attempts to retrieve the password or password hash.

When autocrack ‘n add is used, the MS-CHAPv2 challenge and response are first sent to a cracking rig (local or remote) [5]. The cracked credentials are then appended to the end of the eap_user file. If the challenge and response are cracked fast enough, the cracked credentials are added to the eap_user file before hostapd attempts to retrieve them [5]. Even if the challenge and response cannot be cracked before hostapd accesses the eap_user file, causing the initial authentication attempt to fail, the attack will eventually succeed during subsequent authentication attempts. When weak passwords are used, this process can take seconds [5].

IV.C.2 Second Solution: Crack Offline, Finish Later

For more complex passwords, the Autocrack ‘n Add technique may not be feasible [5]. When this is the case, the attack can be completed by first capturing a RADIUS challenge and response and cracking it offline. When the cracking process is complete, the attacker then adds the credentials to the eap_user file and initiates the hostile portal attack.

Although this dramatically increases the time required to complete the attack, remember from II.C Cracking MS-CHAPv2 that MS-CHAPv2 is vulnerable to a divide and conquer attack which can be used to obtain a password equivalent NT hash in a fairly short amount of time [31][14]. This means that the lifecycle of the attack still falls within the time-frame allotted for most time-boxed infrastructure penetration tests, let alone a time-frame that would be acceptable to an actual attacker.

V. Bypassing Port-Based Access Controls Using Indirect Wireless Pivots

Stealing NTLM hashes is useful, but still does not address the scenario we introduced in III. A Typical Containment Scenario. Let's talk about how we can use the Hostile Portal attack as the start of a kill chain that can be used to bypass port-based access controls on wireless networks. We call this technique an *Indirect Wireless Pivot*.

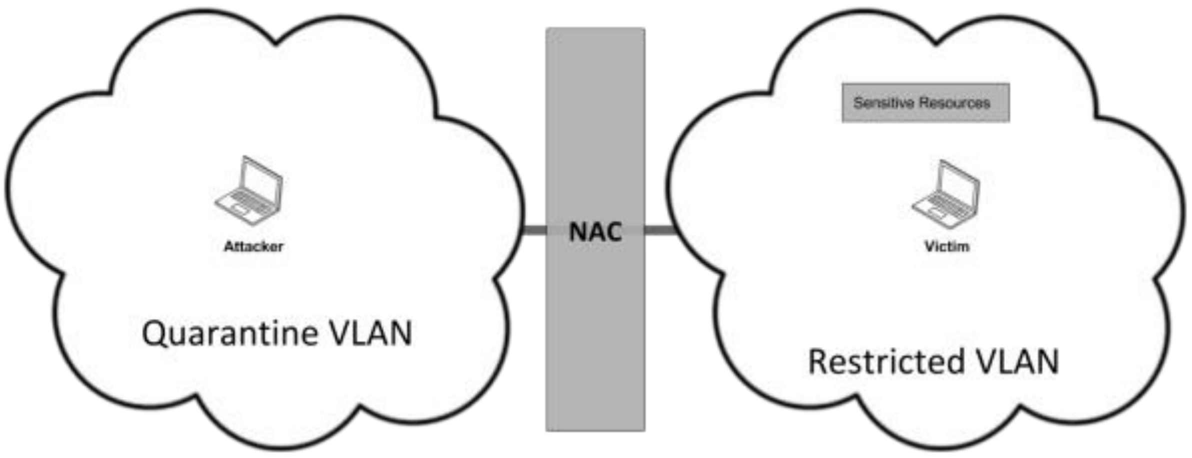

Figure 15 – The containment scenario revisited. The attacker is confined to a quarantine

Let's assume we have two external wireless interfaces at our disposal. Our first interface is currently being used to connect to the target network, which is shown above. However, our second interface remains unused. We begin the attack by using this second interface to perform a rogue access point attack against the target network. This forces the victim device to associate with our second wireless interface.

