|  |
| --- |
|  |
| Capstone Experience  IST 894 |
| Lab 7 – Network Forensics  Scott Finlon |

Table of Contents

[1. General Context 3](#_Toc85145533)

[2. Technical Context 4](#_Toc85145534)

[3. Solution 6](#_Toc85145535)

[References 15](#_Toc85145536)

# General Context

A packet capture (PCAP) is a vital tool used to keep networks operating safely and efficiently. In the wrong hands, it can also be used to steal sensitive data like usernames and passwords (Grimmick, 2021). In this lab we analyze a packet capture that is provided to us to perform some basic network forensics and identify some actions that took place. While we are performing these actions, it’s easy to see just how powerful networking monitoring utilities are, and when used properly can made post-incident investigations much easier, but they are also incredibly easy to abuse and care needs to be taken to ensure their security.

In this packet capture, we first take a look at a few characteristics of it. We analyze the start and stop timestamps, and then take a look at the overall packet count. Wireshark has analysis tools that break down the PCAP into a list of protocols and show the percentage of packets that consisted of different protocols, as well as the total bytes transferred, and the bits per second for each protocol. This can provide a quick at-a-glance view if your top talkers are HTTP or IMAP or SMB. You might not expect large file transfers to be happening, so if a PCAP shows that they are, it can raise quick red flags.

The last half of the lab analyzes some email traffic. It would appear that the user of the machine who generated this PCAP was under an internal investigation, thus the cause of the creation of the PCAP. In Wireshark if we click Tools and then Credentials it shows us, within a few seconds, five different packets where a password was passed in plain text. All five packets, happened to be the same password which was this user logging into their email account via IMAP, but it shows how easily things can be picked up over the wire if they are not encrypted. This ends up presenting an argument that has been playing out over the last several years; security operations personnel need visibility into network traffic to know that nothing malicious is happening, but if things aren’t encrypted and an attacker gets on the network they can see lots of things that they shouldn’t. Where this line gets drawn ends up being a driving force for future organization policies and procedures.

# Technical Context

This packet capture is just under 8000 packets. This, in the grand scheme of things, isn’t an entirely overwhelming amount, but for a new analyst can provide some significant hesitation and make them feel like they are drinking from a firehose. In order to try to filter out specific items and protocols, we can use Wiresharks built-in Display Filters which really are quite powerful. If you have a huge packet-capture of HTTP traffic, you can set a filter to only show you HTTP cookies, or user\_agents, or any other common header field that is commonly passed but seldom seen by end users. Wireshark also has several built-in tools to make looking at packets more digestible by an analyst. For example, the first three bytes of a MAC address is an Organizationally Unique Identifier. This means that those first six characters are directly attributable to who made that network adapter, so in 5C:26:0A:xx:xx:xx the x’s can be any hexadecimal character but you know it was made by Dell. Wireshark can automatically enrich packet capture views to show this manufacturer data, making it easy to filter and search on when looking for specific hardware types (Wireshark · OUI Lookup Tool, n.d.).

More and more services on the internet are taking an ‘encryption everywhere’ approach. For personal devices on their home network this is a great idea, all communications are then only known by the client and the server. However, for corporate networks this can be a challenge. If you’re asked to investigate if a person was browsing to websites that are against company policy, but if they are all over HTTPS it might not be easy or possible to prove that with just a packet capture or with network monitoring. An encrypted tunnel is set up before any data is transferred, so you might be able to see a URL and a port, but you won’t know what was seen or sent on that URL and port. Additional considerations might need to be made regarding proxies. If you restrict what traffic is sent outbound, and force it through a forward proxy, it can be much easier to categorize traffic and monitor that people are going to authorized places. If you don’t restrict outbound traffic, it’s trivial to connect to an external HTTPS proxy, which to your monitoring will show the proxy URL and port only as the HTTPS tunnel is set up, but all other traffic sent over that proxy tunnel will be unknown.

The last step of this lab had us analyzing some IMAP email traffic. We were able to see three emails that appear to have been sent by the user of this machine. One showed Regina, the user in question, attempting to acquire fake passports and Visas, and from the text it’s not the first time. Regina then emailed someone cancelling plans for the following week because they would be out of town. Finally, Regina emailed another address telling them to bring their fake passport and meet at an address that was sent in an attachment. Since we have the full packet capture, we are able to view the full email body and carve out the attachment and Base64 decode it and reconstruct the image to see that they are planning on going to Aruba. If a company has reason to believe that an insider threat is stealing or about to steal sensitive data and has a fake passport and is planning to leave the country, it’s likely extremely time sensitive to uncover all of this information and get it to the proper authorities to stop that person before any of these potentially illegal things can happen.

