International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 01 | Jan 2019 www.irjet.net p-ISSN: 2395-0072

Analysis of Security Vulnerabilities in Wired Equivalent Privacy (WEP)

Kwabena Akomea-Agyin[1], Michael Asante[2]

[1,2]*Department of Computer Science, Kwame Nkrumah University of Science and Technology, Kumasi, State, Ghana*

Abstract:- *This research paper analyzed the security vulnerabilities within the architecture of Wired Equivalent Privacy (WEP) that can be used to hack into a WEP enabled wireless network.*

It was found out that WEP uses static keys for both authentication and encryption. Once the key is compromised during authentication, the same key can be used to decrypt every packet. Secondly, WEP does not support mutual authentication. Thirdly, WEP uses short Initialization Vector (IV) space (24 bits) which leads to IV re-use and keystream re-use attacks. Finally, WEP uses a linear checksum for integrity check which leads to message injection and modification attacks.

Based on the vulnerabilities, a brute force attack was successfully used to retrieve the WEP encrypted password. At the end of the study, it was proven that WEP is completely insecure no matter how complex the WEP key is. Hence WEP should not be configured on any wireless network. Yet, our survey of 1.271 Access Points in Ghana showed that 8.1% of the surveyed networks were still using WEP.

Keywords: Wired Equivalent Privacy (WEP), Vulnerability, Attack, Initialization Vector (IV). Integrity Checksum (ICV).

1. Introduction

According to Nwabude (2008), Jaiaree (2003) and Rackley (2007), WEP was the first encryption scheme made available to Wi-Fi in 1999 (Jaiaree, 2003).

WEP is based on RC4 encryption (Raza et al, 2010; Tanzella, 2003; Fluhrer et al, 2001). RC4 is a symmetric stream cipher (Biham & Carmeli, 2008).

This means that both the transmitter and receiver use the same key to encrypt and decrypt every data (Khan & Khwaja, 2003; Raza et al, 2010). The RC4 Algorithm requires an input key of size 64-bit or 128-bit (Chandra et al, 2009; Vivek, 2011).WEP uses a 40-bit or 104-bit key plus a 24-bit cryptographic salt called an IV as a seed to the RC4 algorithm (Chandra et al, 2009; Vivek, 2011) as shown in figure 1.

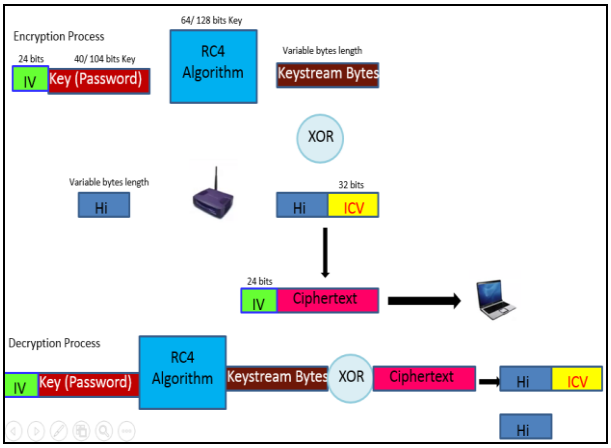

Figure 1: Architecture of WEP

The RC4 algorithm consist of the Key Scheduling Algorithm (KSA) and Pseudo-Random Generation Algorithm (PRGA) (Chandra et al, 2009; IEEE std 802.11, 2012; Souradyuti & Prencel, 2004). The KSA is given by the pseudo code:

i= j= 0;

For i= 0 to 255 do

j= (j + S[i] + K[i]) mod 256;

Swap S[i] and S[j];

The PRGA is given by the pseudo code:

i= j= 0;

i= (i + 1) mod 256;

j= (j + S[i]) mod 256;

Swap S[i] and S[j];

k= S[S[i] + S[j]] mod 256;

The KSA first initializes an S[box] to S[i]=i:

S[box] : S[0]=0 S[1]=1 S[2]=2 S[3]=3 S[4]=4 … S[254]=254 S[255]=255.

Next, it takes the 64 or 128 bit WEP key to repeatedly fill a 256 byte array. If WEP Key is say "KEY" then,

K[box] : K[0]=K K[1]=E K[2]=Y K[3]=K K[4]=E K[5]=Y K[6]=K … K[254]=Y K[255]=K

Next, KSA converts the key into their ASCII characters:

K[box]: K[0]=75 K[1]=69 K[2]=89 K[3]=75 K[4]=69 K[5]=89 … K[254]=89 K[255]=75

Next by going through the pseudo code,

For i=0, j= (previous j + S[i] + K[i]) mod 256.

