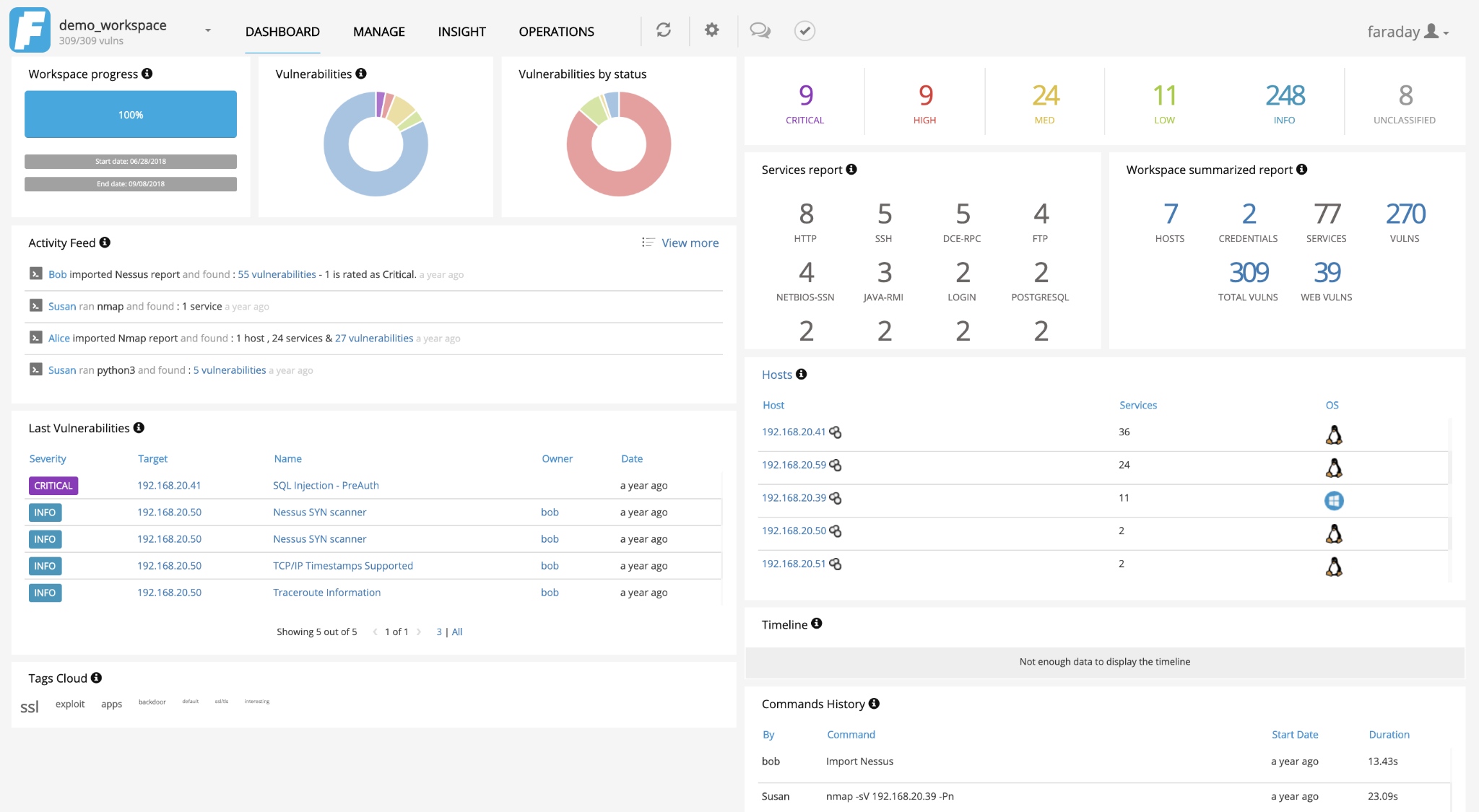
**By: Dylan Fernald**

Pentesting and red teaming is becoming more and more needed in a constantly changing cyber world. Company infrastructure is constantly being attacked, whether it be passively or actively. Therefore, it is crucial to continuously test infrastructure to make sure proper security measures are in place. That being said, the pentesting process has a large intermediary waiting time from the test engagement to the completion of the report. It takes a long time for a pentester to compile and keep track of their findings, and this long period can be shortened with the use of Faraday.

**Overview**

Faraday is a pentesting collaboration tool made by InfoByte. It’s goal is to help expedite the pentesting process in a variety of ways. The Faraday developers consider Faraday to be an “Integrated Penetration-Test Environment, or IPE. It is designed for distributing, indexing, and analyzing the data generated during a security audit [1]”.



