

Technical Assignment Report

Position	Governance, Risk Management & Compliance (GRC)
Executed by	Mark Emelianov, CRISC
Dates	20/10/2022 – 22/10/2022

Contents

Objective.....	2
1. Test setup.....	2
1.2. Host hypervisor.....	3
1.3. Virtual machine.....	3
1.4. Application setup	4
1.5. Vulnerability Scanner setup.....	7
1.6. Code Scanner setup.....	8
1.7. GitHub setup	9
1.8. SonaQube project setup	10
2. Security Scanning.....	11
2.1. Use any opensource tool and setup scanning.....	11
2.2. Perform the security scanning on a test machine (with 5 security loopholes).....	11
2.3. Prepare the scanning result and summary.....	12
2.4. Present the findings and remediation required	13
2.5. Explain the methodologies used	13
3. Code Review	13
3.1. Use any opensource tool and setup code scanning automation	13
3.2. Perform the code scanning on sample source codes (With 5 common type of vulnerabilities)	13
3.3. Prepare the source code scanning result and summary.....	13
3.4. Present the findings and remediation required	13
3.5. Explain the methodologies used	13

Objective

Prepare the testing environment and choose 2 of the following assignments below that best match to your past experiences for one of the following Github repositories ->

- <https://github.com/scalessec/Toast-Swift>
- <https://github.com/joetworkflow/jw-community>

1. Security Scanning

- Use any open source tool and setup scanning
- Perform the security scanning on a test machine (with 5 security loopholes)
- Prepare the scanning result and summary
- Ppresent the findings and remediation required
- Explain the methodologies used

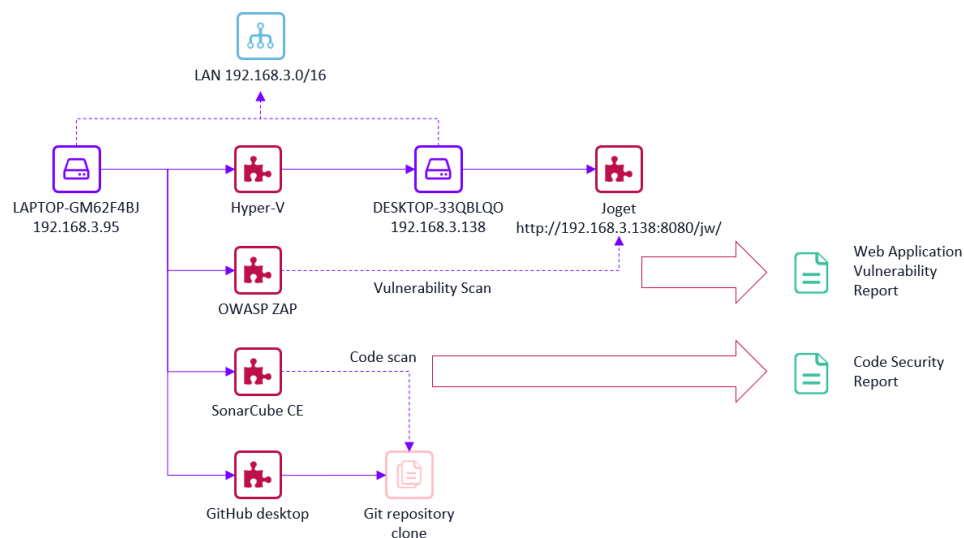
2. Code Review

- Use any open source tool and setup code scanning automation
- Perform the code scanning on sample source codes (With 5 common type of vulnerabilities)
- Prepare the source code scanning result and summary
- Ppresent the findings and remediation required
- Explain the methodologies used

Note: - Once completed, please upload your test scripts & test results to GitHub and share the link.

1. Test setup

The high-level diagram of test setup in virtualized environment was created as below:



Note



In case of production setup, the network should be isolated, OWASP ZAP and Joget should be placed in the isolated network segment to avoid interference with other sites and possible test attack leak from the local network.

1.2. Host hypervisor

Hyper-V host running under Windows 11 Pro with 40 Gb of RAM and 2 Tb of SSD fully satisfies the requirements to build a test environment. Laptop name: **LAPTOP-GM62F4BJ**.

Device specifications	Windows specifications
Device nameLAPTOP-GM62F4BJ	EditionWindows 11 Pro
Processor11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz	Version21H2
Installed RAM40,0 GB (39,8 GB usable)	Installed on27.02.2022
Device IDFEEBFCAB-5806-43A4-BC29-1EF2E94EBC9C	OS build22000.1098
Product ID00355-60677-16355-AAOEM	Serial numberMP25CVDJ
System type64-bit operating system, x64-based processor	ExperienceWindows Feature Experience Pack 1000.22000.1098.0
Pen and touchNo pen or touch input is available for this display	

Host is on the local network with net address 192.168.3.0/16, IP address of laptop is **192.168.3.95**.

