# Incident handler's journal

| **Date:**<br>20/07/2022 | **Entry:**<br>N001 |
|---|---|
| Description | Upon opening a suspicious file, multiple executables files were generated indicating dropper malware activity. In this entry, we will outline the investigation into this incident. |
| Tool(s) used | VirusTotal |
| The 5 W's | • **Who:** The incident is caused by "Clyde West" with an email address "76tguyhh6tgfrt7tg.su"<br>• **What:** The sender emailed a suspicious .exe file claiming to be their CV<br>• **When:** The incident took place at 9:30 AM on 20th of July 2022, online<br>• **Where:** Online/company email system<br>• **Why:** The sender probably tried to access information from the company |
| Additional notes | Suspicious domain ".su" is a red flag because it's commonly used in abusive campaigns. Several executable files were created, indicating probable dropper malware activity. . Recommend blocking the sender's domain and email address. All affected systems need to be quarantined and scanned for additional malware. Need to ascertain if any data was compromised in the breach. |

| **Date:**<br>01/08/2025 | **Entry:**<br>N002 |
| --- | --- |
| Description | Completed packet capture analysis using tcpdump to capture and analyse live network traffic from a Linux virtual machine. This exercise focused on learning network interface identification, traffic filtering, and packet data analysis. |
| Tool(s) used | tcpdump |
| Notes | I started by finding available network interfaces using sudo ifconfig. The eth0 interface was the primary Ethernet connection with IP 172.17.0.2. I also found that sudo tcpdump -D is another way of listing interfaces available for capturing packets.<br><br>To examine live traffic, I used sudo tcpdump -i eth0 -v -c5 to capture 5 packets from the eth0 interface. The verbose output gave me an idea of packet structure, like timestamps, protocols, and TCP flags. It was interesting to see the actual communication flow among different systems.<br><br>Next, I practised sniffing packet data to a file using sudo tcpdump -i eth0 -nn -c9 port 80 -w capture.pcap &. This filtered for HTTP traffic only on port 80 and ran in the background. I generated some test traffic using curl opensource.google.com, so I had actual data to sniff.<br><br>I was able to read the packet capture data with sudo tcpdump -nn -r capture.pcap -v. I could examine the packet headers and read the TCP communication details. |
| Additional notes | I discovered that the -nn option turns off DNS lookups. The activity allowed me to understand basic packet analysis and filtering principles. This tool will be |

| | |
|---|---|
| | useful in network troubleshooting and analysing suspicious traffic patterns in future incidents. |

---

| **Date:**<br>03/08/2025 | **Entry:**<br>N003 |
|---|---|
| Description | I learned how intrusion detection systems use signatures and rules to monitor network traffic by completing a Suricata activity. This was my first experience configuring custom rules and analysing alerts. |
| Tool(s) used | Suricata IDS |
| The 5 W's | I used the cat command to see a custom rule within the custom.rules file. It contained an action (alert), a header setting network traffic parameters, and options for customisation of rules. The rule looked for HTTP GET requests from the home network to other networks.<br><br>The 'alert' action tells Suricata to do something if conditions are met. The header 'http $HOME_NET any -> $EXTERNAL_NET any' specifies protocol and direction of traffic. Rules have options of 'msg' for text of alert, 'flow' for packet direction, and 'content' for matching exact text.<br><br>I tested the rule by running sudo suricata -r sample.pcap -S custom.rules -k none. It processed the sample traffic and made alerts when the conditions in the rule were met. Suricata generated log files in /var/log/suricata. |

| | |
|---|---|
| | The fast.log file contained standard alert data in a simple format. There was timestamp, rule ID, alert message, and source/destination on each line. The eve.json file contained much more verbose data in JSON format. Using the jq command, it was easy to parse and extract such specific fields as timestamps, flow IDs, and destination IPs. |
| Additional notes | This exercise described the working of IDS rules and the significance of proper rule tuning. The JSON format for the output seems more appropriate for automated analysis and network correlation with other security appliances. I can envision custom rules for detecting novel attack patterns or malicious network behaviour. |

---

Reflections/Notes:

**1. Were there any specific activities that were challenging for you? Why or why not?**

The Suricata rule creation was initially challenging because understanding the syntax and rule components required careful attention to detail. Learning how actions, headers, and rule options work together took some practice to fully grasp.

**2. Has your understanding of incident detection and response changed since taking this course?**

Yes, I now appreciate how methodical incident response needs to be with proper documentation and evidence correlation. The 5 W's framework and systematic analysis approaches showed me the importance of thorough investigation processes.

**3. Was there a specific tool or concept that you enjoyed the most? Why?**

I found packet analysis with tcpdump most interesting because seeing actual network traffic helped me understand how data flows work. The ability to capture and examine real network

communications made cybersecurity concepts more tangible and practical.