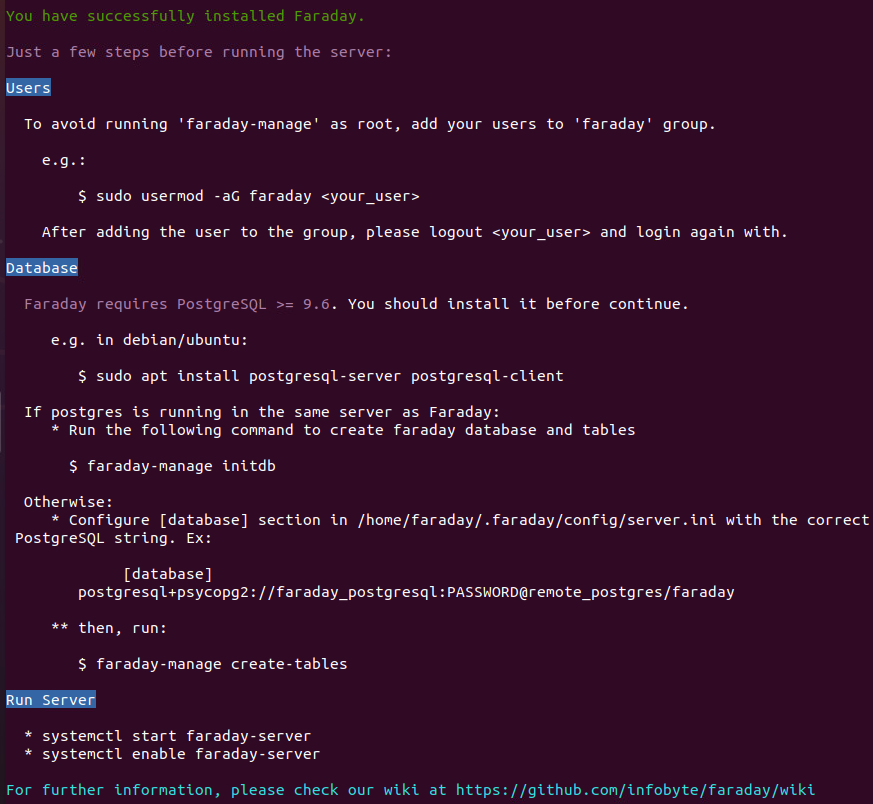
**Installation**

First off, make sure the machine you wish to install Faraday on has Python 3 and a PostgreSQL Database Server installed and configured properly. Then you can begin installing the latest version of Faraday. As of writing this, I am installing version 3.18.1 of Faraday on my Ubuntu 20.04 machine. By using wget, I was able to download the latest installer from Faraday’s GitHub repository.

Next, I used the following command to set it all up.



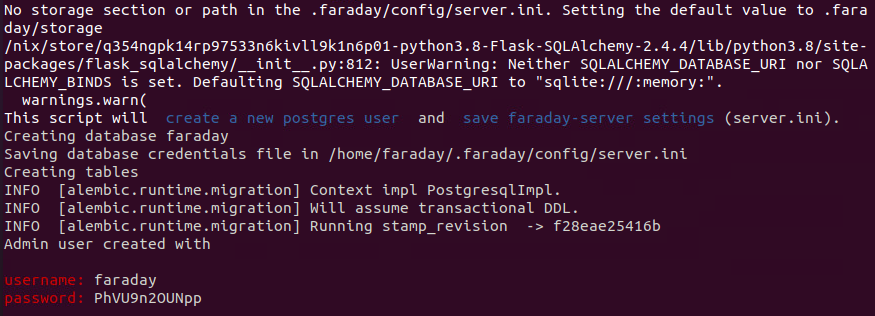
Faraday then gives us some helpful confirmation and information.



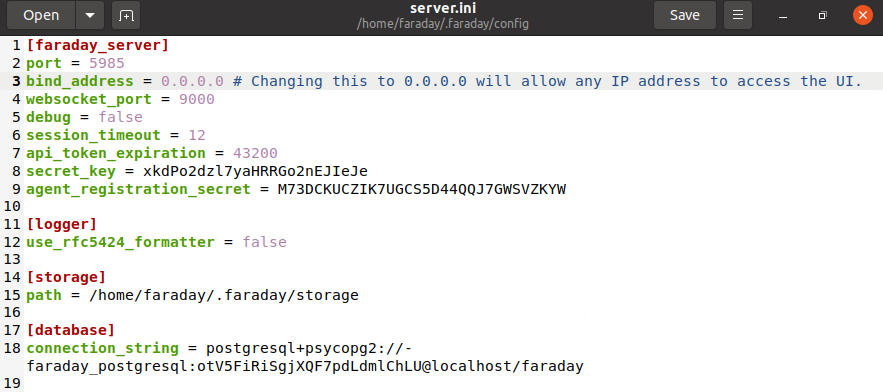
Now we need to initialize the postgres database by using the following command:

