Syeda Mahnur Asif
Emaan Hasan

Introduction

This report outlines an attack made on a target connected to a separate, password protected network. The aim is to take control of the target's browser. The attack is carried out in 3 steps. The first and foremost thing to do is to get on to the network that the target is on. For this we used the tool Aircrack-ng to capture the 4 way handshake and crack the password with a dictionary attack. Next steps include launching a Man-in-the-Middle (MITM) attack to spoof the traffic, this helped redirect the target to one of our own websites. Here social engineering techniques are used to persuade the target to click on a link generated by The Browser Exploitation Framework (BeEF). The link, once clicked on, hooks the target's browser and allows us to manipulate any vulnerabilities it might have.

Methodology

Step 1: Getting on to the Target's Network

For this, as previously mentioned, we use Aircrack-ng here. This is a suite which in itself contains many tools like Aireplay and Airodump. This suite comes pre-installed in Kali Linux and is known for network detail detection, packet sniffing, WEP (Wired Equivalent Privacy) and WPA-PSK (Wireless Protected Access – Pre-Shared Key) cracking, and analysis of wireless LANs. Any wireless NIC which has a driver that supports monitor mode can be used with this suite. Aircrack-ng is a branch of the original Aircrack project that was based on a new attack for a WEP vulnerability, developed by a team at Darmstadt University, Germany, in April 2007.

In this report, Aircrack-ng's password cracking capability is used to try and find out the password of the network we want to join. Our target's network uses WPA/WPA2-PSK security which is Protected Access 2. This uses Pre-Shared Key authentication. Here, the router is configured with a plaintext password. This passphrase along with the network 's service set identifier (SSID) is used to generate unique encryption keys for each client. This key is then known as Pairwise Master Key (PMK).

The 4-way handshake was introduced in WPA2 after patching the vulnerabilities present in WEP. This added security measure made the Access Point and the wireless client prove to each other that both parties knew the Master Key(PMK) without either of them sending it to the other. The handshake occurs so that a session key can be generated. A session key requires 2 Nonces (a random integer) PMK and a MAC address to be created.

A handshake consists of:

1. Access Point generates and sends ANonse to the client.
2. Client generates a Pairwise Transient Key (PTK) using the ANonce it received and the PMK it already has. Client then replies with a CNonse and its own MAC
3. Access Point then sends its MAC and a GTK (Group Temporal Key) to the client.
4. Client sends an ACK with its MAC.

A network which uses this handshake for authentication can have its password cracked with the use of the Message Integrity Code (MIC) in the fourth frame. This fourth frame consists of Key Confirmation Key (KCK) part of the PTK.

Sniffers can sniff out MACs, SSID and Nonces but not the PMK. Thus, to break this, the encryption cipher (AES/RC4) used underneath needs to be broken. A dictionary attack is used to achieve this.

The 4-way handshake crack occurs in the following outlined way:

1. The handshake is parsed to get the MAC address of the wireless client connecting to the Access Point, the Access Point's (authenticator) MAC, STA(Station) Nonces, EAPOL payload and the MIC from the fourth frame. EAPOL-Key packets are used by WPA to distribute session keys to the authenticated station.
2. Using a dictionary, a candidate password is used to generate PMK (Pairwise Master Key, a session authorization token)
3. Pairwise Transient Key (PTK) is computed from PMK, AP, Station MAC address and nonces.
4. KCK from the computed PTK is then used to calculate MIC of the EAPoL. If this calculated MIC matches the MIC parsed from the captured handshake obtained in the first step, then the password has been successfully.

How Aircrack was used

1. The "**airmon-ng check kill**" command kills any processes that might interfere with the normal functioning of aircrack-ng. Network Managers and their related processes like wpa_supplicant are turned off by this.
2. "**airmon-ng start wlan0**" puts the NIC in monitor mode. Here wlan0 is the network interface name. This creates a monitor mode interface callled wlan0mon.
3. A tool in this suite, called airodump is now used. "**airodump-ng wlan0mon**" command is used to show all the wireless networks in the area. By this command, the networks' BSSID (Basic Service Set Identifier, the MAC address of the AP) , the channel it's on, and it's SSID is also revealed. Select and note the BSSID of the target network here for the next steps.
4. "**airodump-ng -c <channel number> --bssid <BSSID of the target network> -w WPAcrack wlan0mon**" focuses on the target network and shows the list of hosts connected to it and their MAC addresses. WPAcrack is the name we give to the file that will save the handshake. This file is automatically created in usually /home or /root (if not specified) when a handshake is caught.

