Automated Infrastructure Security

In this lab, we will demonstrate how to automate infrastructure security testing against the NodeGoat website. The infrastructure security testing will include known vulnerable JavaScript libraries, insecure SSL configurations, and the advanced NMAP NSE script testing technique for web security. At the end, we will also illustrate how to apply the BDD automation framework to SSLScan and NMAP.

The topics that will be covered in this lab are as follows:

- Scan For known JavaScript vulnerabilities
- Scanning with OWASP dependency check
- Secure communication scan with SSLScan
- NMAP security scan with the BDD framework

Scan For known JavaScript vulnerabilities

We will be using RetireJS because it's simple to use and provides several ways of scanning, such as a command-line scanner, Grunt plugin, browser (Chrome/Firefox) extension, and also the Burp and OWASP Zap plugins. In our demonstration, we will be using the command-line scanner in the following steps.

Step 1 -- install RetireJS

RetireJS has been installed in the lab environment using `npm`

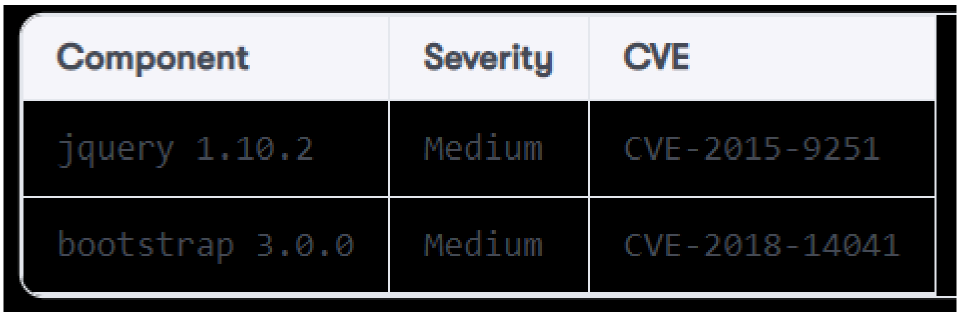
Step 2 -- scan with RetireJS

Once it's installed, we may specify the target project to be scanned. In our example, we will scan the whole project under the [/NodeGoat/] path:

```
cd C:\Users\fenago\Desktop\DevSecOps-course\lab13

retire   --path   ./NodeGoat/    --colors
```

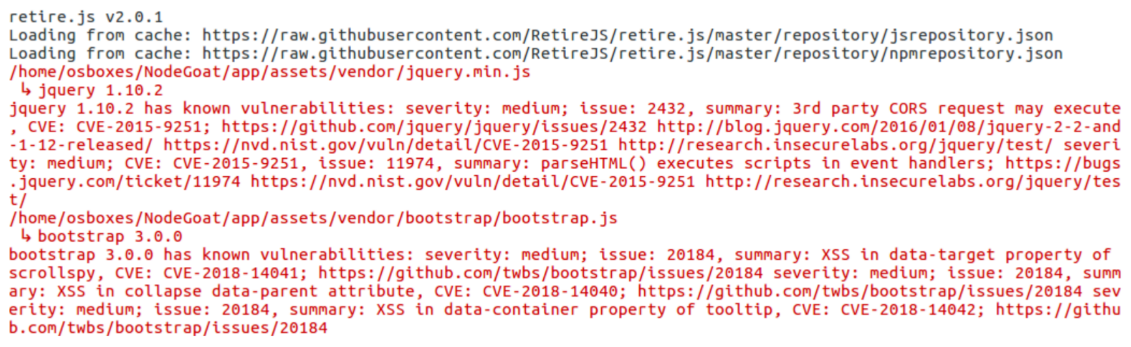
Step 3 -- review the retireJS results

The RetireJS scanning results show critical issues in red. There are two major known vulnerabilities with these JavaScript libraries:



Component	Severity	CVE
jquery 1.10.2	Medium	CVE-2015-9251
bootstrap 3.0.0	Medium	CVE-2018-14041

The following screenshot shows the retireJS scanning results for the NodeGoat project:



```
retire.js v2.0.1
Loading from cache: https://raw.githubusercontent.com/RetireJS/retire.js/master/repository/jsrepository.json
Loading from cache: https://raw.githubusercontent.com/RetireJS/retire.js/master/repository/npmrepository.json
/home/osboxes/NodeGoat/app/assets/vendor/jquery.min.js
 ↳ jquery 1.10.2
jquery 1.10.2 has known vulnerabilities: severity: medium; issue: 2432, summary: 3rd party CORS request may execute
, CVE: CVE-2015-9251; https://github.com/jquery/jquery/issues/2432 http://blog.jquery.com/2016/01/08/jquery-2-2-and
-1-12-released/ https://nvd.nist.gov/vuln/detail/CVE-2015-9251 http://research.insecurelabs.org/jquery/test/ severi
ty: medium; CVE: CVE-2015-9251, issue: 11974, summary: parseHTML() executes scripts in event handlers; https://bugs
.jquery.com/ticket/11974 https://nvd.nist.gov/vuln/detail/CVE-2015-9251 http://research.insecurelabs.org/jquery/tes
t/
/home/osboxes/NodeGoat/app/assets/vendor/bootstrap/bootstrap.js
 ↳ bootstrap 3.0.0
bootstrap 3.0.0 has known vulnerabilities: severity: medium; issue: 20184, summary: XSS in data-target property of
scrollspy, CVE: CVE-2018-14041; https://github.com/twbs/bootstrap/issues/20184 severity: medium; issue: 20184, summ
ary: XSS in collapse data-parent attribute, CVE: CVE-2018-14040; https://github.com/twbs/bootstrap/issues/20184 sev
erity: medium; issue: 20184, summary: XSS in data-container property of tooltip, CVE: CVE-2018-14042; https://githu
b.com/twbs/bootstrap/issues/20184
```

