Technical Assignment Report

Jogit Community

Mark Emelianov, CRISC

Assignment execution plan and results

Deliverables on GitHub https://github.com/markemelianov/TestVuln/

Select scanning tools

- Vulnerability Scanning Tools ZAP
- Source Code Analysis Tools sonarqube

Assignment Report/Assignment_Report_Detailed.docx
(the protocol and detailed explanation of actions)

SecurityTools/Security_Tools_Analysis.xlsx

Set up the testing environment

<<See this report>>

Perform vulnerability scanning and get the report

ZAP Report

Perform source code analysis and get the report

SonarQube Report

Analyze reports, prepare findings and remediation

<<See this report>>

Explain used methodologies

<<See this report>>

Vulnerability Scanning Tools

OWASP

https://owasp.org/www-community/Vulnerability_Scanning_Tools

88 tools

- ✓ Windows-based
- ✓ GUI application
- ✓ Up-to-date

7 candidates

1 winner

Name/Link	Result
GoLismero	Deprecated
Grendel-Scan	Last version: 2012
Nuclei	Only CLI, no GUI
Ride (REST JSON Payload fuzzer)	Last version: 2019
Vega	Last version: 2014
Wapiti	Only CLI, no GUI
Zed Attack Proxy	Last version: 2021 GUI, powerful reporting etc.

ZAP

OWASP ZAP
(Zed Attack Proxy)