Thus j= (0 +0 +75) mod 256 = 75 mod 256 = 75.

Now swapping the content of positions S[i] and S[j],

The content of S[0] which used to be 0 now becomes 75 whilst the content of S[75] which used to be 75 now becomes 0.

Thus after the first iteration, the S[box} becomes

S[box] : S[0]=75 S[1]=1 S[2]=2 S[3]=3 S[4]=4 … S[75]=0 … S[254]=254 S[255]=255.

After 256 iterations, the final output from the KSA is shown in figure 2.

142	42	48	158	245	175	193	161	169	218
6	41	249	146	227	181	44	15	71	176
236	56	122	37	79	106	171	40	89	59
58	33	27	215	53	145	94	22	136	149
162	54	159	7	143	152	205	99	25	195
225	154	13	85	76	125	92	38	117	240
226	196	138	185	134	20	202	189	220	32
126	127	148	140	36	203	84	197	16	104
223	11	90	222	139	180	211	200	121	168
51	167	157	164	228	179	69	224	45	144
66	209	190	103	67	4	19	235	151	150
57	163	72	46	70	253	166	241	102	243
231	186	111	74	86	174	21	61	247	31
18	47	188	135	156	172	217	83	141	219
246	255	105	237	114	238	250	98	251	118
120	93	124	23	24	184	254	97	5	39
165	26	29	199	82	182	232	137	192	183
77	75	35	216	64	201	14	91	160	78
108	133	107	155	8	9	96	3	50	208
95	204	212	88	28	173	55	252	244	177
49	0	65	147	12	130	234	131	109	52
1	129	213	128	60	34	87	110	43	153
206	207	63	85	178	81	198	221	73	
132	233	112	80	62	187	113	239	214	
191	116	119	123	170	10	30	2	230	
68	242	101	17	248	229	210	115	194	

Figure 2: Output from the KSA

Next the PRGA uses the output of the KSA to output a 256-byte keystreams as follows:

i= (previous i + 1) mod 256 = (0 + 1) mod 256 = 1 mod 256 = 1;

j= (previous j + S[i]) mod 256 = (0 + 6) mod 256 = 6 mod 256 = 6;

Swapping the content of positions S[i] and S[j],

The content of S[1] which used to be 6 now becomes 226 whilst the content of S[6] which used to be 226 now becomes 6.

Now generating the first keystream byte (k):

k = S[S[i] + S[j]] mod 256 = S[226 + 6] mod 256 = S[232] mod 256.

But S[232]= 230. Thus k= 230 mod 256 = 230.

Thus the first keystream byte is 230 or 11100110 (in base 2) which will be XORed with the first plaintext byte to encrypt it as shown in figure 1.

After 256 iterations, the final output from the PRGA is shown in figure 3.

11100110	10011110	10111000	11000	10111	10111010
1001	10101111	11110111	10100	11111010	1011111
11100000	11	1101101	0	11001100	11101101
11010011	10110010	10100011	1101110	101100	10111110
10111111	11001011	10001010	11001011	1001	1111100
11001000	11110110	10001011	11010011	100101	10111011
11111	1100	1000100	11101011	11110110	11110101
11001010	10010	1001	101101	11001001	11010101
11000110	11110010	111100	1110110	11011100	110100
111101	1000011	1011000	1001101	10000111	
10110111	11000001	1111110	11000101	11001011	
1001	111011	11011110	1101	101110	
1100001	1101100	10100000	1111000	10100101	
11100	1011001	11110001	11000010	11111010	
11001001	1111100	11110	1110100	1111001	
1010001	11110010	10100010	10001101	1100110	
1000101	11111	1110001	1010000	11101110	
10100100	101001	1010011	10101001	111101	
1011011	11000101	10101000	10010101	1110011	
100001	101100	11010100	1010010	10001011	
10001111	10111011	1001101	10001100	11010111	
10110000	1001111	10010101	1	1000110	
1000	10101000	101100	1	1101011	
11001001	10110001	1100110	1111000	1010100	
11111111	110011	11001001	10000001	101	

Figure 14 c shows resulting PRGA after 125 iterations in binary notations

Figure 3: Output from the PRGA

Next, a 32-bit Integrity Checksum (ICV) is computed and appended to the data prior to encryption (Chandra et al, 2009; Vivek, 2011; IEEE 802.11, 2012) as shown in figure 1. The ICV which is based on CRC-32 is to prevent anyone from tempering with the data in transit (Peterson & Brown, 1961; Chaabouni, 2006). The ICV and the data are concatenated into one block.