WebGoat with OWASP dependency check

In addition to RetireJS, we will also the OWASP dependency check to scan all the files of the NodeGoat project for known vulnerable libraries. Follow these steps for the OWASP dependency check scan.

Step 1 -- prepare WebGoat environment

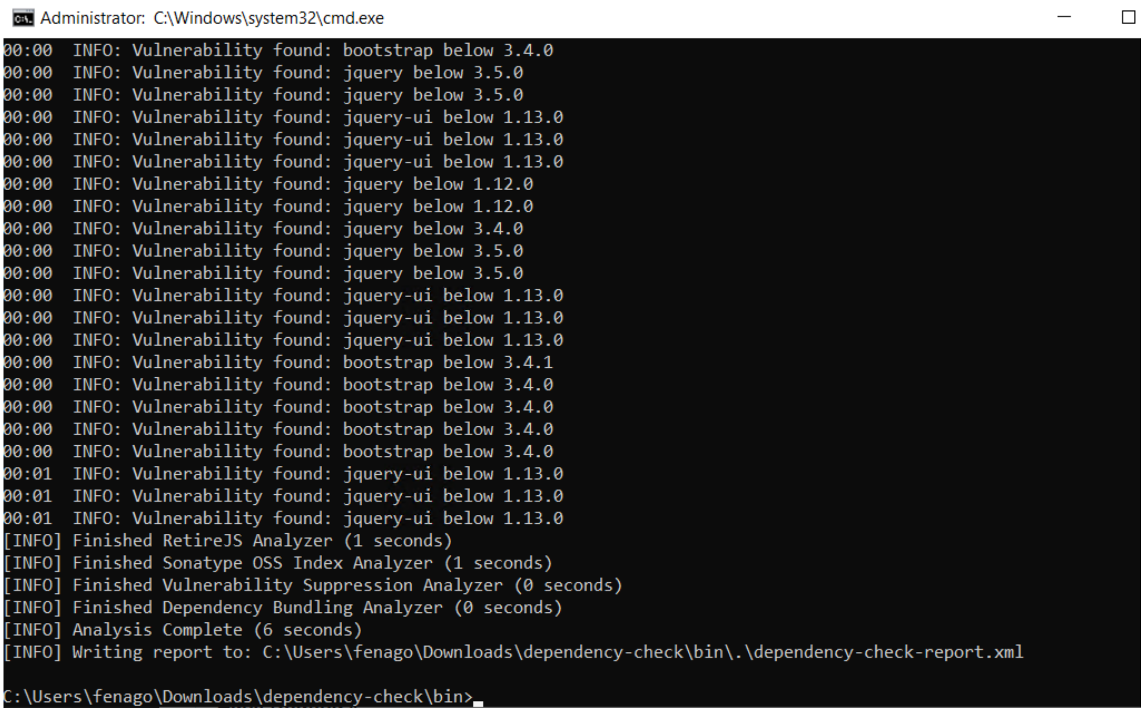
To better demonstrate the scanning results of the OWASP dependency check, we will use the WebGoat project instead of NodeGoat. The WebGoat project can be downloaded from Git. WebGoat is a purpose-built vulnerable web project used to practice security testing:

```
git    clone    https://github.com/WebGoat/WebGoat
```