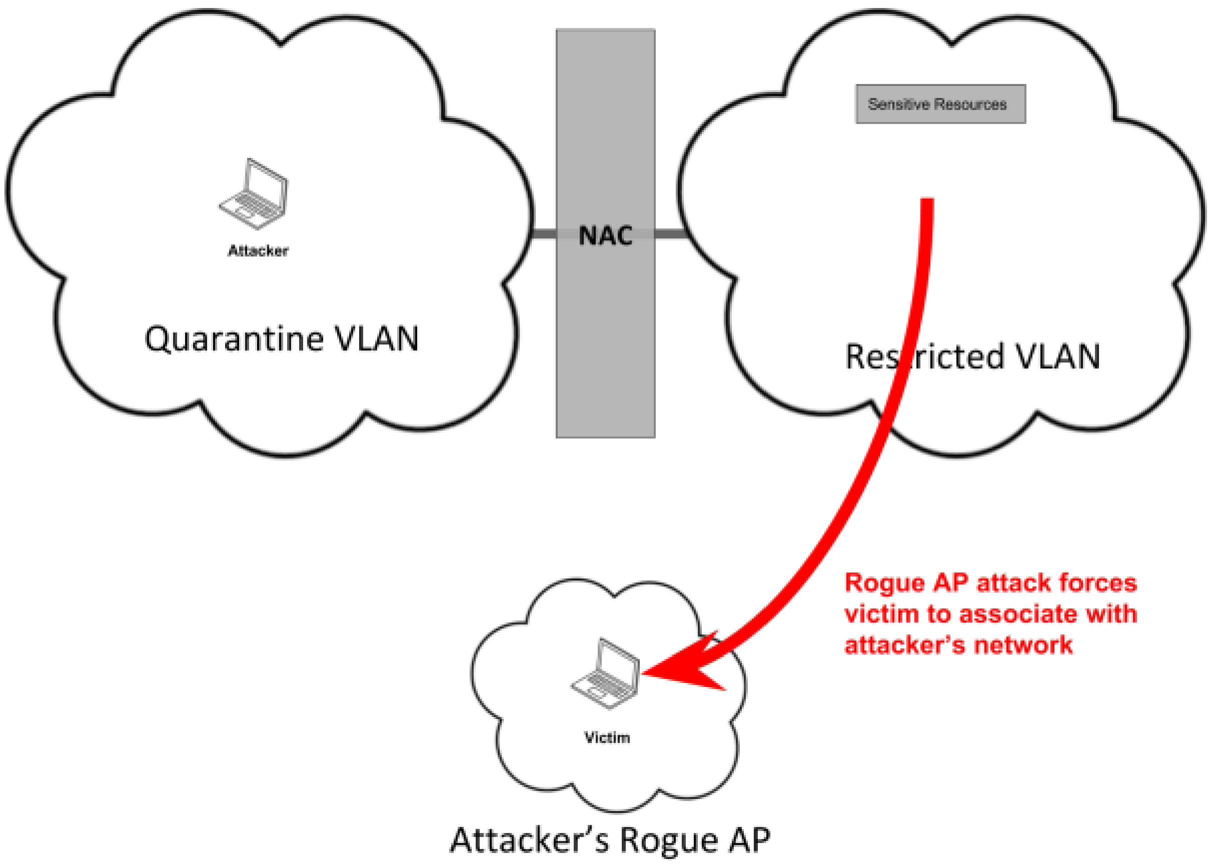

Figure 16 – The attacker uses a rogue AP attack to force the victim to associate with the attacker's second wireless interface. This removes the victim from an environment in which it is protected by the NAC.

The victim device is now associated with a wireless network that we control. At this point, we can use a Hostile Portal attack to force the victim into performing NTLM authentication with our rogue SMB server, as shown below. This allows us to capture the NTLM hash of the victim's password.

