Wireless Exploitation and Mitigation Techniques

Prepared By
Gianfranco Di Santo

In
Partial Fulfillment of the requirements
For
Senior Design – CTC 492

Department of Computer Science
California State University, Dominguez Hills

Fall 2014

Committee Members/Approval

______________  ______________  _____________
Faculty Advisor  Signature         Date

______________  ______________  _____________
Committee Member Signature         Date

______________  ______________  _____________
Committee Member Signature         Date

Dr. Mohsen Beheshti

Department Chair  Signature         Date
## Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approval Sheet</td>
<td>1</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>2</td>
</tr>
<tr>
<td>Abstract</td>
<td>3</td>
</tr>
<tr>
<td>Introduction</td>
<td>4</td>
</tr>
<tr>
<td>Background 1.0</td>
<td>5</td>
</tr>
<tr>
<td>Overview of 802.1X 1.1</td>
<td>6</td>
</tr>
<tr>
<td>Operating Systems and Tools Used 1.2</td>
<td>10</td>
</tr>
<tr>
<td>Section 2.0: Wired Equivalent Privacy</td>
<td>13</td>
</tr>
<tr>
<td>Section 2.1: Exploiting local WEP encrypted Networks</td>
<td>15</td>
</tr>
<tr>
<td>Section 2.2: Mitigation</td>
<td>23</td>
</tr>
<tr>
<td>Section 3.0: WPA/2</td>
<td>26</td>
</tr>
<tr>
<td>Section 3.1: Exploiting WPA/2 Security Protocols</td>
<td>29</td>
</tr>
<tr>
<td>Section 3.3: Mitigation</td>
<td>32</td>
</tr>
<tr>
<td>Conclusion 4.0</td>
<td>35</td>
</tr>
<tr>
<td>References</td>
<td>36</td>
</tr>
<tr>
<td>Appendix</td>
<td>37</td>
</tr>
</tbody>
</table>
Abstract

In a fast paced environment that we reside in today, we need things to travel as quickly as possible. Having an “I need it NOW” attitude has led to many technological advancements. One of these great advancements is Wireless Technology in regards to internet, which constitutes Wireless Fidelity, or, Wi-Fi. The restraints of cables to communicate with one another seems to be a thing of the past. Businesses are more in-tune with using Wireless technology in the work place. Not to mention, Wi-Fi is the “preferred” choice of connecting to the internet amongst end users as well. The absence of extra wires, the restraint free environment, and the now up-to-par wired speeds make a wireless environment more beneficial than a wired one does in today’s world by allowing that extra mobility with still getting what you want and need, when you want and need it. The accessibility factor of using Wi-Fi does come with a toll. Free flowing information is broadcasted through the air, which makes it extremely easy for malicious play to take place. Since the data is already air-borne, all an attacker needs to do is capture that information and decrypt, ultimately allowing them to see all nodes on that same network which means security becomes a great need in businesses with implemented wireless infrastructures. This research paper will be touching base, no pun intended, with wireless security protocols, showing the ease of exploitation to each, and show how to mitigate said attacks to ultimately safeguard your digital assets by keeping a secure wireless infrastructure.
1.0 Introduction

Within my paper, I propose to show the differences, in regards to vulnerabilities, of all the current wireless security protocols. I plan to demonstrate how WEP (Wired Equivalent Privacy), WPA/2 (Wi-Fi Protected Access), WPS (Wi-Fi Protected Setup) can easily be broken into, making them vulnerable, some more than others. I will continue with demonstrating how some wireless security protocols are simply broken by design. By exploiting these wireless security protocols, I will come to a conclusion on how to mitigate these common wireless security vulnerabilities, essentially “patching” the “holes” in these wireless protocols. Wireless exploitation goes beyond the course of vulnerable security protocols, and I hope to show many of the more common Wi-Fi attacks that are most common in the wild, I will essentially also show how to mitigate these problems as well, and stay clear of being a target to malicious wireless activity. I will be using a range of hardware, mostly different types of Access Points and wireless adapter cards designed to allow for packet injection (Both Internal / External) and a wide range of tools scattered across different Linux Distributions that I will explain in further detail within this paper. I hope that my paper will bring a sense of awareness to how easy it is to sniff out information that is free flowing across the air, and also show how to keep yourself protected against malicious activity.
1.1 Brief Overview on Wireless Technologies and Protocols

Wireless connectivity is a growing demand in suburban lifestyle, and as well in the business aspect. Wireless connectivity has technically been around since 1971 when ALOHA.net provided the first public demonstration of a wireless packet data network. After that, it was a somewhat lost field until around the 80’s and 90’s when ALOHA.net’s method of sending packets of data through the air was being perfected. It wasn’t until the late 1990’s to the early 2000’s where Wi-Fi technology started to become more popular.

Businesses everywhere have started to realize actually how important connection mobility is in playing the role of expanding businesses, for example, what if “Joe” needed to present a power point presentation in a room with no computers? He has his laptop and he has a projector, but he is missing a key ingredient in his presentation, the internet. His power point presentation requires to stream off the internet, or use the company’s intranet to show some important financial statistics. Well, with the implementation of a Wireless Local Area Network (WLAN) within the company he can now be mobile within the building with his laptop while still being able to connect to his company’s intranet.

The way wireless works is very similar to the wired Ethernet protocol, it is actually simply an extension! A special router that supports wireless, called an Access Point (AP) ultimately grabs all these packets/frames coming through the wired connection and emits them in a radio frequency which is then captured by the receiving node, and vice versa.

There are different types of transmission frequencies, the network standard is 802.11 but it does come in 802.11a/b/g/n/ac. 802.11ac is the fastest and newest in today’s 802.11 standard, but not commonly used by businesses due to its lack of reliability. 802.11g on the other hand is a bit slower, but offers orthogonal frequency-diversion multiplexing (OFDM) which is a more
efficient coding architecture that reduces interference greatly by splitting that radio signal being sent out into many sub-signals before the signals reach the receiver.

