

WEP (In)Security

Ross Buffington* & Will Proffitt
Faculty Advisor: Dr. William Hooper

Abstract

As more computer networks make the migration to wireless, many factors come into play. Among these, two requirements take a high priority: data encryption and network authentication. One of the first attempts to satisfy these requirements for the IEEE 802.11 wireless standard is known as Wired Equivalent Privacy (WEP).

In 2001, Scott Fluhrer, Itsik Mantin, and Adi Shamir released research detailing a catastrophic weakness in the RC4 encryption algorithm of WEP. By collecting enough data packets and extracting their unique initialization vectors (IVs), the encryption key can be computed statistically with near perfect accuracy.

With a basic knowledge of computer programming, the Linux terminal, and wireless networking, anyone can successfully gain unauthorized access to a WEP protected network and the traffic therein. Because attacks on wireless networks are easy to execute and relatively untraceable, a strong data encryption method is of utmost importance. Therefore, businesses, schools, and even common households should pursue alternatives to WEP encryption.

I. Introduction

For the past ten years, wireless communication has become increasingly pervasive in our lives. Nowhere is this more apparent than in the area of wireless computing. By connecting one's personal wired network to a wireless access point (AP), even the most illiterate of computer users can instantly become the manager of their own WIFI hotspots. Because wireless networking has become so efficient, practical, and easy, it has opened the door to a whole new realm of communication.

But with their benefits, wireless local area networks (WLANs) are not without their risks. Because data packets are sent through the air, the possibility of interception by unintended recipients is uncontrollable. Thus, in an effort to protect WIFI users from eavesdroppers and provide added levels of security, the Wired Equivalent Privacy protocol (WEP) was implemented into the IEEE 802.11 standard for wireless LAN communications in the late 1990s.

It only took a few months for the first research papers on WEP's poor implementation of the RC4 encryption keystream to surface in the scientific community. Fluhrer, Mantin and Shamir (FMS) were the first to submit that by collecting enough data packets from a wireless communication protected by WEP, a computer could calculate, with high statistical accuracy, the secret encryption key and thus break the encrypted ciphertext. [1]

Now approximately seven years after the first implementations of attacks on WEP protected networks and three years since the protocol was officially declared "deprecated" by the IEEE 802.11 committee (due to failure to meet security goals), one would assume the general population would have forsaken the insecure protocol by now.

However, while exceedingly superior alternatives to WEP have been released, many people still use WEP to secure their wireless networks (**Figure 1**). [3]

