NIDS_NIPS AND WEB PROXY ANALYSIS

Introduction

The dynamic revolution of technology has brought about the emergence of different mechanisms to equalize the ever-innovative nature of adversaries in today's cyber world. This paper will focus on some of the popular systems which are IDS (Intrusion Detection System) and HIPS (Intrusion Prevention System).

IDS (Intrusion Detection System) is a security system that is tasked to detect malicious activities on a computer or network. This acts in a similar manner to how alarms work in our vehicles detecting suspicious activity, noting it, cataloging and eventually classifying it (Arthur, 2022). The two main categories of IDS are discussed below.

HIDS (Host Intrusion Detection System) is used to examine activities on a single system such as a mail server, web server or a personal computer. This is limited to only an individual system, and it doesn't have any capability to examine activities on the network systems connected to it. Examples of these systems are Splunk and OSSEC (Open-source security event correlator) (Deshpande, Sharmaa, & Peddoju, 2018).

NIDS (Network Intrusion Detection System) This system is used to examine activities on the network traffic its connected to and it doesn't have the capability to view what's happening on individual systems. Examples of these type of system include Snort and Suricata (Zeeshan, Adnan, cheah, & Johari, 2020).

Intrusion Prevention system (IPS) is used to monitor network traffic for malicious or anomalous behavior and have the capability to block, reject or redirect the traffic in real time.

NIPS (Network intrusion prevention System) is a system that examines network traffic and responds to intrusions hence can prevent unauthorized access. Some of the examples are cisco secure IPS and Snort.

HIPS (Host-based Intrusion Prevention System) is a system that automatically responds to computer intrusions via monitoring activities on one or more personal computers or servers and responds according to a ruleset. Examples include windows defender and Kaspersky endpoint security (Ajay, Abhishek, & Ghalib, 2022).

For a business network I would choose the network-based intrusion and prevention systems which will monitor and detect any malicious activities hence respond to these anomalies.

Background

This paper we are going to delve deep into the investigation by unfolding hidden information from IDS/IPS alert data. The event includes a credit card recycling program that was stored for a long time without being used. This is met by a defensive mechanism deployed by the payment processor company utilizing snort one of the NIDS sensors to detect malicious activities.

Snort is an open-source network intrusion prevention system that can perform real-time traffic analysis and packet logging on IP networks. It is utilized to get alerts for any anomalous detection which is stored in as dataset from what is collected from the tool (Badotra & Surya, 2021).

This investigation concerns an alert that was logged from an external host that tagged an insider, and we will be checking the tcpdump.log comparing it to data from the snort configuration and the snort rules information to understand what was being detected.

IDS/IPS Log Analysis

The initial step was to confirm if the alert did exist as suggested in the background above, which was **true** that the alert did exist.

```
[**] [1:10000648:2] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
05/18-08:01:45.591840 172.16.16.218:80 -> 192.168.1.169:2493
TCP TTL:63 TOS:0x0 ID:53309 IpLen:20 DgmLen:1127 DF
***AP*** Seq: 0x1B2C3517  Ack: 0x9F9E0666  Win: 0x1920  TcpLen: 20
```

Figure 1/Alert

Examination

We did an examination to understand why this alert went off by checking the rules that led to this detection. The rule was included in a deleted rule that showed the details in the screenshot below.

```
# dup of 653
alert ip $EXTERNAL_NET $SHELLCODE_PORTS -> $HOME_NET any (msg:"SHELLCODE x86
0x90 NOOP unicode"; content:"|90 00 90 00 90 00 90 00 90 00 90 00 90 00 90
00|"; classtype:shellcode-detect; sid:2314; rev:2;)
```

Figure 2/Rule

The above rule is used to detect an attempt to execute shellcode on a host in a protected network from an external source. Shellcode is a small piece of cod used as the payload in the exploitation of a vulnerability, this set of instructions once executed will spawn a shell on the target that can be used by an attacker to control or manipulate the target (Wilfrantz, 2022).The content starting with 90 00 is a hex pattern that is matched to detect .This paper gives a high level of NOOP which means no operation in assembly language and it is denoted by the byte 0X90 commonly found in the X86 assembly (MKdocs, 2020).

