Project description:

I am a security analyst working for a company that wants to detect certain network packets on the server, from several protocols. The company wants to test the access to an internal database which is using the RADIUS protocol at the moment, and examine just how safe are websites that uses HTTP with Basic Auth or form-based verification. Finally, as a system administrator: is there a way to see encrypted traffic on our own network (HTTPS)? I am tasked to looking into these things.

## Objective 1 – capture network traffic and view captured traffic through a display filter

Captured traffic had been stored in a pcap file generated in the tool Wireshark. I viewed it. The network protocol of interest was RADIUS. Remote Authentication Dial-In User Services is used for authentication of users logging in to a central database. I played around a bit by submitting bogus passwords to access the central database just to see what the response will be. RADIUS is not secure at all as can viewed in the attached file below.

See also: RADIUS.png

In addition simple capture of ongoing network was used, using the ethernet Applying DNS as a display filter which displayed the DNS traffic. I used ethernet as the interface of interest.

**See also:** **Capturing from ethernet 3.png**

**By utilizing the display filter, the DNS traffic could be filtered out. As shown, DNS traffic is not encrypted.**

**See also: Early DNS traffic.png**

## Objective 2 – Generate and capture RADIUS traffic

**I utilized a public RADIUS server and submitted a username and a password. The RADIUS will then store those two things.**

**See also: Public Radius server**

**In wireshark, to ultiize the capture filter even if one knows the interesting protocol one cannot simply type “RADIUS” there because it is not a display filter. But RADIUS uses the 1812 port, so we can use that to capture the traffic of interest to us. I used a client to the RADIUS server, to send information to it. That traffic will be captured by wireshark.**

**See also: NTRadPing access accept.png**

**I sent the correct username, but different passwords to the RADIUS server using the client. In the attached file below you can see different captured responses from the RADIUS server.**

**See also: Capturing from port 1812 - different responses.png**

**As can be seen, RADIUS is not encrypted at all.**

**See also: Capturing from port 1812 - username clearly visible.png**

**However, the password is not shown in Wireshark. RADIUS uses the field “secret key” and the MD5 hashing algoritm to hide the password behind the server and the client.**

**Wirehsark can be used to decrypt such a weak protected password.**

Edit -> Preferences -> Protocols -> RADIUS**.**

**Shared secret, which can be found in the client.**

**See also: Capturing from port 1812 - password with known secret.png**

## Objective 3 – Analyze a HTTP Basic Authentication

**HTTP is a clear text protocol. A display filter was used, and not a capture filter. Specifically port 80 to capture only HTTP traffic.**

**A HTTP website was accessed, where I was prompted to type my credentials.**

**See also: Http login.png**

**As can be seen this is not secure since Wireshark can see exactly was transmitted.**

**See also: Http login in Wireshark.png**

**Upon a closer look, the credentials can be directly seen in Wireshark.**

**See also: Http credentials.png**

## Objective 4 – Http Form-Based Authentication and DNS

Form based HTTP authentication was created to negate the downsides of basic authentication. It does not send a GET request with the credentials, instead it sends a POST request and passes the credentials with it. Unlike basic authentication, Wireshark will find the credentials in the HTML field.

See also: Display captured packets for HTTP traffic.png

Normally, this would be within TLS, so it does not matter if it is not encrypted. But now the credentials are shown in the form field instead.

DNS traffic, utilizing port 53, are not encrypted either as the attached file below demonstrates.

See also: Capturing DNS traffic with port 53

## Objective 5 – Initiate, Capture and Analyze Telnet Sessions

Telnet is unencrypted, it’s secure form is SSH. Telnet is a computer protocol built to interacting with remote devices, to access and manage a device.

Telnet uses port 23, and I began to set Wireshark to capture packets from that port.

PowerShell was used to generate telnet traffic with the following command: telnet tty.sdf.org

InsideWireshark, rightclick on the first telnet package, choose Follow -> TCP Stream.

See also: Wireshark follow TCP stream.png

It clearly shows that telnet is not secure at all.

## Objective 6 – Capturing and analyzing SSH Sessions

Both telnet and SSH are both used to remotely access a device, but SSH is encrypted and uses port 22. To secure the communication, SSH uses the security protocol TLS.

I demonstrated the differences between captured traffic using SSH relative to telnet. A good way to do would be to use host tty.sdf.org as capture filter. PowerShell can then be use to directly access the site: telnet tty.sdf.org and ssh tty.sdf.org respectively.

Back in Wireshark, there is several conversations going on since the same website was accessed with both telnet and SSH.

Statistics -> Conversations

See also: Conversations capturing both telnet and ssl.png

Clicking in Follow Stream.. shows what any man in middle would see in respective case.

See also: Telnet traffic is not encrypted.png

See also: SSH traffic is encrypted.png

## Objective 7 – Generate, Capture, Analyze and the Decrypt HTTPS Traffic

HTTP/S is just HTTP over TLS. HTTP operates over port 80 while HTTPS operates over port 443.

Too see encrypted traffic (HTTP/TLC) is not possible since that require a private key. But seeing encrypted traffic from one’s own machine is possible when using premaster secret key.

The browser (the client) uses this premaster secret key when generating the master secret key to encrypt all shown content. If it is set locally, in the form of a SSL key log file, Wireshark will be able to decrypt the traffic between that local machine and the server.

Start -> Environment variables -> [System Properties -> Advanced] -> User variables for Administrator -> New

Then I typed SSLKEYLOGFILE in the field Variable name. In the field Variable name I typed the location of the file: C:\Users\Administrator\Desktop\ssl-keys.log

Back in Wireshark: Edit -> Preferences -> Protocols -> TLS

In the field (Pre)-Master Secret log file name field I typed in the address: C:\Users\Administrator\Desktop\ssl-keys.log

To view the conversiona: Statistics -> Conversations

Clicking in Follow Stream.. shows what any man in middle would see in respective case.

And as you can see in the attached file, it is now decrypted.

See also: Unencrypted HTTPS stream.png.

Even if you go back to where we see the packets it is more clear.

See also: Clearer packets.png

Summery:

In this project I went through both HTTP and HTTPS, as well as both Telnet and its more secure twin SSH and compared them to determine which is more secure. And even HTTPS can under certain circumstances be decrypted to great relief our systems administrator. I also had a look at RADIUS, an older protocol used for accessing a common database.