The mobility aspect of Wi-Fi technology sounds pretty amazing, but because data is constantly being passed through the air it could be a great security flaw. There are many different kinds of wireless security protocols to safeguard air-borne data by encrypting it, such as Wired Equivalent Privacy (WEP) [802.11], which is actually the least secure, Wi-Fi Protected Access (WPA/2) [802.11i] which is more secure than WEP, and Wireless Protected Setup (WPS), which seemed to take a step backwards in the evolution of wireless security protocols. All of these could be easily tampered with, regardless of the security type. One may take more time to crack than the other, but nevertheless, they are all at risk. The evolution of these protocols can be seen in Figure 1 down below. One should never assume that their internal network is “safe” just because there is a password set to get in. There are always alternative methods in gaining access. The main flaw of this is that a frequency is always being transmitted no matter what, even if you choose not to broadcast the Service Set Identification (SSID), or also known as the name, of the Access Point (AP) wireless data is still being sent through the airwaves. Anyone could just get near the AP and start grabbing “encrypted” packets flowing through the air and analyzing patterns and such within those packets to essentially decrypt it. Once connected within, the attacker could cause great harm due to the fact that the AP is most likely hard wired throughout the whole business. Although cracking wireless networks may seem easy to one who has knowledge on how to, encryption schemes are becoming more elaborate therefore reducing the amounts of security threats caused due to the fact of wireless transmission.
The figure above, figure 1, shows a quick evolution of the Wireless Security Protocols. As we see, the first of its time, and most exploitable Wireless Security Protocol, as mentioned before, is WEP. The next being 802.1X, in which I will discuss in further detail shortly, as this was, and still is, used as a mitigation technique to the lack of security provided by WEP, and still works for added security measures in an enterprise environment. Shortly after, WPA and WPA-2 were introduced, which is what we currently use today, as it is the most common of all. Wi-Fi Protected Setup (WPS) is not mentioned here as it is technically not an 802.11 Standard, but it is still a wireless security protocol nonetheless. And an exploitable one at that, as will be discussed in the later chapters of this paper as it plays a major role in exploiting WPA/2 networks that happen to have WPS enabled as well.
1.2 Brief Overview on 802.1X / RADIUS as a Wireless Security Protocol

An IEEE Standard known as 802.1X, also known as WPA/2 Enterprise, is definitely worth mentioning, which was developed in response to the lack of security WEP provided due to its flawed stream cipher. This was implemented in business environments which allowed authentication through software on the client side, also known as the supplicant, instead of authentication straight through the Access Point. A RADIUS (Remote Authentication Dial In User Service) server is implemented, which manages the users in a centralized Authentication aspect. The user would authenticate with local software that would talk back to the AP via the Point-to-Point Protocol (PPP), which, in turn, would cause the AP to talk to the RADIUS server by passing along the request in an encrypted MD5 manner to obfuscate any credentials, and essentially grant or deny access to the client depending on the credentials inputted. With the input of authorized credentials, the RADIUS server would then grant the client access to the network by allowing a successful connection to be made to the AP all by using a version of the Extensible Authentication Protocol (EAP), which also includes, but not limited to Challenge-Handshake Authentication Protocol (CHAP) which are authentication schemas in themselves, they work as being the “carrier” of encapsulated messages from the client to the RADIUS server thus cutting out the need for direct AP authentication such as WEP. It’s easy to think of in the following fashion, a client wants to connect to an 802.1X network with specific credentials. Therefore they use their credentials to connect via PPP to the AP using EAP to carry the keying messages, or the credentials. The EAP Authenticator, the AP, makes sure the incoming connection request is being transferred via EAP, which in turn is relayed off to the RADIUS server, which then grants access or denies access according to the credentials being used. The Access-Accept or Reject response is sent back to the AP, which is relayed back to the client, or
supplicant, and if given an “Accept” response, will be allowed to the internal network, with a supplied, unique, authentication key that can now connect to the AP via 802.1X. It’s easy to think of EAP being the delivery carrier of information to the authentication server to safeguard information within route. This cuts out the need for WEP and vulnerable keys. Since connecting to the AP via 802.1X is unique to each client, only after authenticating via the RADIUS server, it leaves little to no vulnerabilities in breaking into the AP, since the AP only acts as a “middle man” in the operation. The way 802.1X “works” by using an Extensible Authentication Protocol (EAP) and the implementation of the RADIUS server is represented in Figure 1 shown below.

802.1X is one of the most secure wireless security protocols, and should be used as a mitigation technique when possible. The accessibility factor when it comes to 802.1X goes way down, as accessibility tends to go down with increased security, therefor it is not widely used in homes and small businesses. Not to mention, the need for additional hardware to be running.
simply to pull 802.1X off, as it needs to contact back to a RADIUS server for authentication to take place. It is used in larger organizations that have more to lose, such as the Dominguez Hills campus. While it is true that it is by far the most secure wireless security protocol, it can still be attacked. It is not perfect, but it is not flawed by design as other Wireless Security Protocols are. I will be discussing the exploitation of more commonly used Wireless Security Protocols (WSP) such as WPA/2, and others mentioned earlier. When mitigation explanations surface, WPA/2 Enterprise (802.1X) will be referenced which the implementation of this protocol is a common mitigation practice. This is why a brief background on 802.1X plays a small, yet important role and is worth mentioning in its own category, and not lumped into the WPA/2 wireless security protocol, even though it technically constitutes as a branch from WPA/2. Let’s begin by talking a little about WEP.

1.3 Brief Overview on Important Tools and Operating Systems

There will be a series of tools discussed in the following chapters of this paper in which are a crucial part of exploiting the wanted wireless security protocols, and to learn how they work to be able to mitigate those attacks. I will briefly discuss some of the tools that will be commonly used. Keep in mind that this is just a quick introduction / overview of the tools to be used. I will be covering the tools more in depth, and exactly how to use these tools, by fine tuning them, in concert with the wireless security protocol it is being used on. The most widely used tool, when it comes to wireless exploitation, is the aircrack suite. This extensive suite of tools is all you will ever need to pentest wireless networks. This suite of tools contain everything you’ll need from cracking WEP, to cracking WPA/2. This suite of tools is considered to be the gold standard of wireless exploitation tools available. The task of using the aircrack suite has become so automated that many have made wireless exploitation “point and click” GUIs based
on aircrack. It allows people to “visualize” what they are cracking, instead of just staying in the command line. One of these great GUI based representations of aircrack is feeding bottle. This GUI tends to put the difficulty of cracking WEP down into “Easy Mode.” It’s a nice little point and click your way through representation that allows for fine tuning of some aircrack features, this tool is mainly for the use of WEP, but could be used for WPA/2 as well, but isn’t as fine-tuned for WPA/2 as it is for WEP.