From the tcpdump.log file we found the packet screenshot below:

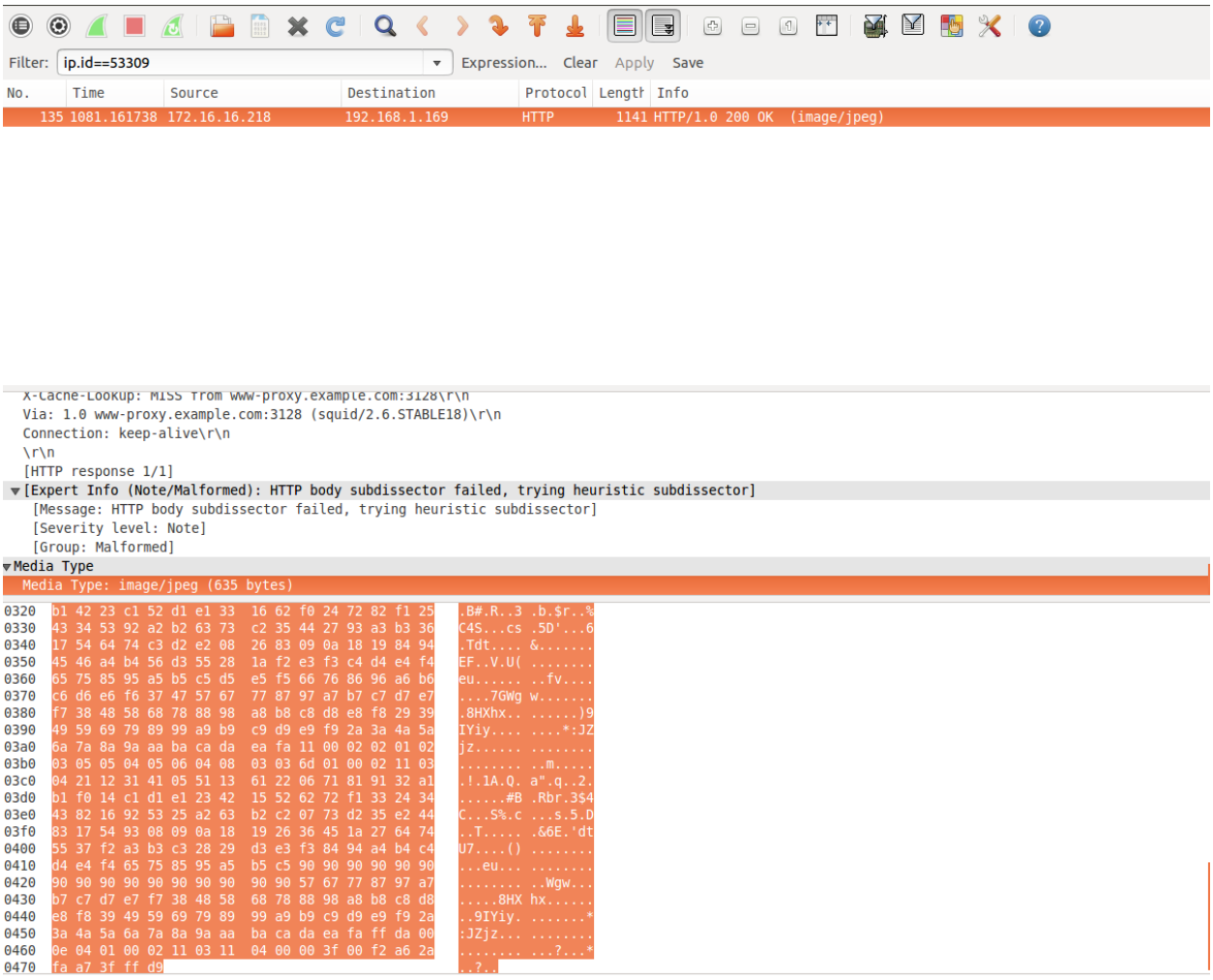

Figure 3/Wireshark

The packet shows a repeated pattern that matches the one indicated in the rule. This is why the alert was fired since it got a match.

