**Blue Team Documentation**

Some notes on the process and self-discovery questions to answer while analyzing ELK Stack logs in Kibana.

**Identifying the offensive traffic.**

Identifying the traffic between Red Team machine and the web machine:

KQL Search:

http.response.status\_code: 201

* When did the interaction occur?
  + July 2nd @ 00:00:44.000
* What responses did the victim send back?
  + 201 response which is a successful PUT request meaning a file was created.
* What data is concerning from the Blue Team perspective?
  + Metricbeat results:
    - This came from a Kali machine which is a known pentesting OS. (agent.name)
    - The PUT request (adding shell.php)
    - Cadaver to access WebDav (weird program/traffic not a typical user)
  + Filebeat results:
    - Apache access
    - 207 confirmed/file created on server
    - Username ryan (someone did this under ryan’s account - compromised account)
    - Source IP 192.168.1.90

**Finding the request for the hidden directory.**

KQL Search:

url.path: "/company\_folders/secret\_folder"

In the Red Team attack a secret folder was found. Time to look at that interaction between these two machines. Some questions to answer:

* How many requests were made to this directory? At what time and from which IP address(es)?
  + 14,355
  + hits were found depending on filebeat or packetbeat. These GET requests happened at 23:34 from one IP being 192.168.1.90.
* Which file(s) were requested? What information did they contain?
  + The access.log by ashton. This file contains records of all requests processed by the server (Apache).
* What kind of alarm would you set to detect this behavior in the future?
  + Base the alarm from the 401 response codes and if there are too many unauthorized attempts.
* Identify at least one way to harden the vulnerable machine that would mitigate this attack.
  + Account lockout after 10 attempts, add 2 factor authentication, restrict devices and IPs.

**Identifying the brute force attack.**

KQL Searches:

url.original (works as well)

url.path: "/company\_folders/secret\_folder" and http.request.method: "get" and http.response.status\_code: 401

url.path: "/company\_folders/secret\_folder" and http.request.method: 401

After identifying the hidden directory, you used Hydra to brute-force the target server. Some questions to answer:

* Can you identify packets specifically from Hydra?
  + The packets are those with user\_agent.original: Mozilla/4.0 (Hydra).
* How many requests were made in the brute-force attack?
  + 10,148 to 14,349
* How many requests had the attacker made before discovering the correct password in this one?
  + 10,147 or 14,348 (one less than because this last was a success)
* What kind of alarm would you set to detect this behavior in the future and at what threshold(s)?
  + User agent Hydra or HTTP Response code of 401 as well limit the amount of bad login attempts over 100.
* Identify at least one way to harden the vulnerable machine that would mitigate this attack.
  + Block any user agent that is Hydra so defined as a firewall rule. Then account lockout after 20 bad login attempts as well as add an alternate authentication method.

**Finding the WebDav connection.**

KQL Search:

url.path: \*webdav\* count

Using the dashboard to answer some questions:

* How many requests were made to this directory?
  + 183 hits (packetbeat)
* Which file(s) were requested?
  + shell.php and passwd.dav (seen in visualize and with added query from Available Fields)
* What kind of alarm would you set to detect such access in the future?
  + For files that are scripted related to the server side. Checking file extensions for something like .php. Also, for when a file is created on the server.
* Identify at least one way to harden the vulnerable machine that would mitigate this attack.
  + Limit the devices/machines that have access to this server. So whitelisting IPs for example or keeping it within the company network.

**Identifying the reverse shell and meterpreter traffic.**

KQL Search:

destination.port: 80 (with both destination ip and destination port -visualization)

destination.ip: 192.168.1.90 and destination.port: 4444

url.path:"/webdav/shell.php"

To finish off the attack, a PHP reverse shell and a meterpreter shell session was started. Some questions to answer:

* Can you identify traffic from the meterpreter session?
  + You can see that the shell.php was PUT there and that there are GET requests for it. However, the meterpreter session is not clear because of how meterpreter encrypts traffic. There is not information on when it was executed other than the GET request for it.
* What kinds of alarms would you set to detect this behavior in the future?
  + Set off an alarm when new outbound traffic is detected as well as when a new machine is connected.
* Identify at least one way to harden the vulnerable machine that would mitigate this attack.
  + IDS to pick up on when there is an original outbound connection to a new port/machine. 4444 was seen with traffic as a port which was the default for meterpreter. A rule can be set to block traffic on that port.