Another great GUI based tool derived from the aircrack suite is minidwep. This tool is used as an auditing tool, to audit wireless networks, so unlike feeding bottle where you can fine tune your attacks, this tool is fully automated. It goes through and tries every single attack possible against the chosen AP until something works and then reports it back to you. You could say it is the Swiss army knife of wireless tools. It is a tool that incorporates all majorly important used wireless exploitation tools into one. It may take a little longer to break into the wireless network with this tool since it can’t really be fine-tuned, and uses a “shotgun” approach, but it doesn’t get easier than “click and go.”

Another great tool that automates the aircrack suite is called wifite. Wifite, unlike the other mentioned tools, is not GUI based, it is command line based, but it is an automated command line based tool. In other words, just like minidwep automates the entire wireless network auditing process, wifite does the same, just not in a GUI. Instead of it being “click and go” you could say it is more of an “execute and go” tool.

Please note that during my process of explaining exploitation processes, I could be using any of the mentioned tools above, but at the end of the day it is just a representation of the real tool being used, which is the aircrack suite. All these tools are just representing the same aircrack
tool, they are not different tools, just a different way to view and manipulate the one tool, being aircrack.

In the later chapters, I will discuss aircrack fine tuning as to what situation you may be in according to the wireless network you are attempting to exploit, or “audit.” While the aircrack suite handles most WPA/2 and WEP related attacks, as well as some client based attacks, it falls short in exploiting and penetrating the WPS wireless security protocol. Tools to use against WPS are a set of two specific tools, one named wash, and the other named reaver. These are very basic tools that can be extremely fine-tuned depending on the WPS network situation you may find yourself in. Wash will simply scans the air for wireless networks around you that will have WPS enabled. Once you find which target network of yours has WPS enabled, you input that information into reaver to begin the WPS attack. This will be covered more in depth in the later chapters discussing WPS exploitation. Wash and Reaver are also implemented in the auditing tool mentioned above, that was coined to be the Swiss army knife of wireless exploitation tools, minidwep.

All these tools mentioned above play a crucial part in the exploitation of wireless networks. These tools are all available natively in penetration testing specific Linux Distributions, such as Kali Linux. A fantastic Operating System that is used for the sole purpose of pentesting wireless networks is called Xiaopan, which is based off of tiny core Linux. Xiaopan also has all the above mentioned tools, including the GUI representations of aircrack natively installed. It is a very lightweight operating system, standing at less than one hundred megabytes, that transforms this operating system into the perfect tool to fire up quickly to test wireless networks on the go. Another fantastic Linux Distribution that focuses on wireless exploitation that is worthy of mentioning is WifiSlax. This operating system has all the
mentioned tools above, and some. It has a huge library of very specific tools, including many different tools that just represent aircrack in different manners. This operating system is not as lightweight as Xiaopan is, and will not be used in this research, but nonetheless, is still worth mentioning.

2.0 Wired Equivalent Privacy (WEP)

We begin with an overview on exactly how the Wired Equivalent Privacy protocol functions. This will come in handy in understanding exactly how it is that we can break such wireless security protocol. This protocol is the most vulnerable, and was decommissioned in 2004 because of its lack of security. First we must understand the encryption details to this protocol. To “safeguard” the connection, WEP is encrypted by using an RC4 key. This Stream cipher is created by taking a 24 bit initialization vector (IV) and concatenates that IV with the password key that you would use to connect to the Access Point. To clear a little confusion, an IV, simpler terms, is equivalent to choosing an arbitrary set of numbers to randomly generate a sum that is equivalent to, and will always be equivalent to, 24 bits (in this case at least when dealing with WEP.) There are many different types of “WEP” that may have seemed to better protect information by elongating the password field. The standard is WEP 64, the next is WEP 128, then WEP 256. Each number represents the bit value of that WEP connection. No matter the WEP type, 24 bits are always reserved for the encryption process to take place. For example, a 64 bit WEP encryption, which is the most common WEP used, uses 10 hexadecimal characters, which would equate to 40 bits being used up for password length, as 1 hexadecimal character is represented in 4 bits. The remaining 24 bits is the IV, and with the key it equates to the stream
cipher that would “safeguard” the wireless connection, so a user would only see the ciphered text being passed through the air. WEP 128 and WEP 256 used the same methods, reserving 24 bits for the encryption schema, and would accommodate longer passwords to make it “tougher” for a hacker to penetrate through the network. With this said, since the WEP password itself was used to encode the stream in the RC4 cipher, given that the cipher was protected in 24 bits, it made it very trivial to actually find the password that was being “safeguarded” in the aerial traffic. The famous “birthday problem” comes into play, where if you were to randomly select 500 people, there will be, 100% of the time, people in the group with matching birthdays. This comes into play when deciphering the 24 bit RC4 cipher. While at first the stream may seem completely random at first, if you listen and capture enough packets being passed through the AP, at 24 bits, after about 5000 captured packets containing that IV, the same “pattern” of randomly generated IVs will begin to be produced. After so, one can then listen in to the patterns of the repeated IVs that arise in the packet capture to make a distinction on what bit is equal to what hexadecimal representation, as the bits that are used in the same pattern more often will lead to an educated guess that those bits are the actual bits used in the representation of the passphrase.

![figure 3, wep cipher text simplified](Provided by Wikipictures)
Figure 3 above shows a simplistic view on exactly what we discussed earlier on how the RC4 stream cipher collaborates with the WEP encryption security protocol.