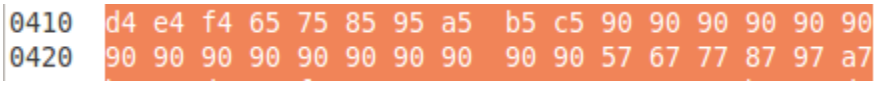

Figure 4/match

Proxy Log Analysis

This section we will be tackling proxy log analysis which provides a record of incoming and outgoing traffic that facilitates detection of threats. This is done by analyzing data to identify unusual patterns or behaviors that might be from malicious activities (Dennis, 2024). From our analysis prior to this we detected a jpeg image that had an unusual binary sequence that was associated with buffer overflow exploits. This is a type of exploit where data written to a buffer exceeds its size and the vital info is overwritten, or a malicious code is executed (Ninad, 2023). We utilized logs from the squid tool which is a caching and forwarding web proxy showing events between clients and servers.

The client system 192.168.1.169 runs a windows operating system and from the screenshot below tried to access the www.evil.evl system as indicated in our prior analysis of the evil system such as MacDaddy payment Processor Local Domain.

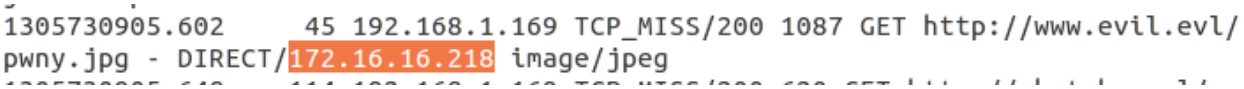

Figure 5/evil

The above access log corresponded to the cache records in storage log as should below

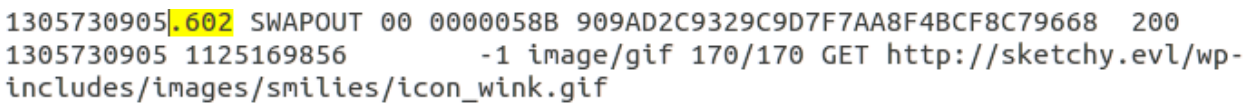

Figure 6/storage log

The Epoch time 1305730905.602 when converted gives the same date as the one from the timeline. The time in this screenshot is converted into GMT and it's vital to note that it was originally in Mountain Standard Time.