Next, a Random Keystream of the same size as the concatenated Data and ICV is picked from the output of the PRGA and XOR with the Data and ICV to produce the encrypted ciphertext (Arbaugh, 2001; Hutton, 2002) as shown in figure 1. For the purpose of easy decryption at the receiving station, the IV is appended to the ciphertext in plaintext prior to transmission (Mantin, 2005).

The XOR operation combines two bytes and generates a single byte (Chandra et al, 2009; Vivek, 2011). If the bits in each byte are equal, the result is 0; if they differ, the result is 1 (Chandra et al, 2009; Vivek, 2011; Peterson & Brown, 1961). Thus

0 XOR 0 = 0

0 XOR 1 = 1

1 XOR 0 = 1

1 XOR 1 = 0

At the receiving station, the receiver takes the IV plus a copy of the secret key and passes them through the RC4 algorithm to generate the same PRGA keystream bytes (Borisov et al, 2001). By performing an XOR of the ciphertext and same length of Keystream byte, the plaintext and the ICV is obtained. The last 32 bit which is the ICV is verified by computing the ICV of the decrypted message. If there is a match, the receiving station knows that the message have

not been tempered with during transit, otherwise it is rejected and the sending station is notified to resend the encrypted message.

2. Methodology

A series of experiments were conducted to find out the Vulnerabilities in WEP. A laboratory was setup that included a victim machine (supplicant), an Access Point, an Authentication server, a hacker machine (running BackTrack 5) and an Alfa AWUS036NH wireless card connected to the attacker's machine as shown in figure 4.

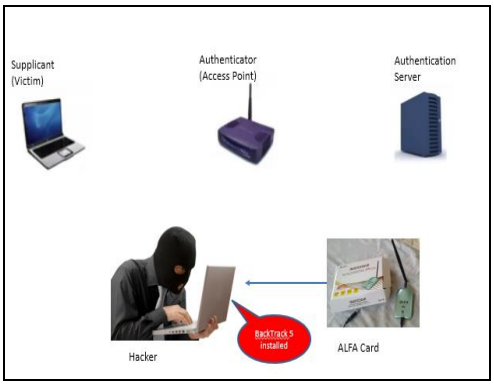

Figure 4:The laboratory setup diagram

3. Vulnerabilities in WEP

3.1 *A hacker can authenticate to the WEP network without knowing the WEP Key:*

The available Wireless Local Area Networks (WLANs) within the vicinity were eavesdropped for one that supports WEP using "airodump-ng" command in Backtrack5 as shown in figure 5

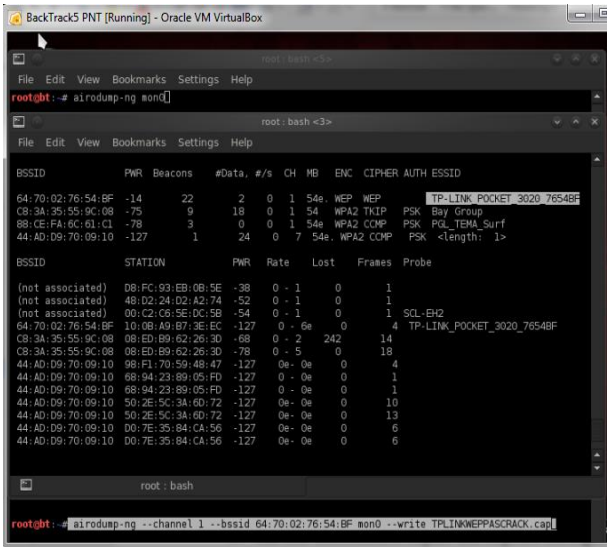

Figure 5:"airodump-ng" command used to monitor WLAN networks

A copy of the authentication request message, challenge plaintext message, challenge response ciphertext message, and authentication success message between the legitimate AP and legitimate client was saved as a .xor extension in Backtrack5 as shown in figure 6.

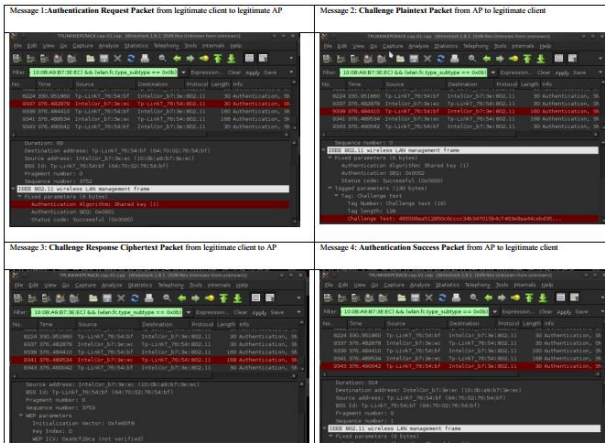

Figure 6: Four authentication request and response messages between the legitimate client and the Access Point captured with Wireshark

Next the attacker sent an authentication request packet to the AP. The AP replied with message 2 which is a challenge plaintext message. The attacker encrypted message 2 with the keystream byte and corresponding IV it had captured from the previous legitimate conversation between the client and AP and replied as message 3 to the AP. The AP successfully decrypted message 3 and granted the attacker authentication access to the network as shown in figure 7.

