Maria Valencia

CSC 154

Lab 8

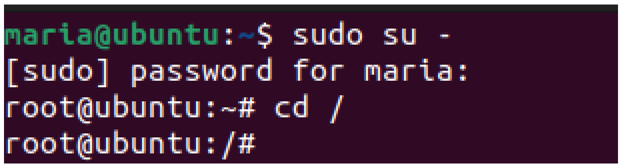
Lab 8: Web Application Defense

Exercise 8.1 Web Server Security

In this task, I installed Apache web server on my Ubuntu VM using Bridge Adapter network mode and secured it with an OpenSSL self-signed cert and Modesecurity WAF.

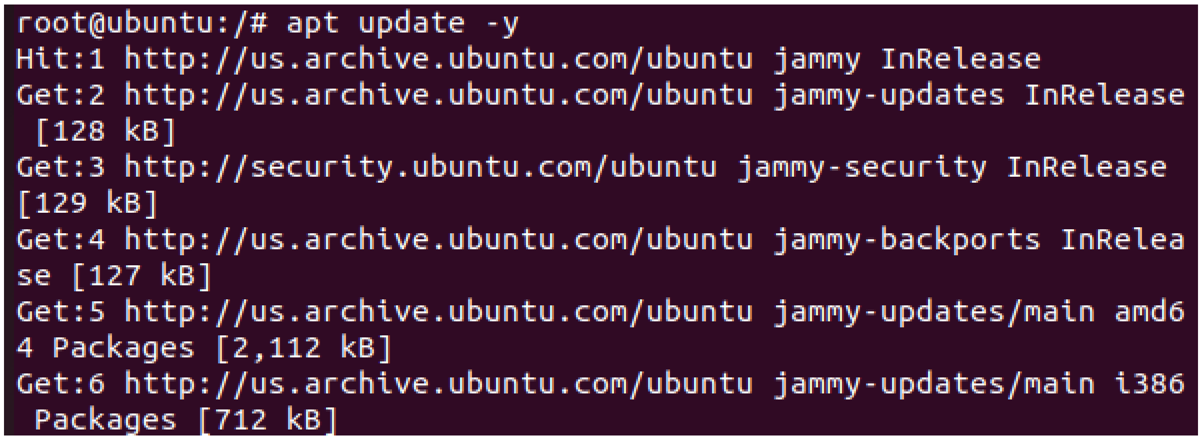
Step 1: Install Apache

I started my Ubuntu VM and opened a bash terminal. I changed the directory to the root folder and switched to the root user.



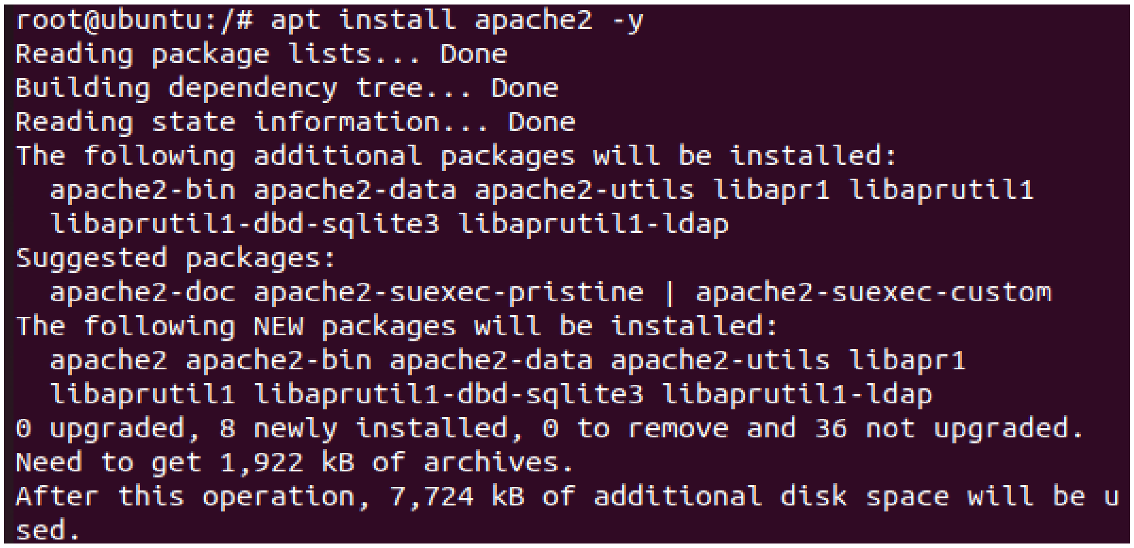
```
maria@ubuntu:~$ sudo su -
[sudo] password for maria:
root@ubuntu:~# cd /
root@ubuntu:/#
```

I then updated the Ubuntu system, so all required packages are up to date.



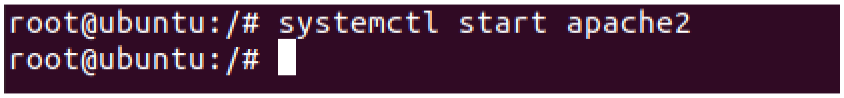
```
root@ubuntu:/# apt update -y
Hit:1 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease
 [128 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease
[129 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelea
se [127 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd6
4 Packages [2,112 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu jammy-updates/main i386
 Packages [712 kB]
```

I installed Apache web server from the Ubuntu apt repositories.



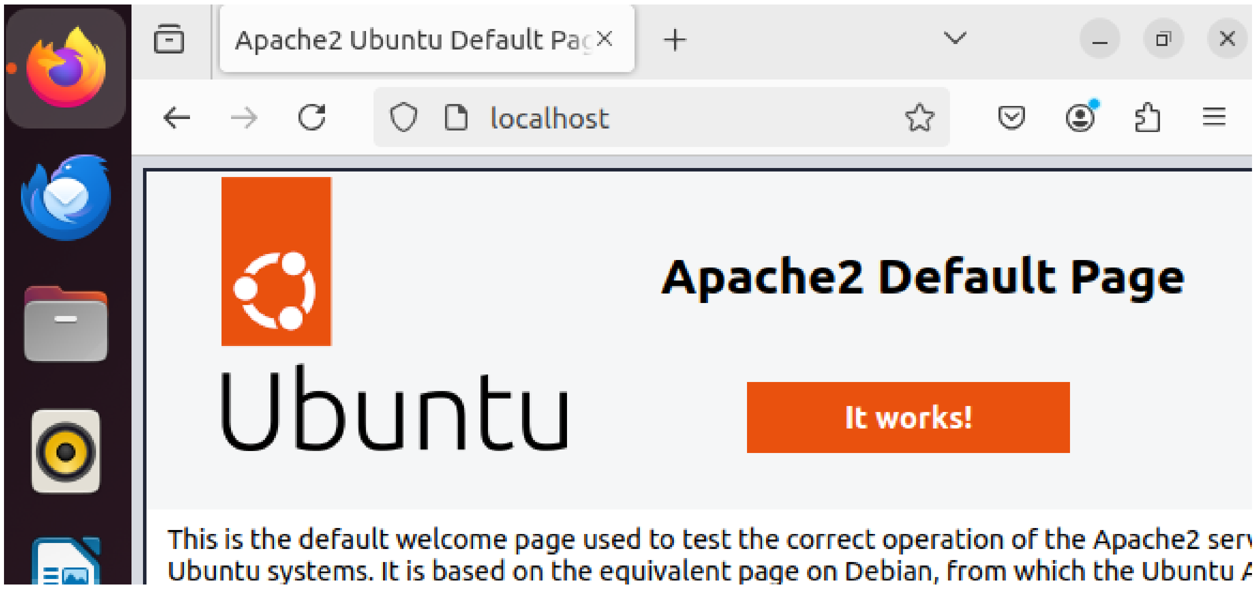
```
root@ubuntu:/# apt install apache2 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1
  libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap
0 upgraded, 8 newly installed, 0 to remove and 36 not upgraded.
Need to get 1,922 kB of archives.
After this operation, 7,724 kB of additional disk space will be u
sed.
```

I then started Apache web server in the Ubuntu VM.

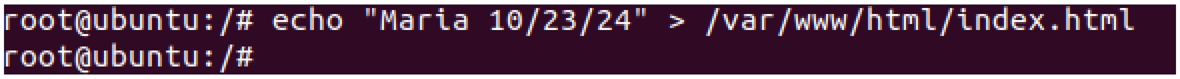


```
root@ubuntu:/# systemctl start apache2
root@ubuntu:/# 
```

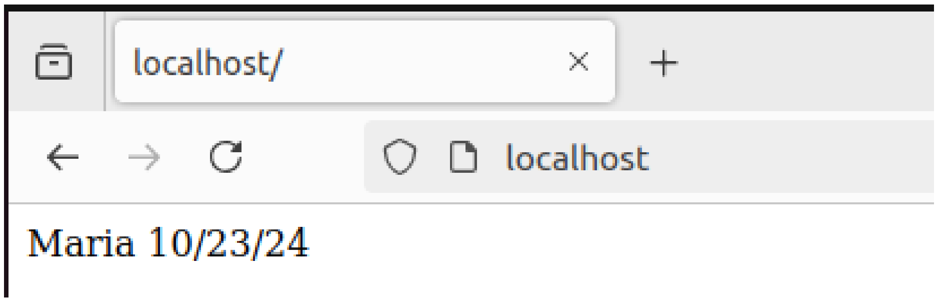
I opened the Firefox web browser within the Ubuntu VM and navigated to http://localhost/. I observed that the default Apache loads. I then updated the default index.html page with my name and the date.



This is the default welcome page used to test the correct operation of the Apache2 serv
Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu A



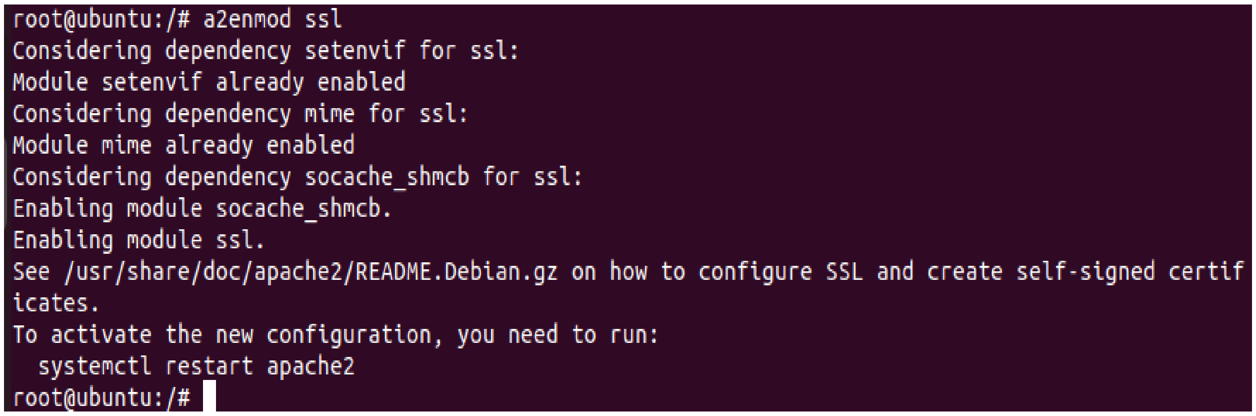
```
root@ubuntu:/# echo "Maria 10/23/24" > /var/www/html/index.html
root@ubuntu:/#
```



Maria 10/23/24

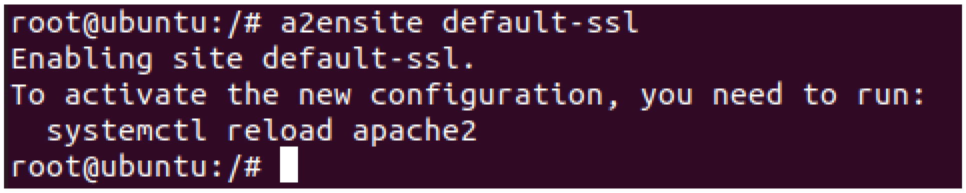
Step 2: Configure Apache SSL

Using my root user terminal, I enabled SSL on the Apache Web Server.



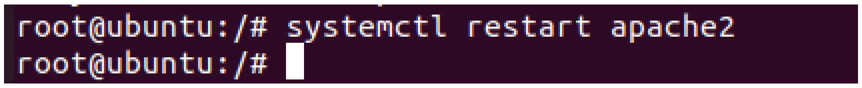
```
root@ubuntu:/# a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certif
icates.
To activate the new configuration, you need to run:
  systemctl restart apache2
root@ubuntu:/#
```

Then, I enabled the default SSL site that install with Apache.



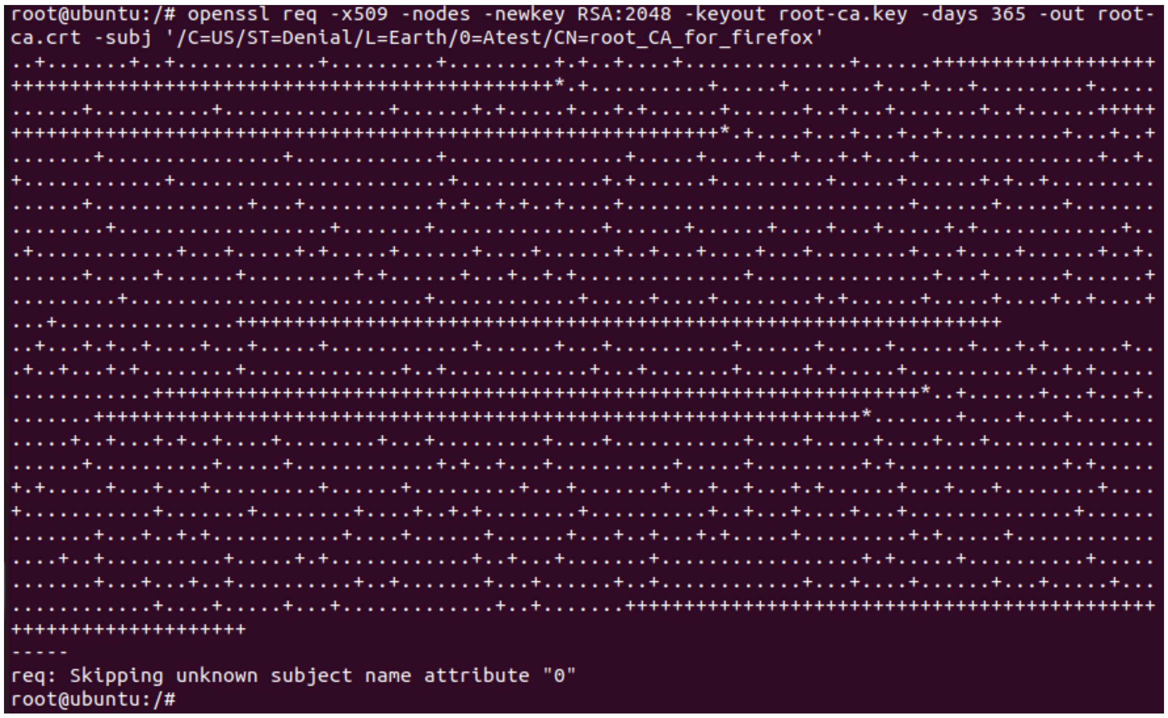
```
root@ubuntu:/# a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
  systemctl reload apache2
root@ubuntu:/#
```

I restarted the Apache web server to enable the new site settings.