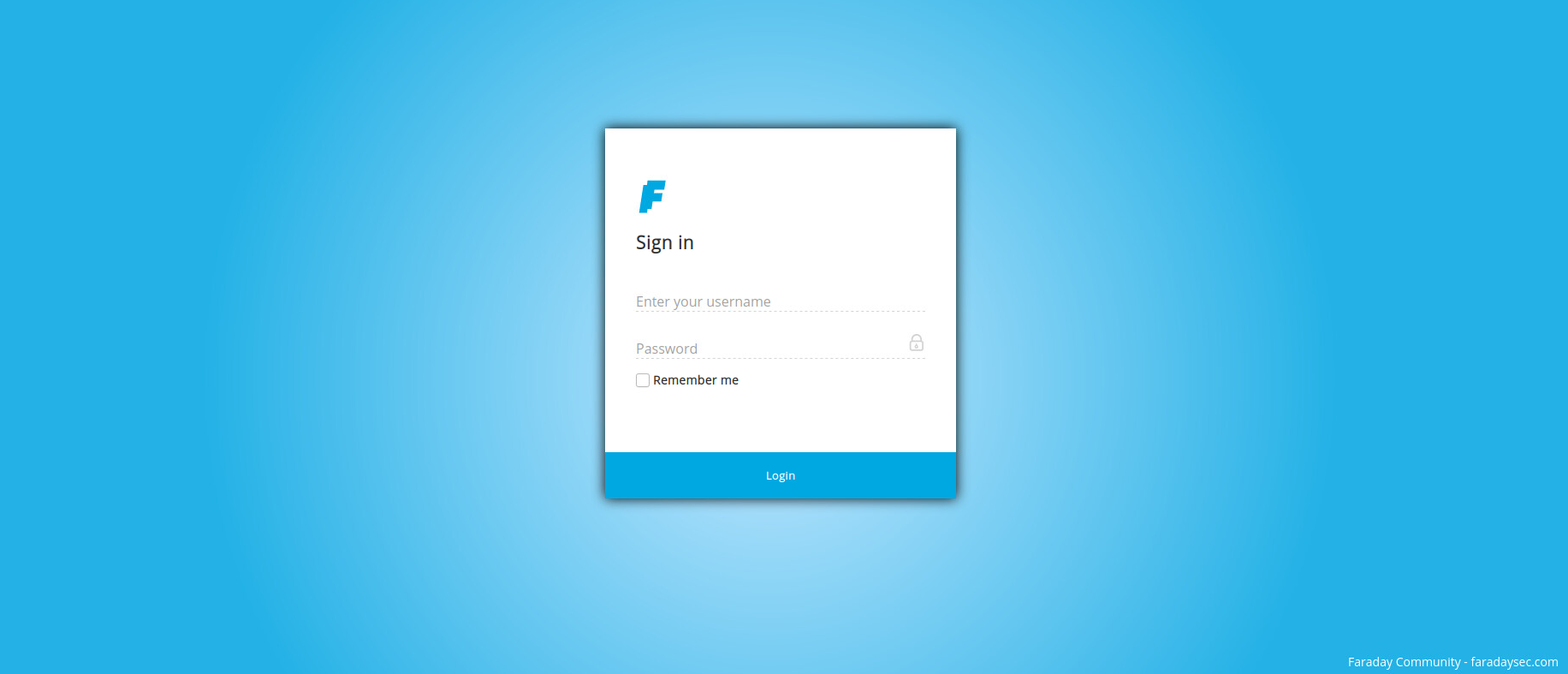


Pay attention to the password that the script creates for you. This will be used to access the Web UI.



By default, Faraday is not configured to be accessed remotely, since it only listens on localhost. This can be changed in the config file located at ***/home/faraday/.faraday/config/server.ini***.

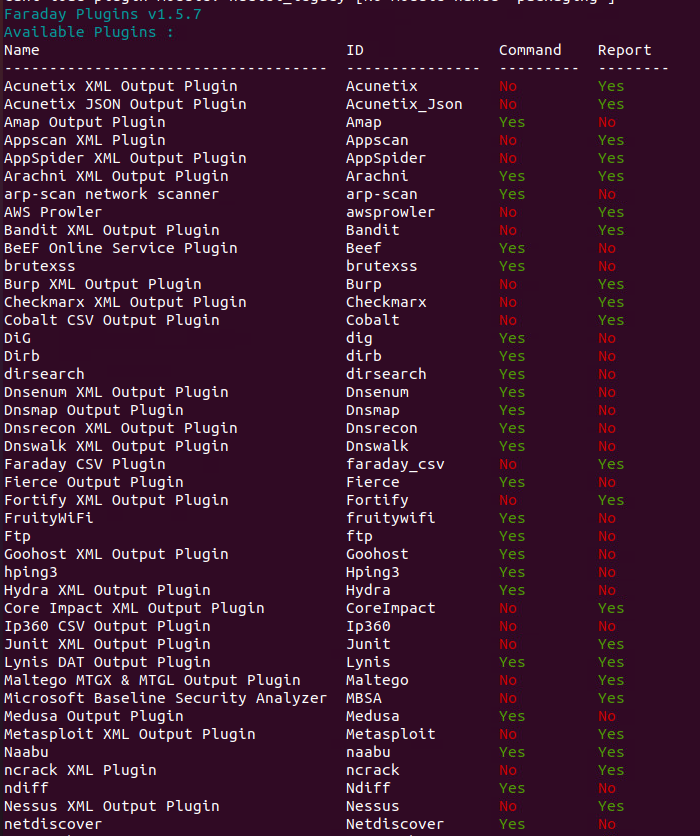


Faraday should now be accessible at ***localhost:5985*** and remotely as well. You can use the randomly-generated password to login with the ***faraday*** account.

Before progressing any further, you’ll want to install ***faraday-plugins*** so that Faraday can recognize tools such as nmap. Use the following command to do so.



You can view the available plugins with the ***faraday-plugins list-plugins*** command. This list will tell you if there is support for the tool’s commands via faraday-cli and if the tool is supported within report generation.

