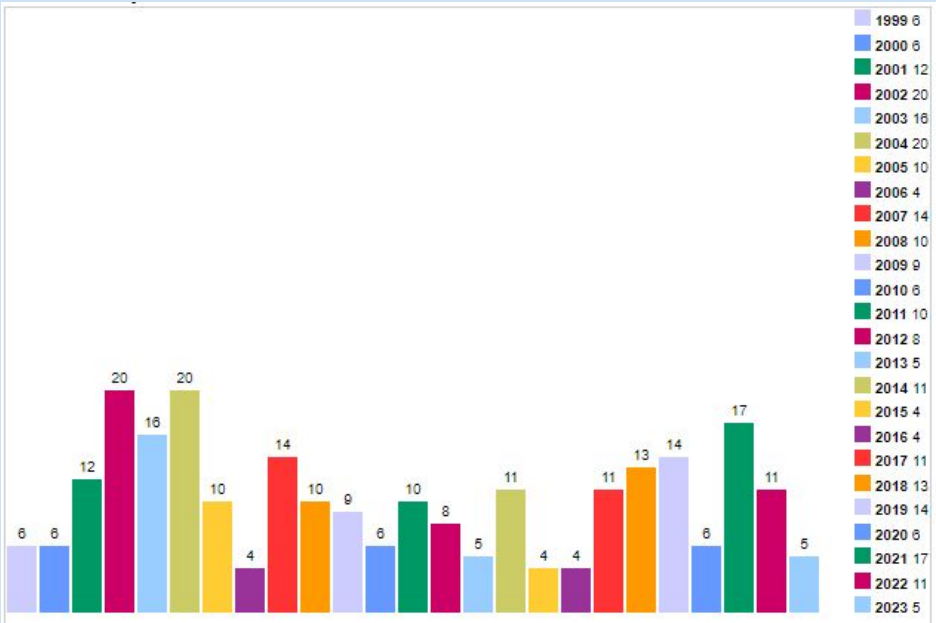
Hardening a Linux Web App

Server Exploits - Module 5

Keeping your Server Patched

It is incredibly important to keep all your technologies patched. This includes your Linux OS, webserver, scripting language and database. Below are the CVE's (documented vulnerabilities) for Apache by year. Keeping your system patched ensures you are not vulnerable.

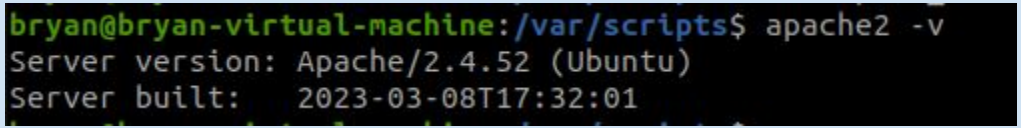


Keeping your Server Patched

The first thing to do is to run `sudo apt update`. This does not update your Linux machine, but rather it updates your apt repository information. This updates apt to see the newest versions of packages available, but does not actually install anything.

The command `sudo apt upgrade` installs the newest packages for services you have, and it will install for all packages. This could lead to things breaking if you upgrade all packages at once, so it's best you upgrade a single service at a time (on a test environment first of course).

Before upgrading, ensure you make backups of all web files just in case. A snapshot is a good idea. I can use the following command to see my apache version.



Upgrading Apache

If we go to Apache's website, we can see that as of this time 2.4.56 has been released, meaning we are a couple versions off. So first things first I update my apt repository with `sudo apt update`. Then, upgrading Apache2 should be as easy as running `sudo apt get upgrade apache2`.

If you are to run these commands and check your Apache version, you should see you're still on version 2.4.52. This is because the current repository our package manager apt is working with isn't updated. If we use the apt list, we can see that the latest version we can get at the moment is 2.4.52.

```
bryan@bryan-virtual-machine:/var/scripts$ apt list -a apache2
Listing... Done
apache2/jammy-updates,jammy-security,now 2.4.52-1ubuntu4.4 amd64 [installed]
apache2/jammy 2.4.52-1ubuntu4 amd64

apache2/jammy-updates,jammy-security 2.4.52-1ubuntu4.4 i386
apache2/jammy 2.4.52-1ubuntu4 i386
```

What can we do?