Step 2 -- dependency check scan

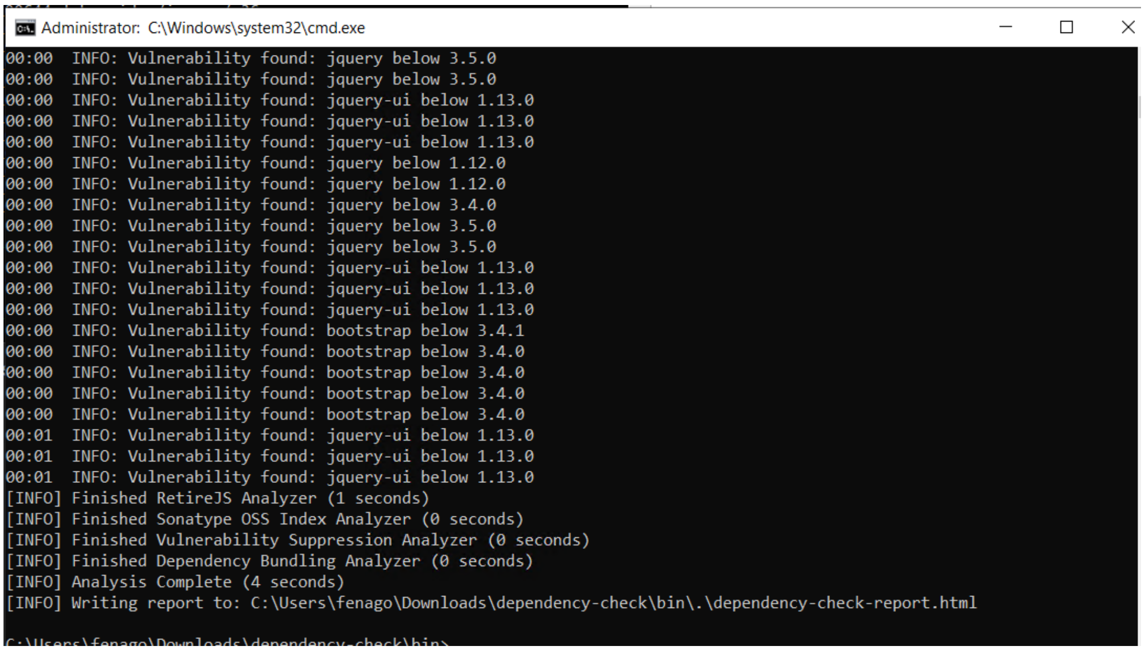
To execute the dependency-check, locate the [\dependency-check\bin\] path. Refer to the following command for the WebGoat project scan:

```
cd C:\Users\fenago\Downloads\dependency-check\bin

dependency-check --project WebGoat --format XML --scan
C:\Users\fenago\Desktop\DevSecOps-course\lab13\WebGoat

dependency-check --project WebGoat --format HTML --scan
C:\Users\fenago\Desktop\DevSecOps-course\lab13\WebGoat
```



Administrator: C:\Windows\system32\cmd.exe

```
00:00  INFO: Vulnerability found: bootstrap below 3.4.0
00:00  INFO: Vulnerability found: jquery below 3.5.0
00:00  INFO: Vulnerability found: jquery below 3.5.0
00:00  INFO: Vulnerability found: jquery-ui below 1.13.0
00:00  INFO: Vulnerability found: jquery-ui below 1.13.0
00:00  INFO: Vulnerability found: jquery-ui below 1.13.0
00:00  INFO: Vulnerability found: jquery below 1.12.0
00:00  INFO: Vulnerability found: jquery below 1.12.0
00:00  INFO: Vulnerability found: jquery below 3.4.0
00:00  INFO: Vulnerability found: jquery below 3.5.0
00:00  INFO: Vulnerability found: jquery below 3.5.0
00:00  INFO: Vulnerability found: jquery-ui below 1.13.0
00:00  INFO: Vulnerability found: jquery-ui below 1.13.0
00:00  INFO: Vulnerability found: jquery-ui below 1.13.0
00:00  INFO: Vulnerability found: bootstrap below 3.4.1
00:00  INFO: Vulnerability found: bootstrap below 3.4.0
00:00  INFO: Vulnerability found: bootstrap below 3.4.0
00:00  INFO: Vulnerability found: bootstrap below 3.4.0
00:00  INFO: Vulnerability found: bootstrap below 3.4.0
00:01  INFO: Vulnerability found: jquery-ui below 1.13.0
00:01  INFO: Vulnerability found: jquery-ui below 1.13.0
00:01  INFO: Vulnerability found: jquery-ui below 1.13.0
[INFO] Finished RetireJS Analyzer (1 seconds)
[INFO] Finished Sonatype OSS Index Analyzer (1 seconds)
[INFO] Finished Vulnerability Suppression Analyzer (0 seconds)
[INFO] Finished Dependency Bundling Analyzer (0 seconds)
[INFO] Analysis Complete (6 seconds)
[INFO] Writing report to: C:\Users\fenago\Downloads\dependency-check\bin\.\dependency-check-report.xml

C:\Users\fenago\Downloads\dependency-check\bin>
```