Ethernet adapter vEthernet (LAN):

```
Connection-specific DNS Suffix . : 
Link-local IPv6 Address . . . . . : fe80::68dc:f8bb:b6a4:3ad7%23
IPv4 Address. . . . . : 192.168.3.95
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.3.1
```

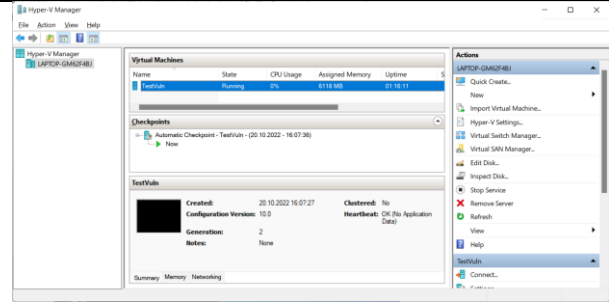
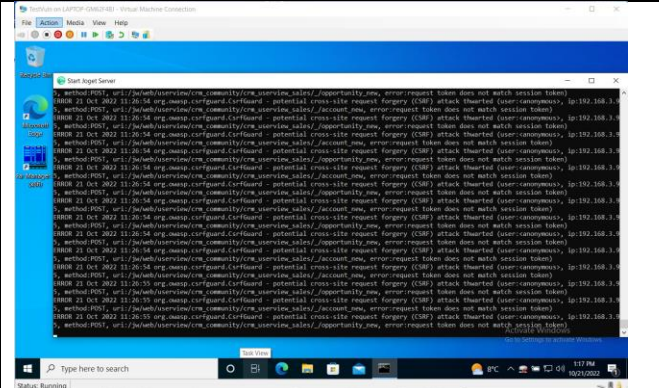
Note



Since virtual machine is connected to virtual Hyper-V switch, no traffic goes out of the host laptop when talking to Joget, so no degradation of speed due to requests going via physical network.

1.3. Virtual machine

Windows 10 desktop (not activated) available for temporary testing purposes installed under **Hyper-V** supervisor on my laptop, virtual machine name: **TestVuln**, windows machine name **DESKTOP-33QBLQO**

	
Device specifications	
Device nameDESKTOP-33QBLQO	
Processor11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz	
Installed RAM8,00 GB (5,51 GB usable)	
Device ID4189006A-660B-4FB5-9F52-3A6F81B1AF12	
Product ID00329-00000-00003-AA658	
System type64-bit operating system, x64-based processor	
Pen and touchNo pen or touch input is available for this display	

Same local network shared 192.168.3.0/16, host address for test virtual machine is **192.168.3.138**

```

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::787b:d43a:e5bf:a2d0%3
    IPv4 Address. . . . . : 192.168.3.138
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.3.1
  
```

1.4. Application setup

Joget community edition was selected: <https://github.com/jogetworkflow/jw-community>

Joget

Joget is a next generation open source no-code / low-code application platform for faster, simpler digital transformation (DX).

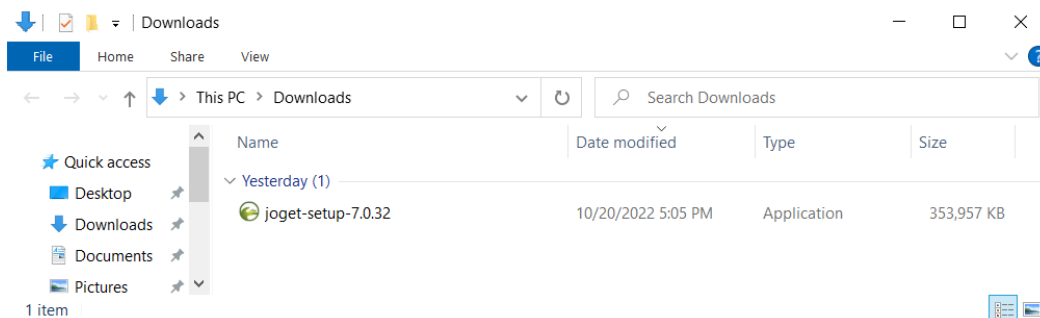
Joget combines the best of rapid application development, business process automation and workflow management in a simple, flexible and open platform. Business and technical teams can collaborate to rapidly build full-fledged enterprise applications visually, anywhere, anytime.

- *Web-based visual approach empowers non-coders to build and maintain apps anytime, anywhere*
- *Reduces time to market, from months to weeks or days*
- *Apps built are mobile ready, cloud ready*
- *APIs for integration and plugin architecture for extensibility*
- *"App Store" for enterprise apps – Joget Marketplace*

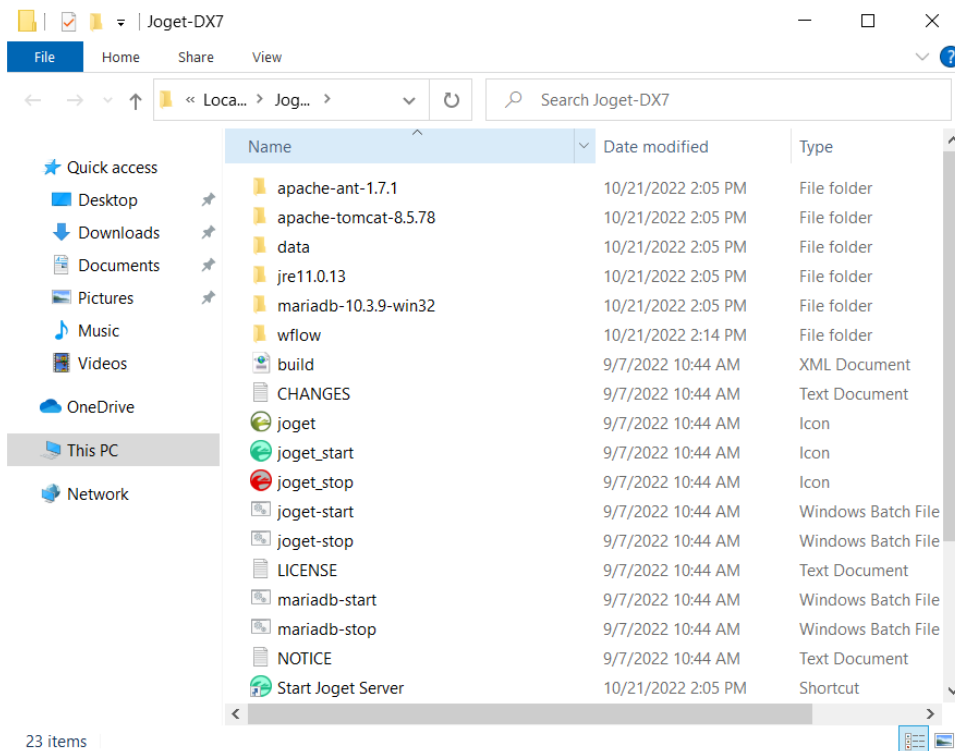
Joget uses Java as backend and JavaScript as frontend technology, with several well-known libraries used (Spring Framework, Hibernate OR, BeanShell scripts etc.)