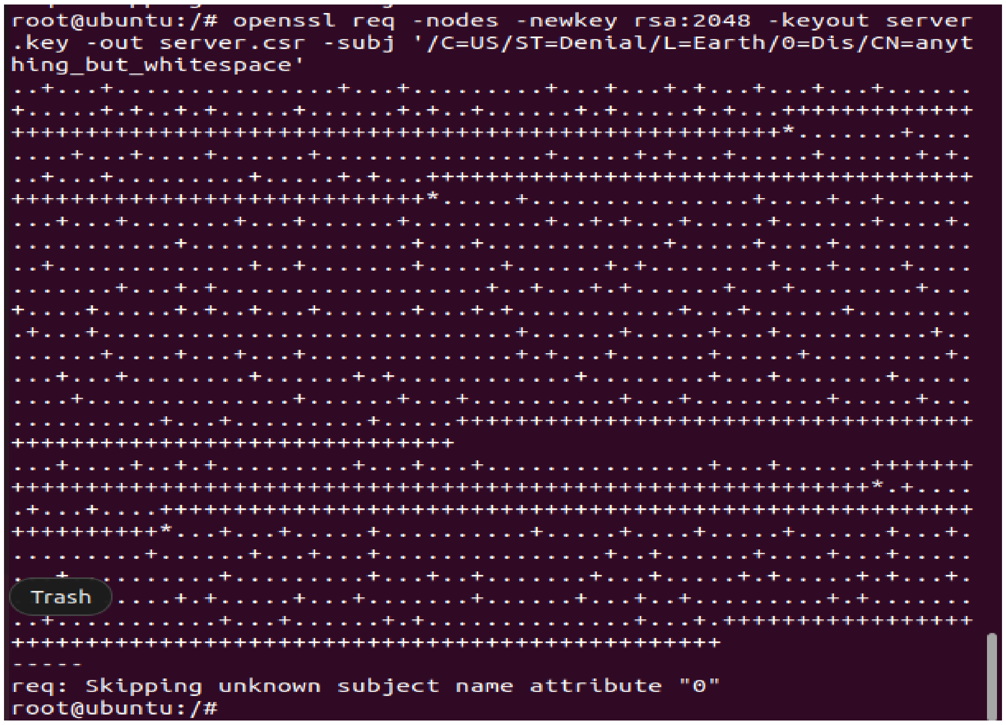
```
root@ubuntu:/# systemctl restart apache2
root@ubuntu:/#
```

Then, created a private certificate authority with the following command and observe root-ca.key and root-ca.crt files are created.



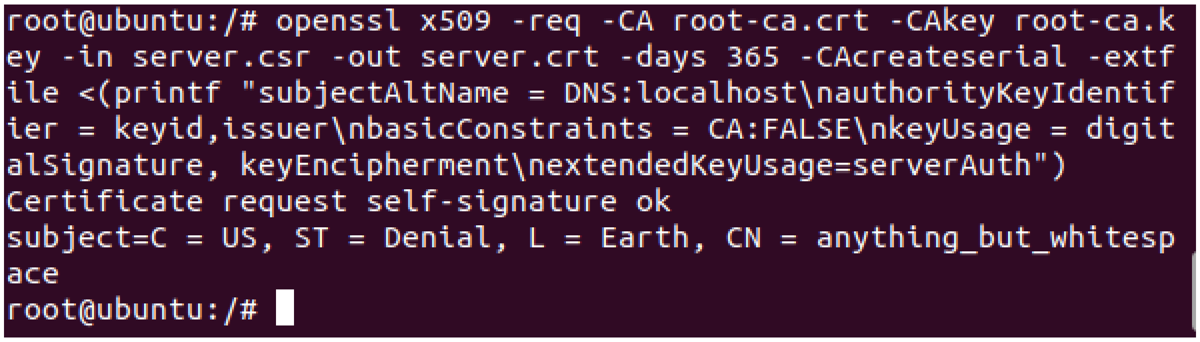
```
root@ubuntu:/# openssl req -x509 -nodes -newkey RSA:2048 -keyout root-ca.key -days 365 -out root-
ca.crt -subj '/C=US/ST=Denial/L=Earth/0=Atest/CN=root_CA_for_firefox'
..+.......+..+............+.........+.........+.+..+....+..............+......+++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++*.+..........+.....+.......+...+...+.........+.....
......+..........+..............+......+.+.....+...+.+......+......+..+...+.......+..+......+++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*.+....+...+...+..+..........+...+..+
.......+...............+............+...............+.....+....+..+...+.+...+...............+..+.
+............+.......................+............+.+......+.........+.....+......+.+..+.........
......+.............+...+...........+.+..+.+..+....+........................+......+.....+.......
........+..................+.......+..............+......+......+....+...+.....+.+............+..
.+............+...+.....+.+.....+......+....+......+..+...+....+...+........+...+....+......+..+.
......+.....+......+.........+.+......+...+..+.+..............+...............+...+......+......+
.........+.........................+............+.....+....+........+.+......+.....+....+..+....+
...+...............+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
..+...+.+..+....+...+.....+............+......+...+..........+......+.....+......+...+.+......+..
.+..+...+.+........+.............+..+............+...+.......+.....+.+.....+..........+..+.+.....
............+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*..+......+...+...+.
.......+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*.......+....+...+.......
.....+..+...+.+..+....+........+...+.........+....+...........+....+.....+....+...+..............
......+..........+.....+............+.+..+...+..........+.....+.........+.+..............+.+.....
+.+.....+...+...+.........+......+.........+...+.......+...+..+...+.+......+...+...+........+....
+...........+.......+........+....+..+.+........+..........+..+...+....+...+..............+......
.......+...+..+.+...........+....+......+......+...+..+...+.+.....+.........+.+.....+............
....+..+..........+.....+.+............+..+...+.......+.................+.+.....+..........+.....
.......+...+...+..+..........+..+.......+...+......+..+............+...+....+......+...+.....+...
............+....+.....+...+.............+..+.......+++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++
-----
req: Skipping unknown subject name attribute "0"
root@ubuntu:/#
```

I created a private key and certificate signing request from the previously created CA certificates. The private key will be used to secure our SSL site. I observed the server.key and server.csr files created.



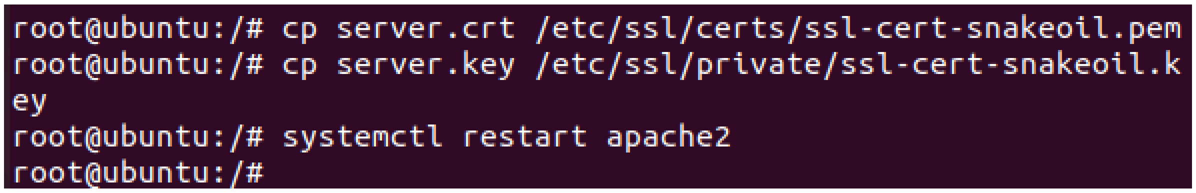
```
root@ubuntu:/# openssl req -nodes -newkey rsa:2048 -keyout server
.key -out server.csr -subj '/C=US/ST=Denial/L=Earth/0=Dis/CN=anyt
hing_but_whitespace'
..+...+...............+...+.........+...+...+.+...+...+...+.....
+.....+.+..+.+.....+......+.+..+......+.+.....+.+...++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++*.......+...
....+...+....+......+...............+.....+.+...+.....+.....+.+.
..+...+.........+.....+.+...++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++*.....+...............+....+..+.....
...+...+.......+...+......+.........+..+.+...+.....+......+...+.
...........+...............+...+............+.....+....+........
..+.............+..+.......+.....+......+.+........+...+....+...
.......+...+.+..................+..+...+.+......+...+........+..
+....+.....+.+..+...+......+...+.+...........+...+......+.......
.+...+............................+......+.....+...+..........+.
......+....+...+...+..............+.+...+......+.....+........+.
...+...+........+......+.+............+........+...+.......+....
....+..............+......+...+..........+...+.........+.....+..
..........+...+.........+.....++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++
...+....+..+.+.........+...+...+...............+...+.....+++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*.+...
.+...+....++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++*...+...+.....+..........+.....+....+.....+......+..+.
.........+......+...+...+...............+..+......+....+...+....
...+..........+.........+...+..+.......+...+.....+.+.....+.+..+.
       ....+.+.....+...+.......+......+...+..+.........+.+......
..+...........+...+......+.+..............+...+.++++++++++++++++
+++++++++++++++++++++++++++++++++++++++++++++++
-----
req: Skipping unknown subject name attribute "0"
root@ubuntu:/#
```

I created a TLS self-signed certificate which will be used in the Apache SSL site configuration. I observed server.crt file created.



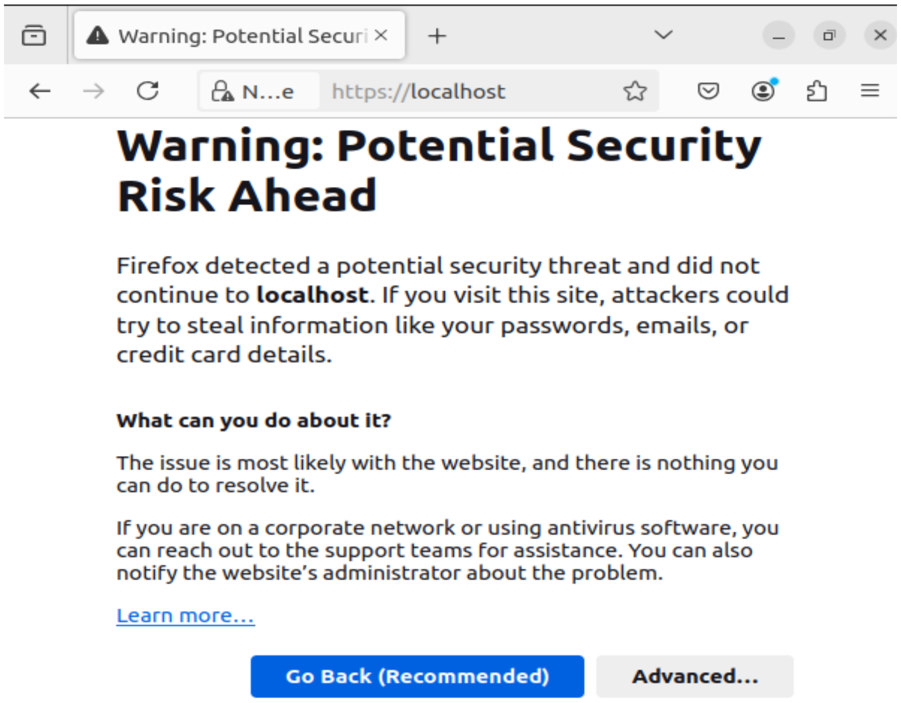
```
root@ubuntu:/# openssl x509 -req -CA root-ca.crt -CAkey root-ca.k
ey -in server.csr -out server.crt -days 365 -CAcreateserial -extf
ile <(printf "subjectAltName = DNS:localhost\nauthorityKeyIdentif
ier = keyid,issuer\nbasicConstraints = CA:FALSE\nkeyUsage = digit
alSignature, keyEncipherment\nextendedKeyUsage=serverAuth")
Certificate request self-signature ok
subject=C = US, ST = Denial, L = Earth, CN = anything_but_whitesp
ace
root@ubuntu:/# 
```

I replaced the default certification and key for our site and then restarted Apache.



```
root@ubuntu:/# cp server.crt /etc/ssl/certs/ssl-cert-snakeoil.pem
root@ubuntu:/# cp server.key /etc/ssl/private/ssl-cert-snakeoil.k
ey
root@ubuntu:/# systemctl restart apache2
root@ubuntu:/#
```

I navigated back to https://localhost/.. I observed the insecure TLS warning and detail by pressing the Advanced button.



Warning: Potential Security Risk Ahead

Firefox detected a potential security threat and did not continue to **localhost**. If you visit this site, attackers could try to steal information like your passwords, emails, or credit card details.

What can you do about it?

The issue is most likely with the website, and there is nothing you can do to resolve it.

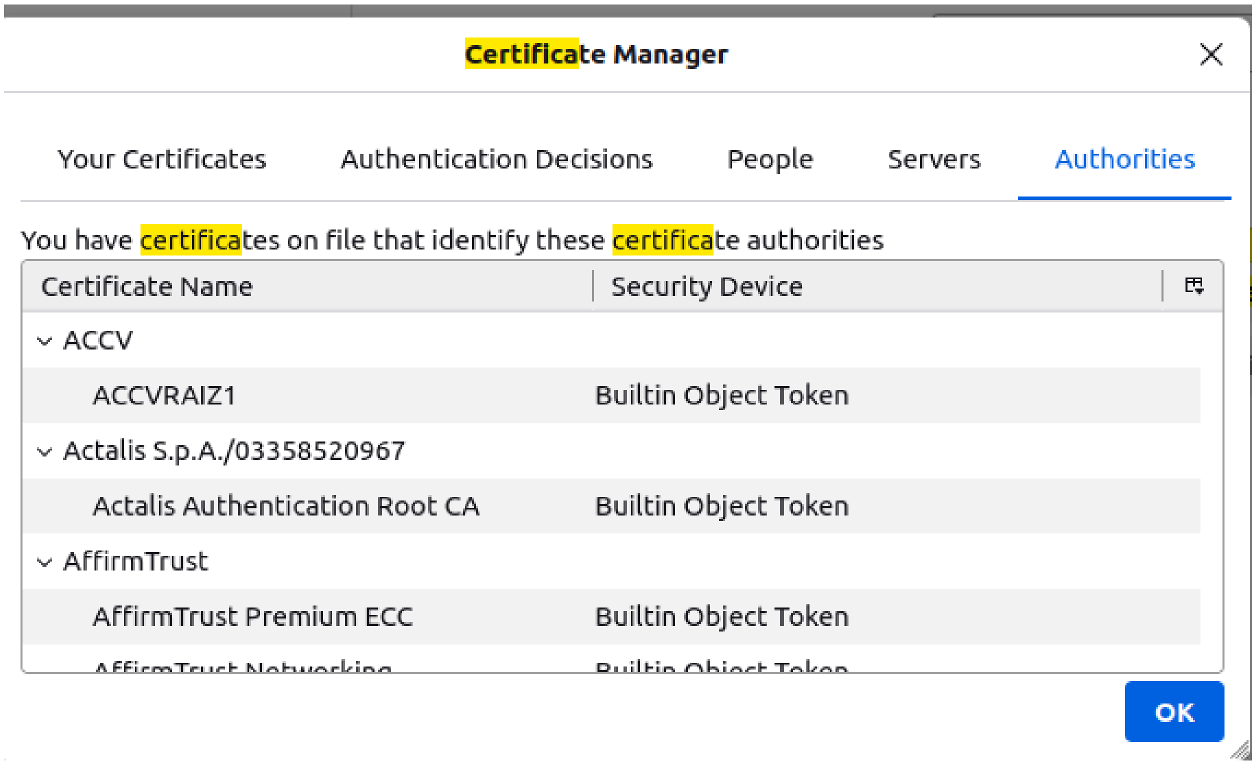
If you are on a corporate network or using antivirus software, you can reach out to the support teams for assistance. You can also notify the website's administrator about the problem.

