MODULE 1 Analyzing Network Protocols With Wireshark

Packet that Matters:

Best way to begin capturing network traffic is collecting our analysis on a span mirror port on a switch near the client.

This will help to keep the traffic flowing and to avoid the traffic being captured by the switch hence making it smaller and easier to analyze.

Another way to create a smaller haystack is to filter the captured traffic to only the traffic that is relevant to the analysis.

We create custom columns, protocols, and custom filters to help us analyze the traffic.

Also saving protocol filterings as button for the quick access to those filters.

Core Protocols

Under the application data there are :

- UDP
- TCP
- IPV4
- IPV6
- ARP
- ICMP
- DNS
- TLS

Custom Profiles

Right click on down right on profile and select "New" and then create the custom profile.

After creating a new profile: go to edit and preferences to start customizing the profile.

Right of the bat: adding Delta time to the profile as title `delta` and type `Delta time displayed` to show the time difference between the packets.

Another way is to right click on any value down and click `add as column` to add the value as a column.

Creating and Saving buttons.

There are some filters that you keep using but you dont want to keep writing them. For example `tcp.flags.syn==1` which is the syn=1 flag where the communication starts.

in order to add it as button, I write this on the filter box and click on `+` to add it as a button.

it is also important to colorize the packets in order to create the visibility. view==>colouring rules, then write the filter you want to filter (like regex) and then do not forget to enable it. Also change the importance level by dragging up and down in the list.

PROTOCOLS IN DETAIL

1- ARP

NOTE THAT ARP does not resolve IPV6 addresses. Meaning IPv6 does not use ARP. IPV6 uses NDP(Neighbor Discovery Protocol) to resolve the addresses which replaces ARP.

Address Resolution Protocol (ARP) is a protocol used to resolve the MAC address of a host. meaning, it bridges the gap between layer 2 and layer 3.

ARP ne ise yarar nasil calisir?

Bir packet gonderecegin zaman, header olusturmak icin destination IP ve MAC address ihtiyacin var. sen baslangicta kendi IP ve mac adresini biliyorsun, bir de serverinkini.

Target'in MAC adresini bulmak icin ARP kullanilir. ilk once local arp cache'e bakar. eger varsa direkt cevap verir. eger yoksa `arp request` gonderir. Networke bunu broadcast olarak gonderir.

Bu broadcast domain icindeki devices will take this up, will check the IP address that is being resolved and will build and send a reply with its own mac address as the source mac address.

Yani aslida soyle:

18 numarali bilet kimde ogrenmek istiyorsun. Elindeki listede varsa zaten sorun yok, yoksa ortaliga bagiriyorsun `18 numarali bilet kimdee` yani `what is the mac address for this IP address`?

Herkes biletine bakip `benimki x`, `benimki Y` diye cevap veriyor.

Sen de sonunda ogrenmis oluyorsun ve header'i yaratip gonderiyorsun.

==> Arp requests are broadcasted to the network but responses are unicasted to the sender.

You check the ARP protocol if you see:

- Problems connecting to an application
- intermittent connectivity.
- Unicast flooding.

If the destination is visible (like Apple or Google,) it means that the destionation was on the Arp Cache already so it is not broadcasted but unicasted.

Hands on Demo:

**Lets disect a ARP request: **

```

```

```
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: BelkinIn_9d:02:73 (94:10:3e:9d:02:73)
    Sender IP address: 192.168.10.1
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.10.108
```

- Hardware type = Ethernet
- Protocol type = IPv4 Meaanign trying to resolve a mac address to IP address.
- Opcode = 1 meaning request
- Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00) so this is the information that is missing and being looked for. 00:00:00... means that the target mac address is not known.
- Target IP address: 192.168.10.108 is the IP address that is being looked for.

Now lets look at the ARP Response to this request:

```
Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: Apple_e7:ce:6d (a4:5e:60:e7:ce:6d)
    Sender IP address: 192.168.10.108
    Target MAC address: BelkinIn_9d:02:73 (94:10:3e:9d:02:73)
    Target IP address: 192.168.10.1
```

- Opcode = 2 meaning reply
- Sender MAC address: Apple_e7:ce:6d (a4:5e:60:e7:ce:6d) so this is the mac address of the sender which was missing.
- Sender IP address: 192.168.10.108

As you can see, sender and target IP addresses match, so the ARP reply is valid and MAC address is resolved.

In some cases we see some weird ARP requests that sends requests for entire network. like couple hundred or maybe more limited or less limited numbers of ARP requests.

This `could be` and indicator to an attack.

The first thing to do is to check the IP origin of the ARP request. For that end, I will create a ARP profile in the wireshark.

as a new coulumn, I will add the `opcode` field which will filter requests and responds. that would be interesting in this case to see if there is any active subnets and actualy any response.

Here is the filter : `arp.opcode==2` This will show all the packets that responded to the requests.

Another very interesting filter is `arp.isgratuitous` which means that we are looking for arps that are gratuitous either as requests or replies. Grattuitous means that the sender and target mac addresses are the same. Meaning sending its own IP and Mac address which means advertising itself.

Once we know that all is fine, we can remove any protocol that is not needed in the trace. For a protocol to be removed we use `!{protocolName}` like `!arp`.