Joget Community Edition is available from Joget web site / Download:

<https://www.joget.org/download/>



Joget is installed in default folder (c:\Joget-DX7)

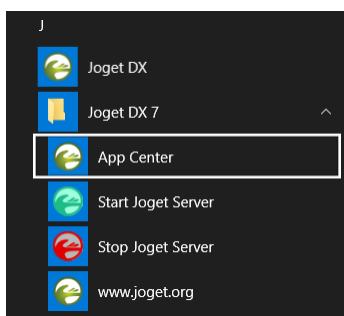


Joget installation includes:

- Apache ANT 1.7.1 (<https://ant.apache.org/>)
- Apache Tomcat 8.5.78 (<https://tomcat.apache.org/>)
- Java Runtime Environment 11.0.13
(<https://www.oracle.com/java/technologies/javase/jdk11-archive-downloads.html>)
- MariaDB 10.3.9 (<https://mariadb.com/>)

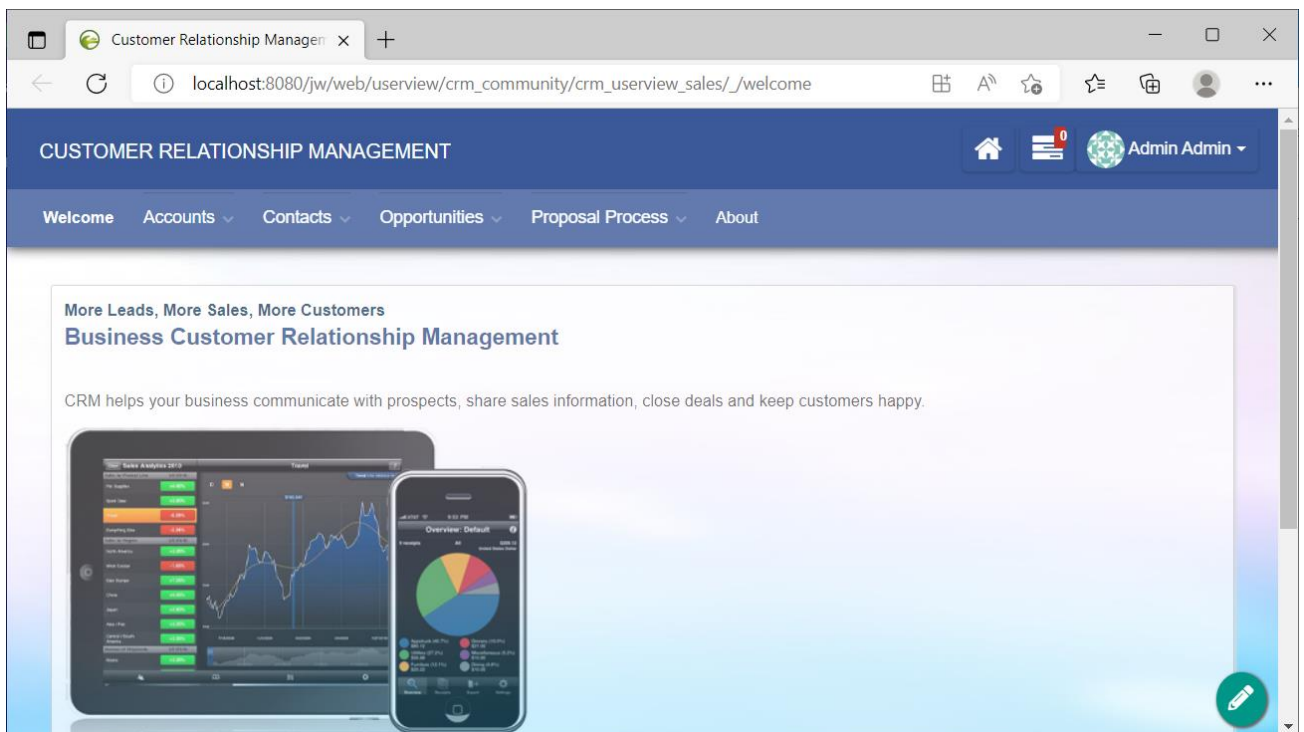
Web application is located in Apache Tomcat folder: C:\Joget-DX7\apache-tomcat-8.5.78\webapps\jw\

Joget has to be started manually on virtual machine via “Start Joget Server” link in the start menu or C:\Joget-DX7\joget-start.bat batch file:

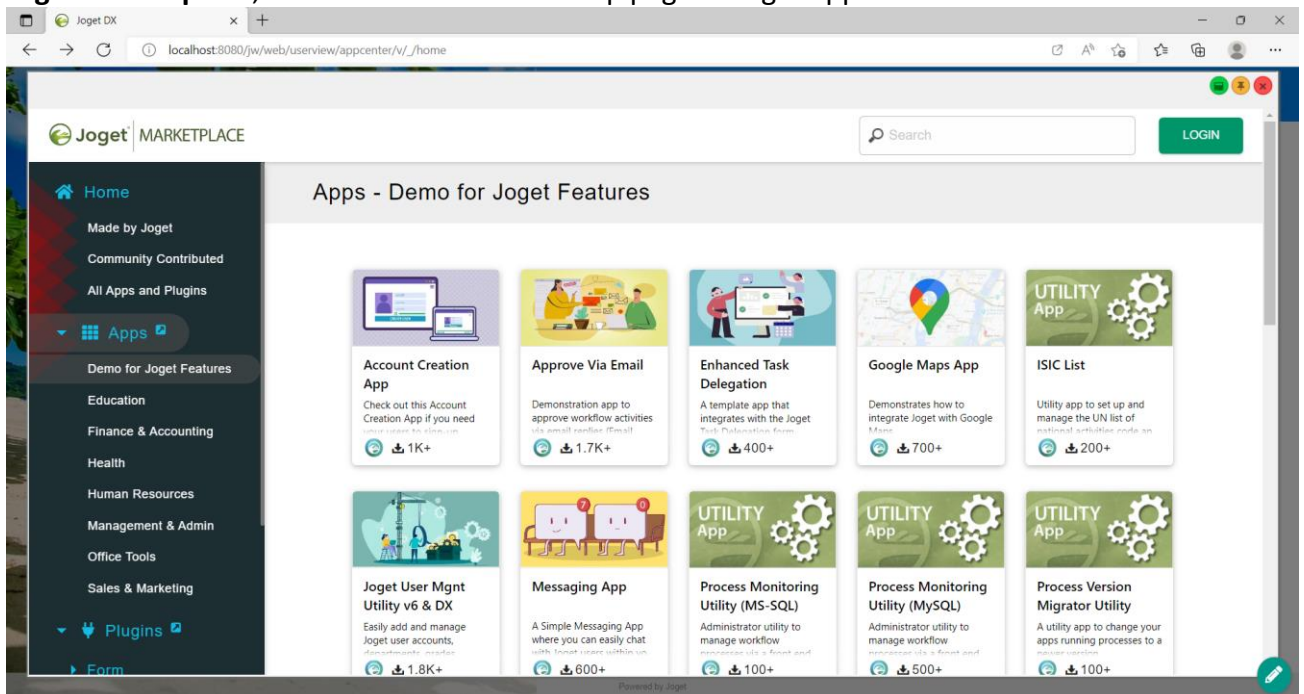


Application will be available from local host as <http://localhost:8080/jw/> or <http://192.168.3.138:8080/jw/> remotely. Windows Defender firewall should be set up respectively to allow web browser to connect to application remotely.

There is one default application built on Joget – CRM Community, which is used as an example of application which can be developed using Joget concept of low code/no code.



Other sample, community or commercial production applications can be downloaded from **Joget Marketplace**, available via link on startup page of Joget application:



I did not use the additional applications, to avoid excessive non-target investigation. The objective is to find vulnerabilities in the **Joget** code / application.

1.5. Vulnerability Scanner setup

The greatest list of vulnerability scanners is located on OWASP site:

[https://owasp.org/www-community/Vulnerability Scanning Tools](https://owasp.org/www-community/Vulnerability_Scanning_Tools)

The objective of selection is to perform vulnerability scanning fast, with readable results and preferably with lowest level of education in the product.

So, the main criteria are to select the

- 1) Windows-based
- 2) GUI application
- 3) Up-to-date.

The final list is as below:

Name/Link	Platforms	Note	Result
GoLismero	Windows, Linux and Macintosh	GPLv2.0	Deprecated
Grendel-Scan	Windows, Linux and Macintosh		Version: 2012 year
Nuclei	Windows, Unix/Linux, and Macintosh	Fast and customisable vulnerability scanner based on simple YAML based DSL.	Only CLI, no GUI
Ride (REST JSON Payload fuzzer)	Linux / Mac / Windows	Apache 2	Last version: 2019 year
Vega	Windows, Linux and Macintosh		Last version: 2014 year
Wapiti	Windows, Unix/Linux and Macintosh		Only CLI, no GUI
Zed Attack Proxy	Windows, Unix/Linux, and Macintosh	Apache-2.0	Last version: 2021 GUI, powerful reporting etc.

And the winner is **OWASP ZAP (Zed Attack Proxy)**.

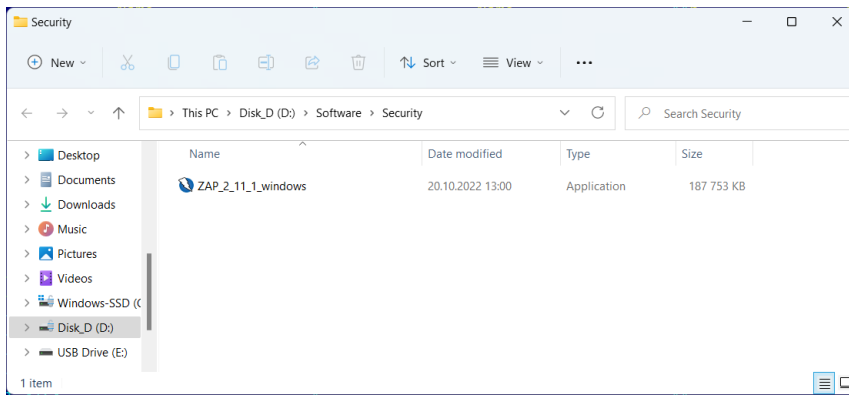
OWASP ZAP (<https://www.zaproxy.org/>) was selected due to its powerful ability to check vulnerabilities and HUP mode allowing it to investigate the issues. OWASP Zed Attack Proxy is an OWASP flagship project, so contains most up-to-date methods of vulnerability detection.

