Cyber Security Documentation: C2 Simulation Exercise

Table Of Contents

Scope

This report documents a C2 simulation exercise conducted to assess the blue team's ability to detect and respond to malicious activity. The exercise utilized a caldera server as the C2 server, with Splunk collecting data from pfsense and Suricata to detect beaconing agents. Two Alpine Linux target hosts were created to simulate compromised systems, and Proxmox was employed to simulate the network environment. Abuse.ch threat intelligence feeds (ThreatFox, MalwareBazaar, and URLhaus) were integrated using a Python script to fetch the CSV files, and Splunk was responsible for parsing and analyzing the data.

Detailed Setup

C2 Server

- A caldera server was configured as the C2 server, listening on port 8888 and using the HTTP protocol for communication.
- No encryption was used.

Splunk

- Splunk was deployed to collect and analyze data from pfsense and Suricata.
- Data sources:
  - pfsense logs forwarded to Splunk on port 5147 (sourcetype: pfsense, index: network)
  - Suricata logs forwarded to Splunk using rsync due to Splunk Universal forwarder segfaulting (sourcetype: suricata, index: ids)
- Abuse.ch Threat Intelligence Feed Integration:
  - A Python script was used to fetch the ThreatFox, MalwareBazaar, and URLhaus feeds from Abuse.ch and save them to a designated directory.
  - Splunk was configured to monitor this directory for new CSV files and automatically ingest them.
  - Splunk's built-in CSV parsing capabilities were used to extract relevant threat indicators from the CSV data.
- Search Queries:
  - Showing Beaconing Tuned for Our Specific Caldera Agent Configuration
  - Detecting Possible Beaconing (More False Positives)
  - Correlating Threatfox IOC Ports with Suricata Logged Ports
- Dashboards:
  - Beaconing Per Hour and Possibly Other Beaconing Activities
  - Most used ports in threatfox iocs

pfsense

- pfsense was used as the network firewall, configured with rule sets to monitor for suspicious activity.
- Intrusion prevention system (IPS) enabled with appropriate rules.
- Remote logging configured to send logs to Splunk on port 5147.

Suricata

- Suricata was deployed as an IDS, using a rule set to detect known C2 communication patterns.
- Logs forwarded to Splunk using rsync due to Splunk Universal forwarder segfaulting.

Target Hosts

- Two Alpine Linux target hosts were created on Proxmox, simulating compromised systems.
- No vulnerabilities were exploited as this was a simulation.
- Agents were installed manually on the target hosts.

Proxmox

- Proxmox was used to create a virtualized network environment.
- Virtual machines:
  - Caldera server
  - pfsense firewall
  - Suricata IDS
  - 2 Alpine Linux target hosts

Achieving Scalability

- Modular design: Each component (C2 server, Splunk, pfsense, Suricata, target hosts) was configured to be independent and easily scalable.
- Automation: Scripts were used to automate data collection.
- Cloud-based infrastructure: Consider migrating to a cloud-based environment for enhanced scalability and flexibility.

Setup Issues

- Splunk Universal Forwarder Segfault: The Splunk Universal forwarder encountered a segfault during startup, preventing it from forwarding Suricata logs directly to Splunk.
- Workaround: Rsync was used to periodically transfer Suricata logs from the pfsense firewall to the Splunk server.

Testing

The Splunk environment was tested by the SOC to assess its effectiveness in detecting beaconing activity. The following test scenarios were conducted:

- Simulated C2 communication: Agents on target hosts beaconing to the C2 server over HTTP on port 8888.
- Splunk correlation and analysis: Evaluating Splunk's ability to correlate data from various sources (including Abuse.ch feed) and identify potential threats.

Conclusions

- The blue team successfully implemented a Splunk environment to monitor and analyze network traffic for potential beaconing activity. The setup was modular, scalable, and automated, with each component designed to be easily manageable.
- While we were successful in detecting beaconing activity, it would be easy for the red team to add peer to peer communication, longer beaconing intervals and/or rotating protocols and ports to make the communications between agents and server much harder to detect.

