

Practical Network Defense

Fourth Assignment - University “La Sapienza”

Group 27: Nicola Bartoloni ***** - Valerio Trenta 1856471

2020-23-06

1 Initial comments and scope

This last assignment is meant to provide an additional level of defense to the **ACME CO. network** by setting up **Intrusion Prevention Systems**, both hosts-based and network-based, in the two firewalls of the network and in the hosts of the **DMZ** subnetwork.

Once these systems are fully operational, a **vulnerability assessment** is then carried out to find, examine and pinpoint the vulnerabilities that are still affecting the services that the network provides.

Given the goal of the **NIPS** that we are going to setup in the two firewalls - that is, to detect and prevent intrusions in the whole network, especially the one set up at the **Main firewall** - and given the fact that we have to consider all the services that the target network provides, the scope of the assignment is the entire network.

Please keep also in mind that the vulnerability assessment has actually its own scope and scenario, which is described in the fourth chapter.

2 Intrusion Prevention Systems setup

It is explicitly required in the assignment to actually set up **Intrusion Prevention Systems**, so that they can not only detect intruders raising an alarm, but also perform actions in order to prevent the intrusion, like dropping suspicious packets. We are going to set up two different systems: in the two firewalls, we exploit the **suricata** service provided by the **OPNSense** administration panel to build a **Network Intrusion Prevention System**, while in the two hosts of the **DMZ subnetwork**, we download and configure the **fail2ban** tool to build a **Host Intrusion Prevention System**.

The differences and analogies between the two systems will be clear in the next two subparagraphs, as well as their configurations and functionalities.

2.1 suricata

The **suricata** service can be enabled in the **OPNSense** administration panel by navigating to *System -> Services -> Intrusion Detection -> Administration*. For both routers,

we enable the service and we also check the box **IPS** to make it possible for the service to actually *drop packets* matching a rule, so that it will not only raise an alarm, but also perform some concrete action in preventing the intrusion.

For both of them we also enable the **promiscuous mode** so that the systems will also monitor and check the rules for packets that are not strictly directed to one of their interfaces - so, obviously, forward packets will also be checked - and the **logging** service.

In the **Main Router**, we enable the service for *all the interfaces, but the WAN interface*. This latter interface indeed is probably the one seeing the most of the traffic, being connected to the Internet, so we decide to run the system only at the interfaces directly connected to our services, so that when packets get there they have already been checked against the WAN interface's firewall rules and the interface itself won't be a bottleneck to the whole network just by checking every packet to every single rule of the **NIPS**.

In the **Internal Router**, the very same logic is applied and thus the service is only enabled on the **CLIENTS** and **SERVERS** interfaces.

Both systems in both firewalls are configured to update their rules daily according to a given schedule. Both of them will drop packets matching against one of the rules that have been selected.

The rulesets that are enabled follow below - and they are almost the same for both to apply defense-in-depth for the most internal firewall:

- all the rulesets from **Abuse.ch**, except *Feodo Tracker*. These rules should detect compromised websites distributing malware, compromised and bad SSL certificates and such;
- **ETOpen Attack-Response** catching common results of successful attacks;
- **ETOpen botcc** for detecting Botnets and C2 hosts;
- **ETOpen DOS** rules for DoS attacks;
- **ETOpen DROP** rules for spam activities;
- **ETOpen Exploit** and **ETOpen Malware** for detecting general exploits and malwares;
- **ETOpen Inappropriate** rules for detecting accesses to inappropriate websites;
- **ETOpen Policy** rules containing checks for things that are usually not enabled by a company policy - that seemed our case;
- **ETOpen Scan** rules to prevent network scans, probes and in general reconnaissance activities;
- **ETOpen Web rules** - *client* in the Internal firewall, *client and server* at the Main firewall, for web application security and prevention of common attacks - such as SQL injection, XSS;

- finally, we also included some rules from the **OPNSense Suricata Application Detection**, namely for **file-transfer** and **mailing** applications - once again, it seemed our case to include these rulesets given the network belongs to a company, and clients are likely to be exploiting such applications.

The two routers do not seem to be overloaded by the rulesets that were applied, even if they are quite a lot and some might also be unnecessary - **botcc** and maybe the **Inappropriate** rules, the latter especially given the proxy and DNS service we set up previously.

2.2 fail2ban

The **fail2ban** tool can be downloaded on both the two machines running in the **DMZ**, namely the **proxy machine** and the **web machine**. We have three different services running on these machines, whose access we want to protect with this tool:

- **SSH** service running on both;
- **apache2** service running on the **web machine**;
- **squid** service running on the **proxy machine**;

so we copy the content of file `/etc/fail2ban/jail.conf` in a new file, `jail.local` under the same directory, which will be now used by **fail2ban** as configuration file for its jail. We leave unmodified the preconfigured values for parameters such as *bantime* and *maxretry* - 3600 and 5 - and we do not specify in neither of the two configurations for the two machines IP addresses that we want to ignore in the field *ignoreip*, apart from the local loop, so that especially in the case of the **squid** service, we will monitor also accesses coming from the Clients subnetwork, so to prevent possible physical intrusions on the machines of the aforementioned subnet which may then lead to a misuse of the proxy service. The tool will then monitor the log files of the *enabled* services in the `jail.local` file to detect possible violations of authentication on those services - e.g., brute force attacks - and **ban** the hosts which made a number of attempts to authenticate that is $> \text{maxretry}$. To each services that we wish to monitor through the tool, we add the string *enabled = true*. These services are:

- **sshd** in both the machines;
- **apache2** and related lines in the `jail.local` file for the **web machine**;
- **squid** for the **proxy machine**.

We enable and restart the **fail2ban service** through command line `service fail2ban restart`. To check the list of jailed hosts per each service, we can execute command `fail2ban-client status <service>`, as pictured in the figures below.

Please keep also in mind that we know that at least one more process is running on both machines - the **rsyslog** process for remote logging at log machine in the Internal

```

root@webserver:~# fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:    2
|  `-- File list:      /var/log/auth.log
- Actions
  |- Currently banned: 0
  |- Total banned:    0
  `-- Banned IP list:
root@webserver:~# fail2ban-client status apache-auth
Status for the jail: apache-auth
|- Filter
|  |- Currently failed: 0
|  |- Total failed:    0
|  `-- File list:      /var/log/apache2/error.log
- Actions
  |- Currently banned: 0
  |- Total banned:    0
  `-- Banned IP list:

zentyal@proxyserver:~$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:    0
|  `-- File list:      /var/log/auth.log
- Actions
  |- Currently banned: 0
  |- Total banned:    0
  `-- Banned IP list:

zentyal@proxyserver:~$ sudo fail2ban-client status squid
Status for the jail: squid
|- Filter
|  |- Currently failed: 0
|  |- Total failed:    0
|  `-- File list:      /var/log/squid/access.log
- Actions
  |- Currently banned: 0
  |- Total banned:    0
  `-- Banned IP list:

```

Figure 1: fail2ban jails for services in the Web machine (apache-auth is just one of the several apache jails we enabled).

Figure 2: fail2ban jails for services in the Proxy machine.

servers - but this does not require any authentication mechanism so it is not monitored by the tool.

3 Engaging the Assessment

4 Vulnerability Assessment

4.1 Results

4.2 Analysis

5 Final Remarks