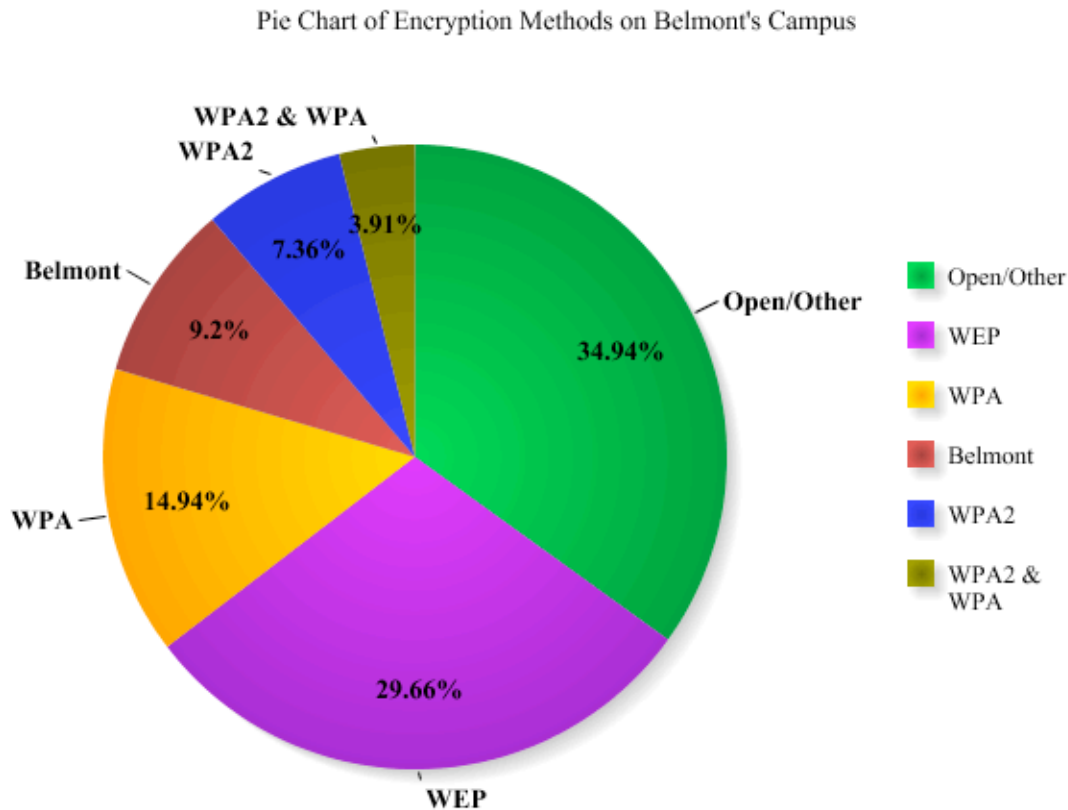


Figure 1: After a survey of the wireless networks on campus using a 7 dbi gain omni directional antenna and an atheros-chipset-based card for passive packet monitoring, we found that almost 30% were secured with WEP. 435 total networks were detected through passive monitoring at the ground level of Belmont's Campus, but additional networks were probably present at higher elevations in dormitories and campus buildings. Additionally, approximately 12 of the 152 open networks were ad-hoc print servers with no assumed connection to the internet or any network. Other open networks may have had radius authentication and encryption techniques, which masked their true level of security. Overall, WEP occupied a plurality of the encryption techniques, despite its status as a deprecated security protocol.

We have researched the WEP protocol and a number of attacks on network traffic secured this protocol for two reasons. First, it is our desire to learn how WEP encryption works, how the decryption aspect works, and how breaking this security protocol works. In all honesty, as undergraduate students of computer science, the idea of “code cracking” seems pretty dang interesting to us. Secondly, we desired to research the WEP

protocol in order to further discourage the general population from using it. A great amount of research has already been conducted into the weaknesses of the WEP protocol's implementation of the RC4 stream cipher and it would not be incorrect to state that our research is essentially equivalent to beating a dead horse. Nevertheless, many common computer users still feel protected behind their WEP networks.

One concept, we bring to the table, has rarely been discussed in popular papers on WEP vulnerabilities. Through our practical lab trials, we discovered a method of inducing large bursts of packets from an AP with little to no original traffic. Since a statistical analysis attack on WEP takes a large quantity of packets, the ability to induce traffic is incredibly important from the time perspective.

II. The WEP Protocol

The IEEE 802.11 committee drafted WEP to meet three major goals: prevent eavesdropping, protect the network infrastructure from unauthorized access, and prevent tampering with transmitted messages. The protocol relies on a secret key, which is shared between the AP and all computers that want access to the WLAN. Originally, WEP called for 40-bit secret keys and later, larger key length of 104-bits was introduced in an attempt to defeat the practicality of brute-force key cracking. [6]

In understanding how WEP works, one must first analyze its underlying method of encryption. The RC4 algorithm creates a cipher-stream (or key-stream) which is exclusive Ored (XOR) with a plaintext message to produce encrypted ciphertext. The algorithm is time-tested, reliable (when implemented properly), and among the most popular methods of encryption in the world today.