2.1 Exploiting Local WEP Encrypted Networks

This is where it starts to get fun, now that we have an understanding on how the RC4 stream cipher in WEP functions, we get to look at different methods that have been developed over time to exploit them. The types of attacks on WEP Networks was a steady incline towards accommodating more and more efficient methods of being able to acquire the WEP Key, the first, however, was the FMS Attack. The FMS Attack, named after the creators, Fluhrer, Mantin, Shamir, brought the idea of using a statistical attack towards the deciphering of WEP passwords. They made the correlation that the RC4 Stream Cipher would “leak” out the key as more packets, or IVs, were captured. Using this known information, the FMS Attack takes the first approach of using statistics in determining what bit represented what hexadecimal representation according to the amount of the same bits being broadcasted were. This technique to crack WEP was about fifty percent effective at the time, and was used to see correlations between packets sniffed through the air on an Access Point that was in current use, and since it only sniffed out for statistical correlations between all cipher text and seed, it required many packets to sniff out correlations, to fully crack a WEP password it took about 6,000,000 packets.

The second attack to come along, that made it even more efficient to crack WEP passwords was the PTW Method, also named after its creators, Pychkine, Tews, and Weinmann. They used the statistical information discovered in the FMS attack method and found further correlations that could be used against ARP request / reply packets. This method is also known
as the “ARP Replay attack.” Instead of inactively sitting by, reading packets across the network, you could see the connected clients on the victim Access Point, and create ARP requests based on their mac addresses, causing a flood of ARP reply’s to be let out by the Access Point. Because there would essentially be more data being emitted by the Access Point, it made it an easy method to speed up the process of capturing packets, or IVs, to use statistical techniques, as described earlier, to decrypt the cipher. The information would only be ARP replies, but it would nonetheless be encrypted ARP replies, which meant we could see those encrypted ARP Replies and find the correlation between the seed, which is the IV + the passphrase, and weed out the common data that resembled to be ARP replies in an encrypted method, which in turn would be speeding up the process of capturing IVs by essentially creating packets from clients on the network, to getting a better statistical percentage of actually finding the encrypted password since there was only one type of request being dealt with, so we could find the correlations between what seemed to be a ciphered ARP Reply, and weed that out to further find the correlation between the rest of the stream cipher with the 24 bit RC4 key to match up the hexadecimal representation, to get the final WEP passphrase. Since we would only be trying to find the correlation with ARP replies, we could figure out what was garbage ARP broadcasts and simply hone in on the RC4 Stream cipher, which made the process much quicker in terms of being able to create data to analyze, and use much less of it as well, especially compared to 6,000,000 IVs. This ARP replay attack is still commonly used as a method of deciphering the key in an environment where there are clients connected to the Access Point.

This leads us to the next WEP Exploitation method which are the Korek attacks. Korek is an unknown individual who found exploits in the WEP protocol itself and posted his discoveries in a netstumbler forum circa 2005. Many of his attacks were experimental, which meant they...
weren’t fully reliable, but his one discovery, the Chopchop attack that he founded stuck as one of the staples in becoming another methodology of breaking WEP. The way the Chopchop attack works is unlike the rest. The other methodologies of breaking WEP focus on retrieving the passphrase due to the correlation of passphrase and IV as the seed within the RC4 algorithm, whereas the Chopchop attack focuses on the CRC32 checksum used for integrity. CRC32 simply provides integrity within a packet being transmitted where the last bit defines the integrity of the packet being sent where 1 passes the checksum, and 0 fails the checksum. Simply by removing the last byte of a packet and decrypting it, we can see what constitutes as a passed or failed encrypted checksum value. After that byte is removed from the packet, we can decrypt the packet itself according to size and relating it to specific types of packets and fixed sizes of said packets by using trial and error of re-injecting that specific packet with a “failed” checksum, so the AP replies with the same packet again, and again, letting us decrypt the packet by finding correlations to the stream. Once we have that packet decrypted, we can then use that packet to inject it back to the access point, causing a flood of data to be emitted, therefore going about of finding the passphrase in the RC4 algorithm. The Chopchop attack is incredibly neat as it has a nature to “decrypt” the packet by telling the AP that it has failed a checksum, and to reply with the same packet, this means that we can actually see the data, in specific to the packet we capture, being communicated across the air in plaintext without even being connected to the AP or knowing the passphrase for the WEP encryption. This Chopchop attack itself does not recover the passphrase for the AP, but helps us decrypt packets so we can use it to replay it and create more traffic within the airspace, ultimately allowing for more IVs to be captured so we can retrieve the passphrase. The Chopchop attack is simply another method of attacking WEP
encrypted networks, but it is one of the slowest methods out there, but is so reliable that is definitely worth mentioning.

Let us continue on with exploration of available WEP attacks with the Fragmentation Attack, which was released in 2005 by Bittau et al. The way fragmentation works is in the method of getting a hold of smaller sized packets in hope that the AP will help decode the packet for them into plaintext, which is very similar to the chopchop attack. This methodology works by sniffing out a packet, once that is sniffed out, the first 8 bytes of the cipher text can be revealed into plaintext as the first 8 bytes in any 802.11 network will always yield the same header. Once the header is decrypted into plaintext, we can use the corresponding cipher text to correlate similarities between the plaintext and cipher text within those 8 bytes, which equals to a specific portion of the produced IV. We can then use those 8 bytes of the key stream to produce 64 bytes of plaintext within 16 fragments and broadcast that packet back within those 16 fragments, once the AP receives those frames that we encrypted each header with the known 8 bytes of cipher text, or key stream, the AP will combine the 16 fragments into a full 68 byte packet, re-encode it, and echo it back to us. Taking out the 4 bytes for the CRC32 checksum, we know exactly what that cipher text says, therefor we can make that correlation with how the key stream is played out. To sum the fragmentation attack up, we are essentially looking for the PRGA, or Pseudorandom Generation Algorithm. This PRGA is what is used to create the cipher text, or the RC4 Key stream in this case, PRGA XOR Plaintext is equal to the cipher text. So this becomes a simple equation, if we are able to get the cipher text, which is the key stream available, and know what the contents of it is in plaintext, then we can reverse the XOR process to get the PRGA key that was used in conjunction with the plaintext to create the stream cipher. Since we know that the first 8 bytes of the header in 802.11 packets are always the same, we then know a small
portion of the plaintext equivalent to the cipher text within the parameters of that small section. We can then use those 2 variables to calculate the PRGA key for those small 8 bytes.

Unfortunately, it would be impossible to use that small calculated PRGA to encode a full plaintext packet to send back to the AP, so instead, we fragment that into 16 frames. Each frame with a bit of packed information suitable to be encrypted with that PRGA that we figured out.