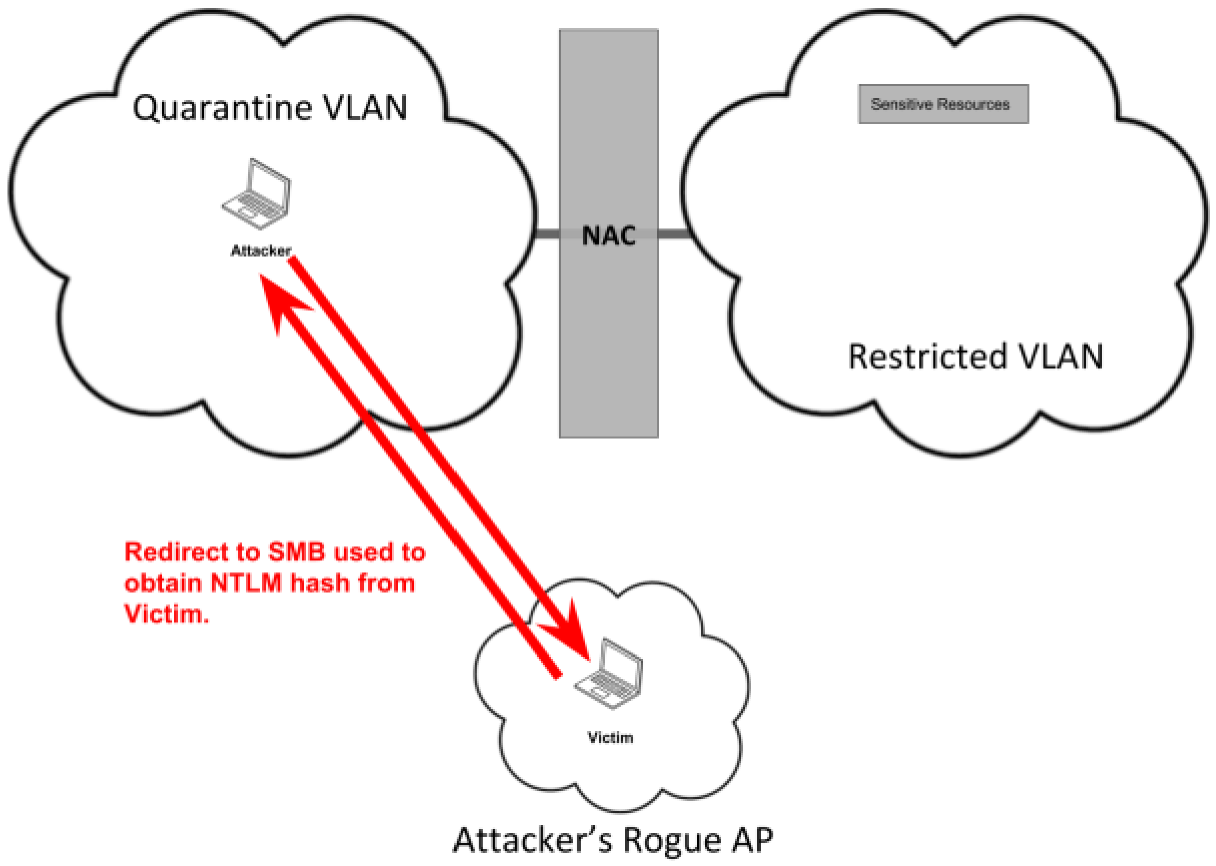

Figure 17 – The attacker's hostile portal redirects all HTTP requests to an SMB share on the attacker's machine. This forces the victim to perform NTLM authentication with the attacker.

We then crack the NTLM hashes offline. When the cracking process completes, we return and resume the attack where we left off. The cracked credentials allow us to place a timed payload on the victim device, as illustrated in *Figure 18* below.

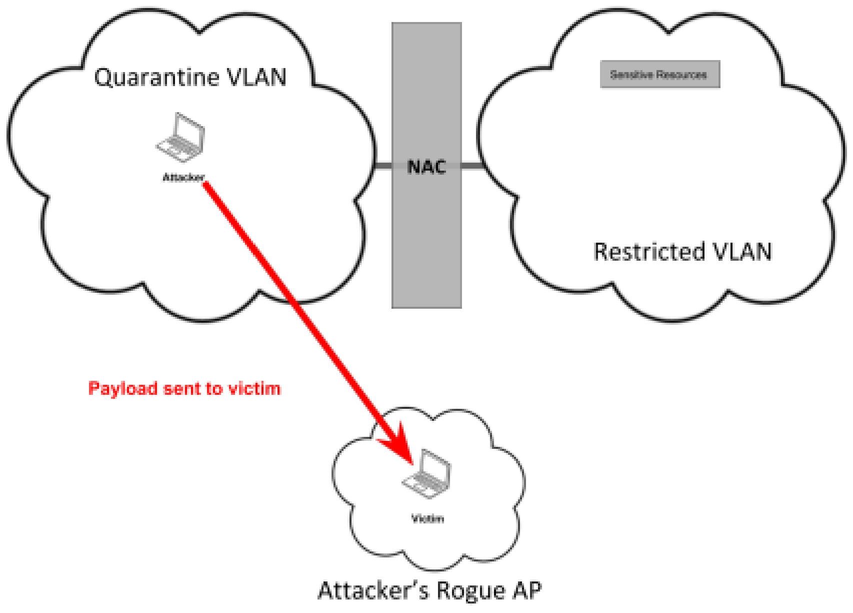

Figure 18 – The attacker uses cracked Active Directory credentials to place a timed payload on the victim device.