Learn more…

To fix this problem, I add the certificate authority file we created earlier to Firefox allowed CAs by following these steps.

1a. Press the "hamburger menu" (three stack horizontal lines icon) in the upper right corner of the browser and select Settings.
2a. In the Settings page, search for Certificate and select "View Certificates..." to launch the Certificate Manager.
3a. With the Authorities tab selected, press the Import button at the bottom of the Certificate Manager window.
4a. Navigate to the root directory where we stored our certificates and keys by selecting Other Locations on the left
navigation menu, and then Computer.
5a. Select the " root-ca.crt " file and press the Select button in the upper right corner.
6a. With the Downloading Certificate window launched, select "Trust this CA to identify websites", press Ok and
then Ok again to close the Certificate Manager window.
7a. Open a new tab in Firefox and navigate to https://localhost.

This is what step 2a would look like.



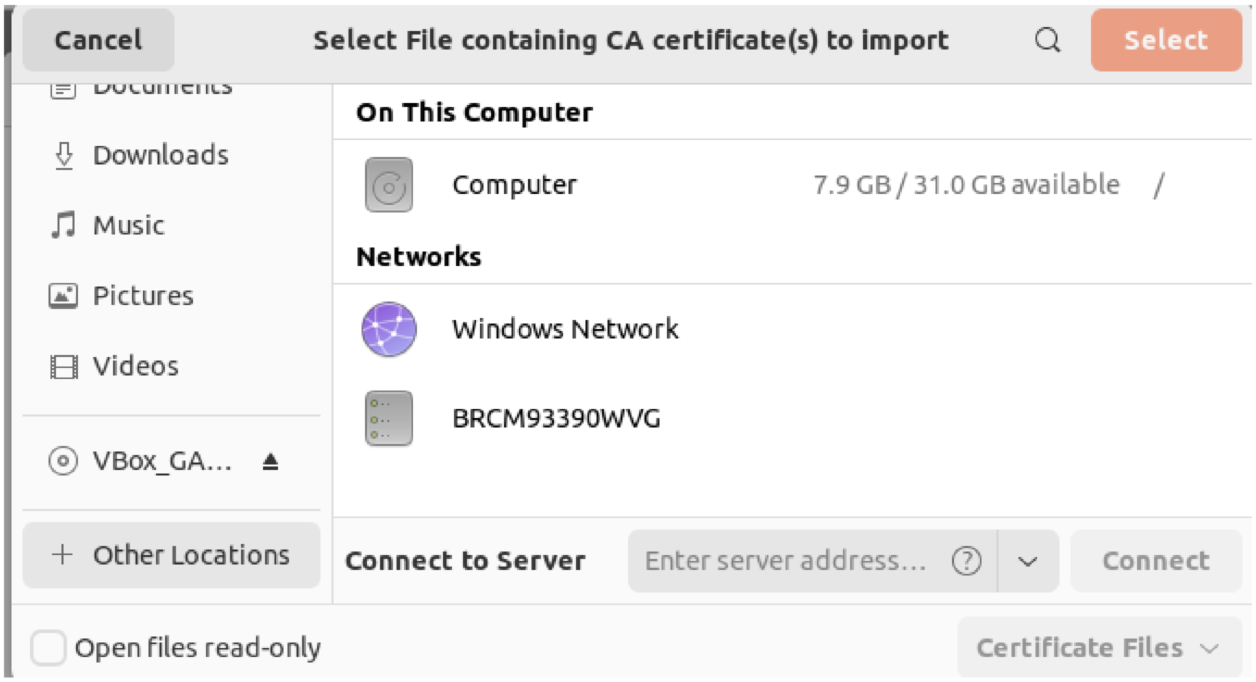
Certificate Manager

Your Certificates | Authentication Decisions | People | Servers | Authorities

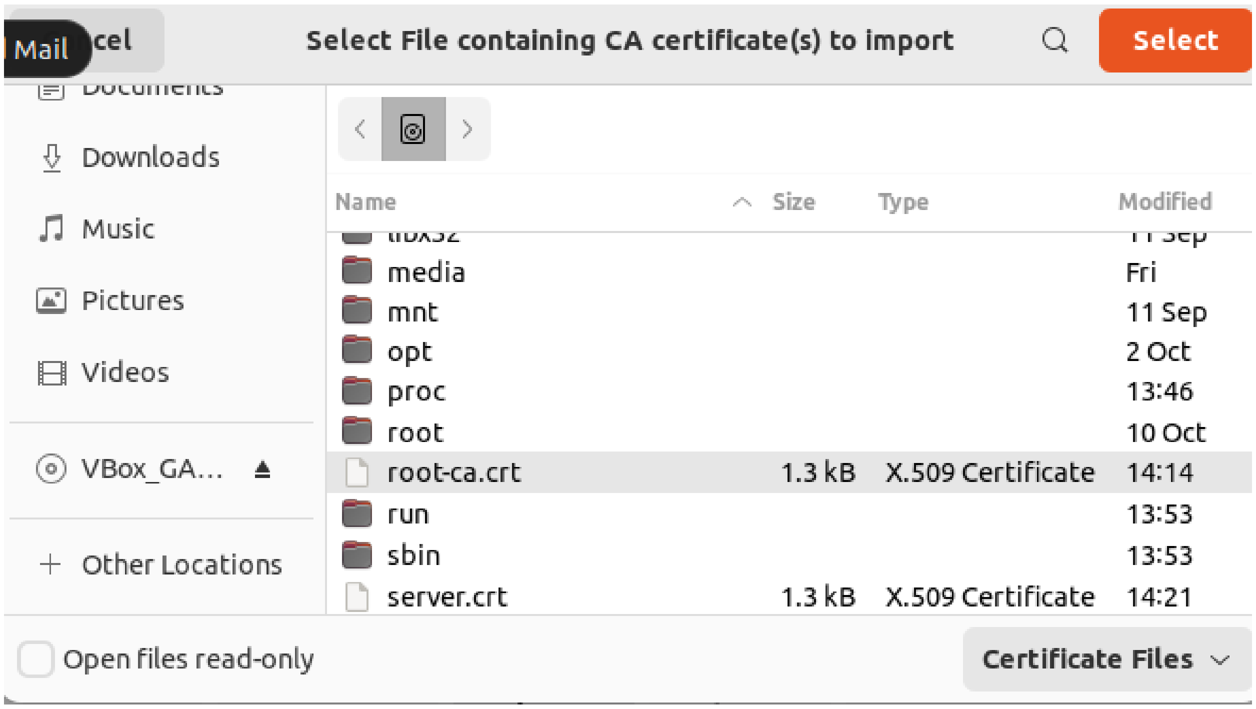
You have certificates on file that identify these certificate authorities

Certificate Name	Security Device
ACCV	
ACCVRAIZ1	Builtin Object Token
Actalis S.p.A./03358520967	
Actalis Authentication Root CA	Builtin Object Token
AffirmTrust	
AffirmTrust Premium ECC	Builtin Object Token
AffirmTrust Networking	Builtin Object Token

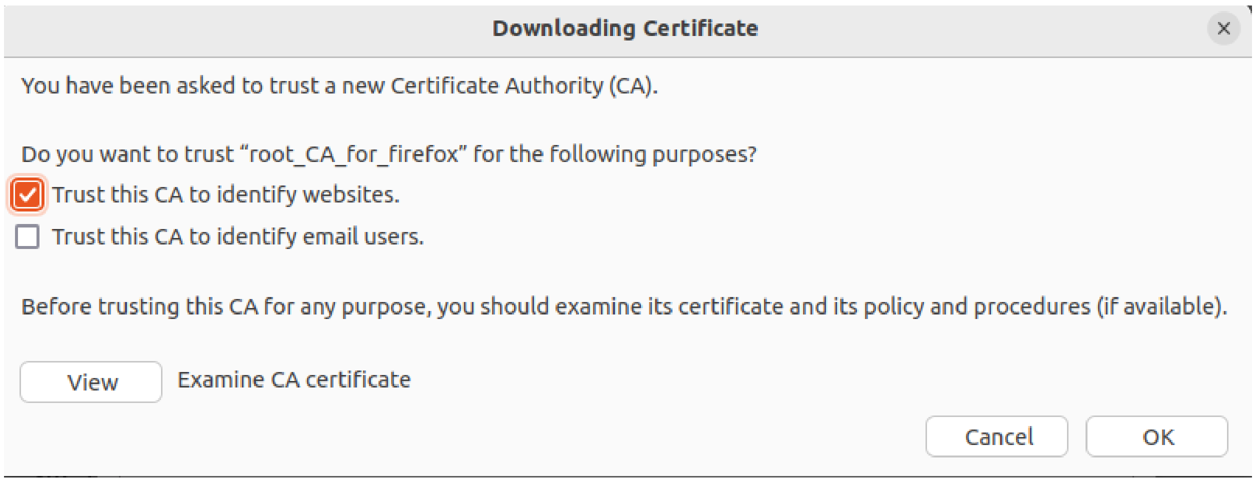
This is what step 4a looks like.



Step 5a


Select File containing CA certificate(s) to import

Name	Size	Type	Modified
media			Fri
mnt			11 Sep
opt			2 Oct
proc			13:46
root			10 Oct
root-ca.crt	1.3 kB	X.509 Certificate	14:14
run			13:53
sbin			13:53
server.crt	1.3 kB	X.509 Certificate	14:21

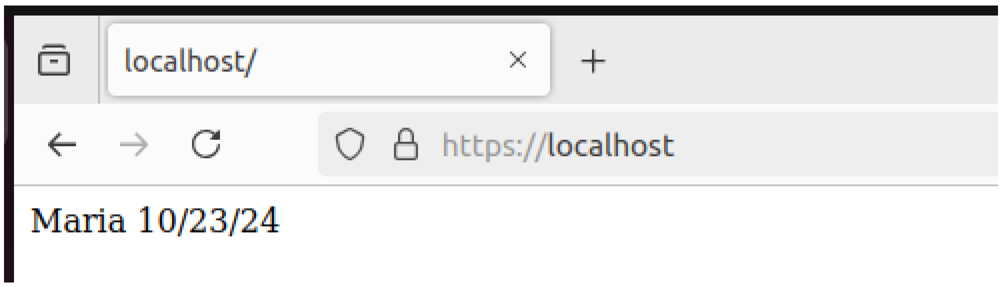

Downloading Certificate

You have been asked to trust a new Certificate Authority (CA).

Do you want to trust “root_CA_for_firefox” for the following purposes?

- [✓] Trust this CA to identify websites.
- [ ] Trust this CA to identify email users.

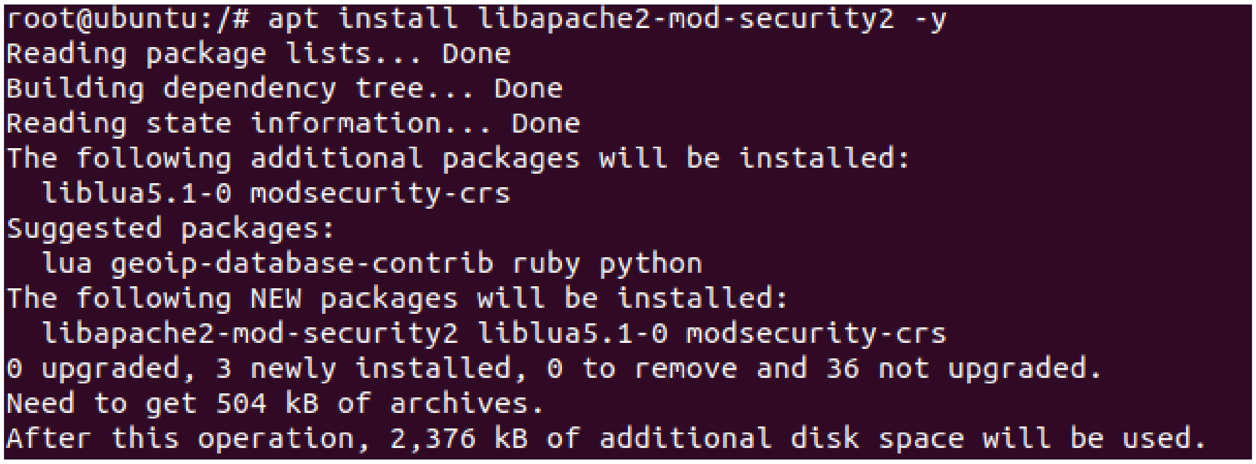
Before trusting this CA for any purpose, you should examine its certificate and its policy and procedures (if available).


https://localhost

Maria 10/23/24

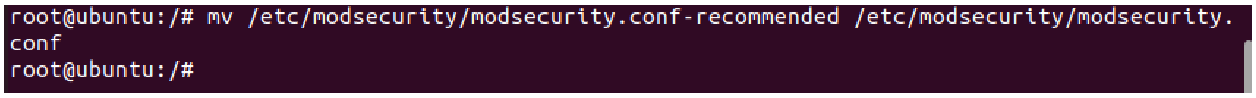
Step 3: Install ModSecurity

I installed Modsecurity using apt, still in the Ubuntu VM root bash terminal.



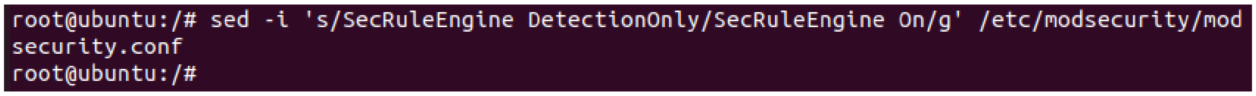
```
root@ubuntu:/# apt install libapache2-mod-security2 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  liblua5.1-0 modsecurity-crs
Suggested packages:
  lua geoip-database-contrib ruby python
The following NEW packages will be installed:
  libapache2-mod-security2 liblua5.1-0 modsecurity-crs
0 upgraded, 3 newly installed, 0 to remove and 36 not upgraded.
Need to get 504 kB of archives.
After this operation, 2,376 kB of additional disk space will be used.
```

I setup the modsecurity configuration file based on the provided recommended config file.