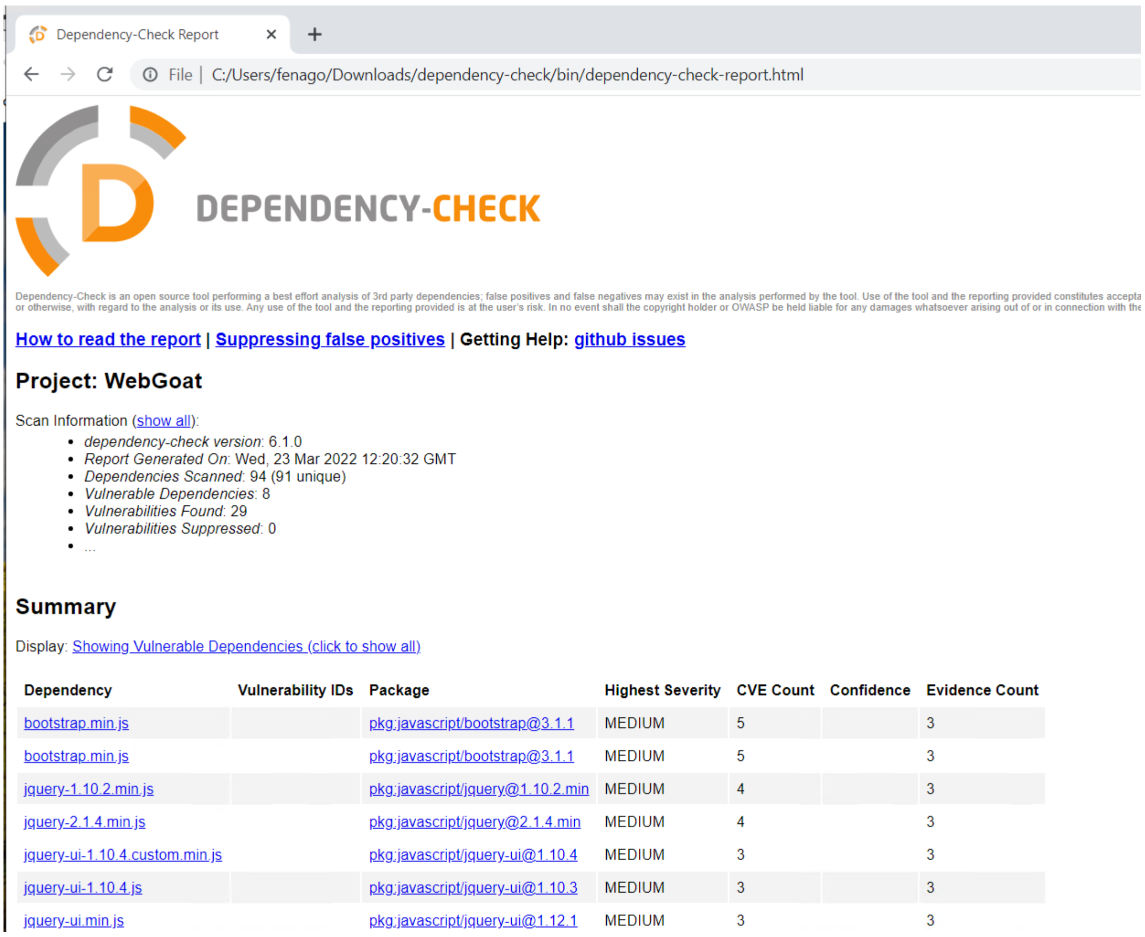
Administrator: C:\Windows\system32\cmd.exe

```
00:00  INFO: Vulnerability found: jquery below 3.5.0
00:00  INFO: Vulnerability found: jquery below 3.5.0
00:00  INFO: Vulnerability found: jquery-ui below 1.13.0
00:00  INFO: Vulnerability found: jquery-ui below 1.13.0
00:00  INFO: Vulnerability found: jquery-ui below 1.13.0
00:00  INFO: Vulnerability found: jquery below 1.12.0
00:00  INFO: Vulnerability found: jquery below 1.12.0
00:00  INFO: Vulnerability found: jquery below 3.4.0
00:00  INFO: Vulnerability found: jquery below 3.5.0
00:00  INFO: Vulnerability found: jquery below 3.5.0
00:00  INFO: Vulnerability found: jquery-ui below 1.13.0
00:00  INFO: Vulnerability found: jquery-ui below 1.13.0
00:00  INFO: Vulnerability found: jquery-ui below 1.13.0
00:00  INFO: Vulnerability found: bootstrap below 3.4.1
00:00  INFO: Vulnerability found: bootstrap below 3.4.0
00:00  INFO: Vulnerability found: bootstrap below 3.4.0
00:00  INFO: Vulnerability found: bootstrap below 3.4.0
00:00  INFO: Vulnerability found: bootstrap below 3.4.0
00:01  INFO: Vulnerability found: jquery-ui below 1.13.0
00:01  INFO: Vulnerability found: jquery-ui below 1.13.0
00:01  INFO: Vulnerability found: jquery-ui below 1.13.0
[INFO] Finished RetireJS Analyzer (1 seconds)
[INFO] Finished Sonatype OSS Index Analyzer (0 seconds)
[INFO] Finished Vulnerability Suppression Analyzer (0 seconds)
[INFO] Finished Dependency Bundling Analyzer (0 seconds)
[INFO] Analysis Complete (4 seconds)
[INFO] Writing report to: C:\Users\fenago\Downloads\dependency-check\bin\.\dependency-check-report.html

C:\Users\fenago\Downloads\dependency-check\bin>
```

The XML report format can be useful to import into other security management tools.