We then kill our rogue access point, allowing the victim device to reassociate with the target network as shown below. Since the victim is still an authorized device, the NAC appliance places it on the Restricted VLAN as before.

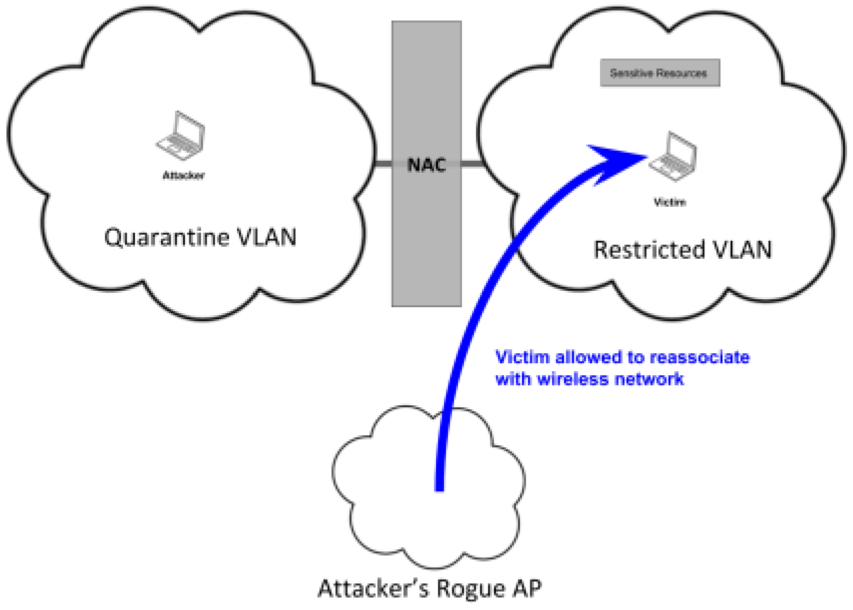

Figure 19 – The attacker allows the victim to reassociate with the target wireless network by killing the rogue AP. Since the victim is an authorized device, it is moved back to the restricted VLAN as before.

Finally, we simply wait for the timed payload execute, sending a reverse shell to our first network interface.

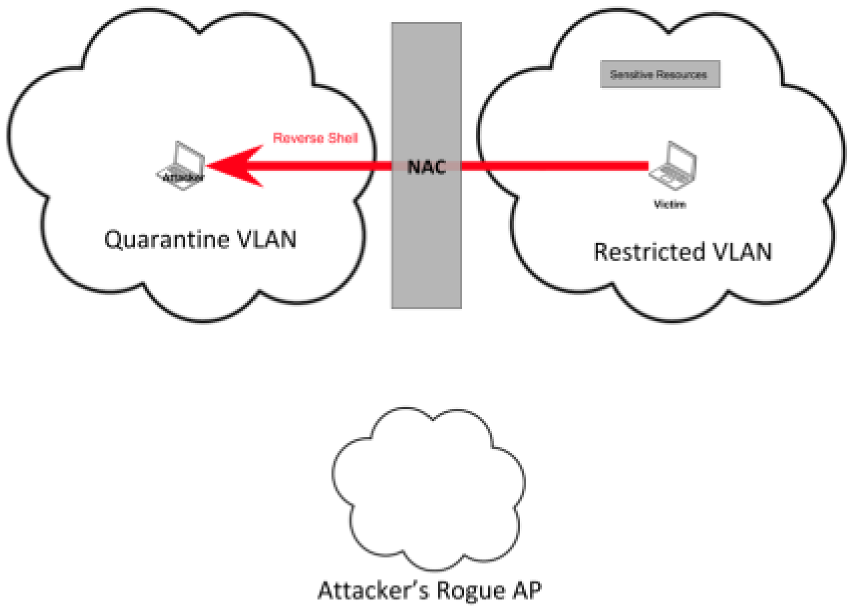

Figure 20 – The attacker waits for the timed payload on the victim device to execute, and receives the resulting reverse shell. This allows the attacker to gain access to the restricted VLAN by pivoting through the authorized device.

Using this technique renders the potential effectiveness of the NAC irrelevant, as we have completely taken it out of the picture in order to perform the attack.