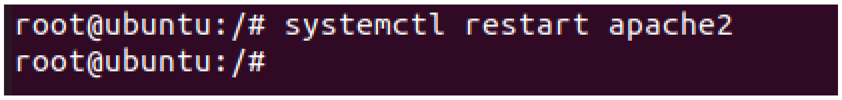
```
root@ubuntu:/# mv /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.
conf
root@ubuntu:/#
```

I updated the configuration file to turn Modsecurity blocking mode on.



```
root@ubuntu:/# sed -i 's/SecRuleEngine DetectionOnly/SecRuleEngine On/g' /etc/modsecurity/mod
security.conf
root@ubuntu:/#
```

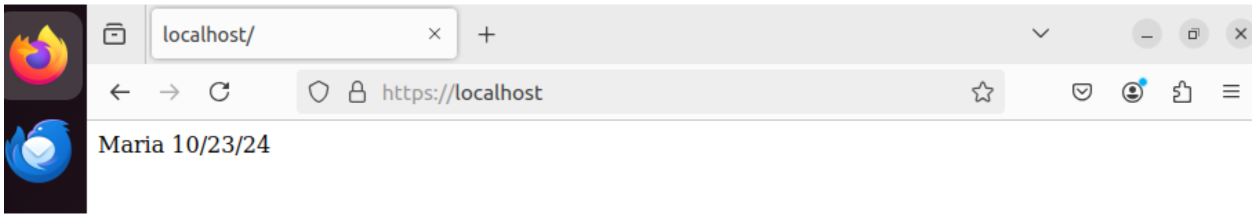
I restarted Apache so the Modsecurity updates take effect.



```
root@ubuntu:/# systemctl restart apache2
root@ubuntu:/#
```

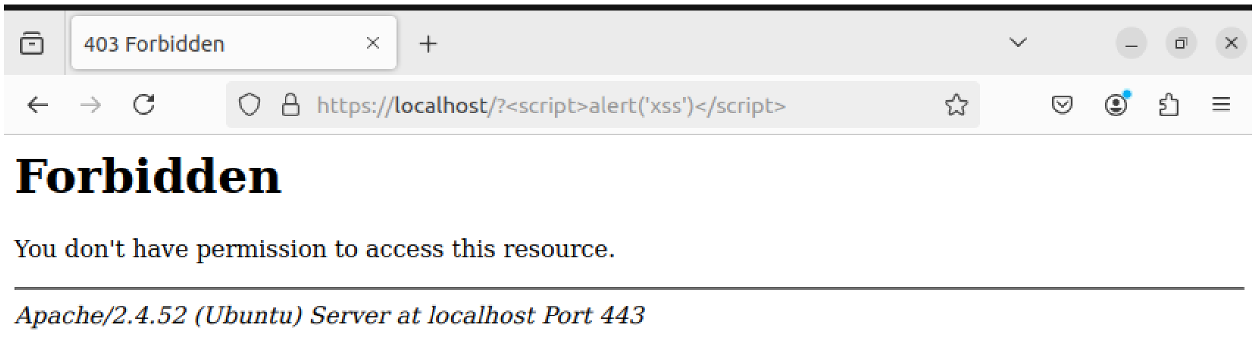
Step 4: test WAF

I navigated to https://localhost/ and observed the page renders without issue.



Maria 10/23/24

I used a cross-site scripting testing payload within the URL. The modsecurity rule detected this malicious string and blocked the HTTPS request.



403 Forbidden

```
https://localhost/?<script>alert('xss')</script>
```

Forbidden

You don't have permission to access this resource.

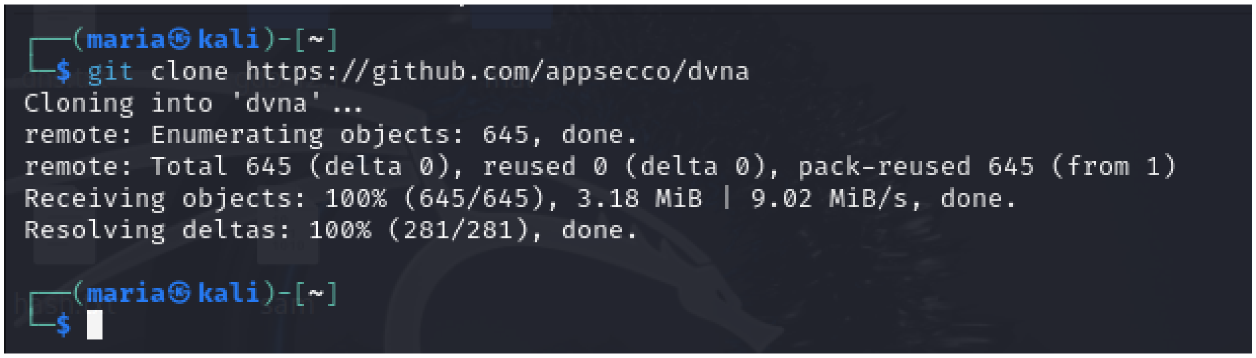
Apache/2.4.52 (Ubuntu) Server at localhost Port 443

Exercise 8.2 Secure Coding

I ran secure code tooling against the DVNA code base in this task using Kali VM using Bridge Adapter Network mode.

Step 1: Install DVNA

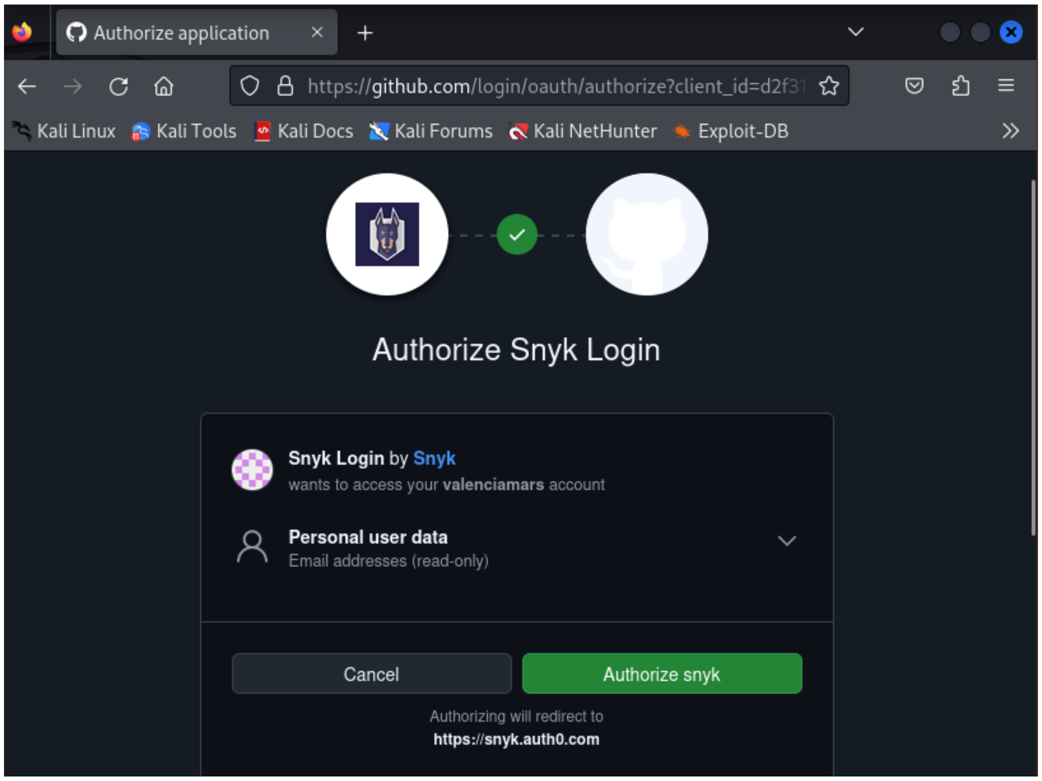
I downloaded the DVNA repository using git from the home directory of my Kali VM user.



```
┌──(maria㉿kali)-[~]
└─$ git clone https://github.com/appsecco/dvna
Cloning into 'dvna'...
remote: Enumerating objects: 645, done.
remote: Total 645 (delta 0), reused 0 (delta 0), pack-reused 645 (from 1)
Receiving objects: 100% (645/645), 3.18 MiB | 9.02 MiB/s, done.
Resolving deltas: 100% (281/281), done.

┌──(maria㉿kali)-[~]
└─$ 
```

Step 2: Install Snyk

I setup Snyk account with my existing github account by navigating to https://app.snyk.io/login/ and using the github button and authorized snyk button.



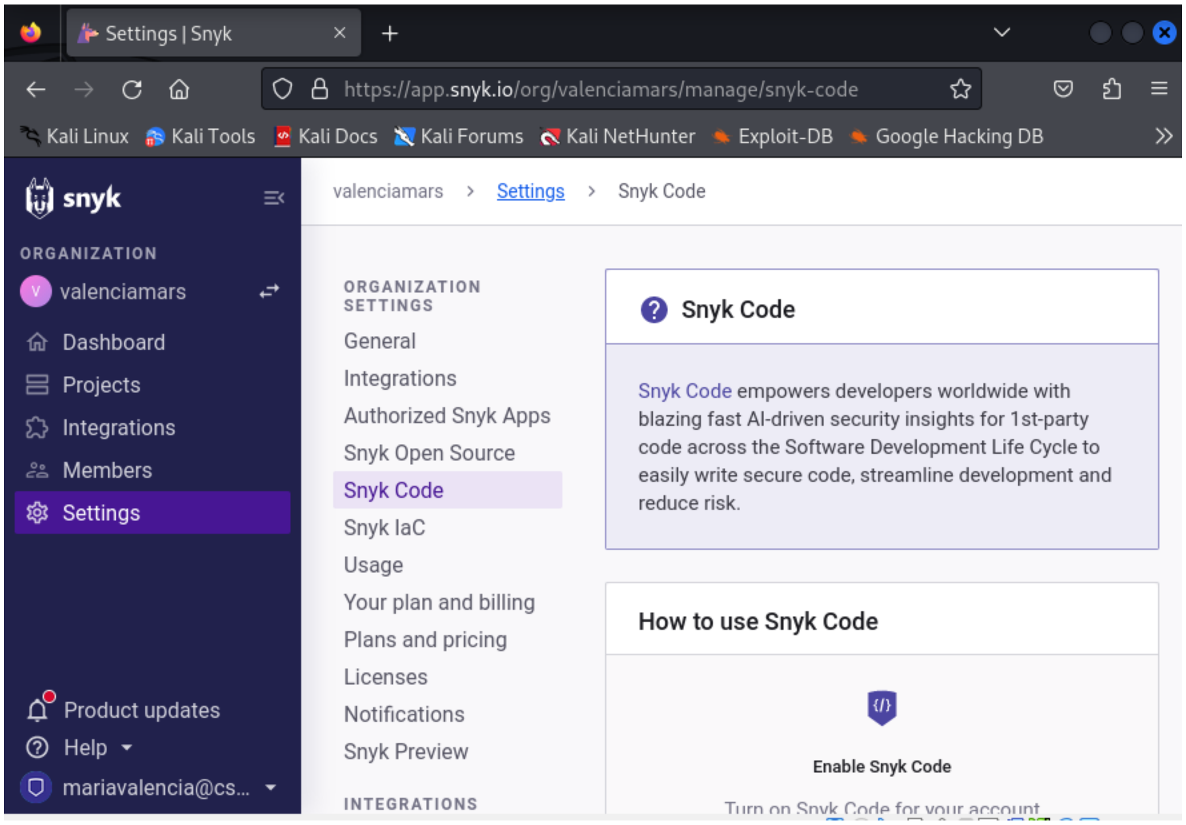
Authorize Snyk Login

Snyk Login by **Snyk**
wants to access your **valenciamars** account

Personal user data
Email addresses (read-only)

Authorizing will redirect to
https://snyk.auth0.com

While logged into the Snyk website, I enabled Snyk Code for remote SAST scanning by navigating to Settings (left menu), Snyk Code (sub menu), Enable Snyk Code (bottom), and hit save.



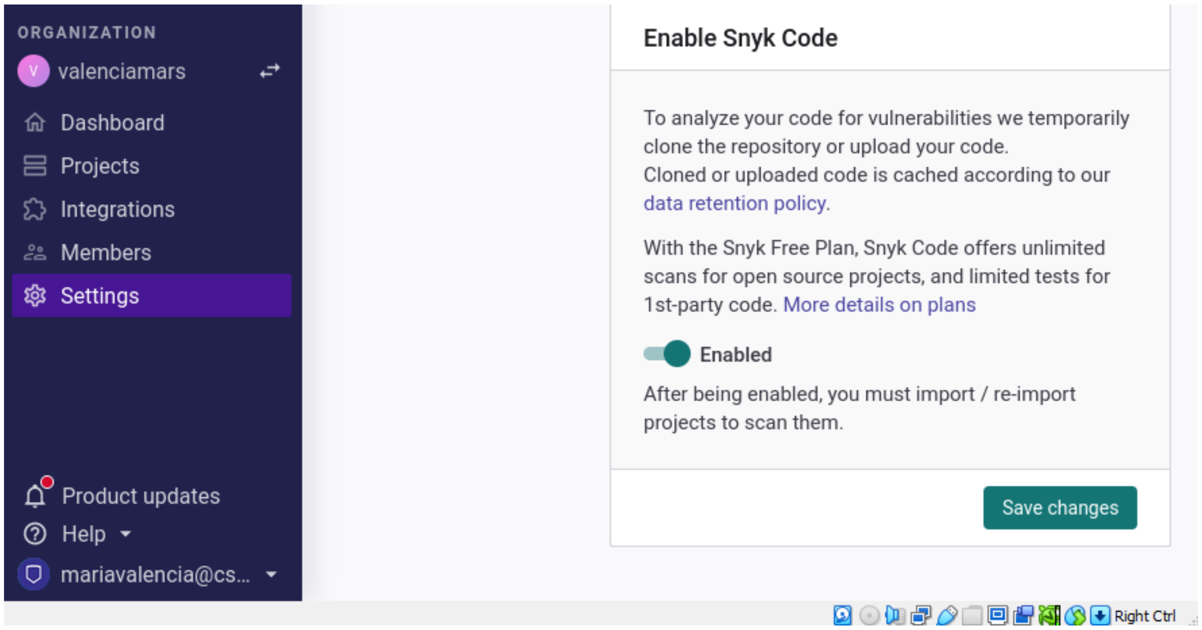
Snyk Code

Snyk Code empowers developers worldwide with blazing fast AI-driven security insights for 1st-party code across the Software Development Life Cycle to easily write secure code, streamline development and reduce risk.

How to use Snyk Code

Enable Snyk Code

Turn on Snyk Code for your account



Enable Snyk Code

To analyze your code for vulnerabilities we temporarily clone the repository or upload your code.
Cloned or uploaded code is cached according to our data retention policy.