Figure 7: "aireplay-ng –fakeauth" command used to fake authentication into WEP network

Analysis:

WEP Authentication was successfully compromised because WEP uses an XOR operation to exchange authentication packets between a client and an access point. The XOR exhibits the associative and distributive properties of mathematics: a xor b = b xor a; and (a xor b) xor c = a xor (b xor c). Hence, by performing xor of a copy of the plaintext challenge message (message 2) with a copy of the encrypted challenge response message (message 3), a copy of the keystream byte that was used to encrypt the challenge response message is obtained. This keystream byte together with its IV was used to forge an authentication into the WEP network and the WEP network granted the attacker success.

Significance:

The significance of this outcome is that any attacker can eavesdrop on the authentication challenge and response

messages, compute the corresponding keystream byte and successfully authenticate to a WEP enabled network without knowledge of the password.

3.2 *WEP does not support Mutual Authentication*

The available Wireless Local Area Networks (WLANs) within the vicinity were eavesdropped for one that supports WEP using "airodump-ng" command in Backtrack5 as shown in figure 8

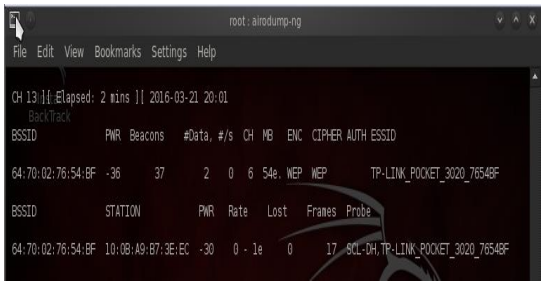

Figure 8: A legitimate WEP AP (BSSID= 64:70:02:76:54:BF) connected to legitimate client (MAC= 10:0B:A9:B73E:EC)

Next a fake soft Access Point was brought up using the "airbase-ng" utility in Backtrack5. This fake AP was configured to have the same SSID, frequency, and channel as the legitimate AP but with a higher transmit power than the legitimate AP as shown in figure 9.

Figure 9: A fake AP powered on with airbase-ng tool in BackTrack5

Next "aireplay-ng --deauth" command was used to send a denial of service attack to the legitimate AP to disconnect all clients from it as shown in figure 10.

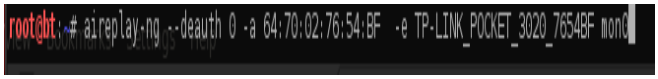

Figure 10: The "deauth" command in BackTrack used to disconnect all clients from the AP

A disconnected client in its attempt to re-establish the lost connection, began to send probe request messages searching for the WEP network it had previously connected to. The fake AP responded to the probe and the legitimate client sent an authentication request to the fake AP. The fake AP responded with a challenge plaintext. Next the legitimate client encrypted the challenge plaintext with its secret WEP key and responded. Because there is no mutual authentication in WEP, the fake AP pretended it was able to check the encryption and sent an authentication success message to the client without knowing the WEP key as shown in figure11.

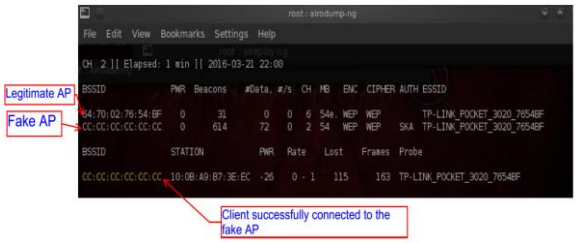

Figure 11: The legitimate client successfully connecting to the fake AP

The legitimate client upon receiving the authentication success message, continued to send encrypted messages to the fake AP which could be saved for later brute force attack.

Analysis:

A client was successfully lured to authenticate to a fake WEP AP because there is no mutual authentication in WEP, the client successfully accepted the access and began sending encrypted WEP packets to the fake AP.

Significance:

The significance of this outcome is that an attacker can lure a client to connect to a fake AP, and collect all the encrypted WEP packets from this client. These encrypted WEP packets can be saved for later offline statistical attacks to retrieve the WEP password without the presents of the legitimate access point.