This technique can be further improved upon by using an SMB Relay attack. Let us imagine our scenario again, this time with two victims instead of one.

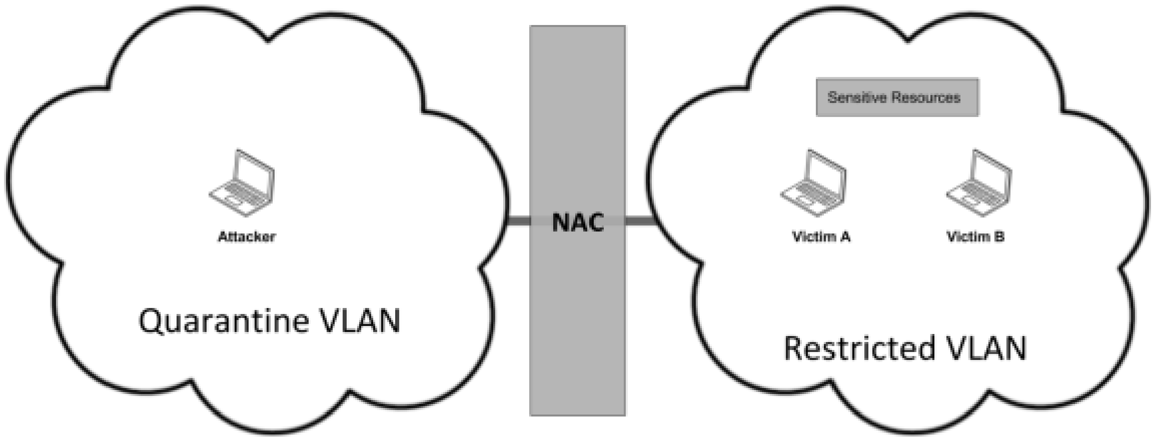

Figure 21 – A modification of our containment scenario. The attacker is restricted to the quarantine VLAN. Two authorized victim devices are present on the restricted VLAN.

As before, we begin by using our second interface to perform a rogue access point attack that forces both victims to associate with our hostile portal as shown in the diagram below.

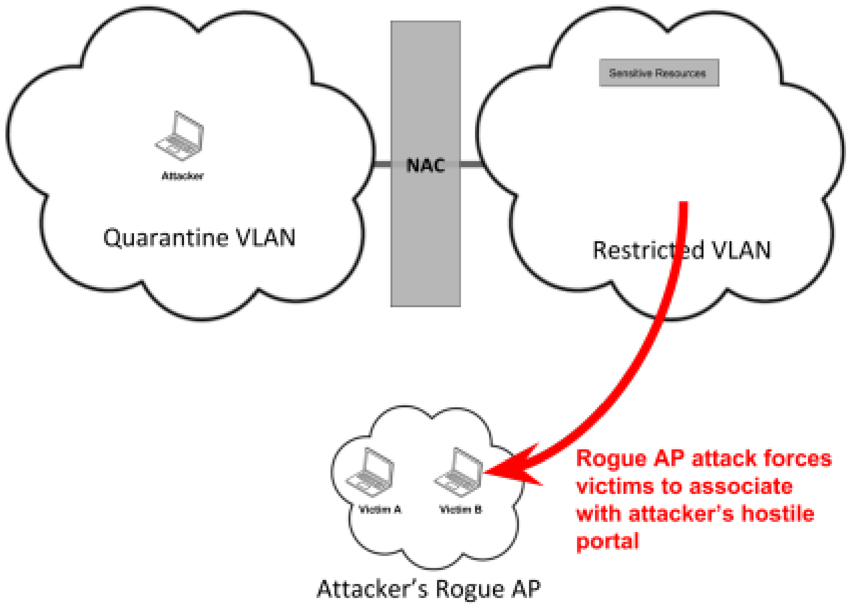

Figure 22 – The attacker uses a rogue AP attack to force both victims to associate with the attacker's second wireless interface. This removes the victims from an environment in which they are protected by the NAC.

We then perform our Hostile Portal attack. However, instead of using the Hostile Portal attack to capture credentials, we use it to perform an SMB Relay attack from one victim to another as shown below. This allows us to place a timed payload on one of the victims.