https://www.zaproxy.org/

Source Code Analysis Tools

https://owasp.org/www-community/Source_Code_Analysis_Tools

112
tools

- ✓ Windows-based
- ✓ Supports Java/JS
- ✓ GUI application
- ✓ Up-to-date

6
candidates

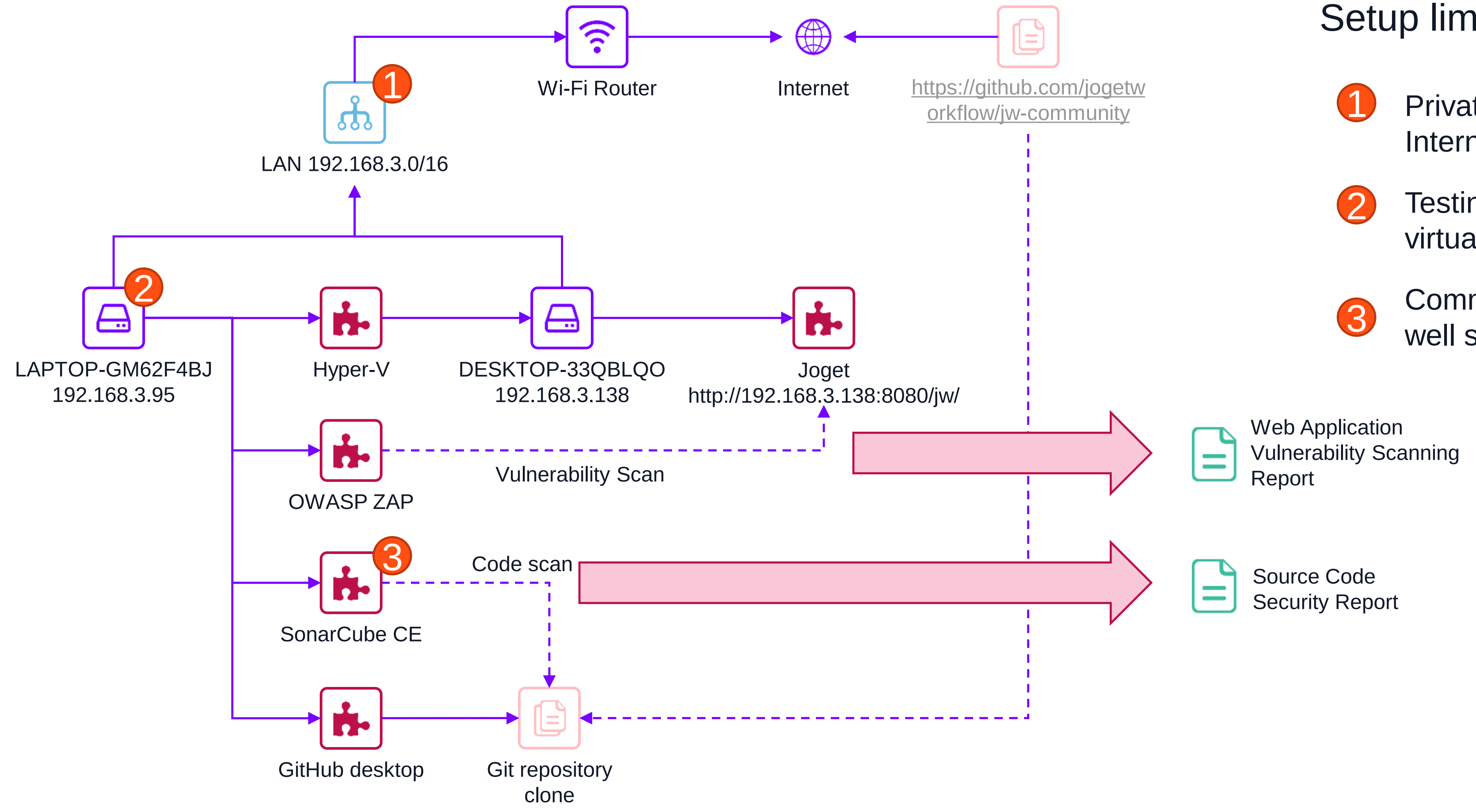
1
winner

Name/Link	Result
Agnitio	Version: 2011 year
MobSF	No GUI
OWASP ASST (Automated Software Security Toolkit)	No GUI
SonarQube	Version: 2022 year, excellent GUI and reporting
Spectral	Not opensource
VisualCodeGrepper	No JavaScript

SonarCube

https://www.sonarqube.org/

Testing Environment Setup



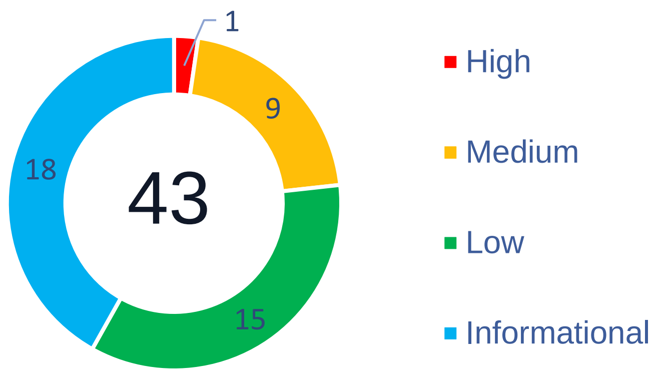
LAN 192.168.3.0/16
1
Wi-Fi Router
Internet
https://github.com/jogetworkflow/jw-community
2
LAPTOP-GM62F4BJ 192.168.3.95
Hyper-V
DESKTOP-33QBLQO 192.168.3.138
Joget http://192.168.3.138:8080/jw/
OWASP ZAP
Vulnerability Scan
Web Application Vulnerability Scanning Report
3
SonarCube CE
Code scan
Source Code Security Report
GitHub desktop
Git repository clone

Setup limitations

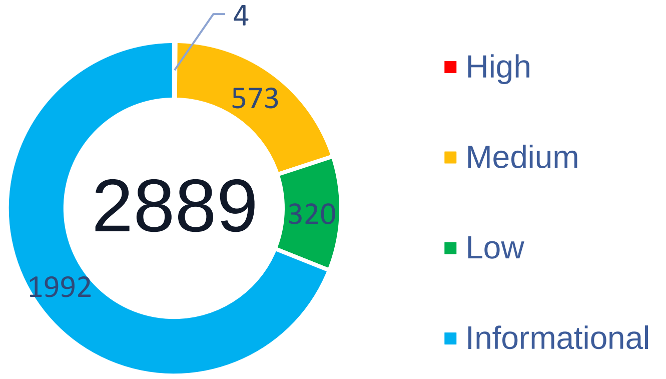
1. Private testing network not isolated from Internet
2. Testing machine with scanning tools is not virtualized
3. Community edition in use, reporting is not well supported

Vulnerability scanning summary

Number of vulnerabilities



Number of Instances



Vulnerabilities in details

Risk Level	Vulnerability	Number of Alerts	Number of instances
High	CORS Misconfiguration	1	4
Medium	Absence of Anti-CSRF Tokens	9	21
	Anti-CSRF Tokens Check		18
	Backup File Disclosure		326
	Bypassing 403		5
	Content Security Policy (CSP) Header Not Set		28
	Insecure HTTP Method - PATCH		66
	Insecure HTTP Method - PUT		66
	Missing Anti-clickjacking Header		25
	Web Cache Deception		18
Low	Cookie Slack Detector	15	127
	Cookie without SameSite Attribute		3
	Dangerous JS Functions		4
	Permissions Policy Header Not Set		47
	X-Content-Type-Options Header Missing		49
Informational	<<All Informational>>	18	1992