Once the AP receives those 16 frames, it will combine all those frames into a full packet, and send it back to us. We then know exactly what the plaintext is, so we go back to the equation of knowing the plaintext and cipher text, now only needing to figure out the PRGA, but on a much bigger scale. Once we can find out a full 1500 byte PRGA from using this method, we can create our own forged packets, and send them into the access point this way. With this information that we have with the full 1500 byte PRGA, or RC4 Stream Cipher, we can manipulate our own packets, forging them, injecting them back into the AP, there for creating traffic that will essentially lead us to be able to capture the IVs that will lead us to sniff out the password.

The last, and most useful to date, exploit that we will be looking at is the interactive packet replay attack, also known as the interactive 0841 attack, defined as 0841 due to the frame control field being modified as “0841” to make the packet being sent look like it is coming from a wireless client on the network straight to the access point. This is extremely useful, because unlike other attacks, where you are initially capturing some sort of traffic being created by clients on the network for your own advancement, you can create packets by your own means to make your advancements within the network, without a single client ever needing to be connected to it at all. With the WEP wireless security protocol, before you actually connect to the AP, you must first “associate” with the AP so the AP knows that you are specifically speaking with it and not some other AP that may be nearby.
Think of the authentication process in a way that a repairman knocks on your door, so you look outside the window to see who it is, but you are unsure exactly who he may be, therefore you ask to see his badge. The repairman shows you their badge, so you call up corporate headquarters in which the repairman works for, the headquarters associates that repairman as working for that company, so then you authenticate him and let him in to do whatever it may be. The same thing takes place with the WEP protocol, each party needs some sort of communication to authenticate each other before proceeding with business, or letting the client connect within the network, and not just to the AP. Now, going back to that analogy, what if the repair man was wearing a mask and so you could not see his face at all? What if he, after being in your home, went outside to go to his truck, and came back into the house? How would you know he was the same repairman as before? This failed authentication comes into play when dealing with mac filtering, which will be discussed in the later sections. But the Authentication process works in this similar fashion on WEP, requiring a 4 way handshake.

When the client connects to the AP, the client “knocks on the door” with an Authentication request. The AP, would want the client to prove himself knowing the WEP key, so a plaintext challenge response is sent to the client, in which the AP wants to receive back making sure it was encrypted with the correct WEP key when sent back, if send back successfully, then the AP sends an Authentication (success) message back to the client, letting the device into the network. So, to pull of an interactive replay attack, what happens is the first half of this authentication process, which is called fake authentication. With this, you, the client, associate yourself with the access point, so the access point knows that you are talking to it by requesting a challenge. In turn, the AP sends you back a challenge request, but instead of sending him back the response, you don’t send anything back. You essentially “stall” hindering at
attempting to “locate” the WEP key to encrypt the response with. Since you haven’t sent anything back to the AP, the AP stays in a state of waiting for that response to know who you really are, therefore you stay associated with the AP during that downtime. Once you stay in a state of pending by being associated with the AP, you are now technically still connected to the AP as a client, even though you aren’t authenticated fully yet, you are still associated with the AP and are recognized as being a client talking to the AP. Now this is where the fun starts, once you’re associated with the AP, you have your own specific MAC address associated with the AP, you then craft a “natural” packet that essentially says, you (Your MAC address) is sending an ARP request to the network. Since your MAC address is associated with the AP, the AP will parse that request freely, since you are associated with that AP, it will think you are part of the network. This still does not reveal any cipher text, or IVs to capture to be able to break the WEP password though, since we don’t know the WEP password we can’t encrypt that information, so what we do, is get the AP to respond to our ARP request by crafting the destination of the frame to be FF:FF:FF:FF:FF:FF since that is the known broadcast destination associated with any router, the AP will then repeat that ARP request, which in turn, we can see the ciphered packets going through the air. We are then able to capture these packets by means of injection, where we inject these ARP request packets, and see what cipher text comes back out, then using previously discussed statistical techniques, we can sniff out the WEP password by decoding the RC4 stream by capturing the IVs we discussed previously.

Please note that all discussed methodologies of breaking into the WEP protocol are all part of the aircrack-ng suite. The aircrack suite is an intensive tool, that when used in command line can be fine-tuned to your liking. For aesthetical purposes, I will exemplify most of the aircrack suite within a GUI based representation, the following picture shows a tool with the
name “feeding bottle” that simply turned the more popular options of the aircrack suite into a GUI representation. This does not mean it is a completely different tool, the tool “feeding bottle” is solely a GUI representation of the tool that does matter, which is the aircrack suite.

The GUI Representation of Aircrack-ng above, Feeding Bottle, Figure 4, shows a visual representation of all the well-known attack methodologies available towards WEP cracking. All the methodologies that we discussed earlier are visually shown, from Fake Authenticating to the Fragmentation attack, and the most common, the Interactive 0841 attack. Figure 5, shown down below, shows the successful end result of Aircrack decrypting the password of a 128 bit WEP network successfully after using a mere 56,752 captured IVs which took no more than three minutes to acquire that sum of IVs.
This section dedicated to WEP mitigation are some of the ways that people attempted to mitigate the problem of WEP being broken by design by adding “layers” of security. The only problem with this, was that, as I stated earlier, WEP was broken by design. The only way to mitigate attacks against WEP networks was to pretty much re-invent the wheel. In the meanwhile, people started implement some “restrictive” features to only allow “authorized” personnel into the networks. One of these attempts was a simple MAC Filtering technique, while WEPAttack was thwarted by the use of WEP 128 and above, people still knew that hackers would get in, so they thought they would add a layer of security by only allowing MAC authorized clients to join to the network. This is a pain to actually manage, and more trouble than
what it is really worth since it would take an attacker all but a few extra seconds to change their MAC address to impersonate some client who is already connected, and has authority to use the network. Like shown in Figure 5 above, you can clearly see the clients connected to the AP, so once the attacker has the password, they can easily type in “macchanger –m “client MAC address here” “interface here” and that’s all. The attacker can now freely impersonate the other person on the network and connect to it freely, MAC filtering proved to do much of nothing at all.