# Solution

We begin this lab by signing into the ‘Cyber Basics’ virtual machine in the U.S. Cyber Range, and downloading the charade\_capture.pcap file to the desktop. We then open a terminal window and run `wireshark ~/Desktop/charade\_capture.pcap` to open the packet capture(PCAP) in Wireshark. The PCAP starts at 12:03:50 on 11/14/2012 and ends at 12:10:10 on 11/14/2012, and was 7779 packets. Running Protocol Heirarchy Statistics on the PCAP show that DNS made up 13.5% of the packets but only 2.3% of the actual traffic passed, whereas 9.1% of the packets and 70.9% of traffic passed is HTTP traffic.

A picture containing graphical user interface

Description automatically generated

Figure 1 - Timestamps of PCAP

A picture containing text

Description automatically generated

Figure 2 - Packet Analysis

Graphical user interface

Description automatically generated with medium confidence

Figure 3 - Protocol Hierarchy Statistics

Next, we actually look at the data in the PCAP. We find the first DHCP request in the capture and see that the hostname that made the request was called ‘reggie-laptop’, and it was assigned 10.3.0.20 and that the first 32 bits of the MAC address are 5C:26:0A which when we look up the OUI shows that the NIC was made by Dell and most likely is a Dell laptop. The endpoint received 10.3.0.1 as it’s DNS server, and it was given a lease of 86400 seconds which is 1 day.

A screenshot of a computer

Description automatically generated

Figure 4 - DHCP request

Graphical user interface, text, application

Description automatically generated

Figure 5 - DHCP assigned IP address and client MAC address

Graphical user interface, text

Description automatically generated

Figure 6 - DNS server address

A screenshot of a computer

Description automatically generated with medium confidence

Figure 7 - DHCP Lease Time

Now that we now the hostname, and IP address that was assigned to this machine we can more easily identify some of the traffic. We look at some of the HTTP requests to see what websites were being browsed. Some of the websites that were browsed include: weather.noaa.gov, maps.google.com, www.monster.com, www.dice.com. Noticing that monster and dice are both job posting sites, we take a closer look and see that this user is likely looking for a new job because they searched monster for ‘Director-Software-Engineering\_5’.

A screenshot of a computer

Description automatically generated

Figure 8 - HTTP request for noaa.weather.gov

Graphical user interface, text, application

Description automatically generated

Figure 9 - HTTP request for maps.google.com

Text

Description automatically generated with low confidence

Figure 10 - HTTP request referred from monster.com job search

Next, we do a quick search for any credentials that were passed in plain text by clicking Tools -> Credentials. It shows that there are 5 packets that contain a credential. All 5 packets happen to be the same password for an IMAP mailbox for ‘in5id3r.thr34t@aol.com’. We dig into the IMAP packets and find 3 emails that were sent from ‘Regina Lampert’. Regina signed two of the emails with ‘R’ and one with ‘Reggie’ which coincides with the hostname of the laptop which is ‘reggie-laptop’. The first email that they sent was at 04:06:14 PST on 11/14/2012 to ‘fakedocs4you@yahoo.com’ and the body of the email show that Regina was attempting to acquire a fake passport and visa, and was set to meet them the next day 11/15/2012. The second email Regina sent was at 04:07:12 PST on 11/14/2012 to ‘hamilton.bartholomew@yahoo.com’ and the purpose of this email was to cancel lunch plans for the following week because Regina was about to head out of town. Finally, the third email that was sent was sent at 04:09:49 PST on 11/14/2012 to ‘sekritskwerl@hotmail.com’ and Regina told them to bring their fake passport and a bathing suit, and there was an image file attached to the email called ‘map.jpg’. We were able to carve out the raw bytes from the attachment, decode it from Base64 and view the image to see that the rendezvous point was the Flying Fishbone, Savaneta 344, Saveneta, Aruba, 584 2506.

Timeline

Description automatically generated with medium confidence

Figure 11 - Email sent to fakedocs4you@yahoo.com

Timeline

Description automatically generated

Figure 12 - Email sent to hamilton.bartholomew@yahoo.com

Text, timeline

Description automatically generated

Figure 13 - Email sent to sekritskwerl@hotmail.com

Graphical user interface, map

Description automatically generated

Figure 14 - map.jpg attachment

# ****References****

Grimmick, R. (2021, August 5). *Packet Capture: What is it and What You Need to Know | Varonis*. Inside Out Security. <https://www.varonis.com/blog/packet-capture/>

*Wireshark · OUI Lookup Tool*. (n.d.). Retrieved October 18, 2021, from <https://www.wireshark.org/tools/oui-lookup.html>