Step 3 -- review the OWASP dependency-check report

After the scan, the report will be generated under the [\dependency-check\bin\] execution path. The filenames are [dependency-check-report.html] and the [dependency-check-report.xml]:


Dependency-Check Report
File | C:/Users/fenago/Downloads/dependency-check/bin/dependency-check-report.html
DEPENDENCY-CHECK

Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes accepta
or otherwise, with regard to the analysis or its use. Any use of the tool and the reporting provided is at the user's risk. In no event shall the copyright holder or OWASP be held liable for any damages whatsoever arising out of or in connection with the

How to read the report | Suppressing false positives | Getting Help: github issues

Project: WebGoat

Scan Information (show all):

- *dependency-check version*: 6.1.0
- *Report Generated On*: Wed, 23 Mar 2022 12:20:32 GMT
- *Dependencies Scanned*: 94 (91 unique)
- *Vulnerable Dependencies*: 8
- *Vulnerabilities Found*: 29
- *Vulnerabilities Suppressed*: 0
- ...

Summary

Display: Showing Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
bootstrap.min.js		pkg:javascript/bootstrap@3.1.1	MEDIUM	5		3
bootstrap.min.js		pkg:javascript/bootstrap@3.1.1	MEDIUM	5		3
jquery-1.10.2.min.js		pkg:javascript/jquery@1.10.2.min	MEDIUM	4		3
jquery-2.1.4.min.js		pkg:javascript/jquery@2.1.4.min	MEDIUM	4		3
jquery-ui-1.10.4.custom.min.js		pkg:javascript/jquery-ui@1.10.4	MEDIUM	3		3
jquery-ui-1.10.4.js		pkg:javascript/jquery-ui@1.10.3	MEDIUM	3		3
jquery-ui.min.js		pkg:javascript/jquery-ui@1.12.1	MEDIUM	3		3

Secure communication scan with SSLScan

In this demonstration, we will inspect vulnerable security configurations with HTTPS. The tool we will be using is SSLScan. Follow these steps to perform the scan.

Step -- SSLScan scan

To execute sslscan, we will specify the output as XML, and also specify the target website's URL:

```
cd C:\Users\fenago\Desktop\DevSecOps-course\lab13

sslscan --no-failed --xml=nodegoat_SSLscan.xml  nodegoat.kerokuapp.com
```

[--no-failed] means only *accepted* connections will be listed in the test results. When reviewing the SSLScan test results, we will only focus on those connections with *accepted*.

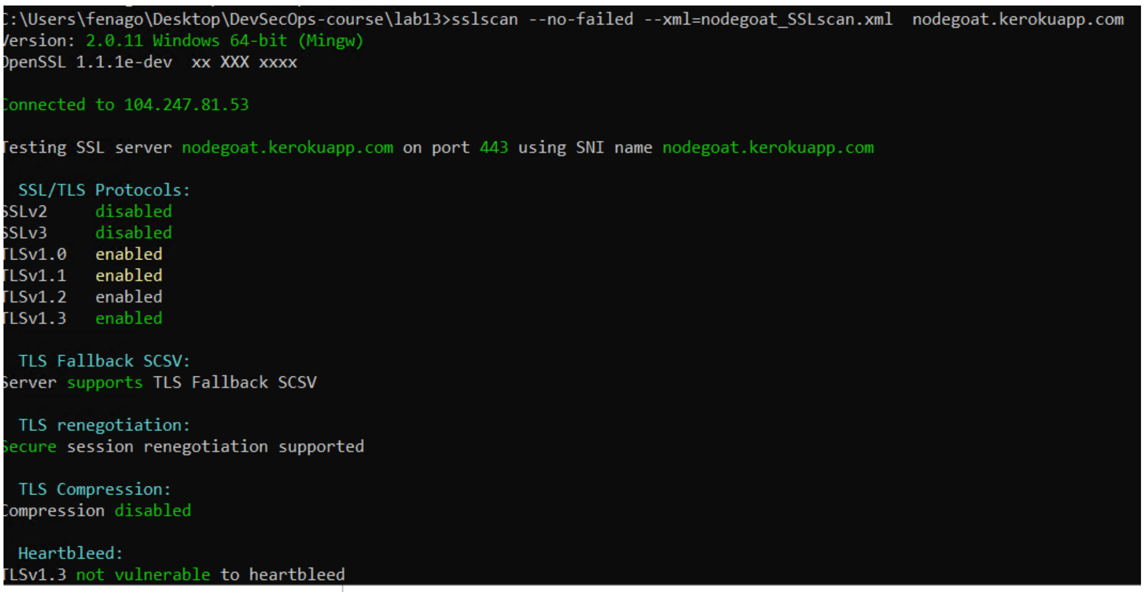
Step 3 -- review the SSLScan results

Here are the SSLScan results without the options of [--no-failed]. Please focus on connections with *accepted* only. When reading the SSLScan test results, we will focus on weak HTTPS protocols and encryption algorithms such as

SSL v3, TLS v1.0, TLS v1.1, and NULL. Generally, the following will be considered as vulnerable:

- SSLv2 and SSLv3
- Symmetric encryption algorithms smaller than 112 bits
- X509 certificates with RSA or DSA keys smaller than 2048 bits
- Weak hash algorithms such as MD5

