Coursework 1: Set Exercises - PCAP Analysis

Github Link: <https://github.com/Mattrfish/COMP3010-SecOps-and-Incident-Management.git>

Youtube Walkthrough Link:

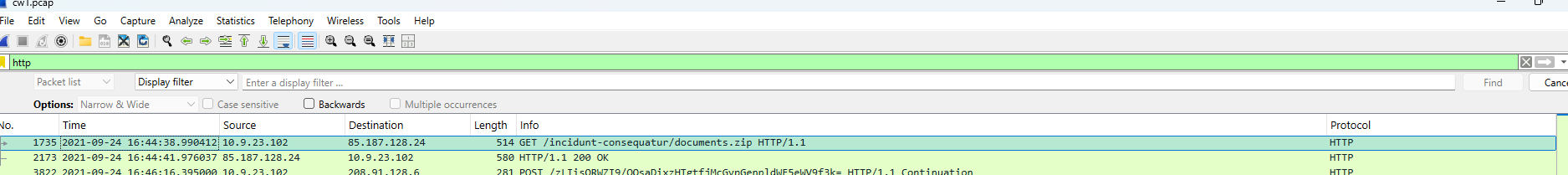
# Introduction

This incident report presents a detailed analysis of a provided PCAP file to investigate a suspected network intrusion. The aim is to identify the infected system, determine how the infection occurred, describe what type of attack was involved, and highlight the key indications of compromise (IOCs).

This report is structured as follows: The methodology section outlines the tools and techniques used and how they were applied to investigate the intrusion. The results section presents the key findings from the PCAP analysis, supported by evidence and screenshots. Finally, the conclusion summarises the findings, discusses prevention strategies to mitigate similar incidents in the future, and reflects on any challenges encountered during the investigation.

# Methodology

Wireshark was used to analyse the pcap file, as it is one of the most effective open source tool for packet-level network traffic analysis.



A screenshot of a computer program

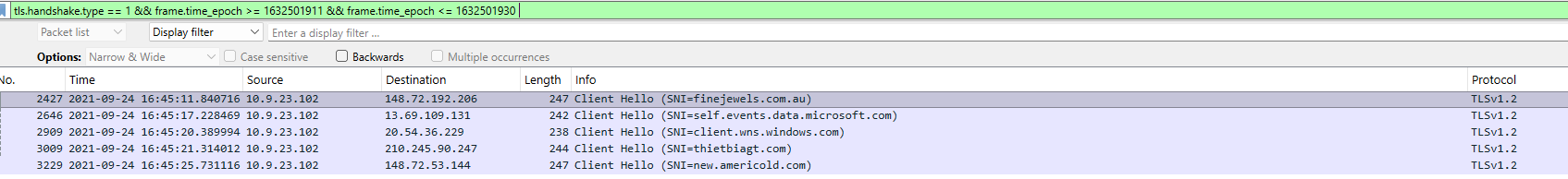
AI-generated content may be incorrect. A screenshot of a computer

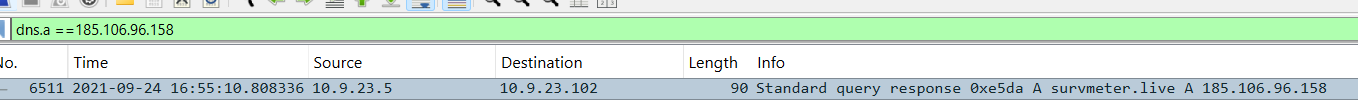
AI-generated content may be incorrect.

The process began by inspecting HTTP traffic using the “http” display filter, to detect the initial infection. I used “follow > http stream” , to inspect communications between the victim and malicious domains, trace the infection process and examine server information, headers, and communication patters. It also enabled me to inspect transferred file types without exporting them, reducing the risk of executing malicious content.

A screenshot of a computer

AI-generated content may be incorrect.



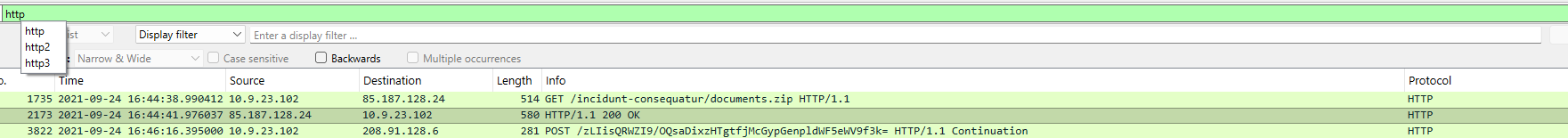


A screenshot of a computer

AI-generated content may be incorrect.

