Milestone 3: Methodology

Omkar Parkhe, Nida Safia Syed, Sri Sai Bhargav Tiruveedhula

In this section, we describe the implementation of a captive portal which comprises of its installation and configuration. We also perform a security analysis by carrying out various attacks including Man-in-The-Middle, DNS Tunneling, and HTTP header injection.

I Experimental Setup

A Resources Used

The following resources are used to perform the security analysis of ChilliSpot captive portal.

A.1 Software:

Ubuntu 16.04, ChilliSpot, FreeRADIUS 3.0, Apache2, Kali Linux, MySQL, PHP7, Firestarter.

A.2 Hardware:

HP laptop, NETGEAR router.

B Selection of the Captive Portal

We use ChilliSpot, an open source captive portal to provide users of a wireless LAN with Internet access. We chose ChilliSpot primarily because of its stability, portability and scalability. It is compliant with Linux, additionally, it is also portable to other platforms. It also supports concurrency in order to improve portability. A client process is created whenever an HTTP authentication request from a client is received, thereby achieving high throughput. Memory is handled conservatively along with error checking, which helps improve stability but at the cost of performance, which can be optimized at a later stage.

Moreover, all other active open source captive portal projects are fully based on ChilliSpot, and even if that is not the case, they use the same enforcement mechanisms as ChilliSpot.

C ChilliSpot Captive Portal

Apart from the ChilliSpot software, the Radius server is hosted on the same PC. We have five main components in ChilliSpot Captive Portal [1] as illustrated in figure 1:

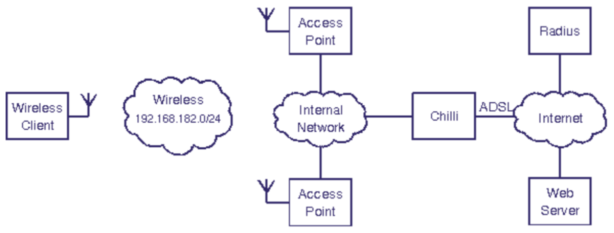

Figure 1: Architecture of a WLAN using ChilliSpot

1. *Chilli:*

   The Chilli software supports the Universal Access Method (UAM) of authentication. When a wireless client requests for an IP address, it is allocated by Chilli using UAM. When the user starts a web browser, Chilli captures the TCP connection and redirects it to the captive portal, where the client is either requested for login credentials or made to accept the terms and policies of the organization. Chilli then forwards the authentication request to a Radius Server. On successful authentication, the Radius Server sends an Access Accept message back to Chilli else an Access Reject message is sent back.

2. *Authentication Web Server:*

   This server is needed to authenticate the users. We used the Apache2 Web Server for this purpose.

   ChilliSpot provides a CGI script for the Web Server which either requests the user for his username and password or asks him to accept the terms and conditions of the associated organization. The credentials are encrypted with the UAM secret key and sent back to Chilli, which forwards the request to the Radius Server.

3. *Radius Server:*

   We used an open source radius server, FreeRADIUS.

4. *Access Points:*

   We used a NETGEAR N600 Dual Band Wi-Fi Router with a data transfer rate of 300 Mbps and 802.11 a/b/g/n wireless compatibility.

5. *Wireless Client:*

   The wireless clients used were Android phones and laptops with Windows and Linux operating systems.

D Captive Portal Configuration:

ChilliSpot creates a VPN or Virtual Private Network (192.168.0/24) by default. It manages the allocation of dynamic IP addresses to clients, so there is no requirement of additional DHCP tools. As ChilliSpot creates a VPN, we need to check whether our Linux kernel supports this.

The installation of ChilliSpot was followed by that of FreeRADIUS 3.0, Apache2, MySql, PHP7 and Firestarter. FreeRADIUS is the most popular open source Radius server, which supports all the common authentication protocols. FireStarter is a complete firewall tool for Linux machines that uses the Netfilter system.

For the successful setup of a captive portal with ChilliSpot, we did the following [2]:

1. Installation of ChilliSpot and FreeRADIUS.
2. Modification of the chilli.conf to configure the Radius servers, login interface and the DNS server.
3. Modification of apache2-hotspotlogin.cgi to specify the browser redirection and manage the actual login.
4. Modification of clients.conf, radius.conf and radiusd.conf to configure FreeRADIUS and MySQL for authentication and accounting.
5. Creation of a Radius database in MySQL.
6. Configuration of FireStarter and creation of the SSL certificate.

II Security Analysis

A Problems with Default Configuration

Many organizations deploy their captive portals with the default configuration. This leads to several problems as elucidated below:

a) A malicious user knows the secret key used for encryption as it is mentioned in the ”uamsecret” (shared secret between the client and server) value. This allows him to decrypt all the data encrypted with this key.

b) The attacker also knows the password for the Radius server, which gives him the authority to manipulate users’ credentials.

c) Default IP address of the router is 192.168.1.1 and that of ChilliSpot is 192.168.182.0/24. If a malicious user changes his IP address to 192.168.2.x, he can bypass the captive portal and access the Internet. The problem here is that, by default, the firewall rules are empty, therefore all the requests are not redirected to the captive portal.

d) ChilliSpot uses a challenge-response protocol, where the server sends a challenge to the client. Based on the challenge and the password entered by the user, the client calculates a response and sends this back to the authentication server. The server performs the same operation as the client and compares it with the client response. If they match, the user is authenticated. A one-way hash function, such as MD5 or SHA1, is used to calculate the response. As these algorithms are already broken, an attacker can come up with new data having the same hash value. Moreover, the challenge/response method is vulnerable to dictionary attacks.

e) Some organizations use ChilliSpot MAC authentication to authenticate their clients. However, this method is vulnerable to MAC spoofing.

f) Some organizations use less than 16 characters for uamsecret. This makes it vulnerable to password guessing attacks.