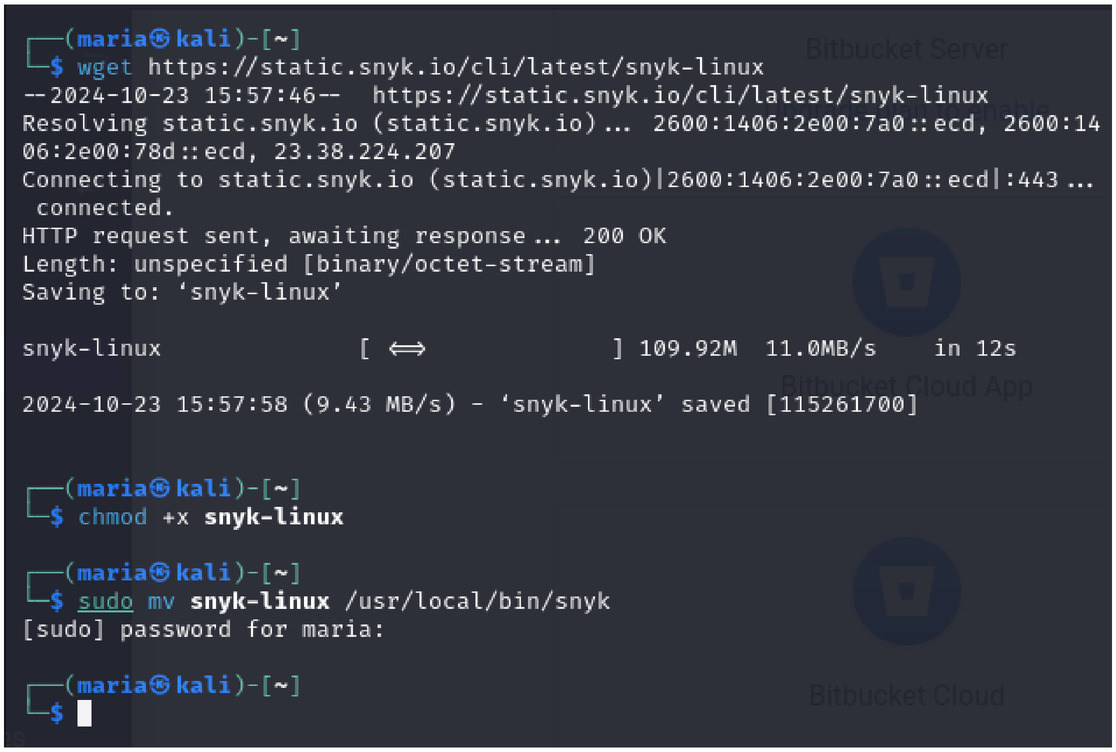
With the Snyk Free Plan, Snyk Code offers unlimited scans for open source projects, and limited tests for 1st-party code. More details on plans

Enabled

After being enabled, you must import / re-import projects to scan them.

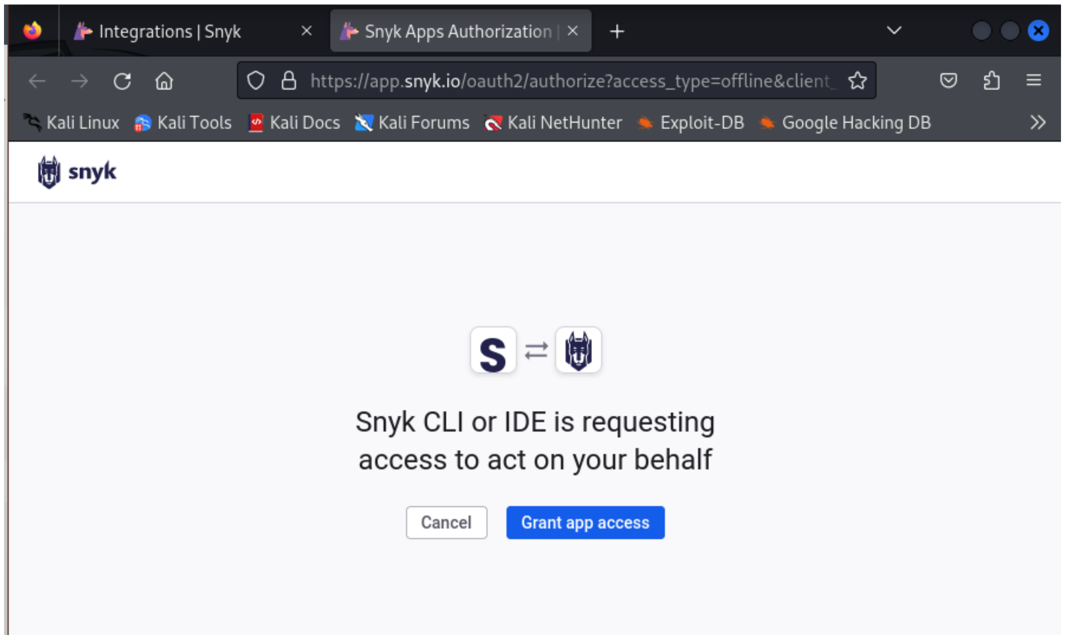
Save changes

After, I installed the Snyk linux binary and configure its use.

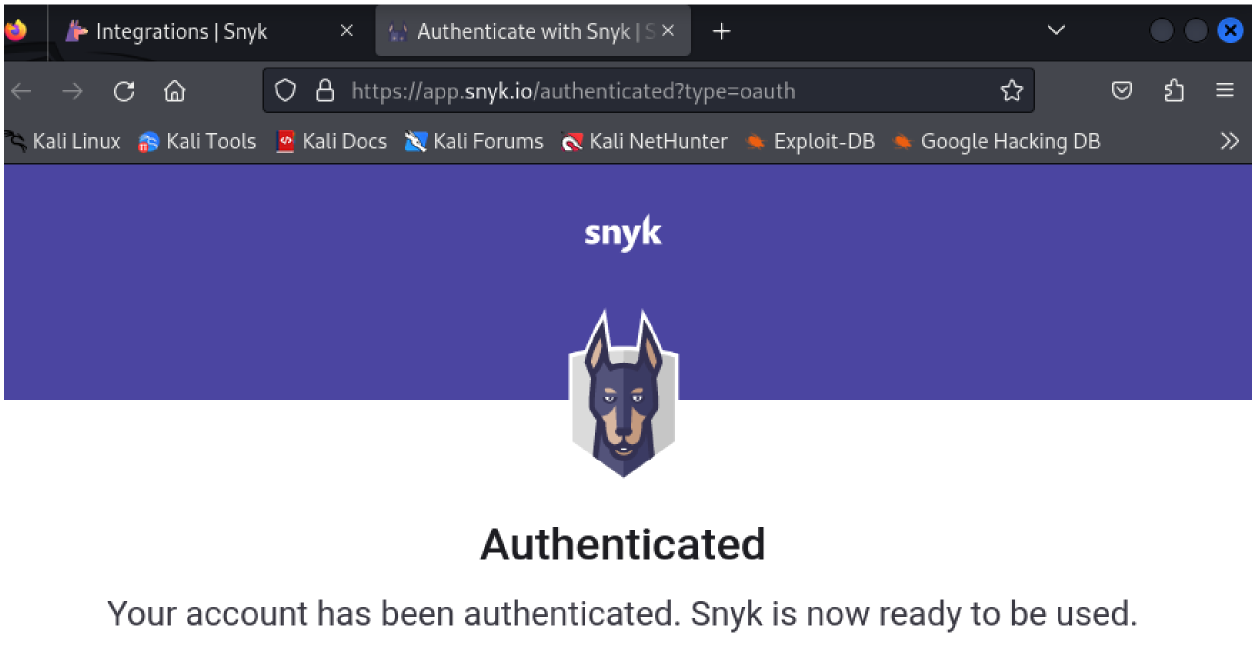


```
┌──(maria㉿kali)-[~]
└─$ wget https://static.snyk.io/cli/latest/snyk-linux
--2024-10-23 15:57:46--  https://static.snyk.io/cli/latest/snyk-linux
Resolving static.snyk.io (static.snyk.io)... 2600:1406:2e00:7a0::ecd, 2600:14
06:2e00:78d::ecd, 23.38.224.207
Connecting to static.snyk.io (static.snyk.io)|2600:1406:2e00:7a0::ecd|:443...
 connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [binary/octet-stream]
Saving to: ‘snyk-linux’

snyk-linux              [ <=>                ] 109.92M  11.0MB/s    in 12s

2024-10-23 15:57:58 (9.43 MB/s) - ‘snyk-linux’ saved [115261700]


┌──(maria㉿kali)-[~]
└─$ chmod +x snyk-linux

┌──(maria㉿kali)-[~]
└─$ sudo mv snyk-linux /usr/local/bin/snyk
[sudo] password for maria:

┌──(maria㉿kali)-[~]
└─$ 
```

Then, I authenticated with Snyk. After running this command, a browser popped up that launched prompting me to login to Snyk using my GitHub account, I signed in and pressed the authentication button.

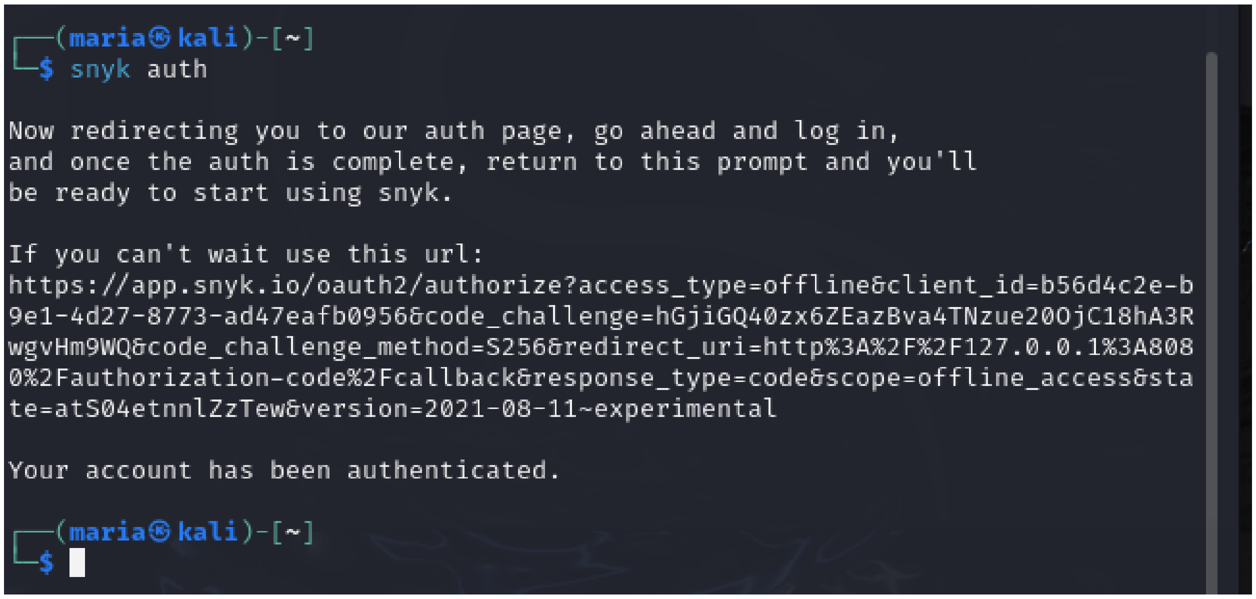


Snyk CLI or IDE is requesting access to act on your behalf



Authenticated

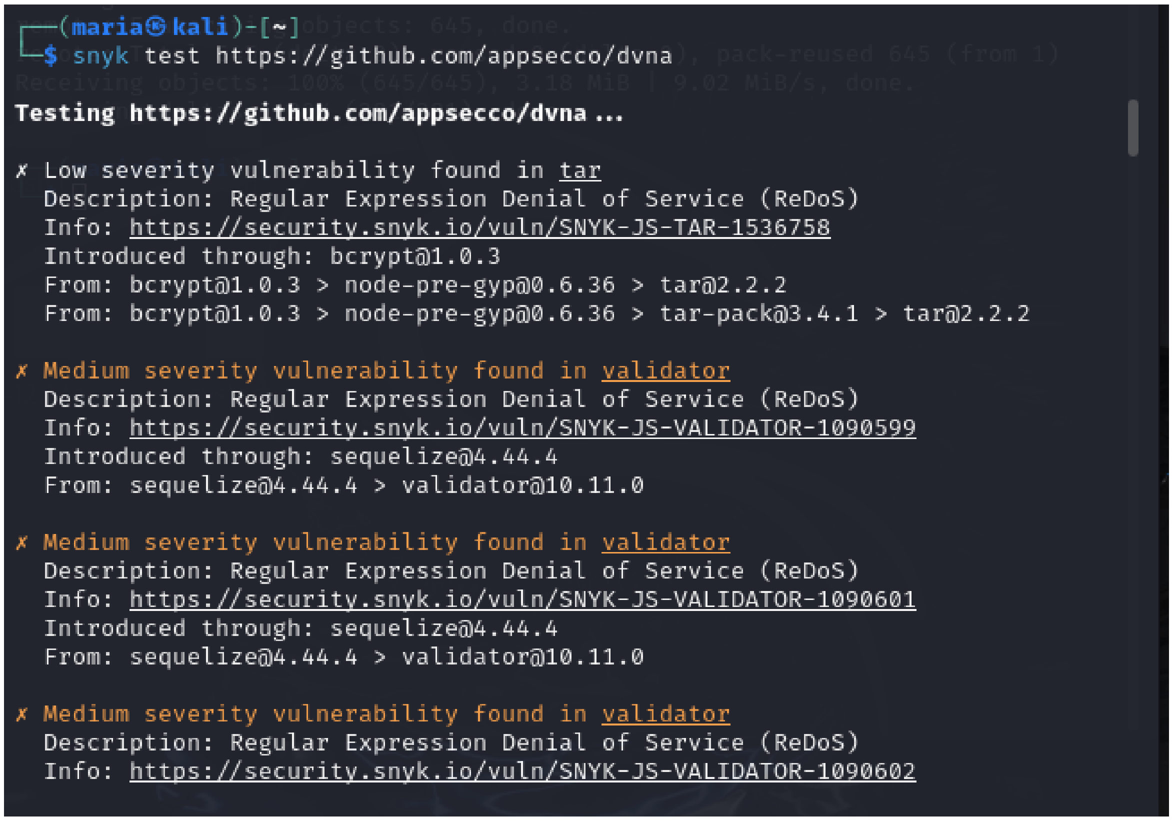
Your account has been authenticated. Snyk is now ready to be used.