3.3 *WEP uses linear ICV which leads to message Modification and Injection Attacks*

"Aireplay –arpreplay" command in Backtrack5 was used to capture a legitimate ARP packet between the legitimate client and access point as shown in figure 12. This captured ARP packet was successfully modified and injected back into the network.

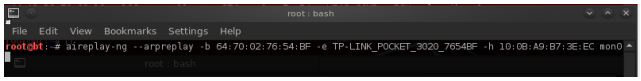

Figure 12:The "aireplay-ng –arpreplay" command

The client replied to the modified ARP request packet and more encrypted ARP packets with IVs were captured and stored for later bruteforce attack as shown in figure 13:

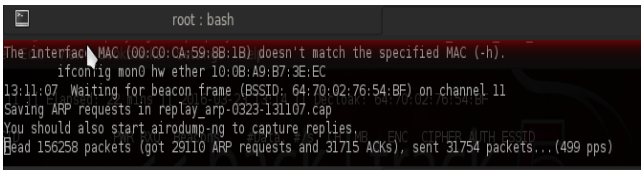

Figure 13:The results of the aireplay-ng --arpreplay command

Analysis:

This attack works by capturing an ARP request packet from the client, modifying it into an ARP Request packet for the same host, and injecting it back into the network. A WEP encrypted packet was successfully captured, modified, and injected back into the network without detection because

WEP uses an ICV which is linear and mathematically distributive: a xor (b xor c) = (a xor b) xor (a xor c).

The reason for the success is explained as follows:

Let C be a ciphertext intercepted by an attacker. Let's assume that C corresponds to some unknown message M as shown in figure 14.

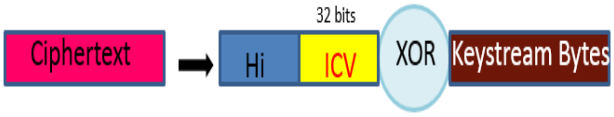

Figure 14: A captured WEP encrypted packet

Now the attacker creates an arbitrary bit-mask (Δ) of the same size as the encrypted data. The attacker then computes a CRC-32 checksum for this bit-mask (c(Δ)) as shown in figure 15.

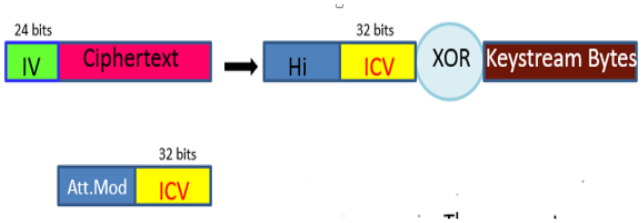

Figure15: A modified bitmask packet with its computed ICV

This bitmask (Δ,c(Δ)) was XORed with the original Ciphertext (C) to produce a new Ciphertext (C′) which is the modified WEP encrypted packet as shown in figure 16.

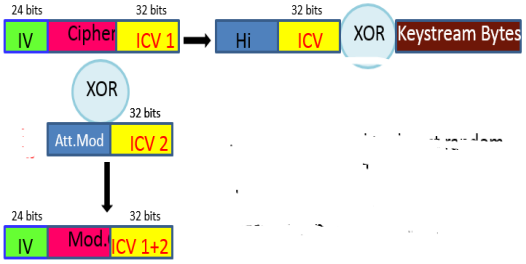

Figure16:The successfully modified WEP encrypted packet with a new computed ICV

Significance:

The significance of this outcome is that an attacker can modify a WEP encrypted packet without knowledge of the WEP password.

These vulnerabilities led to successfully cracking the WEP key.

4. Cracking WEP Password

An Access Point was configured to support WEP security with a password as shown in figure 17. For the purpose of this experiment, password "come123@123co" was chosen.

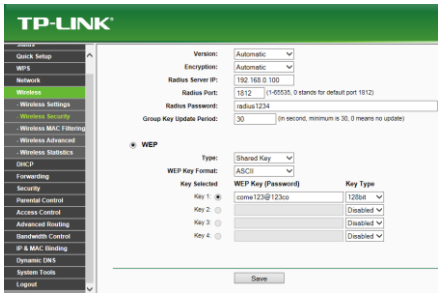

Figure 17: An access point configured to support WEP with password "come123@123co

A legitimate client was also configured to support WEP security with same password as shown in figure 18.