OWASP ZAP 2.11.1 downloaded from <https://www.zaproxy.org/download/> and installed by default on host laptop. Windows 64-bit version is installed.

Note



It is better to use the separate virtual host in isolated network to perform scans. See the note above.



1.6. Code Scanner setup

The greatest list of code security scanners is located on OWASP site:

[https://owasp.org/www-community/Source Code Analysis Tools](https://owasp.org/www-community/Source_Code_Analysis_Tools)

The objective of selection is to perform code security scanning fast, with readable results and preferably with lowest level of education in the product.

So, the main criteria are to select the

- 4) Windows-based
- 5) GUI application
- 6) Up-to-date.

The final list is as below:

Name/Link	Note	Result
Agnitio	ASP, ASP.NET, C#, Java, Javascript, Perl, PHP, Python, Ruby, VB.NET, XML	Version: 2011 year
MobSF	Android Java, Objective C, Swift	No GUI
OWASP ASST (Automated Software Security Toolkit)	An Open Source, Source Code Scanning Tool, developed with JavaScript (Node.js framework), Scans for PHP & MySQL Security Vulnerabilities According to OWASP Top 10 and Some other OWASP's famous vulnerabilities, and it teaches developers of how to secure their codes after scan.	No GUI
SonarQube	Scans source code for 15 languages for Bugs, Vulnerabilities, and Code Smells. SonarQube IDE plugins for Eclipse, Visual Studio, and IntelliJ provided by [SonarLint](https://www.sonarlint.org/).	Version: 2022 year, excellent GUI and reporting
Spectral	Discover, classify, and protect your codebases, logs, and other assets. Monitor and detect API keys, tokens, credentials, high-risk security misconfiguration and more.	Not opensource
VisualCodeGrep	C/C++, C#, VB, PHP, Java, PL/SQL	No JavaScript

And the winner is SonarCube Security Analysis community edition:

<https://www.sonarqube.org/features/security/>

SonarCube is the leading product for Code Quality and Security. Community Edition has the following features:

- Static code analysis for 17 languages
- Java, C#, JavaScript, TypeScript, CloudFormation, Terraform, Kotlin, Ruby, Go, Scala, Flex, Python, PHP, HTML, CSS, XML and VB.NET
- Detect Bugs & Vulnerabilities
- Review Security Hotspots
- Track Code Smells & fix your Technical Debt
- Code Quality Metrics & History
- CI/CD integration
- Extensible, with 50+ community plugins

To enable scanning of Java and Javascript the following additional set up should be made in C:\sonarqube\conf\sonar-scanner.properties file:

1. Oracle Java SDK 19.0.1 to be installed (installation folder: C:/Program Files/Java/jdk-19/) – **sonar.java.jdkHome**
2. Java libraries to be identified and mentioned in the config – **sonar.java.libraries**
3. Java binaries to be identified (WAR file to be taken from production environment if not found from the in the repository) and mentioned in the config – **sonar.java.binaries**
4. Node.js to be installed (<https://nodejs.org/en/>) and the executable to be mentioned in the config – **sonar.nodejs.executable**

sonar-scanner.properties

```
sonar.java.jdkHome=C:/Program Files/Java/jdk-19/
sonar.java.libraries=C:/Users/Mark.LAPTOP-GM62F4BJ/Documents/GitHub/jw-community/wflow-designer/lib/,C:/Users/Mark.LAPTOP-GM62F4BJ/Documents/GitHub/jw-community/wflow-install/src/main/resources/wflow-home/
sonar.java.binaries=C:/Users/Mark.LAPTOP-GM62F4BJ/Documents/GitHub/jw-community/war/
sonar.nodejs.executable=C:/Program Files/nodejs/node.exe
```

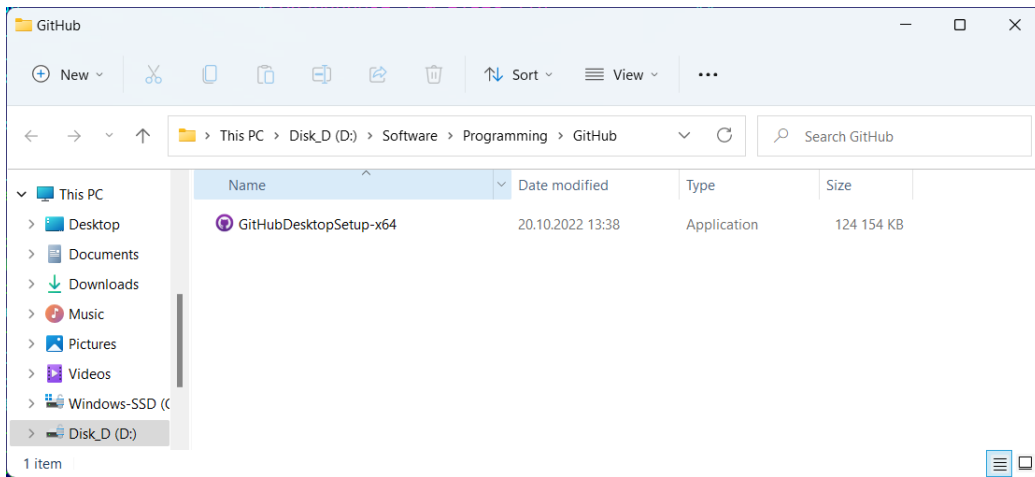
For detailed reporting also CNS Report Plugin should be installed:

<https://github.com/cnescatlab/sonar-cnes-report>. To install, the latest release from <https://github.com/cnescatlab/sonar-cnes-report/releases> to be uploaded to C:\sonarqube\extensions\plugins\ and SonarQube to be restarted.