The next failed Mitigation technique was “Hiding SSIDs” We’ve all needed to connect to a Hidden SSID in our day, it can be somewhat inconvenient, typing the SSID name and then the password. The problem with this, is that SSIDs were not constructed to be hidden, therefore, one could fire up Wireshark, monitor the air around them, and essentially see what the SSID name is via a client connecting to it who is broadcasting probes. The Picture below, Figure 6, shows just that.

![Figure 6 Revealing Hidden SSIDs via Wireshark, Provided by CyberSecurityLabs](image_url)
Soon after, followed proprietary fixes that required proprietary hardware, which made it extremely cumbersome to implement, especially when both the AP, and the client would need this specialized hardware, this was WEP+. Designed by Agere Systems, it attempted to rule out “weak” IVs that would essentially leak out the WEP passphrase, but did not take off at all as it was proprietary hardware that was cumbersome to work with. The next, and possibly closest, mitigation attempt there was while still staying in the realm of WEP was the creation of WEP2. WEP2 was designed by increasing the initialization vector size from 24 bits, which was very prone to collisions, to 128 bits to try to rule out statistical attacks and brute forcing. The brute forcing was correct, but with the advancement of hardware, and the weak algorithm of WEP, since it was still prone to replay attacks, which WEP2 would not stop it, the advancement of WEP2 was dropped altogether.

I’ve actually stumbled upon the best method to mitigate attacks on WEP, while it is still far from bullet proof, I have seem to find that “Dynamic WEP” is the best implementation to work with in securing the WEP protocol. Unfortunately, this is simply the 802.1X standard that is implemented in the form of WEP, where it uses a RADIUS server to dish out a “temporary” WEP password to let the user connect through with, making the WEP password never the same. Unfortunately, just like WPA2 Enterprise, or 802.1X, it is not a home user friendly approach to securing WEP, and is only considered to be a fix in the corporate world. With all these points brought up, I’ve found that the best way to mitigate away from all the problems of WEP is to simply not use it. WEP was deprecated in 2004 due to its lack of security, and WPA took its place. While WPA still uses the same stream cipher as WEP, and is still vulnerable in certain ways, as I will cover in the later chapters, it is still a step up from WEP to secure your system. So to summarize my findings in mitigating WEP vulnerabilities, I have come to the conclusion that
the protocol is so extremely broken by design that the only way to mitigate WEP exploits is to simply not use WEP at all. This has become much easier to do since it’s deprecation in 2004. Newer Access Points don’t even come with an option for WEP anymore, but if you are the few who are still using WEP as a security protocol, I advise you to switch over to WPA, or WPA 2 if your hardware can handle AES properly.

3.0 WPA/2

Now that we have covered the most extensive part of the paper, which was WEP and its vulnerabilities, we can begin by taking a look at WPA / WPA 2 which shouldn’t be as broken as WEP was. WPA stands for “Wi-Fi Protected Access.” First we will be discussing WPA, which is simply the predecessor to WPA 2. They both work in the same fashion when it comes to allowing users to authenticate, but WPA uses a different encryption schema than WPA 2, which works to be compatible with older hardware therefor it is still more exploitable like WEP was in its day.

WPA to begin with, as previously mentioned, was the remedy to WEP. It was the “fix” to bring about a more secure protocol that didn’t have as many exploits as WEP did. Unfortunately, WPA was only a temporary solution, it was something set into play to hold off any attacks for the time being while they worked out the kinks, which is what WPA 2 essentially is, solely WPA with the kinks worked out of it. For WPA, the method of user authentication was a lot different than WEP was. Instead of using the RC4 stream cipher to concatenate the IV with the passphrase in a 24 bit fixed method, they increased that fixed 24 bit IV to a randomly generated 128 bit, which also did not concatenate the passphrase to it. This meant that there was absolutely no way for the passphrase to be leaked out with weak IVs being statistically analyzed to find collisions,
since the 128bit string was always random. This new method of encryption was to be known as TKIP, which stands for Temporal Key Integrity Protocol, where the Pre-Shared Key, or PSK, was not ever concatenated in a way that could lead to exposure, but instead randomly encrypted with a dynamic 128bit string, not a fixed 128bit string. This method of encryption virtually stops any statistical analysis attacks to be performed on the free-flowing data to find the WPA passphrase, this doesn’t mean that there are no other ways to compromise the protocol as a whole, or crack the password. The way WPA works, when authenticating a user, is that the WPA passphrase is kept solely in the hands of the AP, when a client requests authentication, the AP sends A Nonce to the client, in term the client “encrypts” that Nonce with their passphrase, along with other commonly known data such as SSID name, and length, and then sends another Nonce to the AP. The AP makes sure that the Nonce sent back was encrypted accordingly, and if so, acknowledgment would be sent back, to the client authenticating them into the network. This is called the “4 way handshake” in more detail is a small picture showing how the handshake occurs.

**PSK: 4-Way Handshake**

![Figure 7 PSK 4-Way Handshake Provided by Wifidot](image)
The small picture in Figure 7 depicts the process of a user authenticating themselves with a common WPA network, whether it be WPA or WPA 2, the process remains the same, the only thing that is different is the encryption schema, where WPA uses TKIP and WPA2 learned to use AES, which is not as flawed as WPA is. We will now talk about some WPA issues.

The last point to talk about in the WPA world is the announcement of WPS. Once WPS was introduced, it was almost as if we took a step backwards in wireless security. WPS allows convenience for devices to connect to a network, by simply allowing a device to “click a button” and connect to an AP that is WPS enabled. This allows for us to essentially save time in actually configuring the connection settings to the client for the AP, and just allows us to “click and go” so to speak. The way an AP allows the client to connect to the AP is simply through a “pin” method. The client would have to know the secret pin that the AP has implemented, and can use that pin instead of typing in all kinds of information such as the passphrase. This pin is constructed of being eight numbers, where one digit at the end is simply the checksum, then they are halved in the authentication process. Meaning, that it will confirm the first four numbers, and then confirm the last 3 numbers, so by just trying 11,000 combinations total you can recover the WPS key, allowing you to recover the WPA PSK. There are multiple methods of a user connecting via WPS, the most common is the pin method mentioned above. The next is the push button method where the user pushes a button on the device they want connected to the network, does the same on the AP itself and then they associate with each other, this method still requires the AP to accept WPS pins even if it is not directly dishing them out to the clients. The last method is the NFC connection method, where the client has to get into close proximity to the AP with their device, which needs to accept NFC proximity connections as well, where the two devices share information to one another via NFC. This last method, also requires the AP to
allow WPS pins to be accepted at all time, since no matter the way you connect to WPS, a pin will always be used to authenticate the client to the network.