One slightly confusing aspect of WEP is the naming convention of the two most popular key length variants. On most consumer level WI-FI products, encryption with a 40-bit key is known as "64-bit encryption" and a 104-bit key is known as "128-bit encryption." The extra 24 bits tacked onto each of these names represents the 24-bit *Initialization Vector (IV)*, which is concatenated with the secret key and supplied to the RC4 algorithm to create the keystream. The true purpose of the IV is to avoid using the same keystream in two different ciphertexts. If two data packets of a wireless transmission are encrypted with the same keystream, the XOR of the two encrypted packets is equal to the XOR of the two packets, unencrypted. [1][3]

```

if    C(a) = P(a) ⊕ RC4(k)
and   C(b) = P(a) ⊕ RC4(k)
then  C(a) ⊕ C(b) = (P(a) ⊕ RC4(K)) ⊕ (P(a) ⊕ RC4(K))
      = P(a) ⊕ P(b)
where P(message) represents plaintext,
      C represents ciphertext,
      RC4(Key) represents keystream.
```

Every time the key is provided to the RC4 algorithm, a new IV is also provided to augment this key and make it unique. Then, when each packet is encrypted, the ciphertext is a product of XORing a unique keystream with the plaintext. In order to decrypt WEP encrypted packets, one must have the original key and the IV of the packet. Therefore, each packet that is sent across the air contains its encrypted data payload along with the specific IV used for its encryption. One should note that this IV is in plaintext form and is attainable by simply analyzing the encrypted packet (**Figure 2**). [3][5]

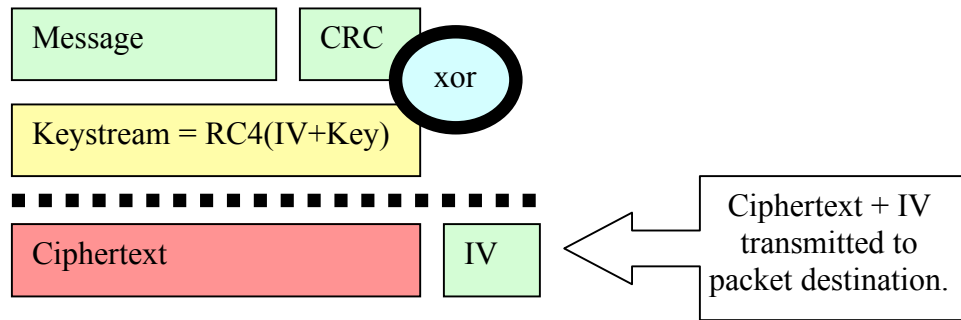


Figure 2: The basic flow of a encryption in WEP. Note: CRC stands for *Cyclic Redundancy Check*. It is a function that handles the integrity checksum, which is calculated and combined with the plaintext message prior to encryption.

The decryption process of a WEP encrypted packet is basically the opposite of the encryption process. The ciphertext is XORed with the keystream to produce the original plaintext.

$$\begin{aligned}
 P' &= C \oplus RC4(IV + Key) \\
 &= (P \oplus RC4(IV + Key)) \oplus RC4(IV + Key) \\
 &= P
 \end{aligned}$$

III. Breaking WEP

In 2001, Scott Fluhrer, Itsik Mantin, and Adi Shamir (FMS) released the foundational paper of WEP cracking. They discussed several weaknesses of RC4 including the possibility of secret key computing in WEP's implementation, by way of statistical analysis. Since then, much research has been conducted to verify and further their original research. It has been shown that WEP is vulnerable to a number of attacks, both active (where an attacker interferes with WLAN traffic) and passive (where an attacker simply monitors WLAN traffic and attacks collected packets).

Originally, based on the FMS research, cracking 128-bit WEP required between 4,000,000 and 6,000,000 data packets. One might ask how difficult it would be to capture a few million packets of WLAN traffic. This depends on the number of clients on the network and the type of applications they are running. We captured packets on our test network as a client downloaded a 100-MegaByte file and obtained 109,795 packets. More

recent research (2007) has shown that a 104-bit WEP key can be cracked with 50% probability using only 40,000 data packets. [1][2][3]

Recalling the purpose of IVs in the WEP encryption process, we remember it is possible to detect correlations between ciphertext and plaintext when an RC4 keystream is reused. The 24-bits for unique IVs is not enough to protect against this vulnerability. On a busy network, the 16,777,216 possible IVs are recycled every few hours or less. Furthermore, it is important to note that while some machines generate IVs pseudorandomly, many use a simple counter. In either case, with enough passive monitoring, an attacker is able to collect packets with duplicated IVs, leaving the WEP secret key vulnerable to deciphering.