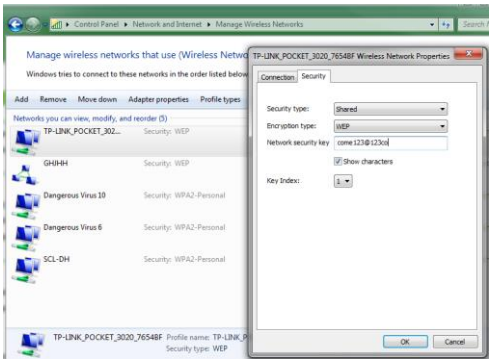

Figure18: A legitimate client configured to support WEP

The client was then connected to the WEP network as shown in figure 19.

Figure 19: The legitimate client connected to the WEP network

The command "airodump-ng mon0" was used to monitor all the available wireless networks as shown in figure 20a. The command was narrowed to the WEP network and the packets were captured to a .cap file as shown in figure 20b.

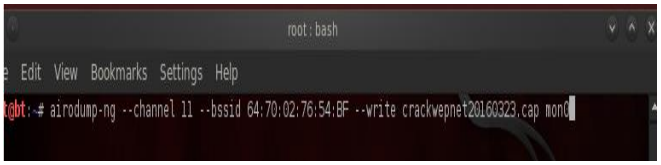

Figure20a:"airodump-ng" command used capture and save the WEP network packets

Figure 20b: The results of the "airodump-ng" command

Next, "aireplay-ng --arpreplay" command was used to capture ARP packet and to spoof the mac-address of the legitimate client as shown in figure 21.

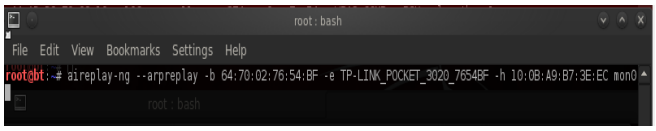

Figure 21:"aireplay-ng –arpreplay" command

Because the ARP packets had already been generated, the "aireplay-ng --deauth" command was used to disconnect all the clients from the AP as shown in figure 22. This caused the clients to re-establish connection and hence re-generated ARP packets. The "Ctrl+c" command was used to stop the "deauth" attack after 1 minute.

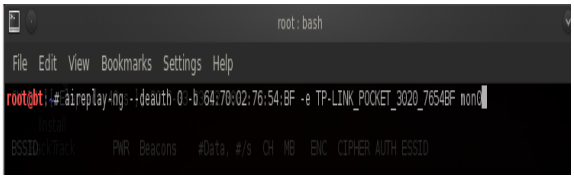

Figure 22:"aireplay-ng --deauth" command

The "aireplay-ng --arpreplay" command was repeated and ARP packets were successfully captured and replayed back into the network as shown in figure 23.

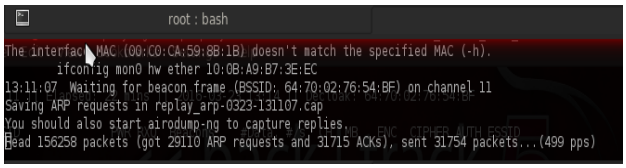

Figure 23:results of the "aireplay-ng –arpreplay" command

The "ls" command was used to locate all the saved .cap file as shown in figure 24.

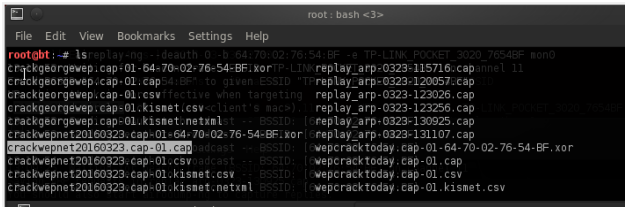

Figure 24:The results of the ls command

The more the WEP encrypted ARP packets generated between the legitimate AP and the attacker machine, the more weak IVs which have a correlation with the WEP secret key were also generated.

The "aircrack-ng" command together with the saved .cap file was used to attempt cracking the WEP key as shown in figure 25.

Figure 25:"aircrack-ng" command

The WEP key was successfully cracked in less than 5 minutes as shown in figure 26.

Figure 26: The successful cracking of the WEP Password

Analysis:

The WEP Key was successfully cracked due to the following reasons:

The WEP Key is either 64-bit or 128-bit long. Thus there are only 8 key-bytes for a 64-bit WEP key and 16 key-bytes for a 128-bit WEP key as shown in figure 27. The first 3 key-bytes are the IVs which are known because they are always sent in clear text.