B Attacks Performed

We perform the attacks described below on our testbed described in section I with the goal of bypassing the captive portal in order to gain free network access [3] [4].

B.1 DNS Tunneling

First, we find the network IP address by sniffing the network traffic using a sniffing tool such as Wireshark, dsniff, tcpdump, etc. In order to sniff the surrounding wireless traffic we use the device’s NIC (Network Interface Controller) in monitor or promiscuous mode. We then find open ports on the network i.e. the ports which accept connections and allow traffic through the gateway. This can be done by performing a port scan using Nmap.

Most of the networks have the UDP/53 port open, which allows all users, authenticated and unauthenticated, to send and receive the DNS requests and responses through the network. We can leverage this to our benefit by tunneling the DNS packets using Iodine [5] to gain unauthorized network access, as follows:

1. We set up a DNS server with a public address (eg. 10.0.0.1), with the help of freedns.afraid.org website. It enables us to create valid DNS records of type ’A’ and ’NS’ for a minimal charge. Now, as we have a valid DNS entry and a public address for our Iodine server, we can run the server using the command, *./iodined -f -c -P <secretpassword> <secrettunnelip> thisis.mydomain.com*. The secret password is used to identify valid DNS tunneling

clients whereas the secret tunnel's IP address is used by the tunnel clients to identify the tunneling server.

2. A client is started by executing the command, *./iodine -f -P <secretpassword> <thisis.mydomain.com>*. After the command executes, the client will have a fixed DNS0 address which is the address of the DNS tunneling server. From this point onwards, the client's DNS requests are forwarded to the DNS tunneling server. The client can verify this by pinging the tunneling server's public address.

3. The command *ssh root@10.0.0.1 -D 8080* creates an SSH tunnel to the DNS tunneling server and a socks proxy on port 8080. Finally, a secure DNS tunnel is established and we gain unauthorized network access by setting a socks proxy in the browser to 127.0.0.1:8080(localhost).

B.2 MiTM using ARP Spoofing and Poisoning

We can sniff the MAC and IP addresses of the authenticated users and the network server present on the network, using sniffing tools such as Wireshark, tcpdump, DSniff, etc. We perform MiTM attack using ARP Spoofing and Poisoning [6] as follows:

1. First, the attacking device impersonates the network server and informs its identity to a victim client, who has already authenticated itself, using the command, *arpspoof <server address> <victim address>*. The victim then tries to connect to this spoofed server.

2. Simultaneously, the attacking device impersonates the victim client, who would like to connect to the network server using the command, *arpspoof <victim address> <server address>*. Both the above mentioned commands poison the ARP tables of the server and the victim by asking them to change the MAC addresses of the victim client and server to attacking device's MAC address.

3. There is a need to communicate with the victim client and the server, as we impersonate both. This can be done by forwarding the packets received from the victim client to the server and vice versa using the command, *echo 1 > /proc/sys/net/ipv4/ipforward*. We can now obtain the victim client's credentials by sniffing the network traffic.

B.3 HTTP Header Injection

It is often the case that networks use HTTP proxies. Many times, Access Control Lists on these proxies are set up only to block basic HTTP requests. To check if there is an open proxy available in the network, we use the Nmap command, *sudo nmap -sS -sV -p 8080 –script http-open-proxy.nse <network_gateway_ip>*. On identifying one, we can bypass the captive portal as follows:

1. HTTP Header Injection or Response Splitting on captive portals can be performed easily using Burp suite. First, we open the captive portal and intercept all the outgoing traffic using Burp suite.

2. In Burp suite, we can identify the sections which can be tampered and add our payload to said sections, using which we can bypass the captive portal. Burp suite provides a variety of intruder payloads, including SQL injection, XSS injection and many more, which can be used to test the security of captive portals.

3. Once we find a response for which some vulnerability of the captive portal is successfully exploited, we can use that payload to gain unauthorized network access.

Some captive portals can also be bypassed using CLRF injection [7] which includes modifying the CRLF to LF in HTTP request.

References

[1] *Chillispot-Open Captive Portal*, 2007. http://www.chillispot.org/.

[2] *WiFi Docs Chillispot-HotSpot*, 2017. https://help.ubuntu.com/community/WifiDocs/Chillispot Hotspot.

[3] *Captive Portal Security part 1*, 2012. http://opensourceandhackystuff.blogspot.com/2012/-02/ captive-portal-security-part-1.html.

[4] Marc Laliberte. *Lessons from DEFCON 2016 Bypassing Captive Portals*, 2016. https://www.secplicity.org/2016/08/26/lessonsdefcon-2016bypassing-captive-portals/.

[5] *Iodine*, 2014. http://code.kryo.se/iodine/.

[6] *ARP cache poisoning / ARP spoofing*, 2003. https://su2.info/doc/arpspoof.php.

[7] *Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting')*, 2017. https://cwe.mitre.org/data/definitions/113.html.