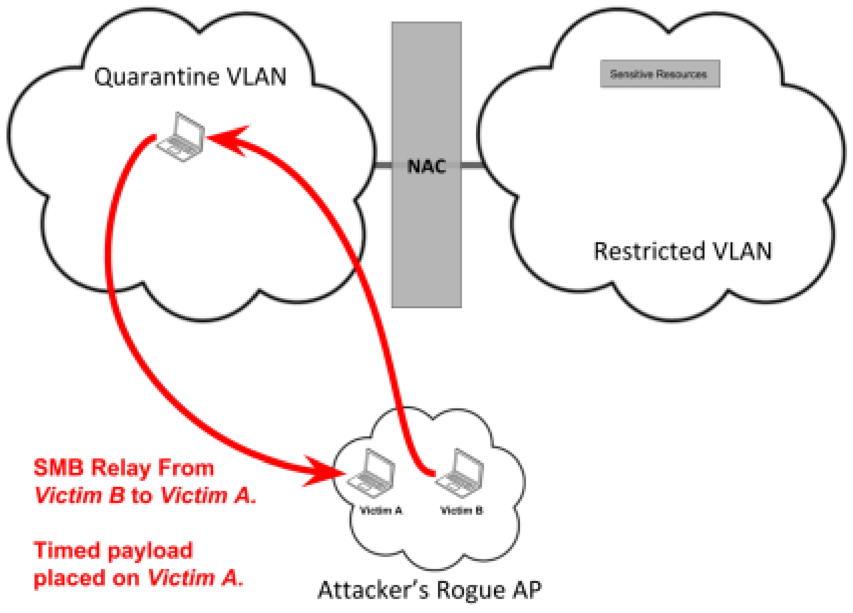

Figure 23 – The attacker uses a hostile portal attack to initiate an SMB Relay from one victim to the other. This allows the attacker to place a timed payload on one of the victim devices nearly instantaneously.

We then kill our rogue access point, allowing the victims to re-associate with the target network as shown below. As before, the NAC places the victims on the restricted VLAN.

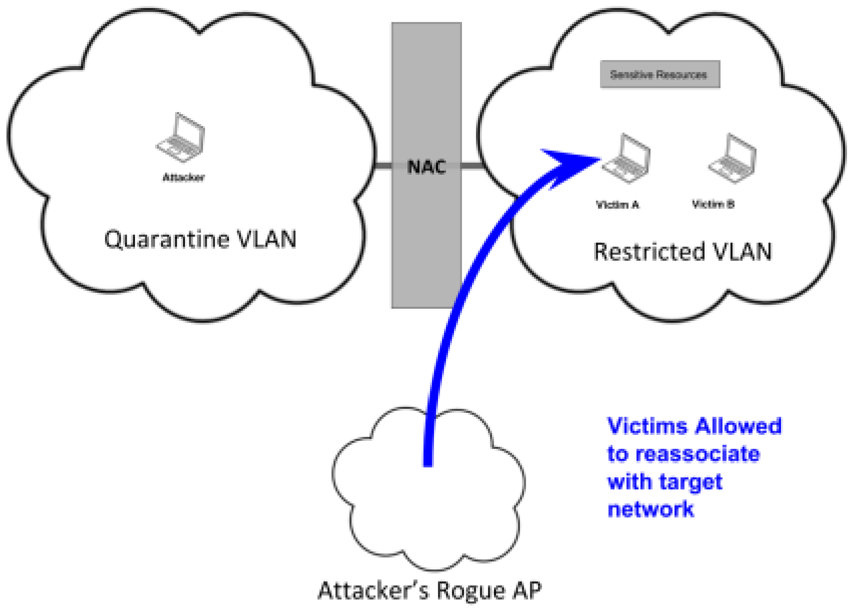

Figure 24 – The attacker allows the victims to reassociate with the target wireless network by killing the rogue AP. Since the victims are both authorized devices, they are returned to the restricted VLAN by the NAC.

Finally, we simply wait for the timed payload to execute, sending a reverse shell back to our first wireless interface as shown below.