3.1 WPA/2 Exploits

So first thing is first, we will discuss the WPA exploits first, since some WEP vulnerabilities seemed to carry over to the realm of WPA. This is due to the fact that TKIP essentially is an improved IV going through the same RC4 stream cipher. We will discuss some exploits that don’t actually deal with the process of exploiting the handshake to gather the passphrase, but some ways that we are capable of potentially compromising the network via other attack vectors. One of which is to exploit the Temporal Key Integrity Protocol, TKIP, within the data streams themselves. Since the TKIP uses the same RC4 stream cipher, just like in WEP, where we could find certain packets and decrypt them, solely to replay them into the network, we can essentially do the same thing here. There is a tool apart of the Aircrack suite called tkiptun-ng which does exactly this to exploit the TKIP through the use of reversing the RC4 stream cipher. Because the stream’s IVs no longer contain the passphrase, we cannot decrypt the packet and use statistical correlation to find said password, but instead, we can still decrypt the packets individually. With this said, this allows us to edit the decrypted packet in a “malicious” manner, and then replaying them back into the network. The problem with WPA is that the AP still needs us to associate with it in some manner, so we may still Fake associate with the AP, which in turn, leads us to be able to replay custom forged packets that we originally decrypted back into the network with arbitrary commands allowing us to do many malicious things. The biggest attack when it comes to this, is to decrypt a sole ARP packet, and modify it to
reroute the traffic from an innocent “victim” PC to the Attacker PC, which is essentially poisoning ARP without ever actually being connected to the AP at all. This is the main and sole reason why WPA2 was implemented, to mitigate away from this fault that WPA was stuck with by using the TKIP that relied on the RC4 stream cipher.

This TKIP exploit soon lead to WPA 2 to use CCMP, which stands for “Counter Cipher Mode with block chaining message authentication code Protocol.” Which is an AES-based encryption schema that is currently used today which makes it much stronger than its TKIP counterpart. Current WPA2 setups allow for the use of CCMP and TKIP to work side by side to handle incoming connections from older clients that may be using TKIP, but still encrypts all the data using AES CCMP encryption unless otherwise stated. The use of CCMP allowed to stop this TKIP exploit of happening, so hackers began to look elsewhere for holes, which we now have a great deal of understanding on exploiting the 4-way handshake that occurs when authenticating a user into the network. One method, which works in a way that will allow for us to manipulate the network without being connected to it, is called “Hole196.” As we know, the AP houses a Pre-Shared key in which the client “knows” to be able to connect to the network, but to transfer that knowledge between client and AP, with that PSK, first, a PTK, or Pairwise-Transient Key is formed to allow the client and AP to talk while being able to protect unicast traffic. A GTK, or a Group Temporal Key, is constructed, as seen in Figure 7 above in step 3 of the Handshake to allow further communication and to protect broadcast data sent to multiple clients in a network. The PTK has the feature where it can detect address spoofing and data forgery, but unfortunately, GTK does not have this feature as mandated in page 196 of the IEEE 802.11 Standard, thus creating “Hole196.” If the attacker is already connected to the network, they can simply sniff out the GTK once a user connects, and use that GTK to forge data and re-
send it back to the client directly without even sending it to the Access Point first ultimately allowing the attacker to perform MITM attacks on a client without even talking to the AP beforehand.

Probably the most common WPA/2 Exploitation out there is simply capturing and deconstructing the 4-way handshake. Basically, this entails it to be a brute force attack, or a dictionary attack. In this situation, the network itself is only as safe as long as the password is. Doing this is the most “common” way to hack WPA and WPA2 personal networks as it will reveal the password to the network itself. If we go back to Figure 7 we can discuss a bit more in depth of the 4-way handshake. Within this authentication technique, each party, the client and AP have a “PMK” or Pairwise Master Key which is simply the WPA passphrase passed through Sha-1 encryption. The AP sends over an authenticator nonce to the client, where the client then creates a PTK, or Pairwise-transient key, that they use to concatenate the PMK along with other information like the authenticator nonce to send back to the AP. The AP now has a sNonce, or supplicatory nonce. The AP goes ahead and derives its own PTK from its PMK to check and make sure that the sNonce that was derived from the PTK created by the client matches, if so then the AP sends through a GTK, Group Temporal Key for encrypted broadcast data and then data can be encrypted, and decrypted in both ways. So, as a hacker, “all” we would need to do is sniff out this handshake happening with the client and Access Point. We may do this by temporarily booting the user off by sending them a de-authenticate packet, sniff out the sNonce sent by the client, break it down to find the PTK, break down the PTK further to find the PMK which is just the SHA-1 encrypted PSK. After we acquire this PMK, we can simply use brute force, or a dictionary attack to crack that SHA-1 encrypted password.
The final exploit against WPA which is possibly runner up most common to the prior just talked about, is the WPS attack. This is a pretty simple hack that can be done to any WPA/2 protected AP that accepts WPS by default. When WPS was first announced, the pin was a hard coded pin that could not be changed, this evolved to being a dynamic pin that could be easily changed at the Administrator’s convenience. Either way, if WPS is enabled on an AP, then it must harbor a pin that can authenticate the client attempting to connect to the network with said pin. As spoken about before, since the WPS pin can be brute forced in about 11,000 tries, it can be done in a matter of half a day, depending on the speed of attempts of course. A quick program called “wash” allows a user to scan the air for access points that have WPS enabled, once the access point is found, one can use a quick program called “reaver” to input the target, or “victim” and fire away pin attempts, essentially brute forcing the pin until the correct pin is found, which then allows you to automatically recover the PSK as an authenticated pin ultimately tells the access point to send over the correct PSK to that user who entered in the valid pin.