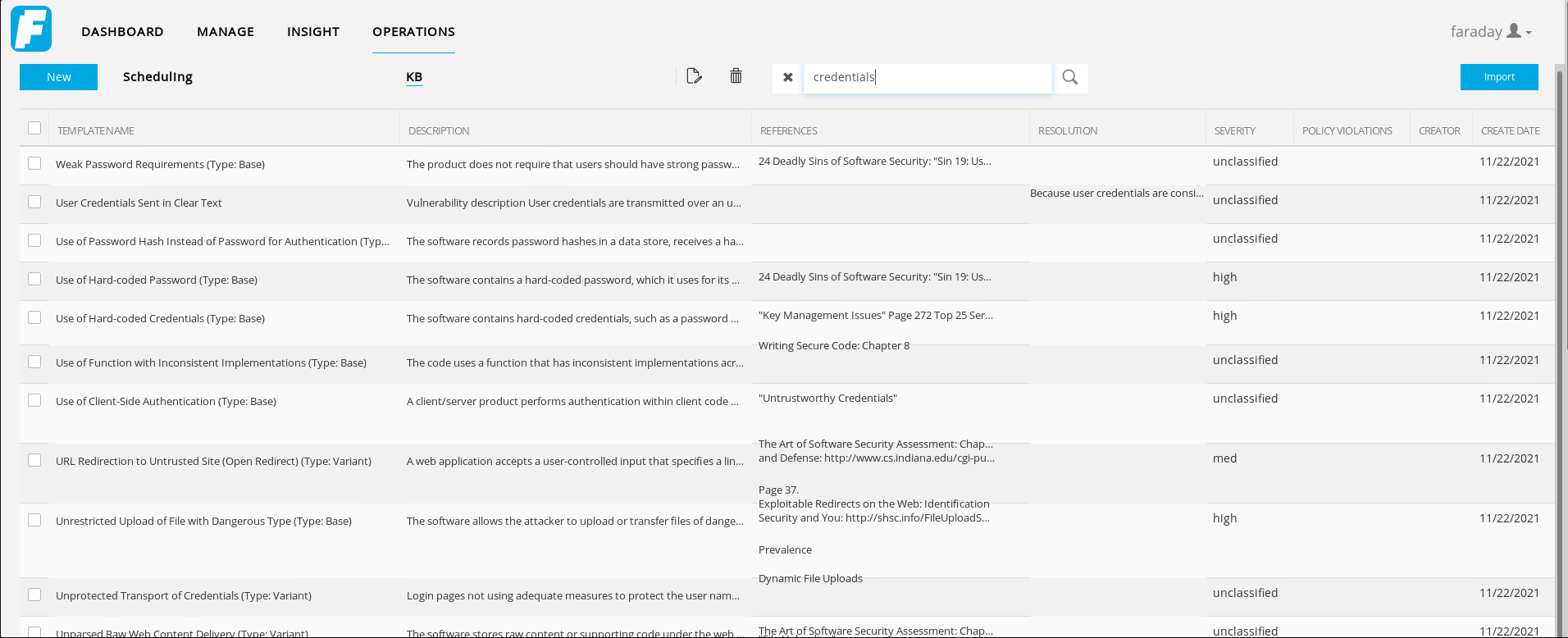


In addition to plugins, Faraday comes shipped with a vulnerability template database. These are pre-made vulnerability templates that can be imported into Faraday for ease of use later. These templates come with a vulnerability name, description, reference(s), resolution(s), severity, and more. You can alsc create your own custom vulnerability templates.

In order to import the vulnerability templates, use the following command.



This will populate Faraday with a vast amount of vulnerability templates, from weak credentials to SQL injection vulnerabilities.

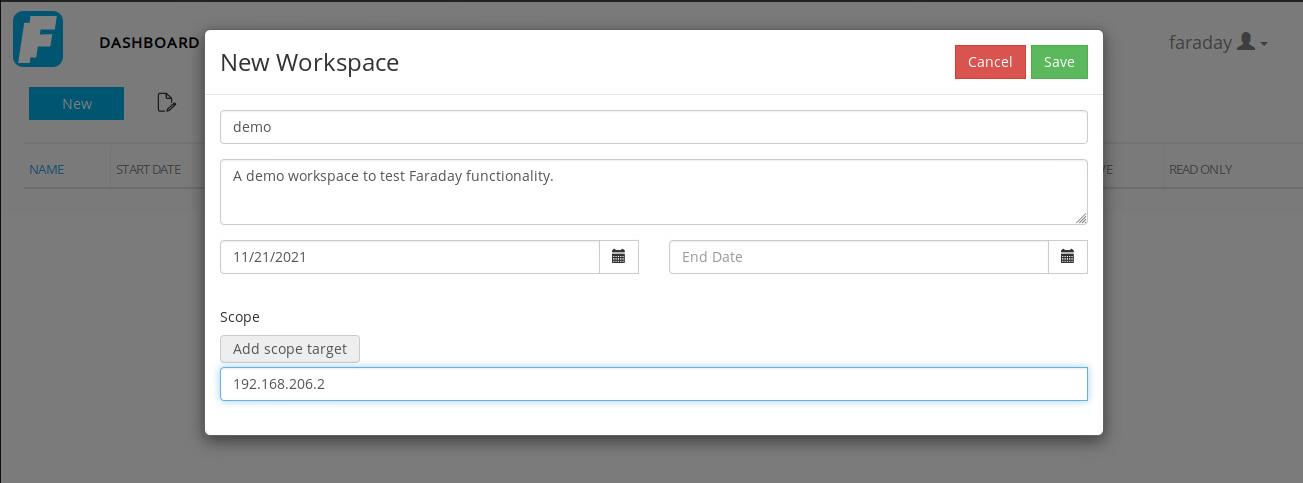


The ***faraday-manage*** tool also comes with other useful commands. Use the ***-h*** option to see them if you are curious.

Once logged into the Web UI, you will notice the dashboard is pretty empty, since we have just begun. Faraday can organize engagements via workspaces. These workspaces are where pentesters would compile an engagement’s intel that has been discovered.

Create a new workspace by clicking on the blue button that says “New” when at the Faraday Dashboard. You can give the workspace a name, description, start date, end date, and scope or targets. From here on out, I am going to move to a Kali machine to resemble how a pentester might interact with Faraday.

Below you will see the workspace configuration that I have chosen for this example. The target scope is going to be the Faraday server for simplicity sake. In reality, this can be any host(s) that you are testing.

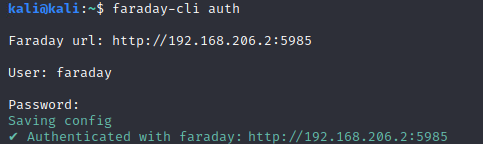


Okay, so now that we have everything regarding the server set up and configured properly, it’s time to move on to the client. Faraday has a separate client tool called ***faraday-cli***. This tool allows a pentester to use Faraday right from the terminal. According to the Faraday docs, ***faraday-cli*** is “the official client that makes automating your security workflows, easier [2]”. The ***faraday-cli*** documentation contains some examples of how you might automate scans on a host. But before we delve too far into the client’s functionality, we have to install it first.