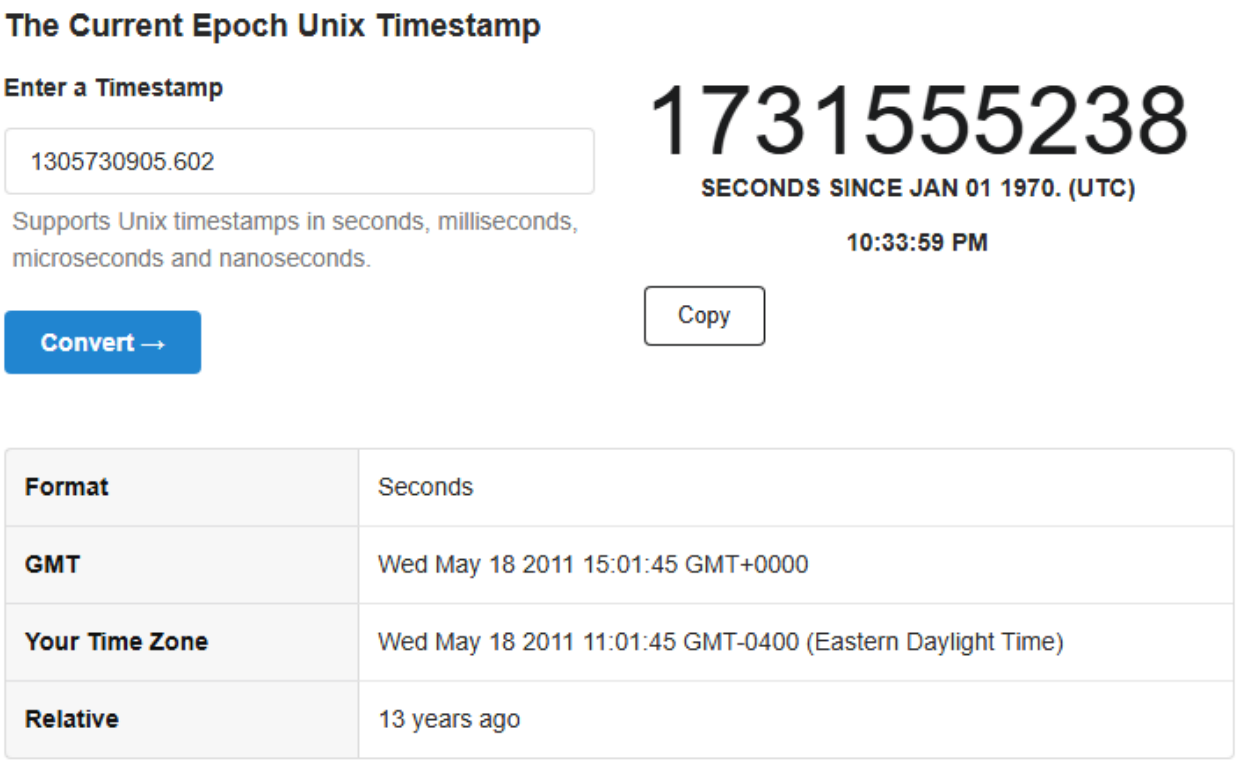

Figure 7/epoch time

There is a consistent log on to clients1 in the logs as shown in the screenshot below which might have been malicious.