If we want to use the latest version of Apache, we can compile it by source. This is highly technical should be used as a last resort.

Another option is to see if another repository has the latest version. Luckily, Ondřej Surý has a good reputation for keeping up with Apache versions on his repository: https://launchpad.net/~ondrej/+archive/ubuntu/apache2

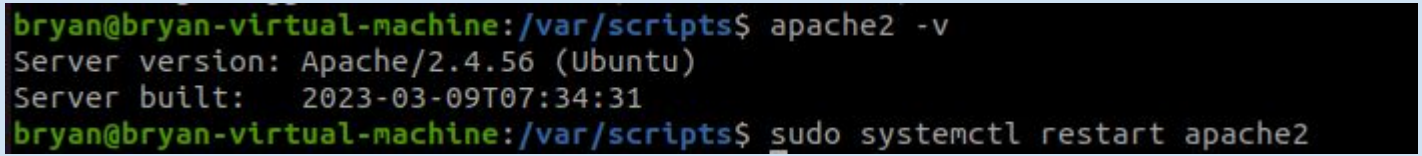
We can add this repository to Ubuntu with the following command. Then, run `sudo apt update` to update apt with our new repository. In general, be careful of repositories you add as ones without a good reputation can contain malware.

```
bryan@bryan-virtual-machine:/var/scripts$ sudo add-apt-repository ppa:ondrej/apache2
PPA publishes dbgsym, you may need to include 'main/debug' component
Repository: 'deb https://ppa.launchpadcontent.net/ondrej/apache2/ubuntu/ jammy main'
Description:
This branch follows latest Apache2 packages as maintained by the Debian Apache2 team with co
uple of compatibility patches on top.
```

What can we do?

Now we can upgrade Apache2 with `sudo apt upgrade apache2`. You can enter N (for no) when prompted if you'd like to modify the apache2.conf file and the 000-default file. We don't want to change our configurations we made when setting up DVWA.

When we check our version, we can see we are now on Apache 2.4.56. Make sure you restart the apache2 service afterwards.



Apache is upgraded, how can we stop attacks?

If we are sick of the attacker exploiting our DVWA application, we can block them from reaching our web application. This is a simple fix to stop a single attacker, but does not solve the vulnerabilities with DVWA (We will get to that a bit later).

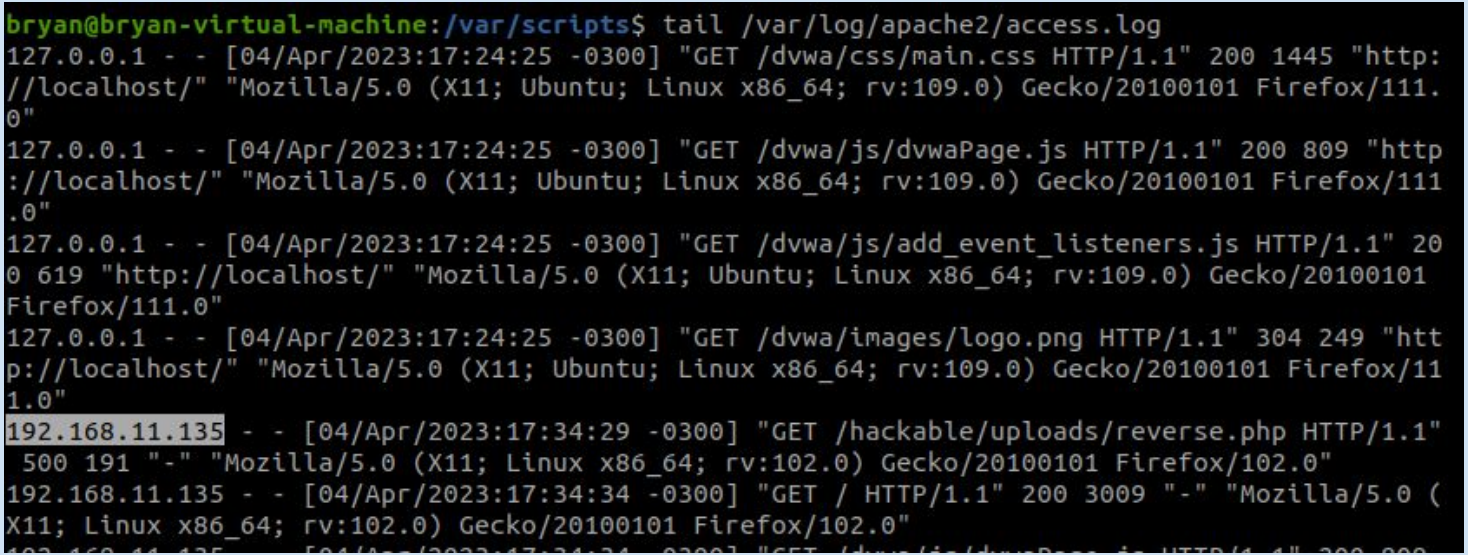
Apache2 can block IP addresses if you create a blacklist.conf file. This is not done by default, so we will have to modify our /etc/apache2/apache2.conf file. To include the following lines:

```
# Block ip addresses in our ipblacklist.conf file
<Location />
   <RequireAll>
      Require all granted
      Include /etc/apache2/blacklist.conf
   </RequireAll>
</Location>

# Add a 403 forbidden message for blacklisted ips
ErrorDocument 403 "<h3>Sketchy activity has been detected from this IP address.</h3><p>Beca
```