This screenshot shows the sslScan results for the NodeGoat website:



```
:\Users\fenago\Desktop\DevSecOps-course\lab13>sslscan --no-failed --xml=nodegoat_SSLscan.xml  nodegoat.kerokuapp.com
ersion: 2.0.11 Windows 64-bit (Mingw)
penSSL 1.1.1e-dev  xx XXX xxxx

onnected to 104.247.81.53

esting SSL server nodegoat.kerokuapp.com on port 443 using SNI name nodegoat.kerokuapp.com

  SSL/TLS Protocols:
SLv2     disabled
SLv3     disabled
LSv1.0   enabled
LSv1.1   enabled
LSv1.2   enabled
LSv1.3   enabled

  TLS Fallback SCSV:
erver supports TLS Fallback SCSV

  TLS renegotiation:
ecure session renegotiation supported

  TLS Compression:
ompression disabled

  Heartbleed:
LSv1.3 not vulnerable to heartbleed
```

In addition to SSLScan, we can also use SSLTest, SSLyze, or NAMP for SSL configuration inspection.

To read the SSLScan test results, focus on the connections with *accepted* or specify [--no-failed] to reduce unnecessary information.

Step 4 -- fix the HTTPS secure configurations

The secure HTTPS configuration of a website can be very tedious and prone to mistakes. It's suggested to use the Mozilla SSL configuration generator. It will help to generate secure SSL configurations based on the web server. Refer to https://mozilla.github.io/server-side-tls/ssl-config-generator/ for details.

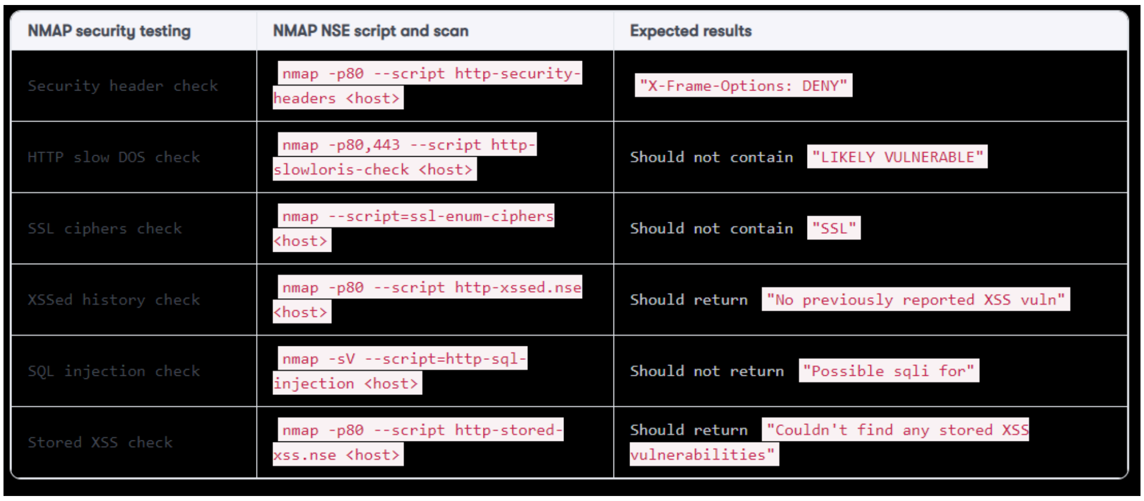
NMAP security scan with BDD framework

The test results for NMAP may be difficult to understand and take time to interpret for non-security professionals. The purpose of integration with BDD and NMAP allows us to define the NMAP execution in plain English. In this example, we will use NAMP to execute some common web security tests with an **NMAP NSE** (**NMAP Scripting Engine**) script. Due to the execution of NMAP and the scanning results can be difficult to interpret, we will apply the Gauntlt framework to execute NMAP. Please be reminded that NAMP web security testing cannot replace a web scanner such as ZAP, due to the limitations of the security payloads and detection engine of NMAP.

NMAP For web security testing

We will be using the NMAP NSE for the following security testing: security header check, HTTP slow DOS check, SSL cipher check, XSSed history check, SQL injection check, and stored XSS check.

This table lists the NMAP security testing scenario, the NSE script we will need, and the expected results:



NMAP security testing	NMAP NSE script and scan	Expected results
Security header check	`nmap -p80 --script http-security-headers <host>`	`"X-Frame-Options: DENY"`
HTTP slow DOS check	`nmap -p80,443 --script http-slowloris-check <host>`	Should not contain `"LIKELY VULNERABLE"`
SSL ciphers check	`nmap --script=ssl-enum-ciphers <host>`	Should not contain `"SSL"`
XSSed history check	`nmap -p80 --script http-xssed.nse <host>`	Should return `"No previously reported XSS vuln"`
SQL injection check	`nmap -sV --script=http-sql-injection <host>`	Should not return `"Possible sqli for"`
Stored XSS check	`nmap -p80 --script http-stored-xss.nse <host>`	Should return `"Couldn't find any stored XSS vulnerabilities"`