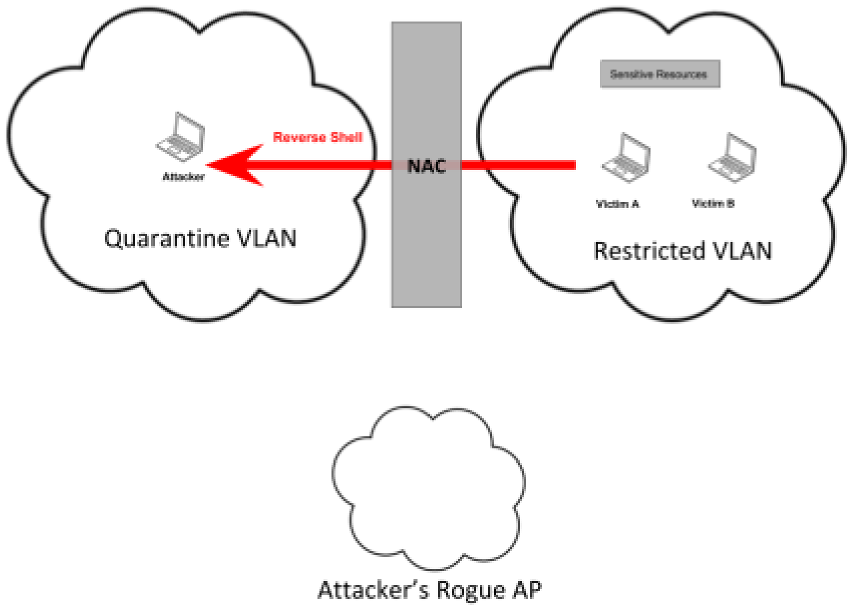

Figure 25 – The attacker waits for the timed payload on the infected victim device to execute, and receives the resulting reverse shell. This allows the attacker to gain access to the restricted VLAN by pivoting through the authorized device.

Conclusion

The reason why the Indirect Wireless Pivot works is that port-based access controls rely on the assumption that the physical layer can be trusted. Port-based NAC appliances work by distinguishing between authorized and unauthorized endpoints after they have been added to the network. Violating an access control policy causes the NAC to impose a *restriction*. In a wired network, this is a *physical* restriction. In a wireless network, this can only be a *logical* restriction.

When we deploy a NAC appliance to protect a wired network, we assume that the physical security controls protecting the network itself are strong enough to prevent an attacker from simply walking into the building, unplugging a trusted endpoint from a switch, attacking it, then plugging it back into the network. In essence, this is what we are doing when we perform a rogue access point attack. In a wireless environment, strong physical security controls are less of a mitigating factor. As such, we are forced to rely on WPA2-EAP as the primary means of protecting the integrity of the network's physical layer. When weak forms of WPA2-EAP are used, the attacker can freely control the physical layer using rogue access point attacks, rendering NAC mechanisms ineffective.

This demonstrates that port-based NAC mechanisms do not effectively mitigate the risk presented by weak WPA2-EAP implementations. It also demonstrates that the addition of a port-based NAC appliance to a wireless network does not make the use of EAP-TTLS or EAP-PEAP any less inappropriate if the network in question is used to grant access to sensitive information (i.e. PCI or HIPAA data).

Fortunately, EAP-TLS is not nearly as difficult to deploy as it was ten years ago. Active Directory can be leveraged to deploy EAP-TLS, enabling administrators to push configurations and certificates to clients via Group Policy. The best approach is to use an internal certificate authority while leveraging Active Directory as much as possible. For mobile devices, certificates can also be distributed to clients using a solid MDM or BYOD onboarding solution [27]. It is even possible to use Let's Encrypt for this purpose, although the developers of Let's Encrypt state that this is far from a perfect solution [27].

As a community, we should question whether reactive approaches to security can ever be as effective as proactive breach prevention. We should question whether it is truly a sound business decision to neglect EAP-TLS in favor of a more reactive approach that focuses on access control and threat containment. Most importantly, we should constantly remind ourselves that the needs for convenience and security are often at odds with one another. Maintain a healthy skepticism for proposed solutions that promise both.

About GDS Labs

Security Research & Development is a core focus and competitive advantage for GDS. The GDS Labs team has the following primary directives:

- Assessing cutting-edge technology stacks
- Improving delivery efficiency through custom tool development
- Finding & responsibly disclosing vulnerabilities in high value targets
- Assessing the impact to our clients of high risk, publicly disclosed vulnerabilities

The GDS Labs R&D team performs security research, with areas of current focus including mobile application security, embedded systems, and cryptography. GDS Labs is a value add service that our clients benefit from on virtually every engagement that we perform.

About Gotham Digital Science

Gotham Digital Science (GDS) is a specialist security consulting company focused on helping our clients find, fix, and prevent security bugs in mission critical network infrastructure, web-based software applications, mobile apps and embedded systems. GDS is also committed to contributing to the security and developer communities through sharing knowledge and resources such as blog posts, security tool releases, vulnerability disclosures, sponsoring and presenting at various industry conferences. For more information on GDS, please contact info@gdssecurity.com or visit http://www.gdssecurity.com.