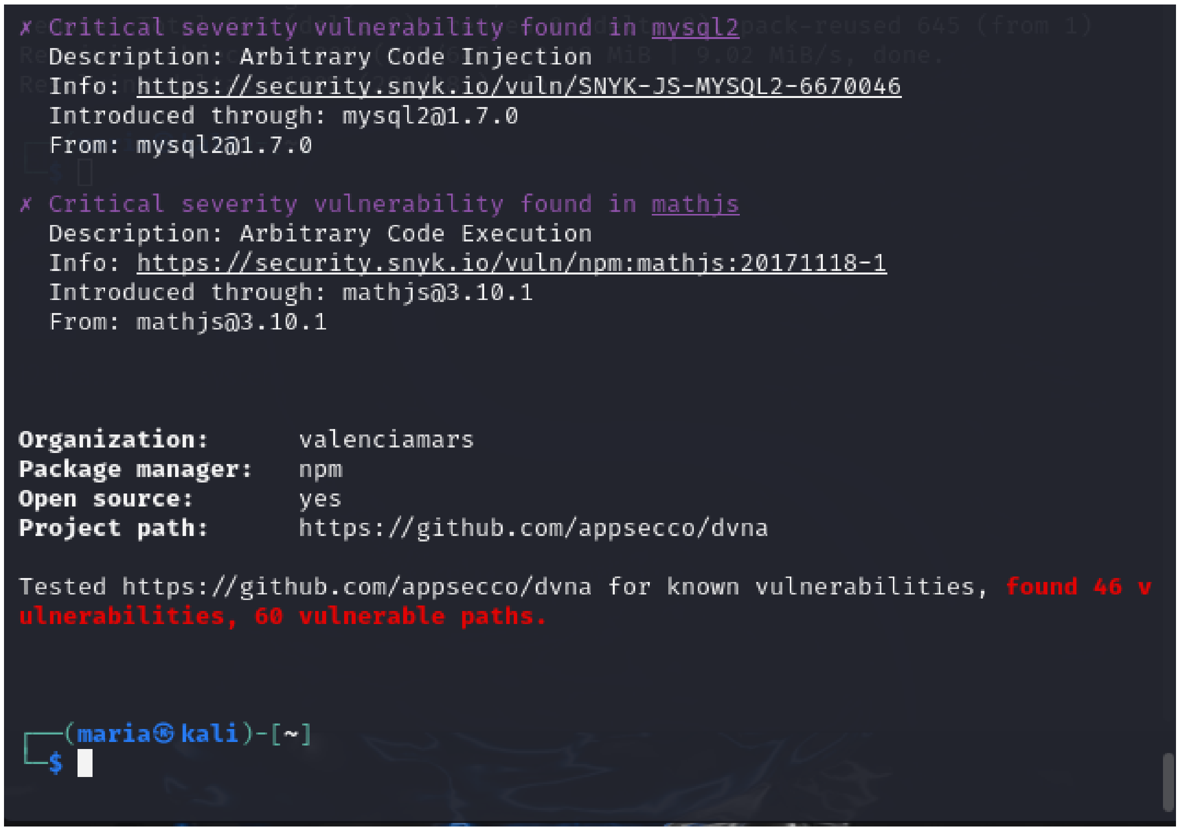
```
┌──(maria㉿kali)-[~]
└─$ snyk auth

Now redirecting you to our auth page, go ahead and log in,
and once the auth is complete, return to this prompt and you'll
be ready to start using snyk.

If you can't wait use this url:
https://app.snyk.io/oauth2/authorize?access_type=offline&client_id=b56d4c2e-b
9e1-4d27-8773-ad47eafb0956&code_challenge=hGjiGQ40zx6ZEazBva4TNzue20OjC18hA3R
wgvHm9WQ&code_challenge_method=S256&redirect_uri=http%3A%2F%2F127.0.0.1%3A808
0%2Fauthorization-code%2Fcallback&response_type=code&scope=offline_access&sta
te=atS04etnnlZzTew&version=2021-08-11~experimental

Your account has been authenticated.

┌──(maria㉿kali)-[~]
└─$ 
```

Step 3: Snyk SCA Scan

I ran a software composition analysis (SCA) scan against the dvna repository on GitHub. I am running this scan locally on the remote repository to avoid having to install NPM packages on my Kali VM. I observed several vulnerabilities are discovered after a few seconds of analysis.



```
┌──(maria㉿kali)-[~]
└─$ snyk test https://github.com/appsecco/dvna

Testing https://github.com/appsecco/dvna ...

✗ Low severity vulnerability found in tar
  Description: Regular Expression Denial of Service (ReDoS)
  Info: https://security.snyk.io/vuln/SNYK-JS-TAR-1536758
  Introduced through: bcrypt@1.0.3
  From: bcrypt@1.0.3 > node-pre-gyp@0.6.36 > tar@2.2.2
  From: bcrypt@1.0.3 > node-pre-gyp@0.6.36 > tar-pack@3.4.1 > tar@2.2.2

✗ Medium severity vulnerability found in validator
  Description: Regular Expression Denial of Service (ReDoS)
  Info: https://security.snyk.io/vuln/SNYK-JS-VALIDATOR-1090599
  Introduced through: sequelize@4.44.4
  From: sequelize@4.44.4 > validator@10.11.0

✗ Medium severity vulnerability found in validator
  Description: Regular Expression Denial of Service (ReDoS)
  Info: https://security.snyk.io/vuln/SNYK-JS-VALIDATOR-1090601
  Introduced through: sequelize@4.44.4
  From: sequelize@4.44.4 > validator@10.11.0

✗ Medium severity vulnerability found in validator
  Description: Regular Expression Denial of Service (ReDoS)
  Info: https://security.snyk.io/vuln/SNYK-JS-VALIDATOR-1090602
```

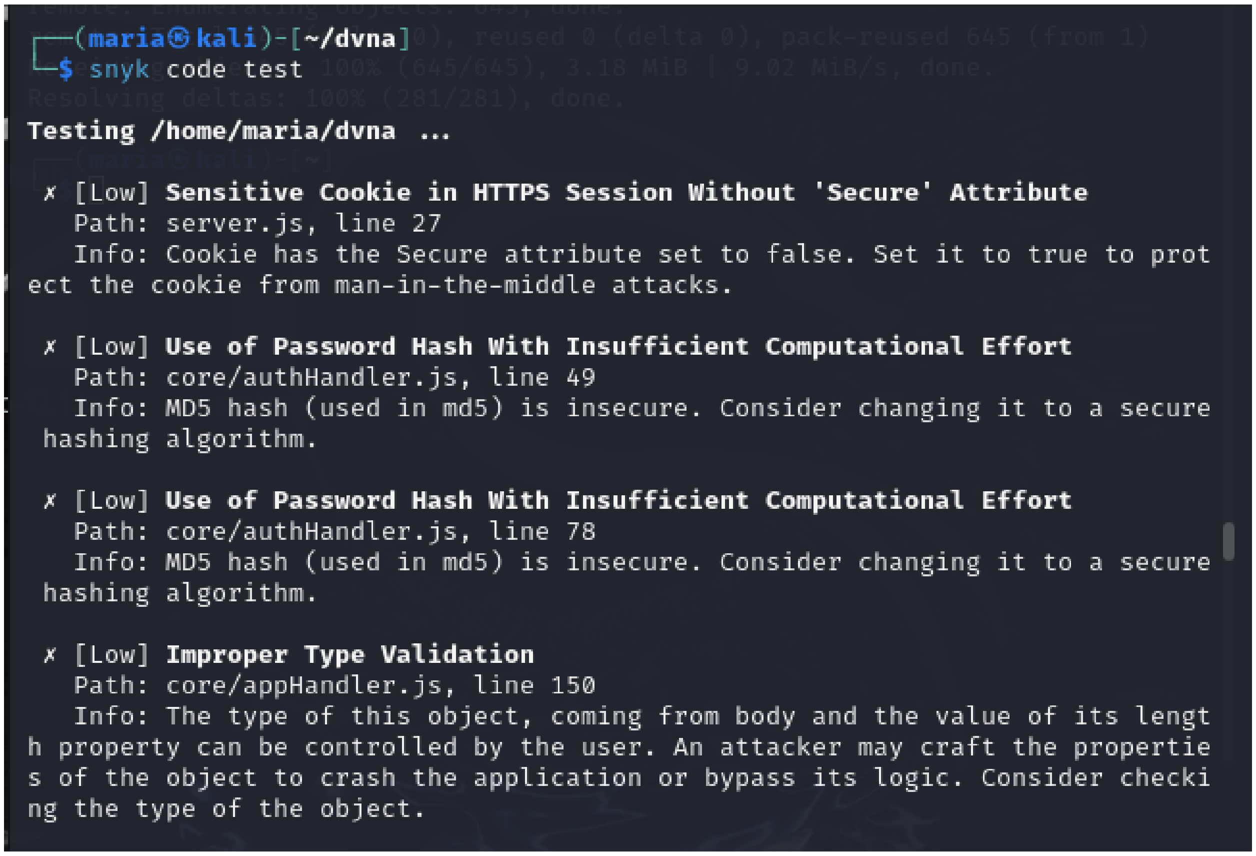


```
✗ Critical severity vulnerability found in mysql2
  Description: Arbitrary Code Injection
  Info: https://security.snyk.io/vuln/SNYK-JS-MYSQL2-6670046
  Introduced through: mysql2@1.7.0
  From: mysql2@1.7.0

✗ Critical severity vulnerability found in mathjs
  Description: Arbitrary Code Execution
  Info: https://security.snyk.io/vuln/npm:mathjs:20171118-1
  Introduced through: mathjs@3.10.1
  From: mathjs@3.10.1



Organization:      valenciamars
Package manager:   npm
Open source:       yes
Project path:      https://github.com/appsecco/dvna

Tested https://github.com/appsecco/dvna for known vulnerabilities, found 46 v
ulnerabilities, 60 vulnerable paths.



┌──(maria㉿kali)-[~]
└─$ 
```

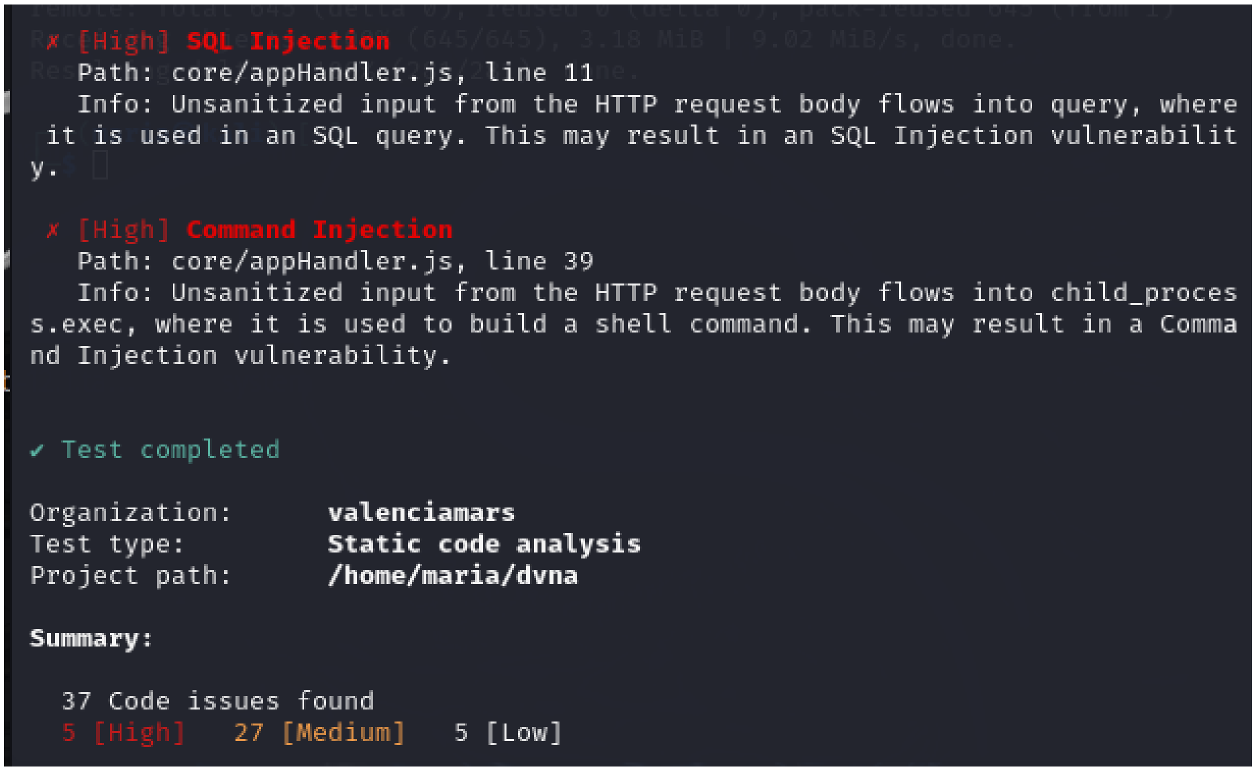
One of the vulnerabilities is “Prototype Pollution” which is of medium severity found in tough-cookie. The cause occurs when an attacker can manipulate the prototype of JavaScript objects, leading to arbitrary code execution, data tampering or denial of service attacks. The impact could be a security breach. Attackers could exploit this to modify object properties globally, leading to unexpected behavior such as application crashes and code execution.

Step 4: Snyk SAST Scan

I executed a static application security test (SAST) on the DVNA local repository using snyk. I navigated to the dvna directory and ran the following snyk comand.



```
┌──(maria㉿kali)-[~/dvna]
└─$ snyk code test

Testing /home/maria/dvna ...

 ✗ [Low] Sensitive Cookie in HTTPS Session Without 'Secure' Attribute
   Path: server.js, line 27
   Info: Cookie has the Secure attribute set to false. Set it to true to prot
ect the cookie from man-in-the-middle attacks.

 ✗ [Low] Use of Password Hash With Insufficient Computational Effort
   Path: core/authHandler.js, line 49
   Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure
 hashing algorithm.

 ✗ [Low] Use of Password Hash With Insufficient Computational Effort
   Path: core/authHandler.js, line 78
   Info: MD5 hash (used in md5) is insecure. Consider changing it to a secure
 hashing algorithm.

 ✗ [Low] Improper Type Validation
   Path: core/appHandler.js, line 150
   Info: The type of this object, coming from body and the value of its lengt
h property can be controlled by the user. An attacker may craft the propertie
s of the object to crash the application or bypass its logic. Consider checki
ng the type of the object.
```



```
 ✗ [High] SQL Injection
   Path: core/appHandler.js, line 11
   Info: Unsanitized input from the HTTP request body flows into query, where
 it is used in an SQL query. This may result in an SQL Injection vulnerabilit
y.

 ✗ [High] Command Injection
   Path: core/appHandler.js, line 39
   Info: Unsanitized input from the HTTP request body flows into child_proces
s.exec, where it is used to build a shell command. This may result in a Comma
nd Injection vulnerability.


✔ Test completed

Organization:      valenciamars
Test type:         Static code analysis
Project path:      /home/maria/dvna

Summary:

  37 Code issues found
  5 [High]   27 [Medium]   5 [Low]
```

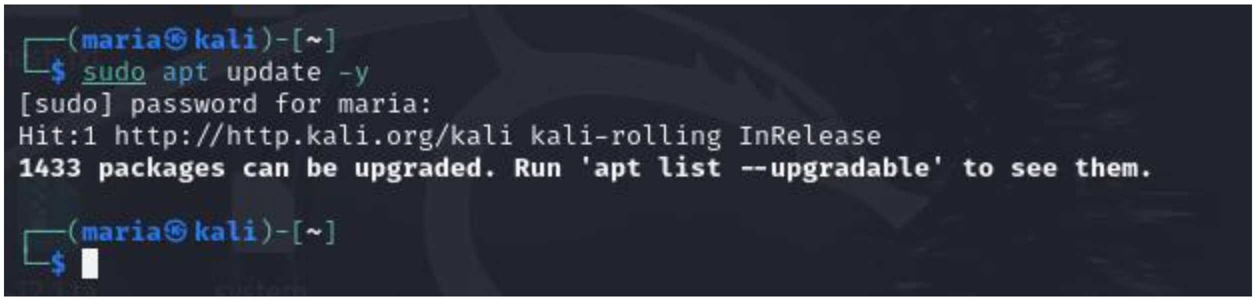
The vulnerability I researched was the Command Injection. This occurs when an application allows users to execute arbitrary system commands on the host machine. The cause can be unsanitized input. Unsanitized input is any data that is provided by a user that has not been properly checked, cleaned or "Sanitized" before it's used by an application. A big impact could be system compromise. An attacker could gain control over the system if an attacker successfully injects commands.

