|  |  |  |
| --- | --- | --- |
| Network Analysis Skills | | |
|  |  |  |
|  | | |

# Description:

**S0184**: Skill in analyzing traffic to identify network devices.

**S0236**: Skill in identifying the devices that work at each level of protocol models.

**S0245**: Skill in navigating network visualization software.

**S0258:** Skill in recognizing and interpreting malicious network activity in traffic.

**S0199:** Skill in creating and extracting important information from packet captures.

To verify the skills listed above, I successfully completed a 50 point challenge on HackTheBox involving an example network intrusion scenario. The prompt was as follows:

**The security team was alerted to suspicous network activity from a production web server.  
Can you determine if any data was stolen and what it was?**

Included in the challenge were the following files:

* bundle.pem
* cap.pcapng
* secrets.log
* conn.log, dns.log, files.log, ssl.log, http.log, packet\_filter.log, weird.log

**Skills Required:**

* Wireshark
* BASH
* Networking Protocols
* Network Security

# Solving the Challenge:

First things first when given a PCAP analysis is to open it up in wireshark and do some of the basics to try to gain some information and filter out some of the easier parts.

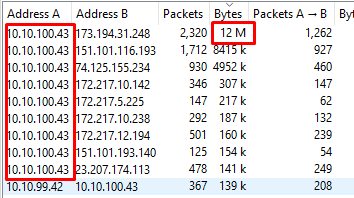
A screenshot of a cell phone

Description automatically generated

This capture file is quite large, and contains a lot of information that isn’t relevant to finding out what information was taken (the flag). So the first thing we’ll do is apply a filter to filter out ARP and DNS traffic, because as of now they’re not useful. Quickly its easy to notice that some of the traffic is encrypted, which makes sense since we were given a PEM file and an SSL log file.

This is again verified when looking at the Protocol Hierarchy (Statistics -> Protocol Hierarchy) in Wireshark. We can see that ~30% of the traffic is encrypted using TLS, so when we find the

A screenshot of a social media post

Description automatically generatedinformation that is stolen, we’ll most likely have to decrypt it using the PEM file given. Taking a look in the Conversations (Statistics -> Conversations) tab we can determine what devices are chatting the most, and with who. As we can see on the left, 10.10.100.43 is talking a lot, way more than any other host, so we can assume this is a regular user on the network. This is verified by looking at the type of traffic which is HTTP, and after removing the DNS filter we see that there are many requests to sites like Youtube, Reddit, etc.

A screenshot of a cell phone

Description automatically generated

To get a little more information on the details of each host, and their traffic, OS, etc. we open up the PCAP in Network Miner. We can see that on the 10.10.20.43 which is our chatty machine is running windows, and that we have a mysql server running on 10.10.20.14, which is where I would look first. Filtering the wireshark traffic for 10.10.20.14 reveals nothing, so we’ll have to try some other methods. To see if we can find anything interesting about communication between this address, we’ll do a search for the hostname.

Running the command ‘strings chalcap.pcapng | grep mysql-ml’ reveals a lot of interesting information. We can see that a user initiated a ping request to the SQL server and got a response, we can assume the attacker is on the 10.10.20.0/24 range which leaves us either ’10.10.20.1’ or ’10.10.20.13’ and we know that the first address is the gateway address, so 10.10.20.13 is most likely our attacker. We can also see that exfildb.sh was ran as the root user on the SQL server, so this is where our data was stolen.

A screen shot of a computer

Description automatically generated

Filtering out the Wireshark traffic for all communication between this address reveals a lot of encrypted TCP communication through TLS (443).

A screenshot of a computer

Description automatically generated

This is where the exfildb.sh command was ran, and it was indeed issued by the machine on 10.10.20.13. We can see that the details of the database were written to a file called dbdump, and the user initiated some POST requests to pastebin with the contents of /etc/passwd, /etc/shadow, and the dumpdb file.

We can see that the request to pastebin created a paste at <https://pastebin.com/dsDhyVZU>, and there is a unique URL for each of the files that were stolen. Navigating to that URL reveals that it no longer exists, however we can see in the request parameters that ‘api\_paste\_private=2’ so my first thought is that is where the flag is, and we cannot see it because it is private.

A screenshot of a cell phone

Description automatically generated

After a little reading on the pastebin API, there is an endpoint (api/api\_raw.php) that given an api\_user\_key, api\_dev\_key, and a valid paste\_id we can retrieve the contents of a paste. We have all of the attackers information, and we have the paste\_id where the data was sent, so we can retrieve the information.

A picture containing object, indoor

Description automatically generated

The script above contains variables for all the parameters needed in the POST request, and the paste\_id’s for each of the pastes sent by the attacker. It runs through each of the ID’s in a loop, and initiates a request to the api/api\_raw endpoint to retrieve the paste information. However we are given the error (Invalid Request, bad API\_USER\_KEY). After some trial and error with testing out different user keys, making a new account for a valid dev key, etc. I determined this is not the way to retrieve the information.

A close up of a screen

Description automatically generated

Instead, we can just follow the message when it was sent to pastebin, however it is encrypted with TLS, and unreadable as is.

A screenshot of a computer screen

Description automatically generated

Since we were given the SSL private key, and the SSL log, we can apply that to wireshark to have the TLS traffic decrypted.

A screenshot of a cell phone

Description automatically generated

Uploading those two files and instructing Wireshark to apply the decryption to the Attacker’s address, and the MySQL server address on ports 443, 80 we can see the decrypted traffic now.A screenshot of a cell phone

Description automatically generated

A close up of a newspaper

Description automatically generatedThis is the decrypted traffic for the /etc/passwd file that was sent off to pastebin. After going through each of the pastes we can find the ’dbdump’ paste, decrypt it, and the flag should be in there.

Put wireshark filters on here