1.7. GitHub setup

To perform local code security scanning and to publish results of the scanning, GitHub desktop should be installed on laptop.

Git Hub desktop available on <https://desktop.github.com/>



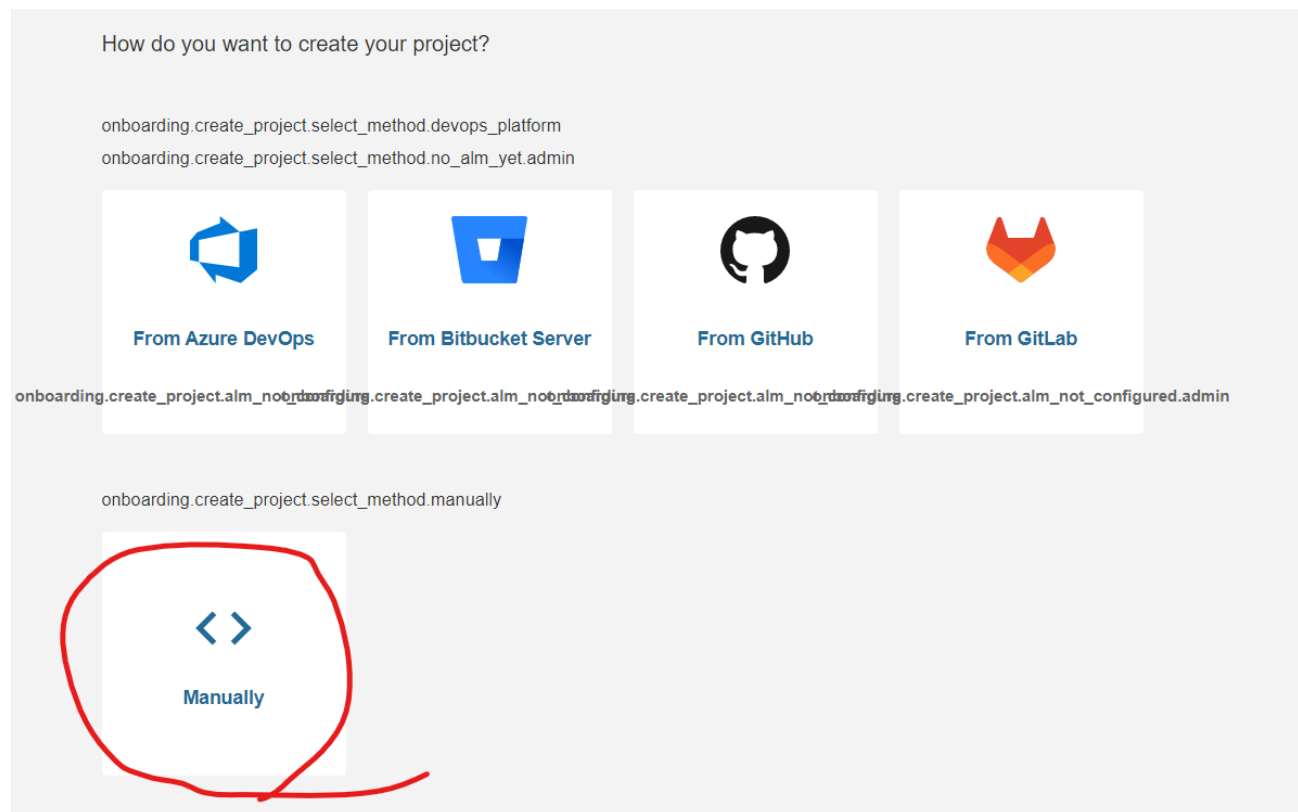
After downloaded, with the GitHub Desktop the following actions to be performed:

1. Install and log in to GitHub
2. Create a clone of jw-community (<https://github.com/jogetworkflow/jw-community>)
3. Create markemelianov\TestVuln project

<https://github.com/markemelianov/TestVuln/> – project which contains all the data related to assignment.

1.8. SonaQube project setup

The project TestVuln created (in manual mode).