Passive monitoring of a WLAN differs from active monitoring in that no probes are sent from the network interface card (NIC) doing the monitoring. The NIC simply “listens” on all WLAN radio frequencies of the 2.4Ghz frequency spectrum. This allowable spectrum is subdivided into eleven different channels to reduce noise and signal disruption on any single channel by giving devices more options for broadcasting. By listening on all channels, one can gather information about every wireless network within range such as: the physical address (MAC address) of the APs, the channel of broadcast per AP, the type of encryption being used per AP, and the amount of traffic each AP is currently experiencing. From this information, an attacker can pinpoint a target AP and begin their attack.

The first step in any WEP attack is monitoring the target AP on its specific channel of operation and collecting the data packets being transferred across the network. Keep in mind that this step is completely undetectable by any security measures put in place by network administrators. At this point, an attacker need only wait until they have recovered 4,000,000-6,000,000 packets and it would be possible to compute the secret key, with absolutely no interaction with the target network, using the techniques described by FMS.

However, collecting millions of packets can take a while and conditions are not always in favor of the FMS attack. What if the wireless load of the network is very low? What if there is only one wireless client on the network and they aren't even downloading anything? Luckily (or unluckily depending on who the owner of the WEP network is), an attack has been developed for such adverse conditions. Andrei Pyshkin, Erik Tews, and Ralf-Philipp Weinmann (PTW) published a paper extending some older theories on RC4 vulnerabilities to work with WEP. The attack works by collecting a specific type of packet, which more information about the plaintext can be inferred. [2]

The Address Resolution Protocol (ARP) enables clients to find each other on a network and build a table of peer addresses by broadcasting ARP request packets, to which all other LAN clients respond to, with ARP reply packets. The basic idea of ARP is to shout “hello” to everyone on a network and record the location, or address, of everyone who responds. Therefore, when an ARP request is sent from a network client, multiple ARP reply packets are sent back to that client from the AP. Furthermore, every ARP packet is

essentially the same, having a large amount of fixed bits in its packet. This provides an attacker with more knowledge of a packet's data payload, allowing secret key cracking techniques to be applied to ARP packets with ease. [4]

Using this knowledge, the PTW attack method allows an attacker to generate a large amount of packets in a short period of time. The basic idea behind this attack is: 1) the attacker monitors the network traffic until an ARP request is generated by a client on the network, 2) the attacker then floods the network with ARP requests using a technique known as *injection*. [2][8][9]

Packet injection occurs with a NIC is made to send custom-tailored packets to a given access point. Not every wireless card has this capability, as it requires open source drivers as well as specific chipsets. However, many common wireless NICs do support it. A NIC can be controlled to send any type of packet and free software is available to “forge” these packets on the fly. [7]

By flooding a network with re-injected ARP requests, the AP begins spitting out hundreds of ARP replies per second. An attacker implementing the PTW attack then collects these replies and runs a statistical analysis crack on them to compute the secret WEP key. This attack has been shown to work with 50% probability when the number of captured packets is 40,000 (or 100% probability with 100,000 packets). This is a huge leap in packet quantity from the FMS attack, which required a minimum of 4,000,000 packets. Where FMS could take hours or days to collect enough packets, PTW takes minutes. [2]

If an attack takes only a few minutes to mount in its entirety, what is stopping any 12-year-old kid with a computer from cracking his neighbor's WEP protected network? We believe the answer is *work*. It takes a fair amount of dedication to learn to use the tools required for mounting a simple WEP attack. And although these tools essentially “do it for you,” the commands used in operating these tools as well as an understanding of the top-level flow of an attack require a good deal of time and effort to function.

Because of this, we wrote a Linux BASH script to automate the PTW attack from start to finish. Our program takes the AP's service set identifier (SSID) as the only parameter. Automated from here, we find the channel the AP is broadcasting on, begin monitoring for ARP requests, and once found, begin re-injecting the packet and capturing the deluge of ARP replies. Once we have 40,000 ARPs captured, the program automatically begins attempting to compute the secret key and if failure occurs, continually re-attempts until success.