### 3.2 WPA/2 Mitigation Techniques

WPA2 is the most secure wireless security protocol that we currently have in existence, using WPA2 should be a mitigation technique in itself for users still harboring in WEP or WPA land. Possibly the only way to mitigate the attack of decrypting TKIP packets, and forging your own from WPA is to not use WPA, and use WPA2 instead. This is to the sole fact that TKIP still uses the RC4 stream cipher, where we saw that in WEP, was extremely easy to reverse XOR to decrypt. Packets forged from decrypted TKIP streams are very niche, such as being those allowing for MITM attacks, by ARP spoofing, or DNS poisoning. If one cannot afford to upgrade their access point from WPA to a WPA2 capable one, then they should statically assign their ARP tables, and statically assign their DNS / IP address (only if known.)
As for Hole196, since it is essentially the “same” sort of exploit where forged packets are sent to individuals, except within the realms of WPA2, one should look into WIPS (Wireless Intrusion Prevention System) for sudden changes in ARP tables, or DNS cache. Many modern IDS/IDPS will seek and prevent MITM complications that may arise, such as ARP Poisoning/Spoofing, DNS Spoofing, or hijacking of any kind, but since this exploit works on a client to client basis these attacks do not go through the main AP or the IDS connected to it. So a WIPS will monitor the air traffic from the AP to make sure no such attacks are happening of the sort that could essentially compromise the clients. There is no patch for this as it is a design flaw in the 4-way handshake. Because Hole196 requires the exploitation of the GTK, which is given out to anyone authenticated within the network, all you would need to do is restrict access to people you do not want potential access to more than what they are intended to get. This is not a fail-safe mitigation technique of course, but simply an overcoat on what was mentioned earlier of possibly integrating some sort of WIPS, or at the least, some sort of client side tool that detects sudden changes in your ARP table, or DNS changes.

The next mitigation technique goes to “administrators” that use weak passwords for their network WPA password. Since the only way to get the PSK from the PMK is by brute forcing the SHA-1 representation of the PSK until there is a match, anything after 20 dynamic alpha-numeric characters will thwart any hacker from being able to successfully perform a dictionary attack, or brute force attack against the WPA password. Once the handshake is captured, the attacker can then go home and crack it offline, so they have all the time in the world to crack your password. This is why you want to make it as long as possible to be able to get rid of the idea of the password being successfully guessed in any set time within a lifetime. This mitigation
technique also applies, not just to WPA2, but to WPA as well. Always change any default passwords.

The final mitigation technique can be used to harden your WPS enabled network. The best method of doing so, is to turn WPS off when it is not in use. Only enable it as you are needing to use it, other than that, have it off at all times. One could set the pin to dynamically change after so many failed attempts, or after so many seconds, this is called a “lock out period” where after so many failed attempts, WPS turns off for a minute, and reactsivate with a new pin. This can definitely slow down the hacker from getting into the network, but that is all it is, a slowdown attempt. The hacker can fine-tune reaver to only try a pin every minute if they wanted to, so it wouldn’t lock the user out, and with only about 11,000 attempts at that very slow pace, an intruder could potentially still get ahold of the PSK in approximately seven and a half days of non-stop WPS brute force attacks. By default, most access points have one pin, and leave it at a constant with unlimited attempts allowed. This should be configured manually within the access point’s administrator page/console to at least not allow for unlimited attempts to happen.

These are all very good mitigation techniques when dealing with a small business, or home environment. In an ideal world, one may want to look into using the protection of 802.1X (Which is WPA2 Enterprise) This is a hardened method of not allowing said GTK exploitation in the Hole196 exploit, and also thwarts the ability to reverse the 4-way handshake to find the user’s password by means of brute force since, like discussed at the beginning of this paper, 802.1X makes the client authenticate with a RADIUS server behind the network before being able to actually get into the network itself.
4.0 Conclusion

In conclusion, the free movement and connectivity that wireless networking provides us with is incredibly useful in today’s rapid moving environment. It allows us to stay connected without being bound to a cable. With this availability, comes a bit of sacrifice, which is security. As discussed, wireless networks are prone to attackers, since anyone can sniff out the free flowing data from the air and make sense of it. We discussed all the wireless security protocols available, saw how some are more efficient than others, also made an exploit for WEP, and proved mitigation techniques to thwart attackers from getting their hands on your or your client’s information by providing a detailed methodology in mitigation techniques. While many of the wireless security protocols can be cracked and have holes in them, we can still come up with methods to plug those holes and make the wireless networks we have a more secure means of sharing private data amongst the recipients you intend that data to be seen by, and no one else. It is easy to maintain the wireless networks secure by practicing the said mitigation techniques so everyone on the network can have a pleasant time online without worrying about the threat of their information being compromised. This documentation, as with many technical mitigation techniques within this field may prove to be obsolete after some time, therefore it will be kept updated as time progresses and new flaws in today’s wireless security protocols are uncovered.
Works Cited


   http://www.airtightnetworks.com/WPA2-Hole196

   http://cybersecuritylabs.files.wordpress.com/2014/02/w02.png


Matthieu Caneill, J.-L. G. (2010). Attacks against the wifi protocols WEP and WPA.


   http://sailjamehra.files.wordpress.com/2010/02/ps3q03-lowerywireless2.jpg


Wikipedia. (n.d.). Wired Equivalent Privacy. Retrieved from Wikipedia:
Appendix

Tools Used:

- Aircrack-ng (For WEP/WPA)
  - Airmon-ng start wlan0
  - Airodump-ng mon0
  - Airodump-ng –bssid FF:FF:FF:FF:FF:FF –w words mon0
  - Aircrack-ng –w /dictionary words
- Feeding bottle (GUI Representation of Aircrack)
- Associcrack (self-made bash script)

```bash
#!/bin/bash

clear

echo "What is the MAC address of your target AP?"
read input_variable

clear

read -p "Would you like for me to put your wireless interface into monitor mode? Say no if you already did so, or want to do it manually. [Y/N]" -n 1 -r
if [[ $REPLY =~ ^[Yy]$ ]]
then

clear

echo "What is your wireless card interface? {i.e. wlan0}"
read input_variable3

airmon-ng start $input_variable3

fi

clear

echo "What is your interface you will be fake associating with? Default should be mon0"
read input_variable4

mdk3 $input_variable4 m -t $input_variable
```

OS Used:

- Kali
- Windows
- Xiaopan