The packet details pane was used throughout to extract domain names, server headers, and SSL certificates. To investigate post-infection activity, I used filters like frame.time and tls.handhsake to isolate specific time windows. The dns.a filter identified the domains associated with c2 servers. I used the “statistics > conversations” to analyse packet counts and beaconing patterns to further understand the c2.

# Results



A screenshot of a computer program

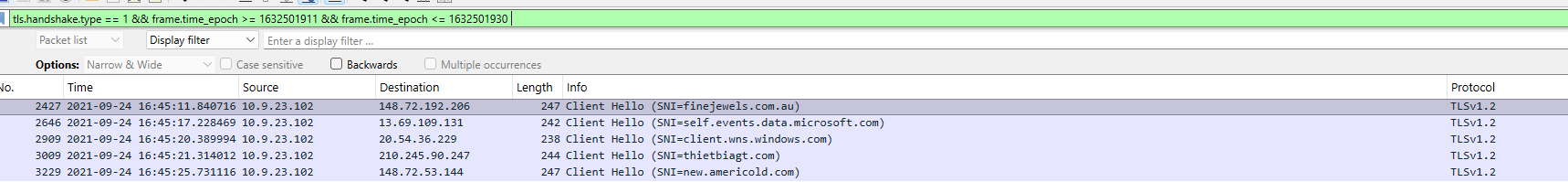
AI-generated content may be incorrect.



I started the analysis by identifying the initial HTTP connection which occurred on 2021-09-24 16:44:38 between the victim host (10.9.23.102) and malicious destination server (85.187.128.24). Following the stream revealed that the victim downloaded a compressed file named “documents.zip” from the domain “attirenepal.com”. Inside contained a malicious file called “chart-1530076591.xls”, which appeared to be the initial stage of infection. The destination server was running “LiteSpeed” with “PHP /7.2.34”, indicating a potentially exploitable server being used to deliver the payload.

A screenshot of a computer

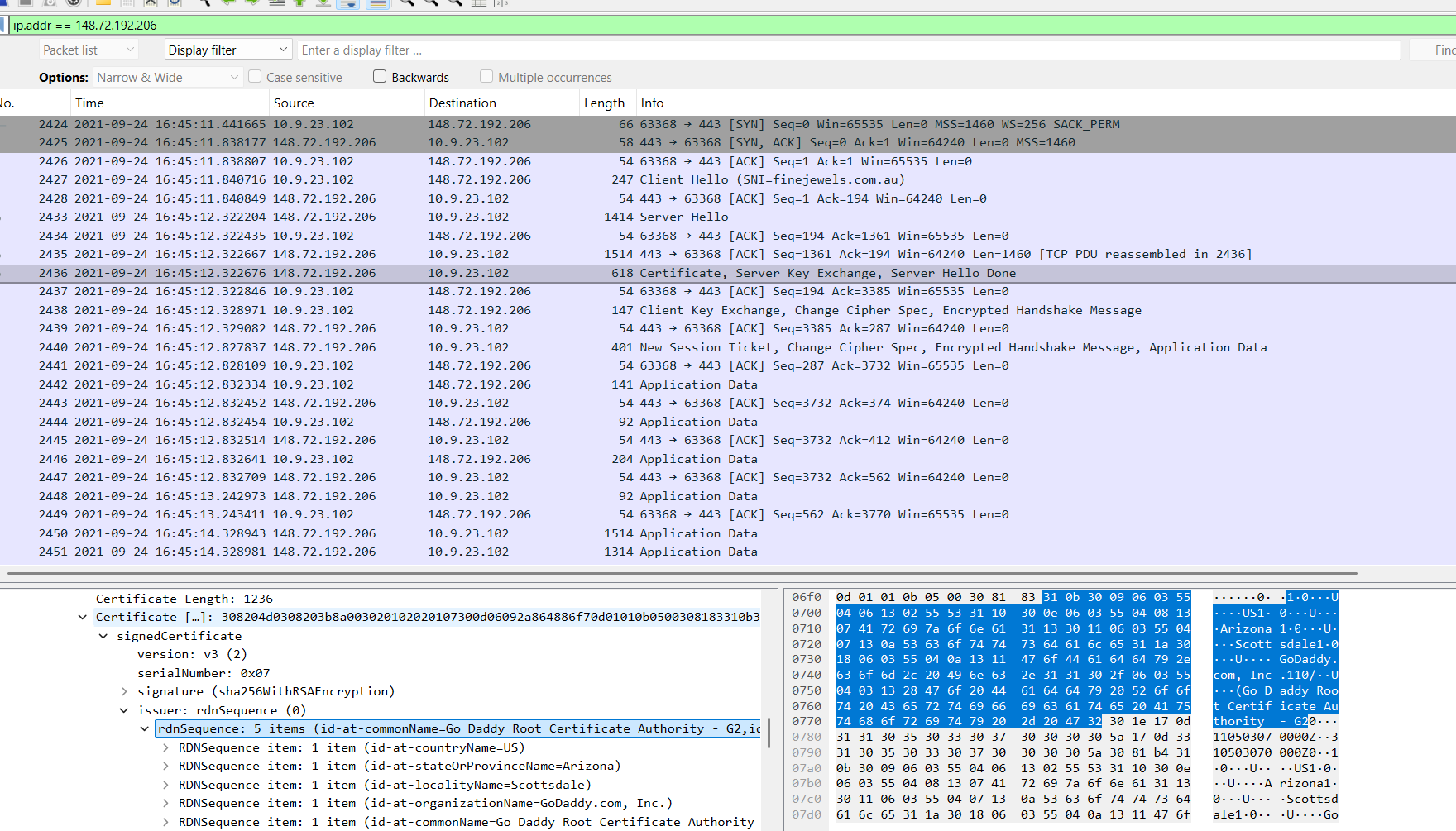
AI-generated content may be incorrect.