IV. Results

With our script, we were able to test different randomized keys for strength and compute an average of how long an automated PTW attack takes to succeed in conditions where one or more wireless clients are on the network. From this data, we present to the reader a normal distribution of attack times (**Figure 3**).

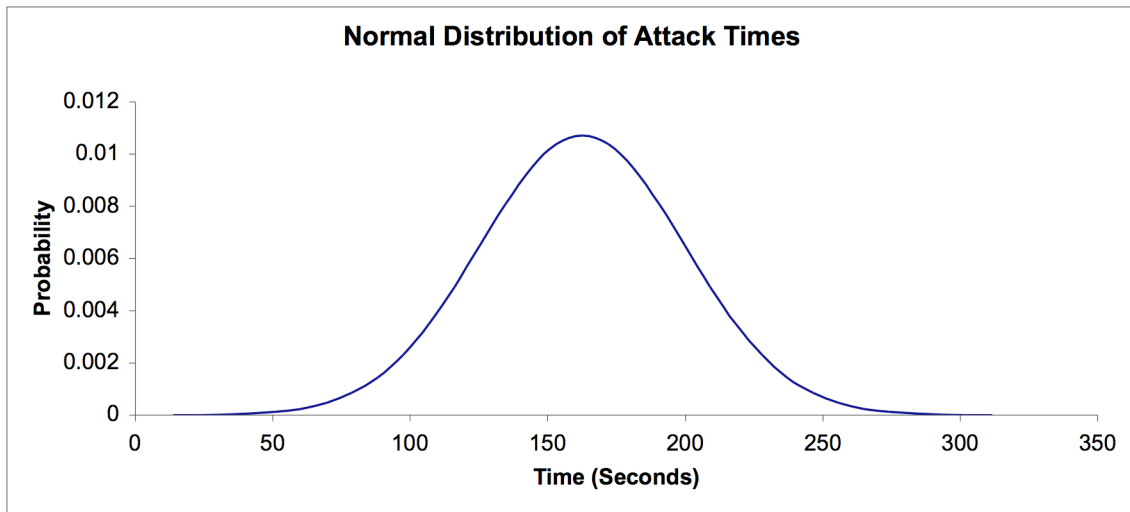


Figure 3: In a real world test of our script, 62% of the attacks took less than 160 seconds, or 2 minutes and 40 seconds. The lowest attack time was 134 seconds; the highest was 309 seconds. The average attack time was 162.58 seconds. All keys were randomly generated and were of the 128-bit variety. The average time to compute the key with the PTW crack method was 15.06 seconds. This is almost three times the value suggested in the PTW paper.

If traffic on the network was low (or non-existent), we found the PTW attack difficult to mount. However, through trial and error (and later through reading the original paper on PTW... doh!) we discovered an interesting way to speed up the process of collecting the original ARP packet required for the attack. With packet injection, we were able to send a *De-authentication Packet* (or de-auth for short) to a client on the network. This type of packet signals the client to disassociate itself with the AP. If a de-auth packet is continually sent, the target client is essentially denied service from the access point. Once the target receives no more de-auth packets, by default on most operating systems, it automatically reconnects to the AP. Upon re-association, an ARP request is broadcasted to update the client's ARP cache. Note: we found that Mac OS 10.4-10.5 is not vulnerable to the de-authentication attack because the operating system does not rebroadcast an ARP request upon re-association with the network. However, Windows XP is perfectly vulnerable.

While our script could be used for other variations of WEP encryption, it is important to note that we have only taken into account 64-bit and 128-bit standard WEP. We do not see this as a hindrance however, because variations such as Dynamic WEP or 256-bit WEP began surfacing around the same time as better alternatives were becoming popular. Furthermore, most of the focus in the scientific community has been on these two common implementations of WEP.

V. Conclusion and Discussion

The Wired Equivalent Privacy protocol was introduced in 1999 to meet three security concerns of wireless networks. Since its release, it has been shown that WEP does a poor job of meeting these requirements and is vulnerable to a number of exploits. WEP has since been officially declared depreciated and has been replaced by a number of alternatives.