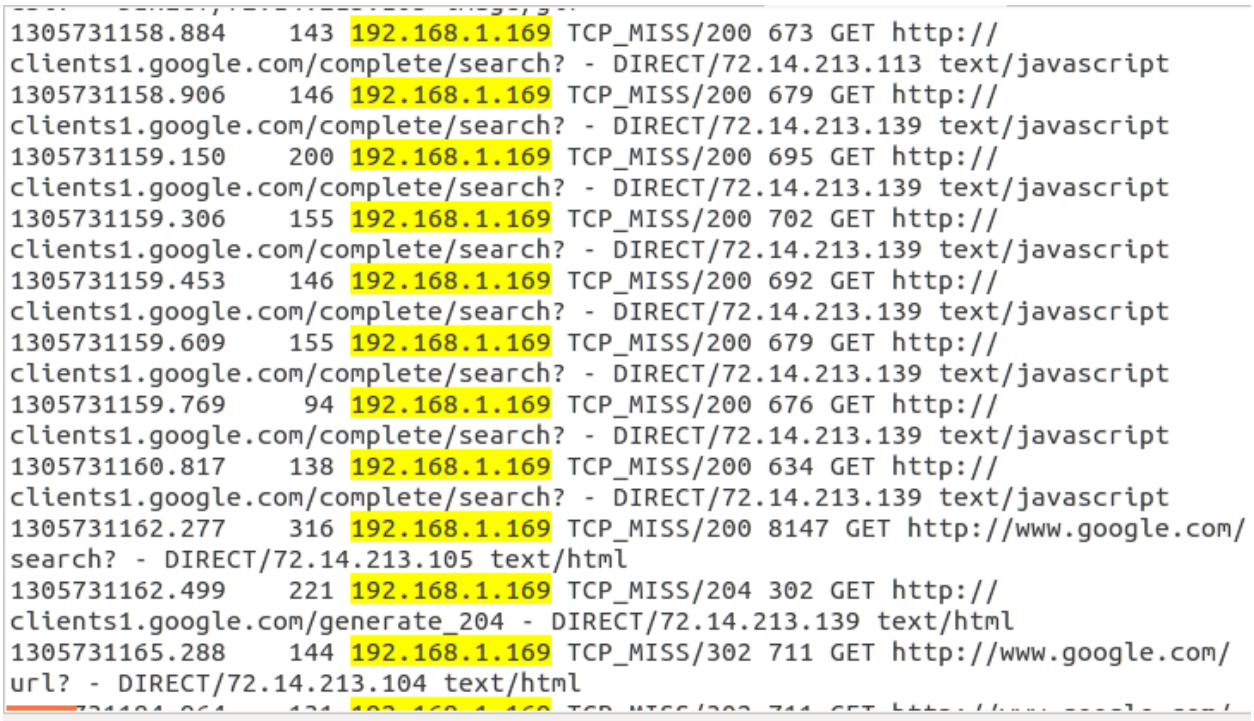

Figure 8/Clients

From our analysis we came across a user who might have been involved in malicious activities.

The screenshot below shows a user by the name of Jonny Evans.

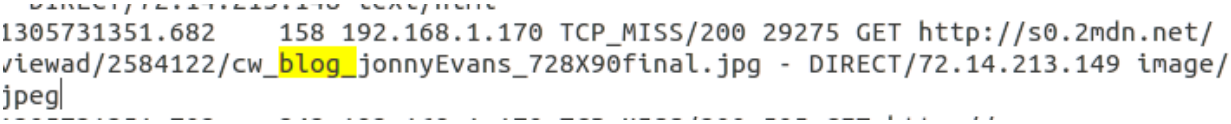

Figure 9/client1

References

Ajay, K., Abhishek, & Ghalib. (2022). Intrusion detection and prevention system for an IoT environment. *Digital Communications and Networks* , 540-551.

Arthur, C. (2022). *Principles of Computer Security.* Newyork: McGaw Hill.

Badotra, S., & Surya, N. (2021). SNORT based early DDos detction system using opendaylight and open networking operatiing system in software defined networking. *Cluster Computing*, 501-513.

Dennis, M. (2024, october 3). *The importance of proxy server logging and monitoring*. Retrieved from cloud infrastructure services: https://cloudinfrastructureservices.co.uk/the-importance-of-proxy-server-logging-and-monitoring/#:~:text=Proxy%20server%20logging%20provides%20a%20record%20of%20incoming,attacks%2C%20and%20even%20attempts%20to%20breach%20the%20system.

Deshpande, Sharmaa, & Peddoju. (2018). HIDS: A Host based intrusion detection system for cloud computing environment. *International Journal of System Assurance Engineering and Management*, 567-576.

MKdocs. (2020). *Buffer Overflows & NOOP sleds*. Retrieved from Computer Network Analysis: https://cted.cybbh.io/tech-college/cttsb/Computer_Network_Analysis/01_Packet_Analysis/04_Host_Scan_Methodologies/02_No_Operation_NOOP_Sled.html

Ninad, M. (2023, April 24). *Exploitnig Buffer Overflow Vulnerabilities*. Retrieved from Cobalt: https://www.cobalt.io/blog/pentester-guide-to-exploiting-buffer-overflow-vulnerabilities

Wilfrantz, D. (2022, November 28). *Understanding shellcode, from exploit to control*. Retrieved from dede.dev: https://dede.dev/posts/Understanding-Shellcode-From-Exploit-to-Control/

Zeeshan, A., Adnan, S., cheah, W., & Johari, A. (2020). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Emerging Telecommunications Technologies*.