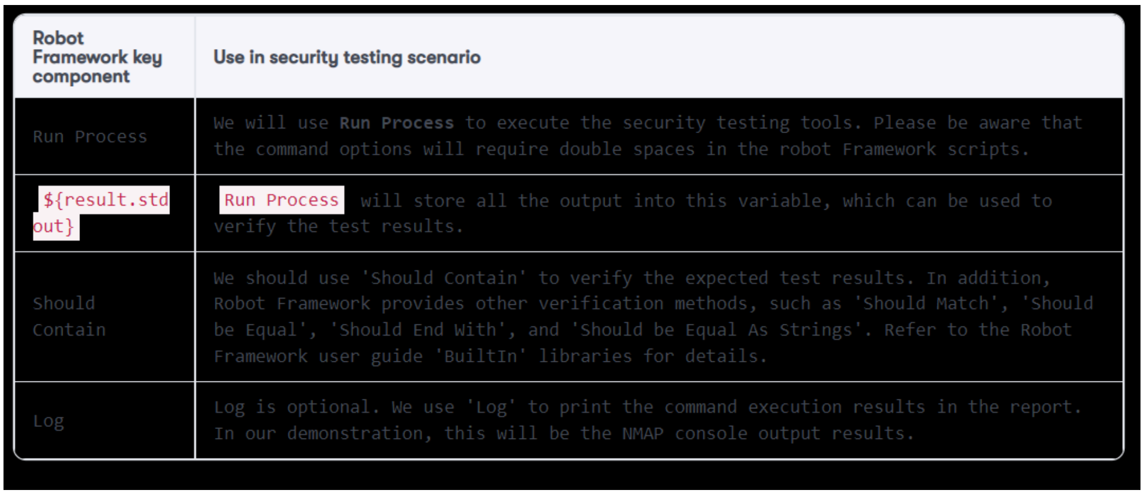
The [-oX] option can be used to generate the output in an XML-format file:

```
nmap -p80 --script http-security-headers nodegoat.kerokuapp.com  -oX
nodeGoat_NmapScan_HTTPheaders.xml
```

We will be using these NMAP security testing commands and expected results to integrate with the BDD framework Gauntlt in the following demonstrations.

NMAP BDD with Robot Framework

We will apply Robot Framework with NMAP in this demonstration. Here are some of the key components to be used in this testing scenario:



Robot Framework key component	Use in security testing scenario
Run Process	We will use **Run Process** to execute the security testing tools. Please be aware that the command options will require double spaces in the robot Framework scripts.
`${result.stdout}`	`Run Process` will store all the output into this variable, which can be used to verify the test results.
Should Contain	We should use 'Should Contain' to verify the expected test results. In addition, Robot Framework provides other verification methods, such as 'Should Match', 'Should be Equal', 'Should End With', and 'Should be Equal As Strings'. Refer to the Robot Framework user guide 'BuiltIn' libraries for details.
Log	Log is optional. We use 'Log' to print the command execution results in the report. In our demonstration, this will be the NMAP console output results.

Step 1 -- define the Robot Framework steps

If you use 'Run Process' to execute NMAP, please be aware that it will require double spaces between each parameter in the command options. Otherwise, the Robot Framework will return a file not found error even if NMAP is installed. In our example, the command options are [-p80 --script http-xssed nodegoat.kerokuapp.com]:

```
*** Settings ***
Library Process

*** Test Cases ***
Testing if the website was previously reported XSS
     ${result} = Run Process nmap  -p80  --script  http-xssed  nodegoat.kerokuapp.com
     Log         ${result.stdout}
     Should Contain ${result.stdout} No previously reported
```

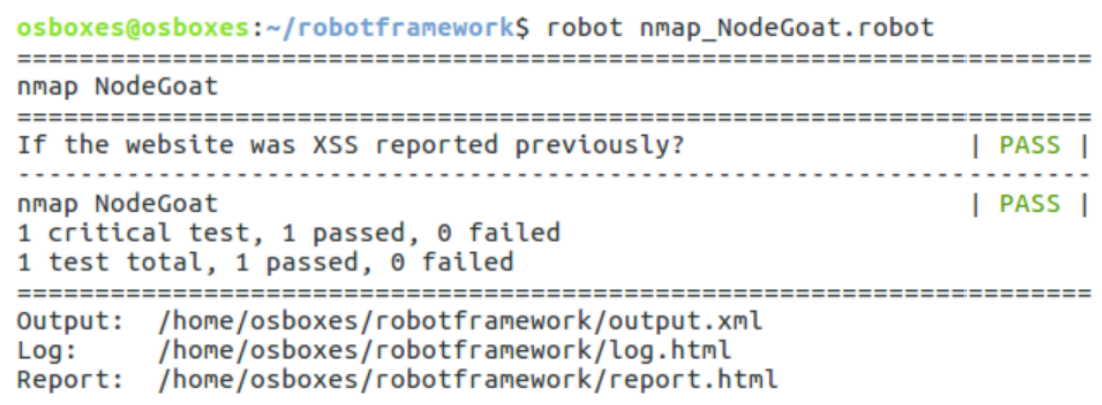
Robot Framework has several verification keywords built in, such as 'Should Be Equal', 'Should Start With', 'Should End With' 'Should Not Match', 'Should Match Regexp', and so on. Refer to http://robotframework.org/robotframework/latest/libraries/BuiltIn.html#Should%20Be for more details.