Where before, mounting an attack on WEP took a good deal of time and effort, to learn the tools required, to mount the attack by collecting millions of packets, and to understand how to mount the attack, by using our script, anyone can crack a WEP encrypted network in a minimal amount of time, provided there is at least one client connected to the access point. We do not plan on releasing our program to the public. Instead, our true purpose and hope is to show the public that this protocol has been annihilated and anyone still using WEP places their network in danger of compromise. In our opinion, if your network is secured with WEP, it is already compromised.

As we have provided an understanding of how to break WEP encrypted networks, we would also like to conclude with tips to secure WLANs from current attacks. Firstly, wireless networks should always be encrypted and require authentication. WI-FI Protected Access (WPA) was released as an immediate solution to the insecurities of WEP. It is much more secure due to the way the encryption keystream is generated and because it does not allow duplicated IVs. Statistical attacks described by FMS are not possible with WPA packets for this reason.

An access point can be configured to be “hidden,” which means its SSID is not visible to the general public. While this does not fool the packet-sniffing tools we used in our script because the packets are not actually hidden, the average user will not even know they are surrounded by WLANs if the APs are hidden.

We also recommend implementing a Remote Authentication Dial In User Service (RADIUS) server in all wireless LANs. RADIUS provides a centralized means of authorization for a WLAN. Every user is required to provide a secondary set of credentials to gain access to network resources. This means that even if an attacker gains access to your WPA secured network, they would still have to crack another level of security, which is often beyond the ability of the average WI-FI stumbler.

Finally, for private WLANs in which all clients are known, an access control list can be configured based on the network interface card MAC address of each authorized user. Additionally, network traffic can be routed through a service like Secure Shell (SSH) or a Virtual Private Network (VPN) to provide another layer of encryption. Any single method of securing a WLAN can be vulnerable to attack but when combined, the probability that a network will be compromised is greatly reduced.

In the future, we hope to further our research in wireless network security by focusing on WPA vulnerabilities as well as developing a deeper understanding of what an attacker has access to once a WLAN has been compromised. Additionally, our hope in continued

research is to see a need and use our growing understanding of technology and software development to create software to solve current security problems in WLANs.

It is our belief that as society becomes more dependent on wireless networks, research into vulnerabilities and security will become increasingly important. By understanding the fundamental flaws in early versions of wireless security, we will be more apt to understanding newer flaws.

VI. Acknowledgements:

I would like to thank all who have helped with this project. Specifically, thank you Will Proffitt for providing much of the required equipment for our project as well as the spectacular charts in this paper. Thank you Dr. William Hooper for providing guidance and understanding along the course of this project. Lastly, I'd like to thank all of the individuals living in my apartment complex who own WEP protected access points (muhahaha!).

VII. Resources:

- [1] Fluhrer, Mantin, Shamir. “Weaknesses in the Key Scheduling Algorithm of RC4.” http://www.drizzle.com/~aboba/IEEE/rc4_ksaproc.pdf. 2001.
- [2] Pyshkin, Tews, Weinmann. “Breaking 104 Bit WEP in less than 60 seconds.” <http://eprint.iacr.org/2007/120.pdf>. 2008.
- [3] Goldberg, Ian. “Intercepting Mobile Communications: The Insecurity of 802.11.” <http://www.isaac.cs.berkeley.edu/isaac/mobicom.pdf>. 2001.
- [4] Silberschatz, Galvin, Gagne. “Address Resolution Protocol.” *Operating System Concepts with Java*. p676. 2007.
- [5] Bittau, Andrea. “WiFi Exposed.” <http://www.acm.org/crossroads/xrds11-1/wifi.html>. 2004.
- [6] “How 802.11 Wireless Works.” <http://technet2.microsoft.com/windowsserver/en/library/370b019f-711f-4d5a-8b1e-4289db0bcafd1033.msp?mfr=true>. 2003.
- [7] “Aircrack-ng Resources.” <http://aircrack-ng.org/doku.php?id=links>. 2008.
- [8] “ARP Amplification.” http://aircrack-ng.org/doku.php?id=arp_amplification. 2007.
- [9] “Tutorial: Simple WEP Crack.” http://aircrack-ng.org/doku.php?id=simple_wep_crack. 2008.