To install ***faraday-cli***, simply use the following command.



The next step is to authenticate your client with your Faraday server.

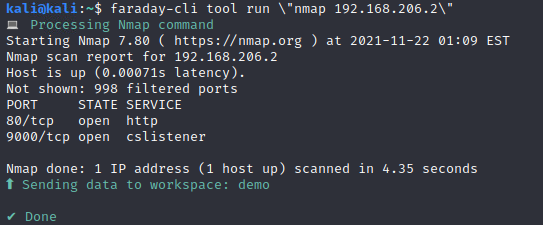


Then, select the demo workspace we made.



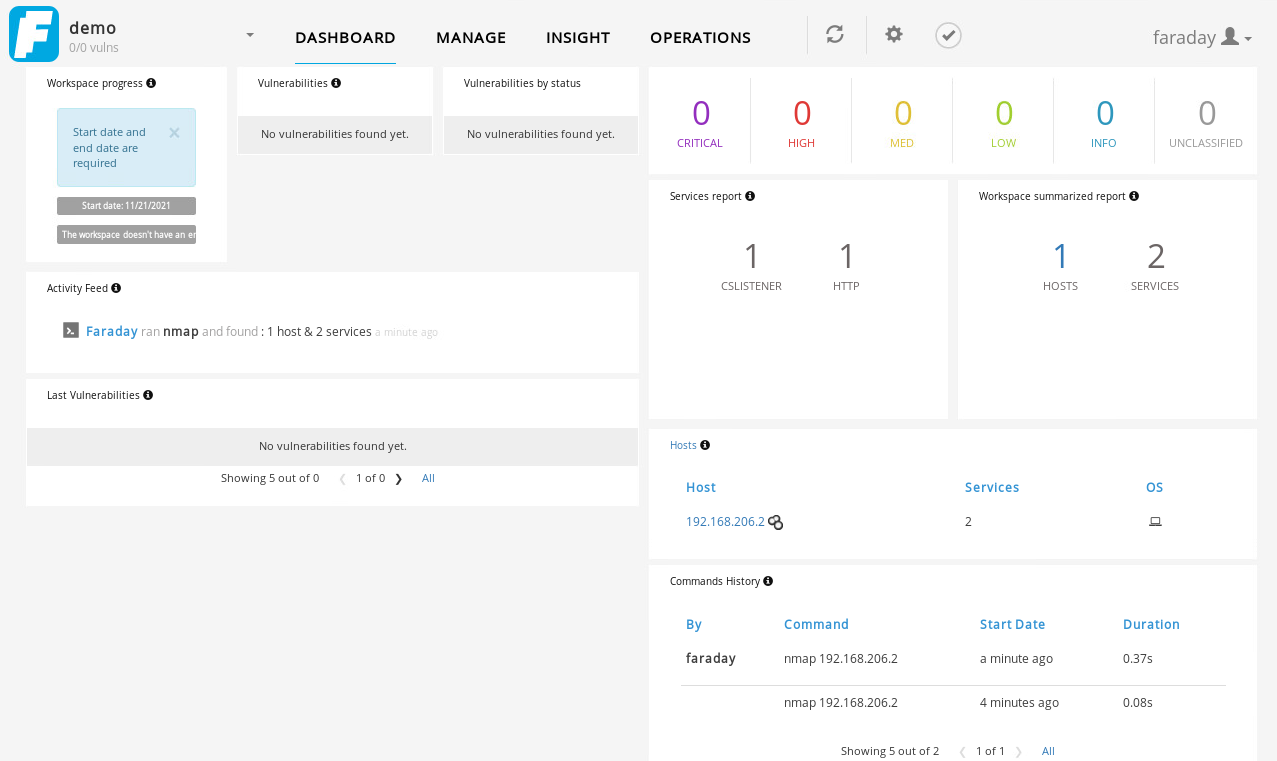
**Using Faraday**

Now we are ready to use Faraday to its full potential. The Faraday Client documentation covers a lot, so I’m going to highlight some of my favorite features. First off, you can either use ***faraday-cli*** with individual commands, or as a shell. For instance, if you wanted to nmap a host via an individual command, you could do something like this.



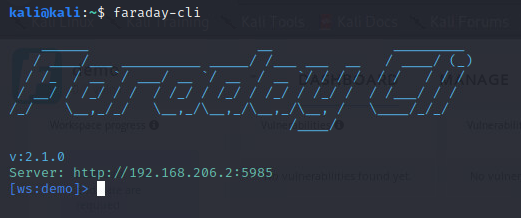
The client processed the nmap command and sent the output to the Faraday server. Let’s check it out.

On the Faraday Dashboard we can see what commands were run, what time they were run, and how long they were running for. Faraday kept track of the scanned host, and how many services were exposed.