Apache is upgraded, how can we stop attacks?

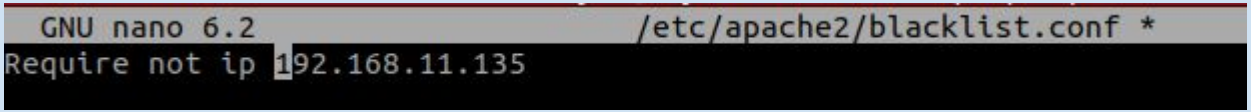
Next we have to create the blacklist.conf file. We can see the IP of the attacking machine from our access.log under /var/log/apache2/access.log trying to activate a PHP reverse shell. First, visit DVWA from you Kali machine, then tail the access.log on your Ubuntu machine to see the activity from your Kali IP. You logs will look different from mine depending which pages you visit from you Kali machine.



Apache is upgraded, how can we stop attacks?

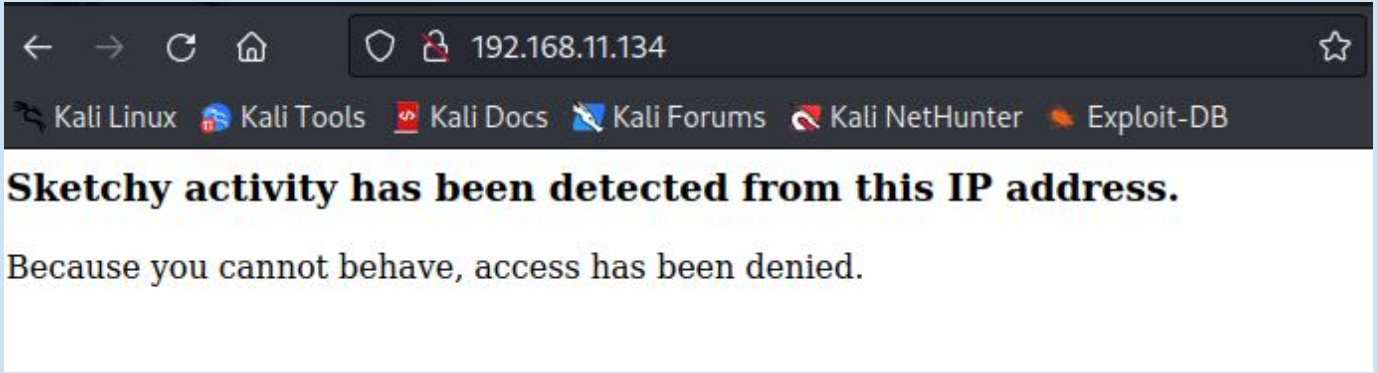
Now we can create /etc/apache2/blacklist.conf and add the following line to block our Kali machine. Make sure to use your Kali IP.