Optional step:
Since connected devices have already authenticated themselves to the AP when originally connecting to it, there can be a possibility of no handshake capture. A handshake is caught when a device is authenticating itself to the AP. Thus, there is choice between waiting for any already connected device to either disconnect and then reconnect, or wait for a new device to connect to the network. Instead of the latter choice, the first is chosen, but waiting for disconnection and reconnection is not done. Here, the attacker, can bump any of the connected devices off the network by forcibly de-authenticating them, which will then make them request reconnection to the AP during which the handshake can be captured.
For de-authenticating, another terminal instance is opened. "**sudo aireplay-ng -0 <number of packets> -a <BSSID of AP> -c <any host's MAC address> wlan0mon**" is used to send a specified number of deauth packets.. At re-connection, the handshake is successfully captured.

When a handshake is captured it is displayed in the terminal. By this point all external work is complete. The WPAcrack file is located, along with the file path to the dictionary that will be used for the dictionary attack.

5. “**aircrack-ng -a2 -b <BSSID of AP> -w <file path of the dictionary> <file path of for the saved handshake>**.”This starts the dictionary attack. The time can vary widely between a few seconds and hundreds of years to it not working at all depending on the dictionary being used, its size, and whether it contains the password used to protect the network.

Assuming the dictionary attack was successful, the password of the network is cracked and now we can connect to the network where our target is by simply entering the password.

Step 2: Man in the Middle Attack with Ettercap

A MITM attack is essentially when a third party comes in between the communication of two parties. This is usually a silent attack and both parties believe they are communicating privately and remain unaware of this intrusion on the channel. The third party could be simply eavesdropping, that is capturing information being transmitted, or could be altering the information exchanged.

MITM is also known as Address Resolution Protocol (ARP) poisoning. ARP is a network layer protocol that is used to resolve the link layer address associated with a given IPv4 address. The device that needs to obtain the link layer address sends out a broadcast request onto the network, requesting the resolution of an IPv4 address. The host with the IPv4 address then replies with its link layer address which the device stores in a table. The vulnerability here is that ARP, unlike DNS, will accept updates to its ARP table at any time. Hence, an attacker can send the source device an ARP reply and force it to change its ARP table entry to make a link layer address resolve to a different IPv4 address. An entry explicitly marked permanent however cannot be changed.

ARP poisoning, is also referred to as ARP cache poisoning or ARP poison routing. In this, the ARP cache of two devices is “poisoned” with the MAC address of our own NIC card. A fake ARP reply in response to the request for a physical address corresponding to a legitimate IPv4 address on the network. The fake reply(gratuitous ARP reply) has our own MAC address. Now the device that requested the physical address, sends all its communication to our machine instead of the machine which actually has the IPv4 address required assigned to it. We can now monitor and also alter any information between the two devices.

Using Ettercap

The tool here used for ARP poisoning is called Ettercap. According to the official website, *“Ettercap is a comprehensive suite for man in the middle attacks. It features sniffing of live connections, content filtering on the fly and many other interesting tricks.”* It is mainly used for network analysis and security auditing.

To ARP poison via Ettercap, we first start Unified scanning on the network. Ettercap supports two kinds of scanning unified and bridged. In unified scanning, Ettercap sniffs all the packets passing on the network and forwards the ones not directed to the host it is running on, using layer 3 routing.

Bridged sniffing on the other hand, uses two network interfaces and forwards data from one network interface to the other while sniffing and content filtering.

Ettercap builds a lists of hosts on the network while sniffing. From this list, the two devices that we want to Man-in-the-middle in can be selected. In our case, we choose the host we want to eventually poison as 'Target 1' and add the default gateway as 'Target 2.' Having done so, we ARP poison the connections, by selecting the 'sniff remote connections' option in Ettercap. MITM attack has now been successfully executed and we are now in the middle of the communication between the target and the Access Point. At this point, if our target was using a particularly secure browser such as the latest version of Mozilla Firefox or Opera, they would have received a security warning and would not have been allowed to access HTTPS IPs. However, lesser browsers such as Internet Explorer, while giving some warning do not stop the user from browsing.