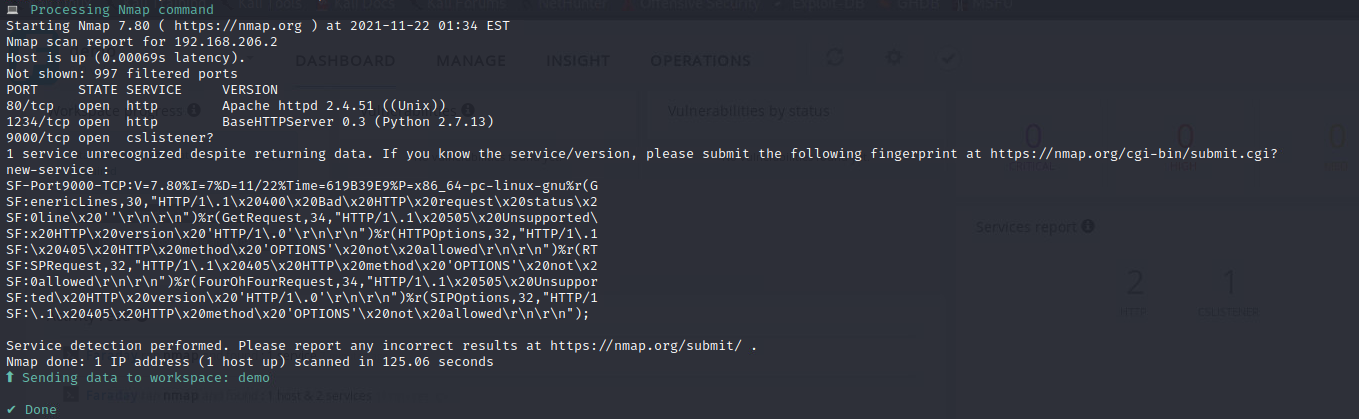


Now, I am going to set up a vulnerable web application (courtesy of AppSecco [4]), and see how Faraday keeps track of vulnerabilities automatically.

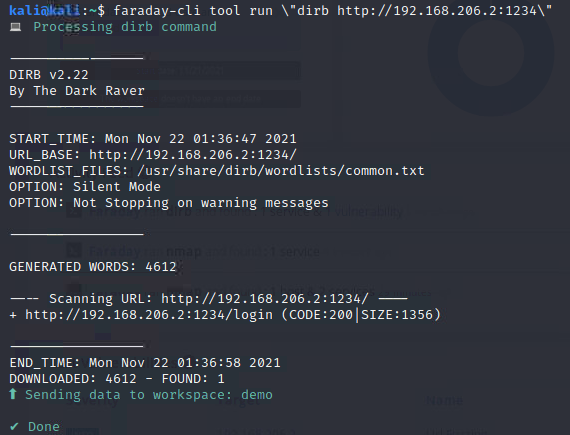
Instead of using individual commands, you can also use the shell feature of ***faraday-cli***. To enter the shell, using the ***faraday-cli*** command without any options.



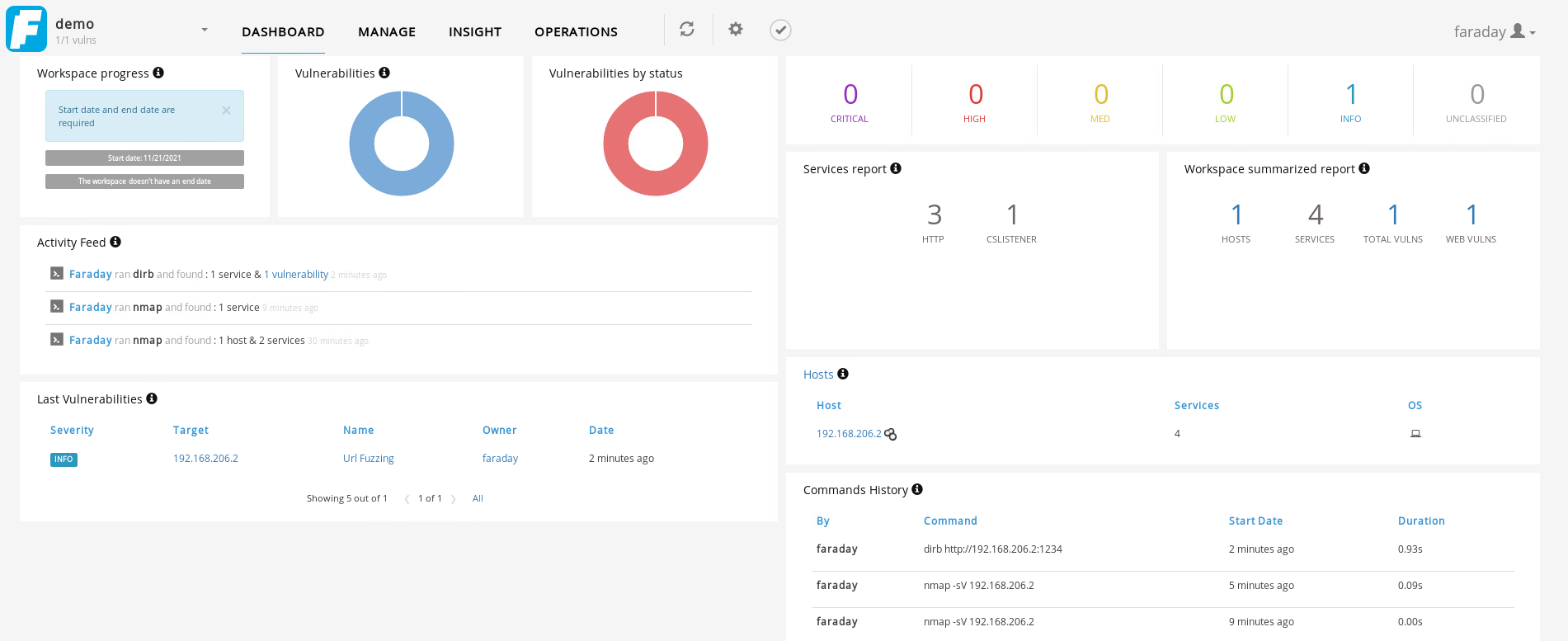
Notice how the client recognizes the Faraday Server we are authenticated with, in addition to the workspace we are currently using. Now, I’m going to do another simple nmap scan on the same host.



