Security

Byte-Sized Decryption of WEP with Chopchop, Part 2

Last updated Jun 16, 2006.

Inverse Arbaugh Attack

NOTE

Editor's Note: For those of you jumping in directly to this page, be sure to read Part 1 *on the previous page.*

KoreK took the widely known weak ICV problem and deduced that if an encrypted packet is truncated by one byte, a new valid message can be created because its new ICV is tied to that one missing UNENCRYPTED byte. Technically this is known as an Inverse Arbaugh Attack, which leverages the weak ICV against itself via a series of calculations to deduce, or guess, the missing byte. Again, the only concern is whether or not the packet is valid, as determined by the ICV. The data in the packet is not relevant.

In short, KoreK took a very close look at how the ICV was created and found a mathematical relationship between the truncated plaintext byte of the message and the value used to turn the invalid shortened message into a valid one. The following breaks down the process.

The first step is to take the captured packet(M) and remove the last byte of the packet to create a value, M-1. Since we do not know what the plaintext value of the truncated byte was, we next have to start guessing. Fortunately, there are only 256 possible values (00, 01, ...254, 255) for that plaintext byte. So, we first assume the truncated byte was 00 and plug that value into an equation (see note below) to create a bitmask that attempts to turn the invalid M-1 packet into a valid M-1 packet. We then test this new M-1 packet by injecting it into the network. Once the packet is received by the access point, it is checked and either validated or rejected. Again, each packet is based upon a guess of the truncated byte, so it only has 1/256 chance of being correct. However, if the guess is correct, then the AP will validate it, and resend it back out to the wireless network. Assuming an attacker is monitoring the wireless network, they will see this packet and know that they made the right guess. As a result, they now know the last plaintext byte of the message (M-1). It is then possible to calculate the PRGA value used to encrypt the byte via yet another XOR operation (P XOR M = RC4 PRGA). In other words, an attacker can crack a packet one byte at a time and obtain both the plaintext value of the packet and the streaming key used to encrypt that packet.

NOTE

Note: This mathematical equation that is used is outlined in the DOC file that KoreK included with Chopchop. However, unless you enjoy dealing with polynomials, inverse values, and coefficients, then trust me when I say that it is possible to create a bitmask value that can be XORed with the truncated message to produce a new message (M-1).

So, let's review this again.

First, the message (P+P(ICV)) is truncated by one byte (P-removed) to create "M-1." Second, we create a "Value" via a mathematical equation that is directly tied to P-removed. Next we XOR M-1 with "Value" to create a new M-1v that is hopefully valid. Finally, the M-1v packet is injected. If M-1v is retransmitted by the access point, it is valid, which also means the "Value" was correct and therefore means the guessed plaintext byte was correct. Using this process, the attacker can deduce the plaintext byte and the PRGA byte that was used to create each encrypted byte of the packet. If the AP does not retransmit the byte, then the guessed byte was not correct and the attacker needs to try the next byte value.

If the byte value was correctly guessed, then the whole process starts anew. Except this time the attacker starts with the new packet that was just injected and validated by the AP. Again, it doesn't matter what is in the packet, as long as the ICV is valid. The attack will eventually yield the entire PRGA used to encrypt the original packet, which can then be XORed with the original encrypted packet to produce the plaintext data.

Chopchop

Thankfully, KoreK put all these flaws together into one package known as Chopchop. This nifty piece of code does several things. First, it monitors incoming data for specific packets that are injected during the testing routine. The program then takes a previously captured valid packet and starts the decryption process. It truncates the last data byte, calculates a "Value" based upon the guessed byte, XORs this "Value" with the captured packet to create a new packet, spoofs the destination MAC address for tracking purposes, rebuilds the packet, and then injects it into the wireless network. Since the program is monitoring all the network traffic, it will detect the valid packet when it is retransmitted by the AP. Once this packet is received, it looks at the destination MAC address and determines the corresponding plaintext byte that was used to create the "Value" that was used to create the encrypted packet. This byte is recorded, along with the PRGA value that would be used to encrypt the byte, and then this whole routine is repeated with the `new and valid captured packet` until the entire PRGA has been deduced. Finally, the deduced PRGA is XORed with the original cyphertext to create the original plaintext. While this somewhat simplifies the attack, clearly illustrates that any WEP packet can be cracked despite the use of dynamic keying.

Using Chopchop is much easier than understanding how it works. First, setup a wireless network that is using WEP. Then load up a laptop with the Auditor LiveCD from http://www.remote-exploit.org (please donate!). Open up ethereal and capture a single data packet and save it to memory. Locate the file and run "tethereal –nr file.pcap" to obtain a valid MAC address. For the best results, you will need a "target" to be online, which should be exposed via the tethereal command. Finally, configure your wireless network card for monitoring and run the following command:

```
chopchop –burst 13 –m mac -b bssid -p packet.pcap
```

The burst option defines how quickly to inject the packets. The –m value is the MAC address of the target. The –b value is the BSSID of the access point. And –p is the single packet you are trying to crack. Once you launch the attack, the program will output the current byte and how many frames were written to the screen. Expect it to take anywhere from several seconds, to a couple minutes, depending on the size and content of the packet. Once it has completed, you can reopen the packet in Ethereal or tethereal to view its contents! Figure 2 illustrates what Chopchop looks like while cracking a packet.

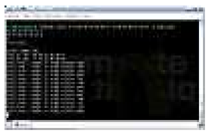

Figure 2: Chopchop in action

Summary

Chopchop is a complex piece of work that targets WEP's implementation of XOR and ICV. By attacking a WEP protected packet one byte at a time an attacker can bypass the whole encryption process and gain access to the hidden data underneath. Gone are the days where dynamic keying offered adequate protection for its users. In fact, WEP is broken in so many ways it should be outlawed. If you currently deploy this "protection," please find an alternative. Program like Chopchop only highlight the dangers and weakness of this protocol — it is up to you to put this knowledge to use!

***Thanks to KoreK for writing this program and including some details in the DOC file included with the package. I doubt this article would have been possible without it!

***Thanks to Joshua Wright for reviewing the article, making a few suggestions, and clearing up a few misconceptions about dynamic

keying for me!

References and Resources

WEP Inverse inductive attack reveals plaintext:
http://www.wirelessve.org/entries/show/WVE-2006-0037

Chopchop Information:
http://www.wirelessve.org/entries/show/WVE-2006-0038

Chopchop (DOC file contains some very details information on the attack):
http://www.netstumbler.org/showthread.php?t=12489 (must be member to download)

Arbaugh, William A. "An Inductive Chosen Plaintext Attack against WEP/WEP2:"
http://www.cs.umd.edu/~waa/attack/v3dcmnt.htm