The problem

Performing a security audit of your infrastructure, website or services once a year is a great first step, but it's probably not enough. Even doing it every six months could lead to several security holes. And besides that, if the audit only involves one or a few tools, it could lead to vulnerabilities discovered by other programs untouched and uncovered.

The solution

That's why the Faraday platform offers you the possibility to do continuous scanning using almost all the auditing tools on the market. The goal of this page is guiding you through setting up Faraday to do a scan every week or after some event, all with different tools and obtaining all the results on your Faraday platform.

While this is not (*yet*) a replacement for regular manual security audits (the software is not quite there yet), by implementing continuous scanning you can pick a lot of the low hanging fruit, leaving only the really interesting stuff for real people.

The guide

Setting up continuous scanning

The script is located in `$FARADAY/scripts/cscan/`. The main script is `cscan.py`.

Mind you may need to install the dependencies (`pip install python-owasp-zap-v2 w3af-api-client`) and to use burp you need the include the plugin plugin/carbonator/carbonator.py, necessary to adapt it to our implementation.

The following tools are available:

- w3af
- nmap
- nikto
- burp
- zap
- nessus
- openvas

The idea is to use a set of scripts together with these tools and Faraday so we can export all the data to our databases. Each report must be copied to `$HOME/.faraday/report/[workspace_name]`. Faraday will then convert all the reports into valuable information, as it does when using it manually. This is all done automatically by cscan.

To give you an idea of how cscan works, you can think of it like this:

```
./cscan.py: #execute each script inside ./scripts/network/ and ./scripts/web/
./scripts/web #directory for web tools
```

```
./scripts/network #directory for network tools
./output #temporary directory where the reports are generated
./websites.txt #Website list
./ips.txt #IPs/Networks list
./plugin #plugin or library necessary for ./scripts/
./config.py #global configuration
```

And this is the nmap script:

```
./scripts/network/nmap


NAME="nmap_$(date +%s).xml"


${CS_NMAP:=nmap} -iL  -oX ./scripts/network/nmap NAME="nmap_$(date +%s).xml"
${CS_NMAP:=nmap} -iL $1 -oX $2$NAME $NAME
```

As you can see, cscan is much like a wrapper for a lot of smaller, simpler scripts that take advantage of a tool.The ips.txt and websites.txt are files where the intended targets should be listed.

After listing the intended targets and reviewing the config.py file, you should set a schedule. Cron is an excellent choice to schedule the audits. This is a simple example of a crontab line where you run an automatic audit every day at midnight:

```
# crontab -l


0 0 * * * bash /root/dev/cscan/cscan.py ; mv /root/dev/cscan/output/*
/root/.faraday/report/workspace_name/
```

Another options is to configure the scripts with Jenkins. For example, you could set it up so every time a new merge or release is detected, Jenkins would run the scripts.

Using the new information

Each time the scan is run Faraday will incorporate the new information. This can be overwhelming at first, so you should try to organize it. Using the Web UI, this is a breeze. You can set tags for every vulnerability, mark it as confirmed or a false positive, establish the priority, and pretty much everything you want.