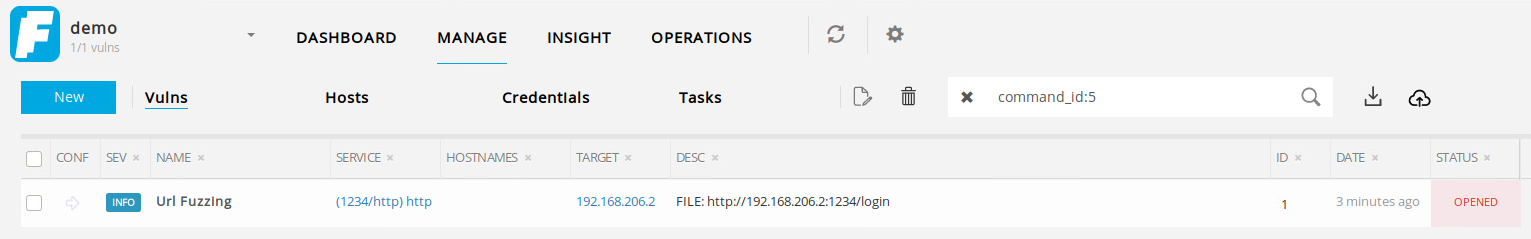
Now that we have discovered the vulnerable web application on port 1234, let’s use dirb to try and find some directories.



We were able to find a login page of some sort. Let’s see what information was sent to Faraday.

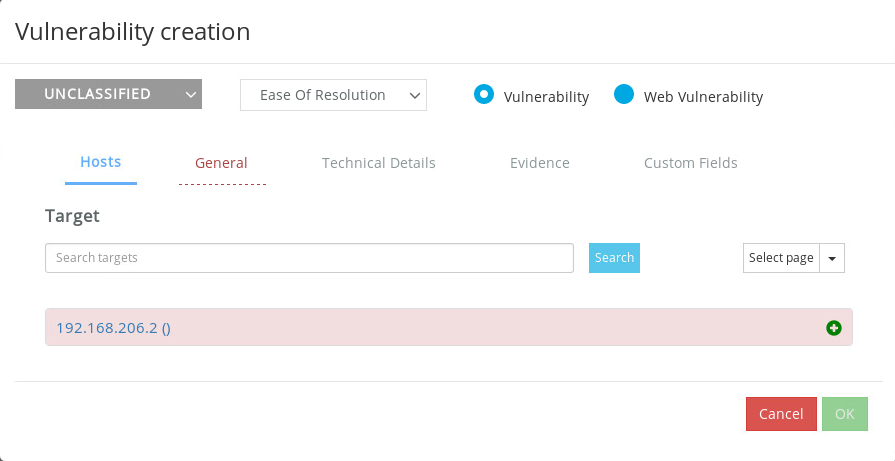


The URL Fuzzing vulnerability that we found was automatically imported into Faraday, with the host, location, service, and more already compiled and saved.

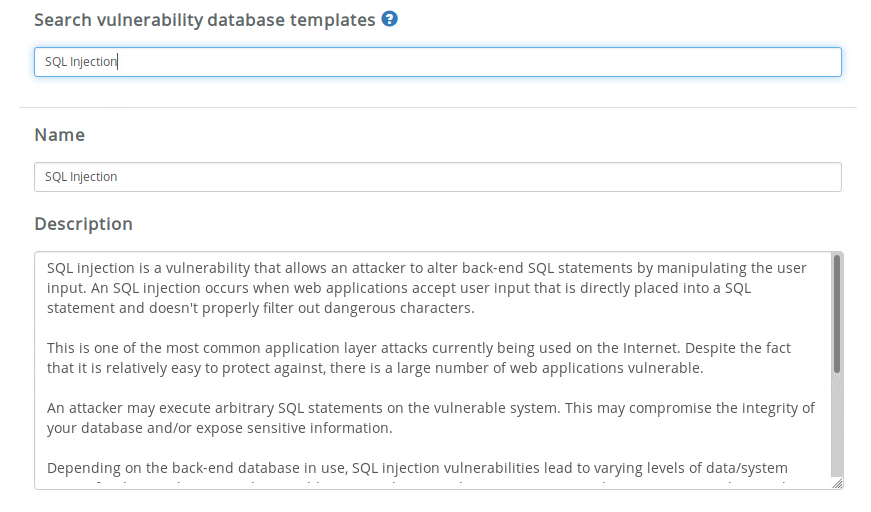


Now, using commands straight through ***faraday-cli*** only works if there is a corresponding plugin. Currently, Faraday has over 700 plugins, but that doesn’t cover every tool. In this case, what should a pentester do if they use, say, sqlmap to find a SQL Injection vulnerability? Well, they could create their own vulnerability within the Faraday workspace.

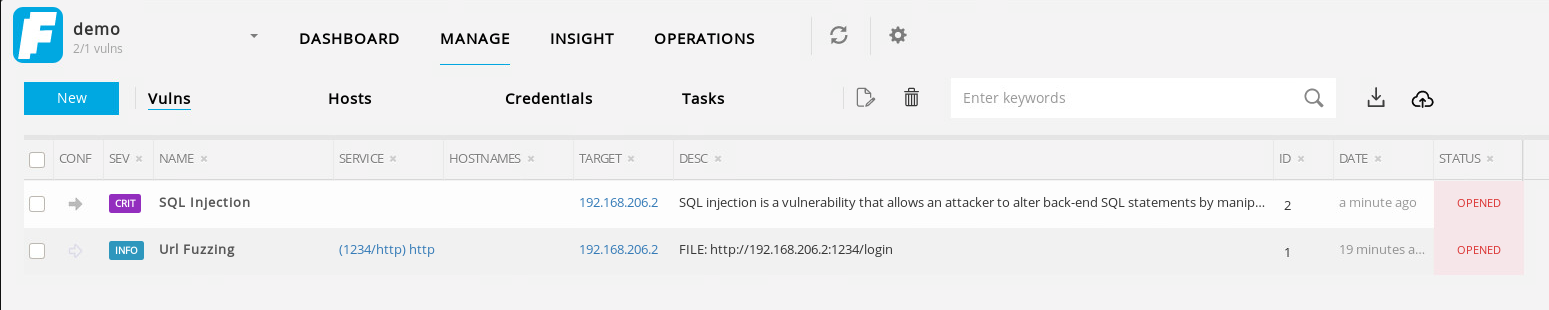
By navigating to the “Manage” tab of the Web UI, you can create a new vulnerability for the workspace. Not only does the vulnerability creator allow you to make a custom, brand new vulnerability, but it also has access to the template we imported earlier.