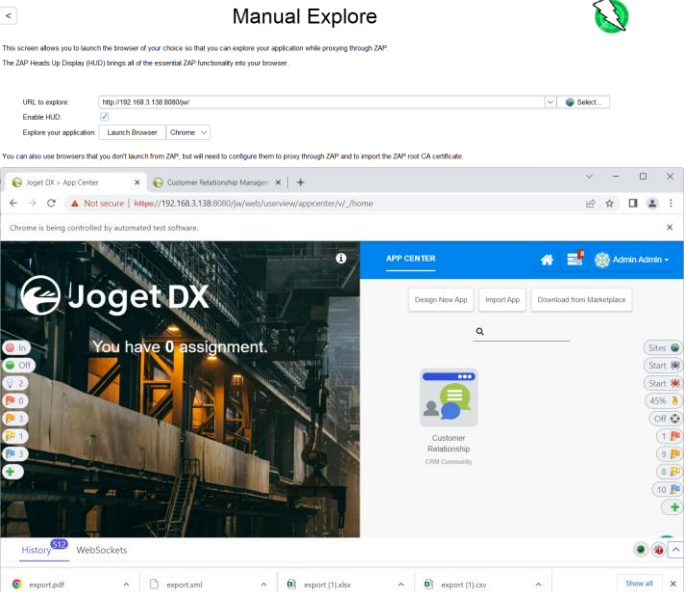
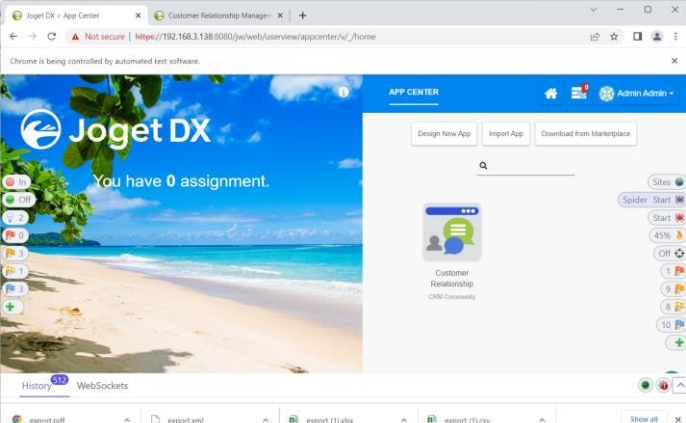
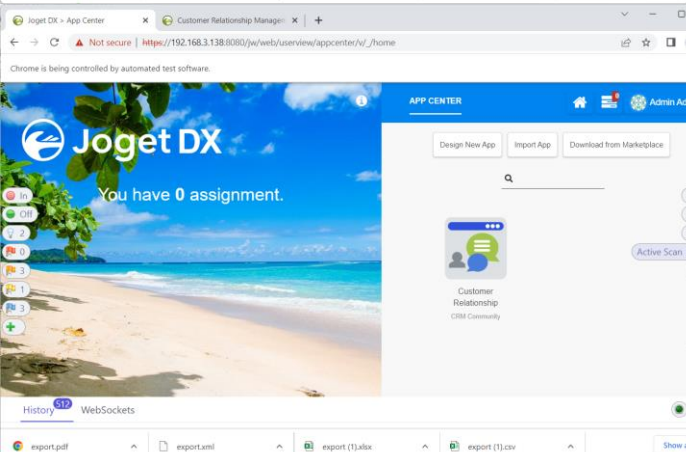
2. Security Scanning

2.1. Use any opensource tool and setup scanning

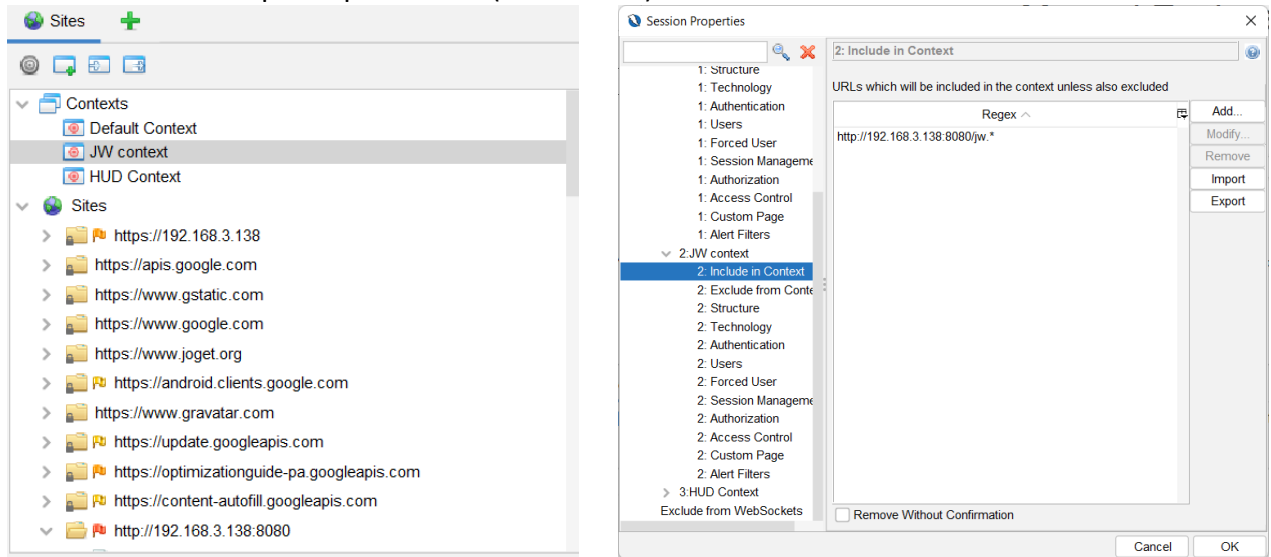
OWASP ZAP (Zed Attack Proxy) is installed and used according above setup.

2.2. Perform the security scanning on a test machine (with 5 security loopholes)

Security scanning was performed using the following methods:

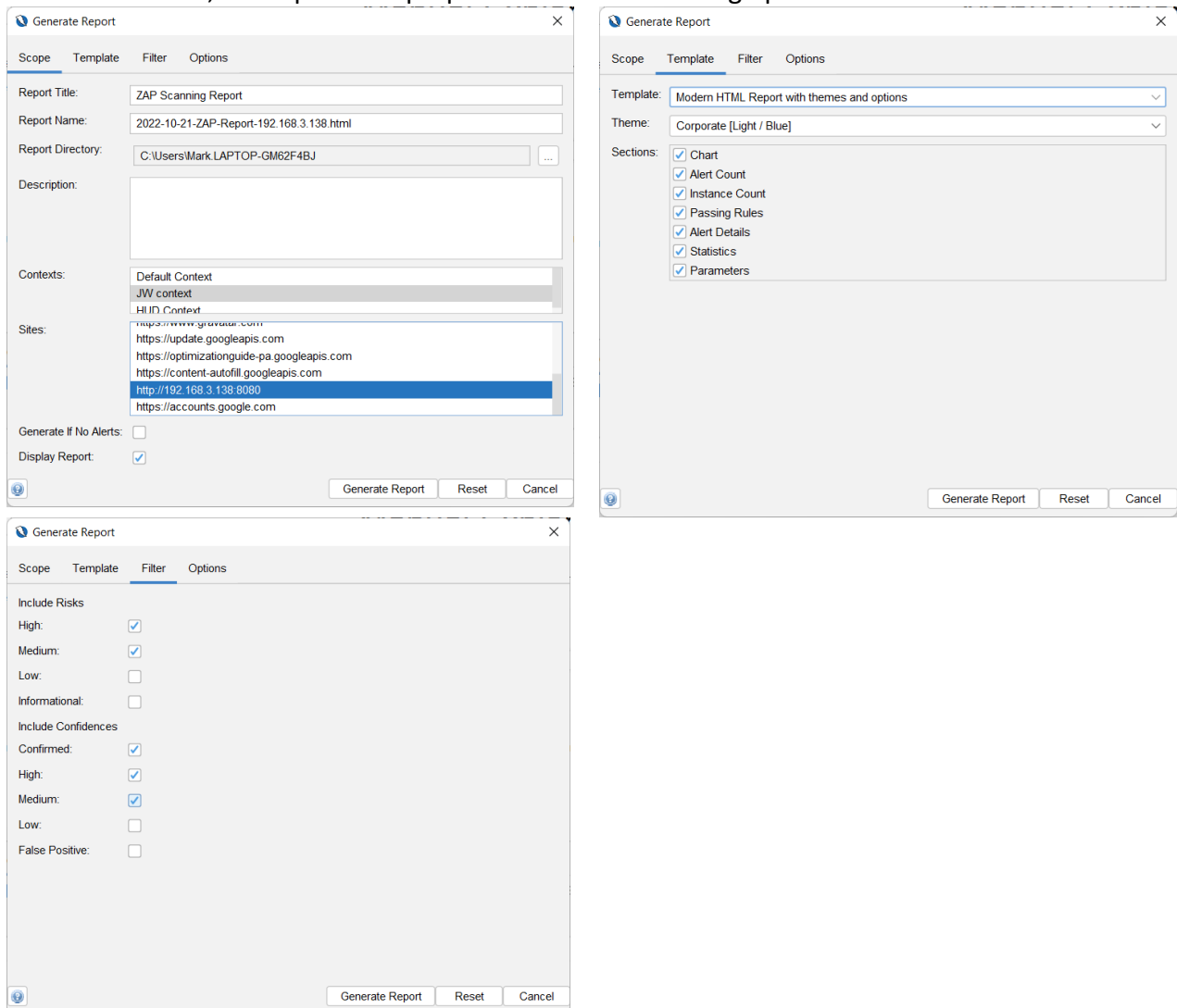
Manual Explore	The method allows enable HUD (Heads Up Display) that provides the functionality of ZAP directly in the browser. In manual explore mode pen tester should walk through all the possible pages of the application and execute all the possible functions of the application.	
Spider	To collect all the structure of the site for the next steps	
Active Scan	To perform the extensive scan for known vulnerabilities	

Context set up was performed (JW context):



2.3. Prepare the scanning result and summary

As a result, the report was prepared with the following options:



As a result, the following report created: **ZAP report\2022-10-21-ZAP-Report-192.168.3.138.html**

2.4. Present the findings and remediation required

Analysis of the report was made and the findings and remediations represented in the final presentation file: **Presentations\Mark_Emelianov_Technical_Assignment.pptx**

2.5. Explain the methodologies used

Methodologies used are explained in the final presentation file:

Presentations\Mark_Emelianov_Technical_Assignment.pptx

3. Code Review

3.1. Use any opensource tool and setup code scanning automation

SonarCube Security Analysis community edition was set up according above procedure.

3.2. Perform the code scanning on sample source codes (With 5 common type of vulnerabilities)

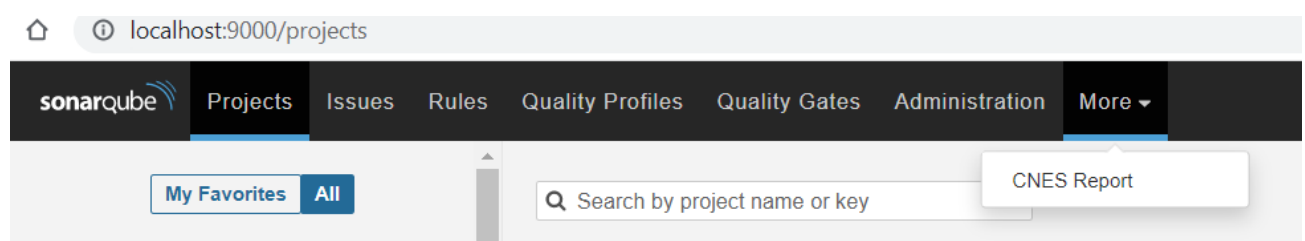
Code scanning to be executed. The command like

```
C:\sonarqube\bin\sonar-scanner.bat -D"sonar.projectKey=TestVuln" -D"sonar.sources=." -D"sonar.host.url=http://localhost:9000" -D"sonar.login=sqp_XXXXXXXXXXXXXXXXX"
```

Executed in the folder where github project **jw-community** clone is located.

3.3. Prepare the source code scanning result and summary

The source code scanning result generated using CNES report: **SonarQube Report\2022-10-22-TestVuln-analysis-report.docx**



3.4. Present the findings and remediation required

Analysis of the report was made and the findings and remediations represented in the final presentation file: **Presentations\Mark_Emelianov_Technical_Assignment.pptx**

3.5. Explain the methodologies used

Methodologies used are explained in the final presentation file:

Presentations\Mark_Emelianov_Technical_Assignment.pptx