Recommendations

- Investigate Forwarder Issue: Conduct further analysis to determine the root cause of the Splunk Universal forwarder segfault and implement appropriate solutions to prevent future occurrences.
- Enhancements:
  - Implement additional security measures on target hosts to harden their defenses.
  - Optimize Splunk searches and dashboards for improved efficiency and faster detection.
  - Consider integrating threat intelligence feeds from other sources to broaden coverage.
- Training:
  - Provide training for the SOC team on advanced threat detection techniques, Splunk capabilities, and effective use of threat intelligence.
  - Conduct regular exercises to maintain the team's proficiency in incident response.
- Continuous Improvement:
  - Regularly review and update security configurations based on emerging threats.
  - Monitor and evaluate the effectiveness of the security infrastructure.
  - Implement a continuous improvement process to identify and address areas for enhancement.

- Iterate over the testing scenarios to make the detection more difficult for the red team. Consider adding more complex scenarios.
- Automate testing scenario set up with infrastructure as code tools such as ansible or terraform.

Second Iteration: Encrypted Beacons

Scope

For the second iteration we decided to focus on beacons over https.
We added a new test scenario to the SOC testing plan, which involves beaconing activity over HTTPS.

Initial assessment

We updated the Splunk environment to include a new correlation search that detects beaconing activity over HTTPS, but we were not very successful in detection without a lot of false positives.

ja4 Hashes

We added ja4 hashes to our detection capabilities through zeek installed on the pfsense firewall machine. Suricata has the capability to add ja4 hashes in the latest version, but is not yet available in the package manager. In contrast zeek supports the full suite of ja4+ hashes out of the box.

Modifications

- ja4 Package Installation: We installed the ja4 package for Zeek, enabling the tool to generate ja4 hashes for HTTPS connections. Ja4 hashes are cryptographic fingerprints that capture the cryptographic configuration of SSL/TLS connections.
- ja4 Hash Analysis: We analyzed ja4 hashes generated by Zeek to identify potential indicators of malicious activity. By examining the cipher suites, elliptic curves, and other cryptographic parameters captured in the ja4 hashes, we could identify patterns associated with known beaconing frameworks or threat actors.
- ALPN Protocol Analysis: We analyzed the Application Layer Protocol Negotiation (ALPN) protocol used in HTTPS connections present in ja4 hashes. We found that searching for ALPN protocols that were not commonly used by legitimate applications (such as "00" or custom protocols) proved effective in identifying potential beaconing activity.
- TLS Certificate Analysis: We implemented techniques to analyze TLS certificates used in HTTPS connections. By examining certificate details, we could identify suspicious patterns, such as self-signed certificates or certificates issued by unknown authorities.

- Domain Name Analysis: We analyzed domain names used in HTTPS connections to identify potential indicators of malicious activity. This included checking for suspicious domain names, typosquatting, or domain names associated with known threat actors.

Through these modifications, we were able to effectively detect encrypted beaconing activity over HTTPS, including the beaconing traffic originating from the Caldera C2 server. However, we also identified benign beacon-like behavior coming from legitimate applications such as Splunk and Tailscale, highlighting the possibility for false positives.

Recommendations

To improve the accuracy of our encrypted beaconing detection and reduce false positives, we recommend the following:

- Establish a Baseline of Known Good Hashes: Create a baseline of ja4 hashes generated by legitimate applications and trusted servers. This baseline can be used to filter out benign traffic and focus on more suspicious activity.
- Contextual Analysis: Consider the broader network context when analyzing ja4 hashes, ALPN protocols, and other indicators. Examine factors such as source and destination IP addresses, network traffic patterns, and other relevant information to determine if the detected activity is consistent with known benign behavior.
- Threat Intelligence Integration: Leverage threat intelligence feeds to identify known malicious ja4 hashes, ALPN protocols, or domain names. This can help filter out benign traffic and focus on more suspicious activity.
- Continuous Monitoring and Updates: Regularly review and update detection methods to stay current with emerging threats and best practices. This includes updating threat intelligence feeds, refining detection rules, and evaluating the effectiveness of detection techniques.
  By implementing these recommendations, we can enhance the accuracy of our encrypted beaconing detection and reduce the number of false positives, ultimately improving our ability to protect our network from malicious attacks.