Exercise 8.3 DAST Scan

In this task, I ran a DVNA web application and scan it using Dastardly from my Kali VM in Bridged Adapter network mode.

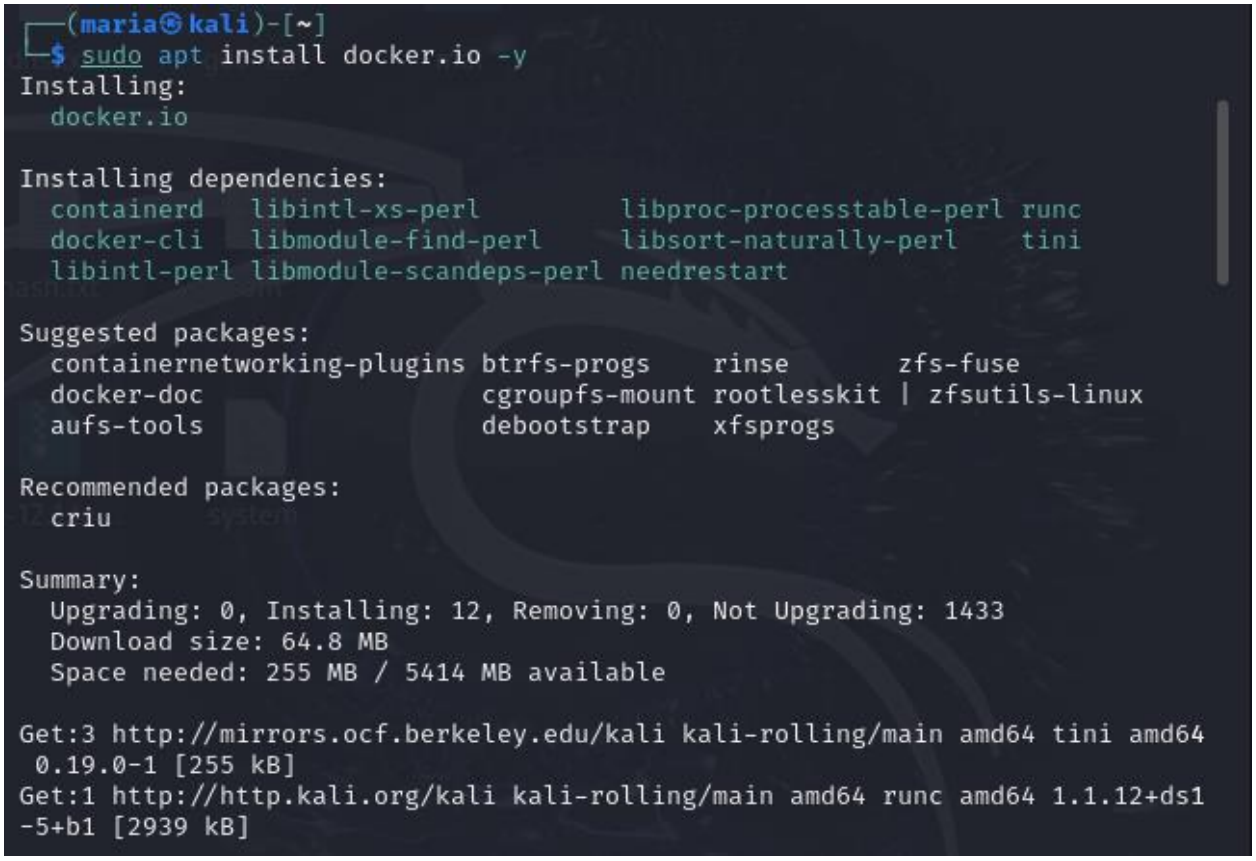
<u>Step 1: Install Docker</u>

Both Dastardly and DVNA will run in containers using docker. I updated my Kali VM to ensure all required packages are on the needed versions.



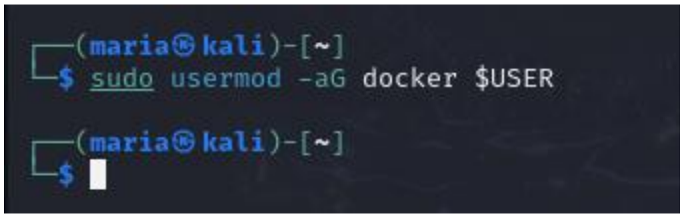
```
┌──(maria㉿kali)-[~]
└─$ sudo apt update -y
[sudo] password for maria:
Hit:1 http://http.kali.org/kali kali-rolling InRelease
1433 packages can be upgraded. Run 'apt list --upgradable' to see them.

┌──(maria㉿kali)-[~]
└─$ 
```

After the update was done, I installed the Docker packages.



```
┌──(maria㉿kali)-[~]
└─$ sudo apt install docker.io -y
Installing:
  docker.io

Installing dependencies:
  containerd    libintl-xs-perl          libproc-processtable-perl runc
  docker-cli    libmodule-find-perl      libsort-naturally-perl    tini
  libintl-perl libmodule-scandeps-perl needrestart

Suggested packages:
  containernetworking-plugins btrfs-progs    rinse       zfs-fuse
  docker-doc                  cgroupfs-mount rootlesskit | zfsutils-linux
  aufs-tools                  debootstrap    xfsprogs

Recommended packages:
  criu

Summary:
  Upgrading: 0, Installing: 12, Removing: 0, Not Upgrading: 1433
  Download size: 64.8 MB
  Space needed: 255 MB / 5414 MB available

Get:3 http://mirrors.ocf.berkeley.edu/kali kali-rolling/main amd64 tini amd64
 0.19.0-1 [255 kB]
Get:1 http://http.kali.org/kali kali-rolling/main amd64 runc amd64 1.1.12+ds1
-5+b1 [2939 kB]
```

Once the docker was installed, I added my Kali VM to the docker group so it can run Docker commands without root.

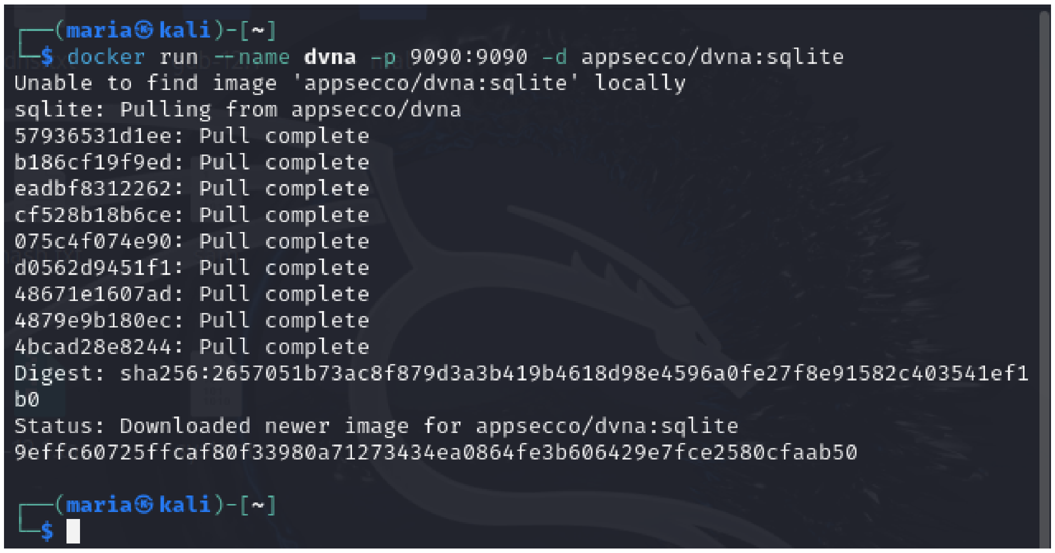


```
┌──(maria㉿kali)-[~]
└─$ sudo usermod -aG docker $USER

┌──(maria㉿kali)-[~]
└─$ 
```

I rebooted my Kali VM so that the changes could take effect.

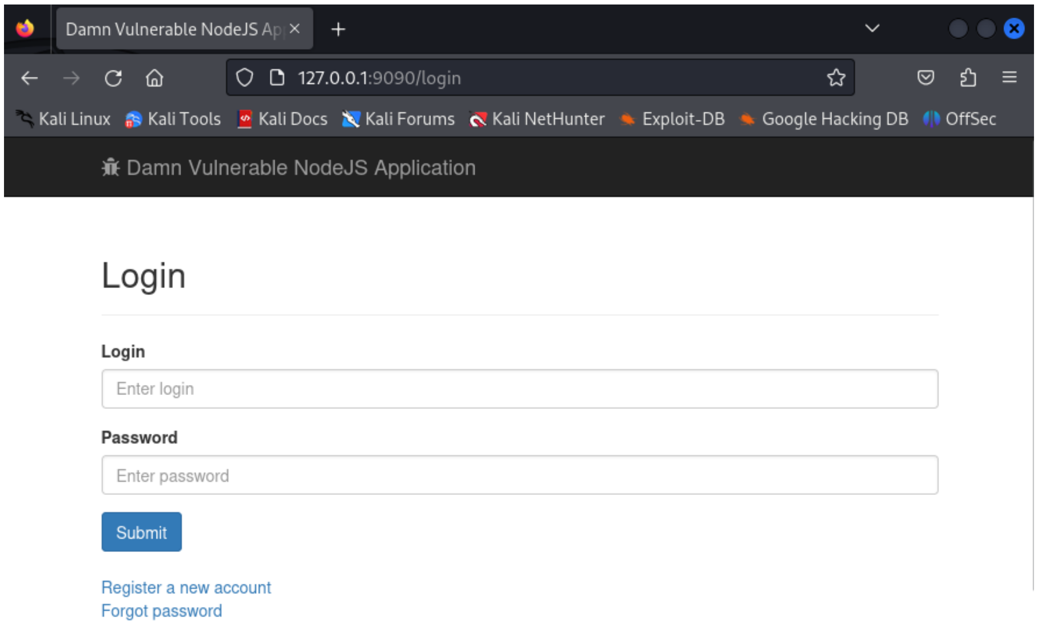
Step 2: Run DVNA

Once my Kali VM account has logged in again with docker group permissions, I opened a terminal and ran the DVNA docker container. The image downloaded and the container will launch while forwarding port 9090 to the Kali host.



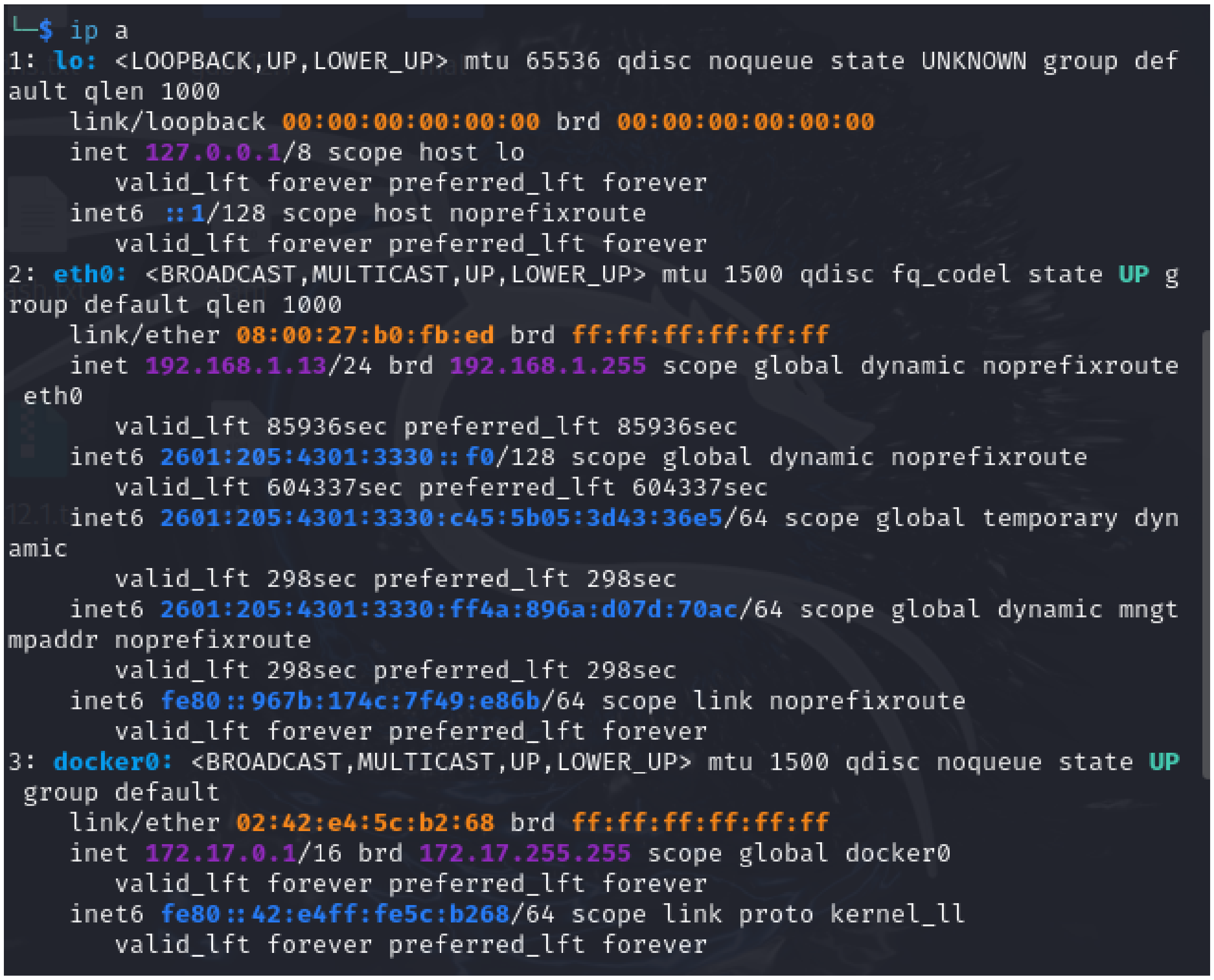
```
┌──(maria㉿kali)-[~]
└─$ docker run --name dvna -p 9090:9090 -d appsecco/dvna:sqlite
Unable to find image 'appsecco/dvna:sqlite' locally
sqlite: Pulling from appsecco/dvna
57936531d1ee: Pull complete
b186cf19f9ed: Pull complete
eadbf8312262: Pull complete
cf528b18b6ce: Pull complete
075c4f074e90: Pull complete
d0562d9451f1: Pull complete
48671e1607ad: Pull complete
4879e9b180ec: Pull complete
4bcad28e8244: Pull complete
Digest: sha256:2657051b73ac8f879d3a3b419b4618d98e4596a0fe27f8e91582c403541ef1
b0
Status: Downloaded newer image for appsecco/dvna:sqlite
9effc60725ffcaf80f33980a71273434ea0864fe3b606429e7fce2580cfaab50

┌──(maria㉿kali)-[~]
└─$ 
```

I opened the Firefox browser within the Klai VM and navigated to http://127.0.0.1:9090. I observed the DVNA is up and running!