Having successfully ARP poisoned our victim machine, we now have to DNS spoof it in order to redirect the target browser to an Internet link of our choice. DNS spoofing, as the name suggests, exploits vulnerabilities in the Domain Name System. The DNS is a way to convert human readable urls such as www.google.com into machine readable IP addresses. When a user wishes to visit www.google.com, a request is sent to the DNS servers to resolve the URL into an IP address. The DNS servers form a layered, hierarchal system themselves, with the user’s router serving as one layer, the user’s Internet Service Provider as the next and so on. These layers, as well as the user’s own computer, maintain a cache of DNS lookups already performed for easy and rapid access. DNS spoofing works by sending the wrong IP address in return of a lookup by the target host, taking them to a different page, or one to one of our own. The average Internet user should have no idea that this has occurred.

In Ettercap, dns_spoof is a plug-in that enables a host already in the middle of a communication between two other hosts on the network, to spoof any DNS look up. Before enabling the plug-in, we go into the etter.dns file in the Ettercap directory and enter the URL we want to spoof and the IP it needs to be spoofed to. Having saved this, the plug-in can now be enabled. Another plug-in to be enabled is the auto-add one, which “*will automatically add new victims to the ARP poisoning MITM attack when they come up. It looks for ARP requests on the LAN and when detected it will add the host to the victims list if it was specified in the TARGET. The host is added when an ARP request is seen from it.*” Having enabled both these plug-in, the DNS specified has now been spoofed.

Step 3: Hooking the Target's browser with the Browser Exploitation Framework (BeEF)

We now have the capability to take the victim to which ever link we require, enabling us to hook their browser and gain access to it and to the other functionalities connected to it. For this purpose we now utilize the Browser Exploitation Framework (BeEF). BeEF is a penetration testing framework specializing in detecting and exploiting browser vulnerabilities. By exploiting vulnerabilities, BeEF takes over a victim’s current session using modules and payloads.

BeEF generates a link that once clicked on hooks the browser which was used to access that link. We now need to persuade the target user to click on that link. We embedded the link into a page we send to our user via DNS spoofing. Once the target clicks on the embedded link the target’s browser is taken to a webpage, stored in our (the attacker’s) computer. That link should ideally be such that keeps the victim onto the site for the longest amount of time. Since the only thing keeping the connection open is the tab in which the web page with the hook has just opened. If that tab closes, the connection to the target is lost. There are some built-in “Persistence” commands in BeEF such as Man-in-the-browser and the Pop-Up option that serve to rectify this problem. Man-in-the-browser makes every link accessed from the original hooker page such that the hook isn’t lost. So unless the target user manually erases the hooker URL and types a different one, it stays hooked. Pop-Up opens pop-up windows that, if passed unnoticed, will keep the victim user hooked even if the tabs are closed. Pop-Up is however easily stopped if the victim has an advanced pop blocker. Another way to keep the user on for longer are Social Engineering techniques. If we run Wireshark after getting onto the network we can see the traffic on the network. It also shows us what web pages the victim uses most often. Armed with this information, we can alter our BeEF hooker page and also the page we DNS spoof our victim towards. We can alter it to look like a fake copy of the website. This would firstly not alarm the user to there being any interference and secondly would keep them on longer, or make them click a link that is supposed to be there anyway. If the Man in the browser command is running this click will keep the victim hooked.

Now that the target is hooked, there are many commands that BeEF allows us to execute in the target’s browser. The commands are divided into many sub folders such as Browser, Networking, Persistence, Plug Ins, Miscellaneous etc. Each command exploits some vulnerability the browser might have. In our case, we choose to access the victim’s webcam through the “Webcam” command in the Browser subsection of the Command tab. The vulnerability exploited here is the Adobe Flash which enables us to take pictures off of the webcam. A warning does appear but if this is worded cleverly enough, the victim can be tricked into accepting the prompt that displays in lesser browsers such as Internet Explorer.