When you create this file, restart apache2 afterwards to ensure the changes get loaded.



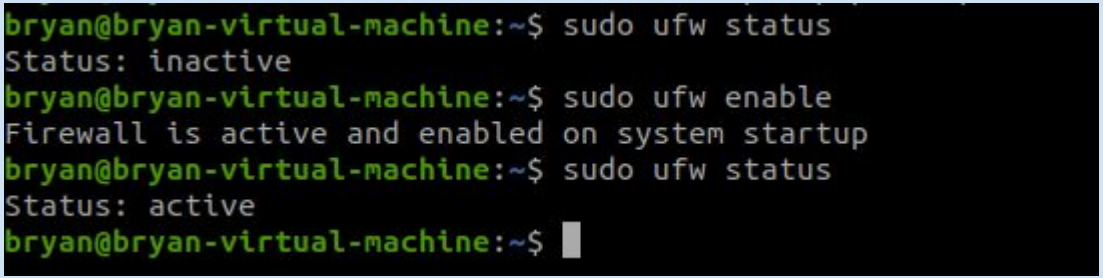
Apache is upgraded, how can we stop attacks?

When you try to visit the page from your Kali machine, you should be presented with the message you wrote in your apache2.conf file. However, if you try to visit from another machine like your host machine, you should be able to access DVWA just fine.



What about SSH?

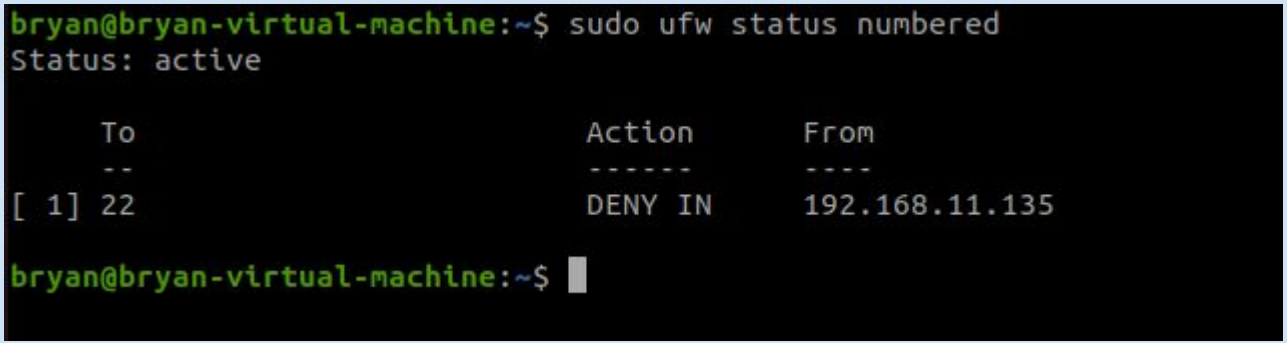
Unfortunately, Apache's blacklisting capabilities only work on our web application. We can use UFW (Ubuntu Firewall) to handle other ports. By default UFW is disabled, so we need to enable it.



What about SSH?

We can then deny access to port 22 from our kali IP with `sudo ufw deny from <kali_ip> to any port 22`. This denies access from our Kali's IP address to any IP address on our Ubuntu machine port 22. I then use `sudo ufw status numbered` to see a numbered list of rules. I can delete rules by their number with `sudo ufw delete 1`.

Ideally, we wouldn't even specify the port with a real attacker, we would just deny their IP to all ports. However, it is important to know how to block a single port.



What about HTTP?

We can also ensure that port 80 (which is what is running DVWA) only accepts HTTP traffic. Using `sudo ufw allow http` we can allow HTTP traffic on port 80. It will automatically create a rule for IPv6 addresses on the machine, but because we haven't set up our Ubuntu machine to use IPv6, we can get rid of the last rule.