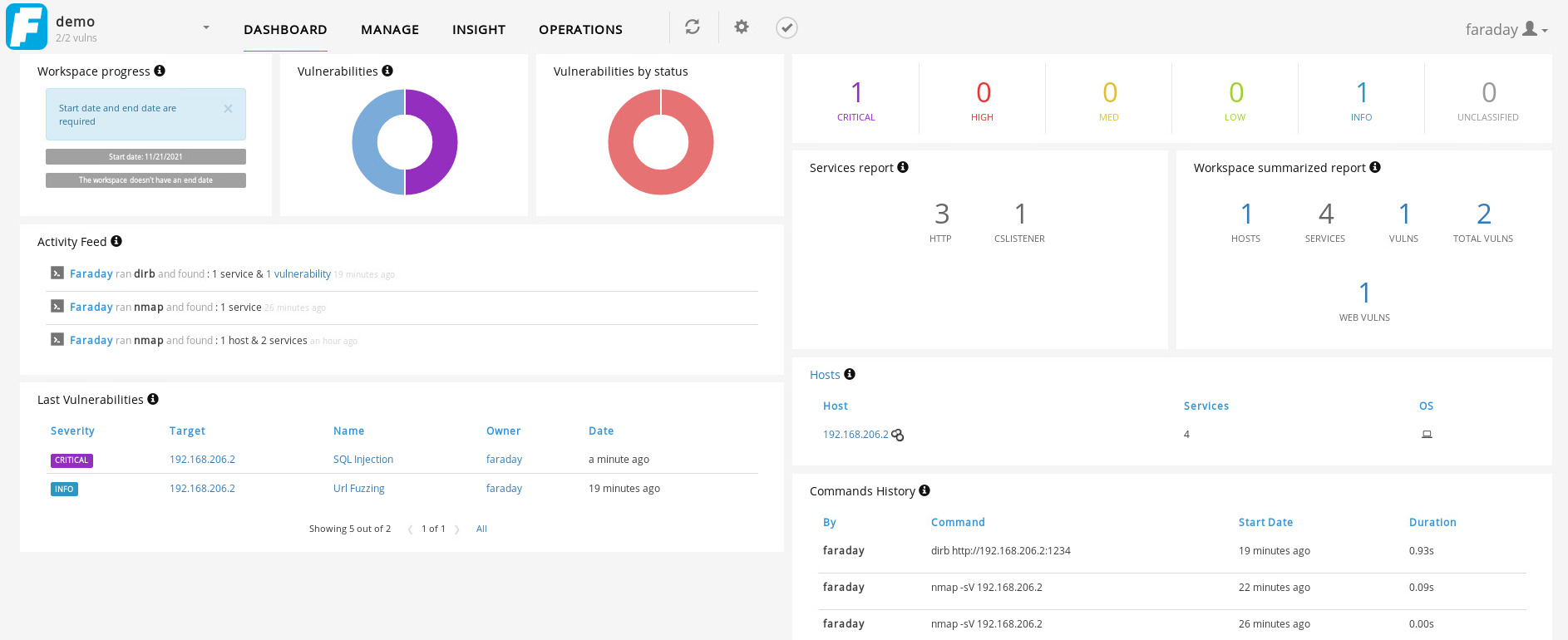
The “General” tab allows you to search for vulnerabilities by name, so let’s search for a SQL Injection vulnerability.



The SQL Injection vulnerability template gets loaded into the creator complete with a description, resolution method, and impact tags such as accountability, availability, confidentiality, and integrity. You can also select the ease of resolution, and severity of the vulnerability. Below you will see the vulnerability within the workspace.



If we go back to the dashboard we can see the charts have been updated accordingly.



**Commercial Version**

By now you can probably see the benefit to using Faraday as a pentester and how this collaboration tool helps pentesters be more efficient. Well, that’s not all Faraday is capable of.

All of what I’ve shown so far has been part of the community version of Faraday. The commercial version of Faraday offers even more features, such as multiple user accounts and report generation. With multiple user accounts, a team of pentesters is able to keep track of what areas of testing have already been covered in the engagement. The report generation greatly reduces the time it takes pentesters to write both a technical and high-level report. Since I am only using the community version, I am unable to show off these features. Faraday’s website has more information regarding these features. [5]

**Conclusion**

Pentesting is crucial to ensuring an up to date and protected environment. However, the overall length of engagements is not ideal. Faraday seeks to improve this by enhancing security tools and scaling teamwork. It allows pentesters to normalize and integrate security tools, collaborate, automate vulnerability response to remediate faster, and automatically generate reports based on their findings. All of these features are why Faraday is useful for expediting the pentesting process.

**Sources**

1 - <https://github.com/infobyte/faraday>

2 - <https://docs.faraday-cli.faradaysec.com/>

3 - <https://github.com/infobyte/faraday-cli>

4 - <https://hub.docker.com/r/appsecco/dsvw>

5 - <https://faradaysec.com/>