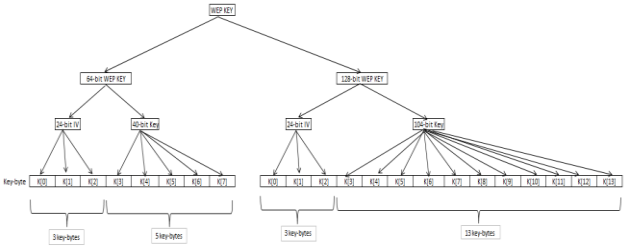

Figure 27: The 64-bit and 128-bit WEP key-bytes

During cracking of the WEP key, the first true key-byte (K[3]) was correctly guess before obtaining the remaining key bytes. This makes the attack statistical in nature as each weak IV gives about 5% chance of guessing the correct key-byte and 95% chance of guessing wrongly. However, by analyzing a large number of these weak IVs and the key bytes they reveal, a bias towards the true key bytes was expected. That is why an ARP packet was captured and replayed back into the network to generate more IVs containing these weak IVs. Each weak IV provides a statistical vote for each unknown key byte as shown in figure 28.

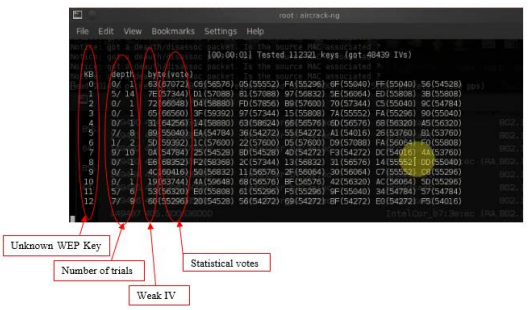

Figure 28: The "aircrack-ng" WEP cracking process

All the obtained weak IVs with their corresponding key-bytes were ranked based on their statistical votes from most probable key-byte to the least probable key-byte as shown in

figure 35. The correct key-bytes (except for the last key-byte) are displayed in the first column of figure 28; The numbers next to the key-bytes are the votes for these key-bytes, The numbers right to these values are the alternative candidates for the key-bytes and their votes.

The correct key is found by using a few IVs and the key to generate the corresponding PRGA. If the generated PRGA matches the ones returned by the captured packets, the key is assumed to be correct with a very high probability. If not, then at least one of the decisions for one of the key-bytes must have been incorrect. The attack now start looking for a decision for a key-byte that it suspects to be wrong. It could choose a decision where the difference in the number of votes between the most voted value for the key-byte and the second most voted value for the key-byte is minimal. The attacker now assumes that the correct key-byte is the second most voted one and continue the computation of the PRGA with the substitute key-byte. This is repeated until the correct key has been found or a time limit has been exceeded.

In this case, the WEP password was successfully cracked in less than 5 minutes after capturing 66,560 IVs and trying 541 possible keys as shown in figure 26.

5. Research Findings

1. WEP key cracking does not depend on the size of the key: It takes apparently the same time to crack a 64-bit and a 128-bit WEP key.
2. WEP key cracking depends on the number of weak IVs. It is suggested that between 60,000 to 70,000 IVs must be gathered for experimentation in which case a total of 66,560 used for this study was with 66,560 IVs.
3. It is faster to generate more weak IVs by capturing a gratuitous Address Resolution Protocol Packet (ARP) packet, modifying it for the same host, and injecting it back into the network. This will generate more ARP request and response messages with IVs and keystream bytes.
4. WEP key can also be cracked offline. After gathering enough packets with weak IVs from the network, the data can be written to a file and brute force attack can be conducted on it offline.

6. Conclusion

From this thesis work, it have proven that there are indeed vulnerabilities in Wired Equivalent Privacy (WEP) security protocol. These vulnerabilities can be easily exploited by an attacker to gain unauthorized access into a Wireless Local Area Networks that has been secured with WEP.

7. Recommendation

1) WEP uses static keys for both authentication and encryption. It is recommended that the keys be frequently changed to avoid key compromise.
2) Due to WEP's vulnerabilities in the use of short IV space and linear checksum which leads to message injection, modification, and keystream attacks; it is recommended for administrators to completely stop using WEP. WEP can be broken no matter the key length used.
3) Organizations should create and enforce wireless network security policies that address all the known vulnerabilities. Such policies should include which users are allowed to use the WLANs and what level of information is allowed to be transmitted over the WLANs.
4) Security assessments or audits are essential for checking the security posture of an organizations' WLAN infrastructure. It is important for organizations to perform regular audits of their WLANs to identify rogue APs and unauthorized access. Organizations can also outsource the audit to a third-party who have the tools and the technical expertise to do a more detailed penetration testing and fix all discovered issues.

8. Future work

Future work includes conducting further study into WPA/WPA-2 to identify if there are any vulnerabilities that can be used to compromise a WPA/ WPA-2 enabled network, finding vulnerabilities in WLANs as they are prone to attacks using Man-in-the Middle Attacks, Denial of Service Attacks, patching the flaws in the WEP security protocols of WLANS, investigating and the development of a robust and secured centralized management solution for large enterprises and also investigation into the pros and cons of Network Intrusion Detection Systems(NIDS).