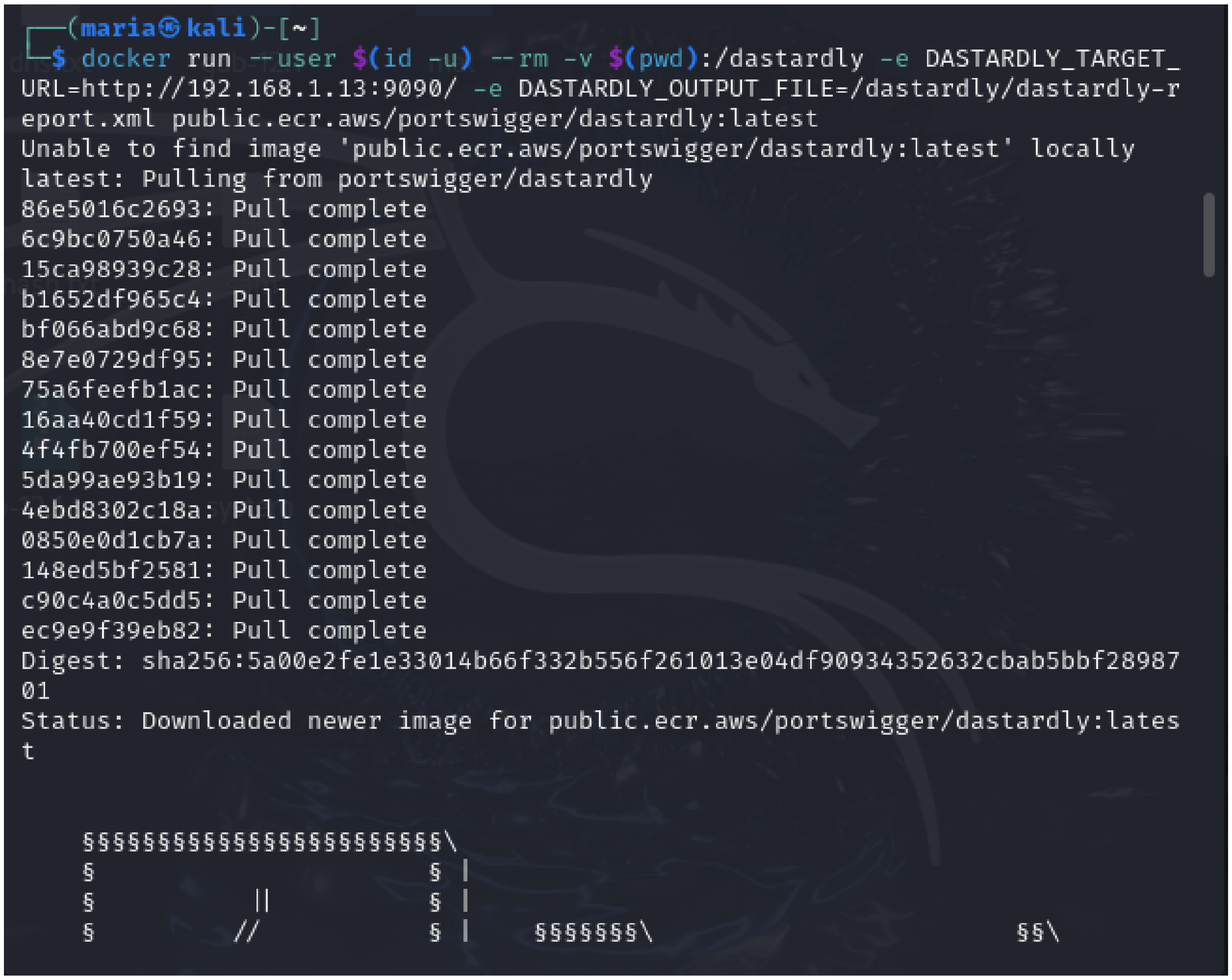
Step 3: Run Dastardly Against DVNA

From my Kali VM terminal, I looked up the VM's IP address under interface eth0 to use as a target for Dastardly.



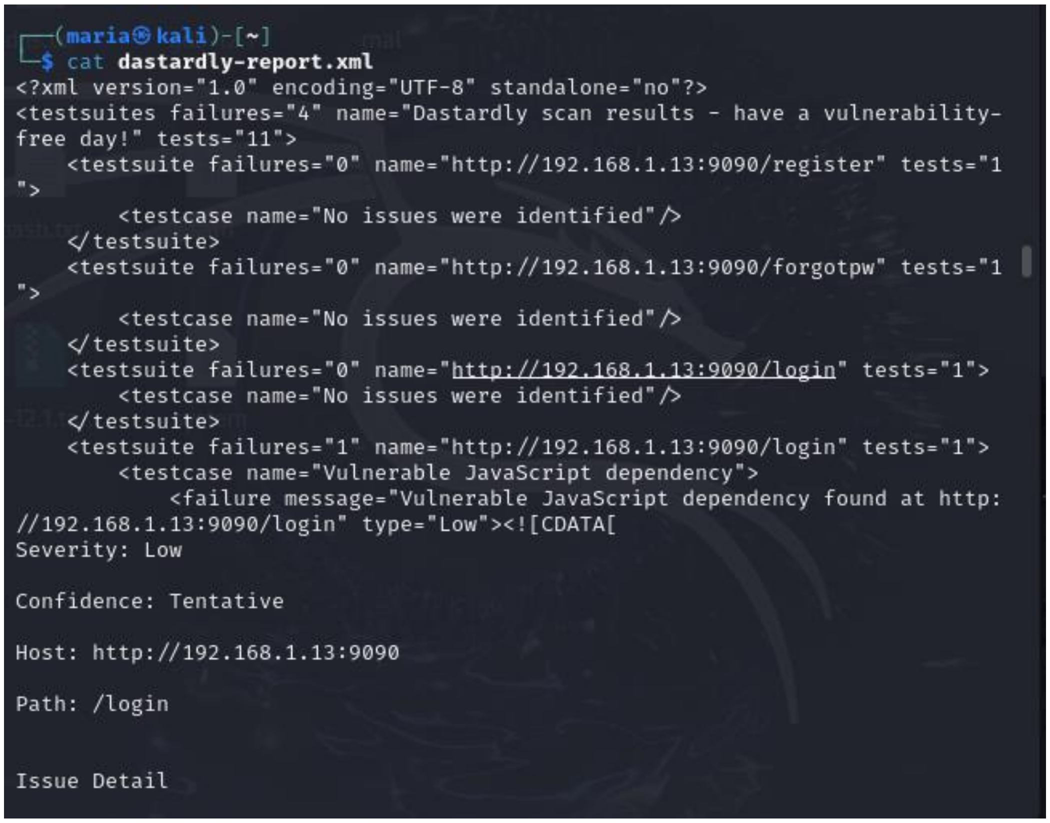
```
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group def
ault qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g
roup default qlen 1000
    link/ether 08:00:27:b0:fb:ed brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.13/24 brd 192.168.1.255 scope global dynamic noprefixroute
 eth0
       valid_lft 85936sec preferred_lft 85936sec
    inet6 2601:205:4301:3330::f0/128 scope global dynamic noprefixroute
       valid_lft 604337sec preferred_lft 604337sec
    inet6 2601:205:4301:3330:c45:5b05:3d43:36e5/64 scope global temporary dyn
amic
       valid_lft 298sec preferred_lft 298sec
    inet6 2601:205:4301:3330:ff4a:896a:d07d:70ac/64 scope global dynamic mngt
mpaddr noprefixroute
       valid_lft 298sec preferred_lft 298sec
    inet6 fe80::967b:174c:7f49:e86b/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
 group default
    link/ether 02:42:e4:5c:b2:68 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:e4ff:fe5c:b268/64 scope link proto kernel_ll
       valid_lft forever preferred_lft forever
```

While the DVNA application is running, I launched a Dastardly container targetting the local DVNA server. I made sure to replace the IP_Address with the IP address of the Kali VM.

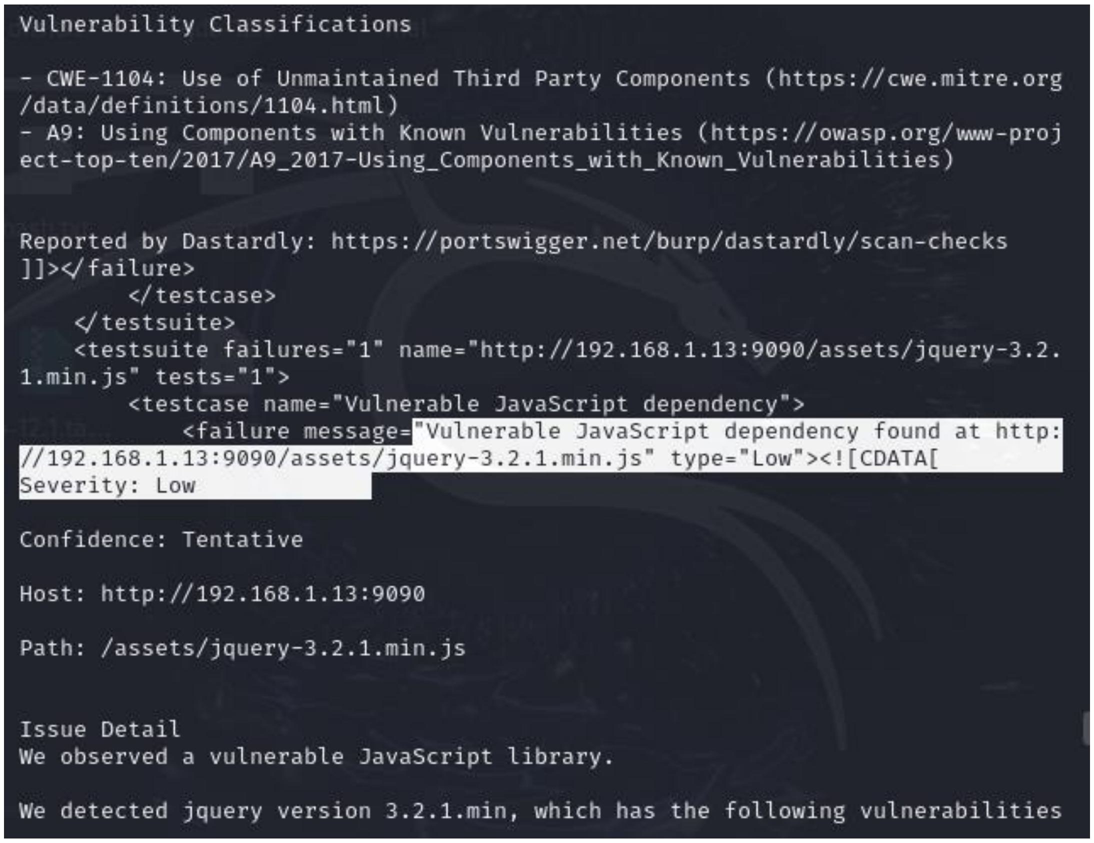


```
┌──(maria㉿kali)-[~]
└─$ docker run --user $(id -u) --rm -v $(pwd):/dastardly -e DASTARDLY_TARGET_
URL=http://192.168.1.13:9090/ -e DASTARDLY_OUTPUT_FILE=/dastardly/dastardly-r
eport.xml public.ecr.aws/portswigger/dastardly:latest
Unable to find image 'public.ecr.aws/portswigger/dastardly:latest' locally
latest: Pulling from portswigger/dastardly
86e5016c2693: Pull complete
6c9bc0750a46: Pull complete
15ca98939c28: Pull complete
b1652df965c4: Pull complete
bf066abd9c68: Pull complete
8e7e0729df95: Pull complete
75a6feefb1ac: Pull complete
16aa40cd1f59: Pull complete
4f4fb700ef54: Pull complete
5da99ae93b19: Pull complete
4ebd8302c18a: Pull complete
0850e0d1cb7a: Pull complete
148ed5bf2581: Pull complete
c90c4a0c5dd5: Pull complete
ec9e9f39eb82: Pull complete
Digest: sha256:5a00e2fe1e33014b66f332b556f261013e04df90934352632cbab5bbf28987
01
Status: Downloaded newer image for public.ecr.aws/portswigger/dastardly:lates
t


    §§§§§§§§§§§§§§§§§§§§§§§§\
    §                      § |
    §          ||          § |
    §         //           § |    §§§§§§§\                        §§\
```

After some minutes, the scan completed with a few low findings. Dastardly is unable to scan authenticated pages and tests for only a few vulnerability classes. Using cat, I displayed the vulnerabilities found from the DAST scan.



```
┌──(maria㉿kali)-[~]
└─$ cat dastardly-report.xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<testsuites failures="4" name="Dastardly scan results - have a vulnerability-
free day!" tests="11">
    <testsuite failures="0" name="http://192.168.1.13:9090/register" tests="1
">
        <testcase name="No issues were identified"/>
    </testsuite>
    <testsuite failures="0" name="http://192.168.1.13:9090/forgotpw" tests="1
">
        <testcase name="No issues were identified"/>
    </testsuite>
    <testsuite failures="0" name="http://192.168.1.13:9090/login" tests="1">
        <testcase name="No issues were identified"/>
    </testsuite>
    <testsuite failures="1" name="http://192.168.1.13:9090/login" tests="1">
        <testcase name="Vulnerable JavaScript dependency">
            <failure message="Vulnerable JavaScript dependency found at http:
//192.168.1.13:9090/login" type="Low"><![CDATA[
Severity: Low

Confidence: Tentative

Host: http://192.168.1.13:9090

Path: /login


Issue Detail
```



```
Vulnerability Classifications

- CWE-1104: Use of Unmaintained Third Party Components (https://cwe.mitre.org
/data/definitions/1104.html)
- A9: Using Components with Known Vulnerabilities (https://owasp.org/www-proj
ect-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities)


Reported by Dastardly: https://portswigger.net/burp/dastardly/scan-checks
]]></failure>
        </testcase>
    </testsuite>
    <testsuite failures="1" name="http://192.168.1.13:9090/assets/jquery-3.2.
1.min.js" tests="1">
        <testcase name="Vulnerable JavaScript dependency">
            <failure message="Vulnerable JavaScript dependency found at http:
//192.168.1.13:9090/assets/jquery-3.2.1.min.js" type="Low"><![CDATA[
Severity: Low

Confidence: Tentative

Host: http://192.168.1.13:9090

Path: /assets/jquery-3.2.1.min.js


Issue Detail
We observed a vulnerable JavaScript library.

We detected jquery version 3.2.1.min, which has the following vulnerabilities
```