GDS, Stroz Friedberg, and Aon

GDS was acquired by Stroz Friedberg in April 2016, the premier global cyber security firm with leading experts in digital forensics, incident response, security science, investigations, eDiscovery and due diligence. This provides GDS with unparalleled access to leading Threat Intelligence, up-to-date information on indicators of compromise, and an accurate real-time picture of the tactics, techniques, and procedures (TTPs) actively being used in the wild. This in turn allows GDS to provide the most relevant testing services to our clients as part of the overall strategic goal of cyber resilience.

Aon Risk Solutions, the global risk management business of Aon plc (NYSE: AON), completed its acquisition of all of Stroz Friedberg and GDS in November 2016. Aon is a leading global provider of risk management, insurance brokerage and reinsurance brokerage, and human resources solutions and outsourcing services. The combined resources of Aon, Stroz Friedberg, and GDS will provide clients a comprehensive suite of cyber security services to better prepare for the evolving cyber threat landscape and reduce their overall risk profile.

New York Office (Global Headquarters):	***Charlotte Office:***	***London Office:***
Gotham Digital Science LLC	Gotham Digital Science LLC	Gotham Digital Science Ltd
32 Avenue of the Americas – Fourth Floor	1000 NC Music Factory Blvd	161 Drury Lane – Fourth Floor
New York, NY 10013	Charlotte, NC 28206	London, WC2B 5PN
United States	United States	United Kingdom

References

[1] http://dl.acm.org/citation.cfm?id=1360099
[2] http://asleap.sourceforge.net/asleap-defcon.pdf
[3] http://theta44.org/karma/aawns.pdf
[4] http://www.willhackforsushi.com/presentations/PEAP_Shmoocon2008_Wright_Antoniewicz.pdf
[5] https://sensepost.com/blog/2015/improvements-in-rogue-ap-attacks-mana-1%2F2/
[6] https://tools.ietf.org/html/rfc3579
[7] https://tools.ietf.org/html/rfc4017
[8] https://tools.ietf.org/html/rfc5281
[9] http://www.willhackforsushi.com/?page_id=67
[10] https://tools.ietf.org/html/rfc5216
[11]https://4310b1a9-a-93739578-s-sites.googlegroups.com/a/riosec.com/home/articles/Open-Secure-Wireless/Open-Secure-Wireless.pdf
[12] https://www.blackhat.com/presentations/bh-dc-07/Arkin/Presentation/bh-dc-07-Arkin-ppt-up.pdf
[13] https://www.sans.org/reading-room/whitepapers/analyst/securing-personal-mobiledevice-next-gen-network-access-controls-35627
[14] http://crack.sh/bsideslv2017.pdf
[15] https://blogs.technet.microsoft.com/josebda/2010/12/01/the-basics-of-smb-signing-covering-both-smb1-and-smb2/
[16] https://blog.gdssecurity.com/labs/2013/2/5/resurrecting-wifitap.html
[17] http://sid.rstack.org/static/articles/w/i/f/Wifitap_README_202c.html
[18] https://www.aircrack-ng.org/doku.php?id=airtun-ng
[19] https://www.aircrack-ng.org/doku.php?id=tkiptun-ng
[20] http://www.ietf.org/rfc/rfc1001.txt
[21] http://www.rfc-editor.org/rfc/rfc1002.txt
[22] https://msdn.microsoft.com/en-us/library/dd240328.aspx
[23] https://www.trustwave.com/Resources/SpiderLabs-Blog/Introducing-Responder-1-0/
[24] https://www.cylance.com/redirect-to-smb
[25] https://technet.microsoft.com/en-us/library/dd283093(v=ws.10).aspx
[26] https://msdn.microsoft.com/en-us/library/dd759173(v=ws.11).aspx
[27] https://framebyframewifi.net/2017/01/29/use-lets-encrypt-certificates-with-freeradius/
[28] https://crack.sh/
[29] https://technet.microsoft.com/en-us/library/cc957983.aspx
[30] https://www.helpnetsecurity.com/2017/04/26/lure10-exploiting-wi-fi-sense/
[31] http://web.archive.org/web/20160203043946/https://www.cloudcracker.com/blog/2012/07/29/cracking-ms-chap-v2/