References

[1] Chandra Shekar Gambiraoput, "Security Threats and Intrusion Detection Models in WLAN", January 2010.

[2] Thoetsak Jaiaree, "The Security Aspects of Wireless Local Area Network (WLAN)", September 2003.

[3] Nwabude Arinze Sunday, "Wireless Local Area Network:Security Risk Assessment and Countermeasures", August 2008.

[4] Abdul Qudoos Memon, Ali Hassan Raza and Sadia Iqbal, "WLAN Security", April 2010.

[5] Arbaugh W.A. "An Inductive chosen Plaintext Attack against WEP/WEP2".

[6] Rackley Steve, "Wireless Networking Technology. From Principles to Successful Implementation". May 2007.

[7] Biham E & Carmeli Y. "Efficient Reconstruction of RC4 Keys from Internal States". In Kaisa Nyberg, editor, FSE, volume 5086 of Lecture Notes in Computer Science, pages 270-288. Springer. November 2008.

[8] Chandra Praphul, Bensky Alan, Bradley Tony, Hurley Chris, Rackley Steve, Rittinghouse John, Ransome F.James, Stapko Timothy, Stefanek L.George, Thornton Frank, & Wilson John. "Wireless Security, Know it all". ISBN 978-1-85617-529-6. Elsevier Inc. May 2009.

[9] Vivek Ramachandran. "BackTrack 5 Wireless Penetration Testing. Biginners Guide". ISBN 978-1-849515-58-0. September 2011.

[10] Khan J & Khwaja A. "Building Secure Wireless Networks with 802.11". Canada. Wiley Publishing, Indianapolis, Indiana. June 2003.

[11] Fluhrer S.R, Mantin I, & Shamir A. "Weaknesses in the Key Scheduling Algorithm of RC4". In Serge Vaudenay and Amr M. Youssef, editors. Selected Areas in Cryptography 2001.Vol 2259 of Lecture Notes in Computer Sciences, pages 1-24. Springer. August 2001.

[12] Tanzella F."Wireless LAN Intrusion Detection & Protection" http://www.airdefense.net. August 2003.

[13] IEEE, IEEE 802.11 Standards documents. http://standards.ieee.ogr/wireless/ . January 2004.

[14] Peterson W.W & Brown D.T. "Cyclic Codes for Error Detection". Proceedings of the IRE. January 1961.

[15] Hutton D. "Practical Exploitation of RC4 Weakness in WEP Environments". Presented at Hiver in 2002. June 2002.

[16] Mantin Itsik. "A Practical Attack on the Fixed RC4 in the WEP Mode". In Bimal K. Roy, editor, ASIACRYPT. Volume 3788 of Lecture Notes in Computer Science. Pages 395-411. Springer. July 2005.

[17] Souradyuti P & Prencel B. "A New Weakness in the RC4 Keystream Generator and an Approach to Improve the Security of the Cipher". In Bimal K. Roy & Willi Meter, editors, FSE. Volume 3017 of Lecture Notes in Computer Science. Pages 245-259. Springer. July 2004.

[18] Borisov N, Goldberg I, & Wagner D. "Intercepting Mobile Communications: The Insecurity of 802.11". In Proc. ACM Mobicom, Rome, Italy. July 2001.

BIOGRAPHIES

Kwabena Akomea-Agyin is a cofounder NSA Consortium, a company that specializes in preventing and responding to network security incidents. Kwabena is a strong evangelist of security with about 8 years working experience in managing networking, security, and transmission portfolios within the telecommunication space. His research interest is in digital forensics and cyber security as a defensive tool to helping organizations and individuals achieve confidentiality, integrity, and availability over the intranet and internet. Kwabena holds an MSc in Information Technology from Kwame Nkrumah University of Science and Technology.

Michael Asante is a Senior Lecturer in Computer Science and holds a PhD in Systems Engineering from the University of Reading and also Master of Science in Scientific Computing and Information Technology from the London South Bank University, London all in the United Kingdom. His areas of Specialization are Data Communication, Computer Networks and Security and Distributed systems. Michael's research interests are in the areas of Network Protocols, Network Architectures, Wireless Networks, Mobile Internet Protocol Convergence and Ubiquitous Systems, Network Security, Distributed Systems, Computer System Security, Multimedia Compression Techniques and Standards, Wireless Multimedia Transmission, Cross- layer Optimization and Voice-over Internet Protocol.