Step 2 -- execute and review the results

To execute the robot Framework script, use the following command:

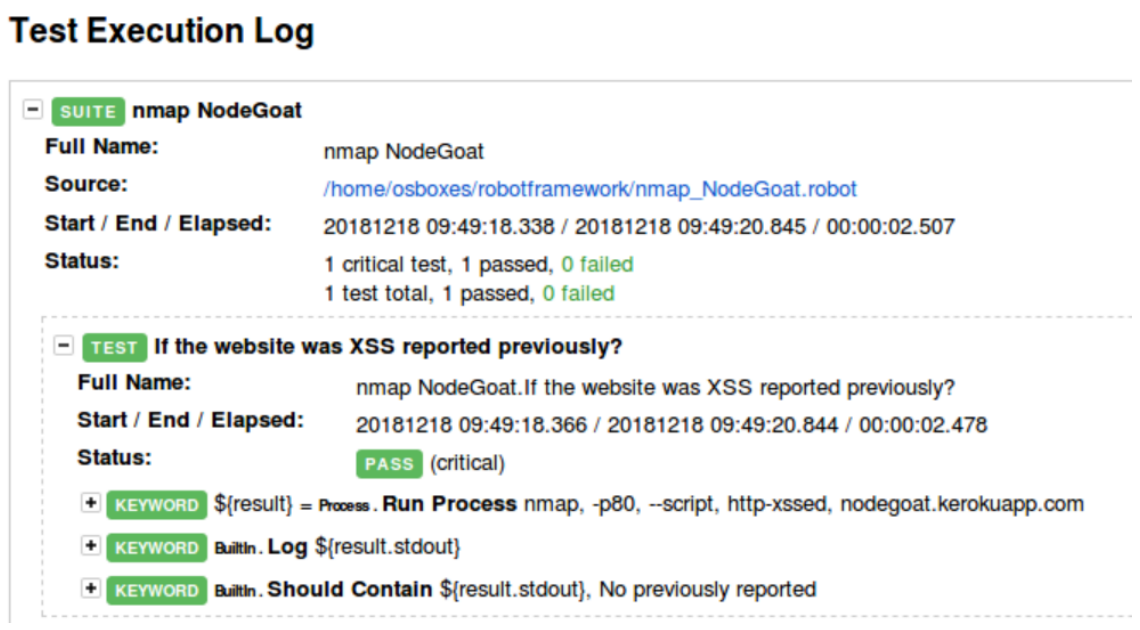
```
robot  nmap_NodeGoat.robot
```

The following screenshot shows the test results for robot framework. It also generates HTML reports:



```
osboxes@osboxes:~/robotframework$ robot nmap_NodeGoat.robot
==============================================================================
nmap NodeGoat
==============================================================================
If the website was XSS reported previously?                           | PASS |
------------------------------------------------------------------------------
nmap NodeGoat                                                         | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed
==============================================================================
Output:  /home/osboxes/robotframework/output.xml
Log:     /home/osboxes/robotframework/log.html
Report:  /home/osboxes/robotframework/report.html
```

Here is one of the Robot framework HTML reports, [log.html]:



Test Execution Log

```
SUITE nmap NodeGoat
Full Name:            nmap NodeGoat
Source:               /home/osboxes/robotframework/nmap_NodeGoat.robot
Start / End / Elapsed: 20181218 09:49:18.338 / 20181218 09:49:20.845 / 00:00:02.507
Status:               1 critical test, 1 passed, 0 failed
                      1 test total, 1 passed, 0 failed

  TEST If the website was XSS reported previously?
  Full Name:            nmap NodeGoat.If the website was XSS reported previously?
  Start / End / Elapsed: 20181218 09:49:18.366 / 20181218 09:49:20.844 / 00:00:02.478
  Status:               PASS (critical)
  KEYWORD ${result} = Process . Run Process nmap, -p80, --script, http-xssed, nodegoat.kerokuapp.com
  KEYWORD BuiltIn . Log ${result.stdout}
  KEYWORD BuiltIn . Should Contain ${result.stdout}, No previously reported
```

Summary

In this lab, we demonstrated infrastructure security testing against the NodeGoat website using RetireJS, OWASP dependency check, SSL communication configuration, and integration with BDD frameworks. These security testing scenarios cover the known vulnerable libraries check, secure SSL configuration check, and basic web security check.

The RetireJS and OWASP dependency check can identify the known CVE of libraries, based on a scan of their project source files. SSLScan is used to inspect any vulnerable HTTPS configurations such as SSL, short encryption keys, and weak HSA or encryption algorithms.

We also illustrated how NMAP NSE can be used to do basic web security inspections such as XSS and SQL injection. An NMAP security scan with the integration of Robot Framework was also demonstrated. BDD testing techniques can help to make infrastructure security testing easier to understand and can even be maintained by a non-security team.