Reviewing the HTTP timeline further showed a POST request at 16:46:16, indicating possible command-and-control (c2) activity. Between the initial HTTP ok response and this POST request, I identified several suspicious TLS requests coming from the victim host. By applying a filter to isolate only the “ClientHello” packets, I revealed three additional domains involved in downloading malicious files to the victim host: finejewels.com.au, thietbiagt.com and new.americold.com.





I looked up the IP on the first domain to identify the Certificate Authority (CA) that issued its SSL certificate. I determined that the certificate was issued by GoDaddy Root Certificate Authority. This information was valuable as it confirmed the use of a valid certificate to make the malicious traffic appear legitimate.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

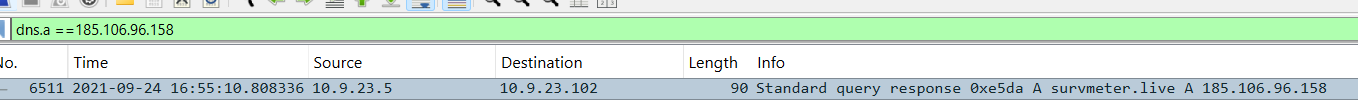
A screenshot of a computer

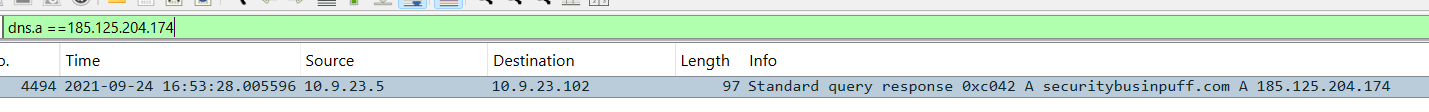
AI-generated content may be incorrect.

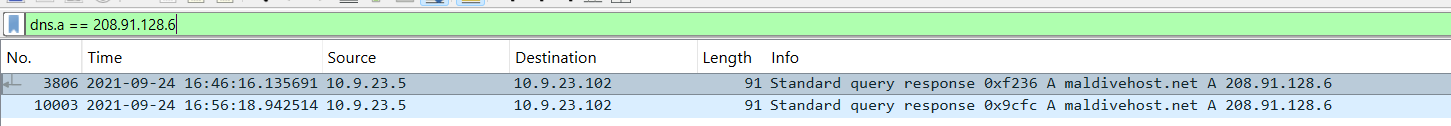
Revisiting the earlier POST request, I noticed multiple similar requests were being continuously sent every 30 seconds to a suspicious domain. To investigate further and determine whether it might be beaconing, I opened the conversations section in the statistics. There I identified two additional IPs (185.106.96.158 and 185.125.204.174) that had a similar behaviour with a large packet count. Applying display filters to both these Ips confirmed that they were C2 servers, due to their high request frequency and large packet count.

A screenshot of a computer

AI-generated content may be incorrect.







Further inspection showed that the first IP contained a host header which was ocsp.verisign.com\r\n, likely to disguise the connection as legitimate traffic. Using DNS filters, I mapped the Ips to their domains (survmeter.live and securitybusinpuff.com). Additionally, I identified another domain (maldivehost.net) used for the previous POST requests, which appeared to handle exfiltration rather than beaconing.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

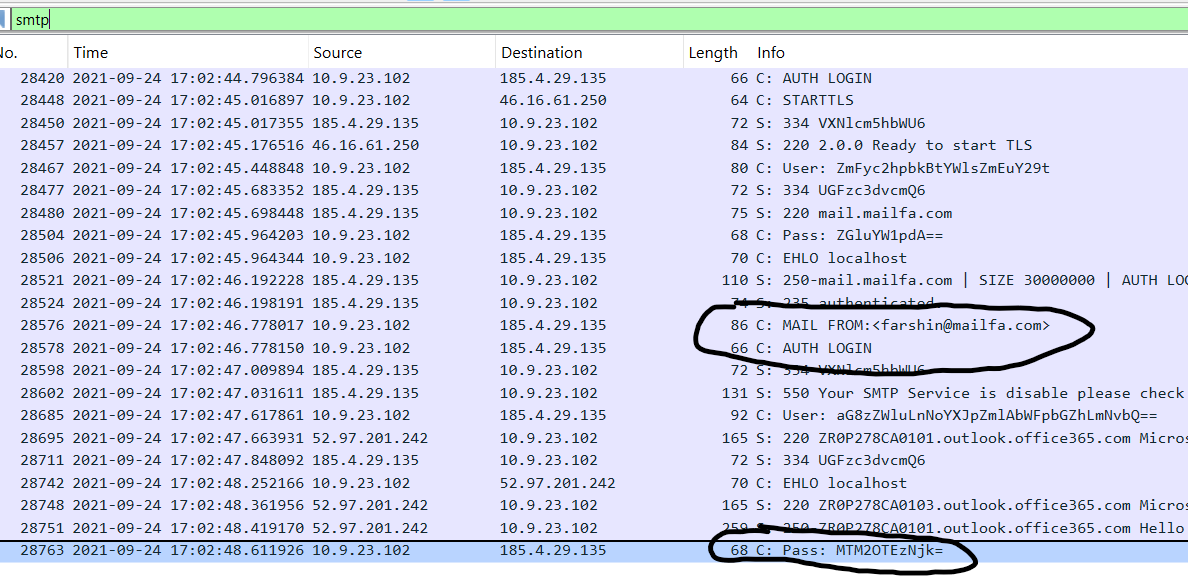
AI-generated content may be incorrect.

I examined the first request to the c2 server in greater detail and found that the payload was 281bytes in length, with the message beginning with “zLIisQRWZI9”. Following the HTTP stream further revealed the sever header as “Apache/2.4.49 (cPanel) OpenSSL/1.1.1l mod\_bwlimited/1.4”. These details confirmed that it was hosted on a standard web stack that had possibly been hijacked or intentionally configured for malicious use.

A screenshot of a computer

AI-generated content may be incorrect.

During the final stage of analysis, I performed a DNS query to identify any IP lookups on the victim host that the malware performed. I found a query at “2021-09-24 17:00:04” for the domain “api.ipify.org”, which is a commonly used IP-checking service.



A screenshot of a computer

AI-generated content may be incorrect.

Finally, I filtered SMTP traffic to identify any potential data exfiltration through email. This revealed the first “MAIL FROM” address as “[farshin@mailfa.com](mailto:farshin@mailfa.com)”. I followed the stream further and authentication attempts from “[ho3ein.sharifi@mailfa.com](mailto:ho3ein.sharifi@mailfa.com)”, using the password “MTM2OTEzNjk=”. This indicates the attacker using potential compromise or disposable email accounts for communication or data exfiltration.

# Conclusion

The analysis successfully confirmed that the cause of infection was from downloading a malicious file from attirenepal.com. This triggered communication with multiple malicious domains that delivered additional payloads and established C2 connections. The continuous beaconing to 185.106.96.158 and 185.125.204.174, along with exfiltration activity through maldivehost.net, showed a typical post-incident behaviour pattern. Furthermore, the malware performed an external IP check via api.ipify.org and attempted to send credentials through SMTP, further indicating exfiltration and communication from the victim host.

One of the main challenges encountered during the analysis was examining HTTPs- encrypted data, as it hid payload contents and limited visibility of certain indicators within the traffic. It meant I had to rely on timing and TLS handshake details to search malicious behaviour.

To prevent similar issues in the future, users should be trained on phishing to help them identify and avoid clicking on malicious links that lead to unsafe websites. Additionally, implementing antivirus software and other tools like Norton can help detect and block access to malicious websites. These tools can also alert users to potentially harmful downloads, allowing to contain it early and prevent further data exfiltration or system compromise. Finally, the use of properly configured firewalls and network monitoring systems can help block known malicious domains and IOCs before they reach internal hosts.

# Appendix