Top 5 vulnerabilities

- ✓ CORS Misconfiguration
- ✓ Backup File Disclosure
- ✓ Insecure HTTP Method - PATCH
- ✓ Insecure HTTP Method - PUT
- ✓ Content Security Policy (CSP) Header Not Set

Vulnerability scanning - Methodologies

ZAP

Automated scan	This option allows you to launch an automated scan against an application just by entering the URL. If you are new to ZAP, it is best to start with Automated Scan mode.
Manual scan	This functionality is very useful when your web application needs a login or contains things like registration forms, etc.
Passive scanning	Passive scans only scan the web application responses without altering them. It does not attack or insert malicious scripts to the web application, so this is a safe scan; you can use it if you are new to security testing. Passive scanning is good at finding some vulnerabilities and as a way to get a feel for the basic security of a web application.
Active scanning	Active scan attacks the web application using known techniques to find vulnerabilities. This is a real attack that attempts to modify data and insert malicious scripts in the web application. Active scans put the application at risk, so do not use active scanning against web applications you do not have permission to test.

Top 5 vulnerabilities – 1/3

Risk Level	Alert	Number of instances
High	CORS Misconfiguration	4

Description

This CORS misconfiguration could allow an attacker to perform AJAX queries to the vulnerable website from a malicious page loaded by the victim's user agent.

In order to perform authenticated AJAX queries, the server must specify the header "Access-Control-Allow-Credentials: true" and the "Access-Control-Allow-Origin" header must be set to null or the malicious page's domain. Even if this misconfiguration doesn't allow authenticated AJAX requests, unauthenticated sensitive content can still be accessed (e.g intranet websites).

A malicious page can belong to a malicious website but also a trusted website with flaws (e.g XSS, support of HTTP without TLS allowing code injection through MITM, etc).

Solution

If a web resource contains sensitive information, the origin should be properly specified in the Access-Control-Allow-Origin header. Only trusted websites needing this resource should be specified in this header, with the most secured protocol supported.

Risk Level	Alert	Number of instances
Medium	Backup File Disclosure	326

Description

A backup of the file was disclosed by the web server

Solution

Do not edit files in-situ on the web server, and ensure that un-necessary files (including hidden files) are removed from the web server.

Top 5 vulnerabilities – 2/3

Risk Level	Alert	Number of instances
Medium	Insecure HTTP Method - PATCH	66

Description

This method is now most commonly used in REST services, PATCH is used for **modify** capabilities. The PATCH request only needs to contain the changes to the resource, not the complete resource.

Solution

Ensure that only the required headers are allowed, and that the allowed headers are properly configured.
Ensure that no workarounds are implemented to bypass security measures implemented by user-agents, frameworks, or web servers.

Risk Level	Alert	Number of instances
Medium	Insecure HTTP Method - PATCH	66

Description

This method was originally intended for file managemant operations. It is now most commonly used in REST services, PUT is most-often utilized for **update** capabilities, PUT-ing to a known resource URI with the request body containing the newly-updated representation of the original resource.

Solution

Ensure that only the required headers are allowed, and that the allowed headers are properly configured.
Ensure that no workarounds are implemented to bypass security measures implemented by user-agents, frameworks, or web servers.

Top 5 vulnerabilities – 3/3

Risk Level	Alert	Number of instances
Medium	Content Security Policy (CSP) Header Not Set	28

Description

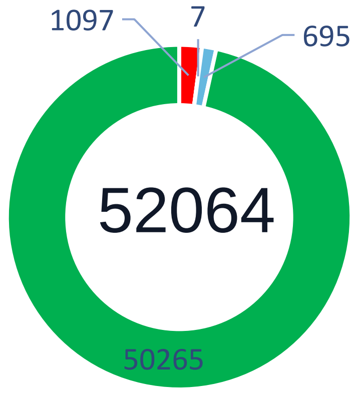
Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.

Solution

Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header, to achieve optimal browser support: "Content-Security-Policy" for Chrome 25+, Firefox 23+ and Safari 7+, "X-Content-Security-Policy" for Firefox 4.0+ and Internet Explorer 10+, and "X-WebKit-CSP" for Chrome 14+ and Safari 6+.

Source Code scanning summary

Number of issues by type

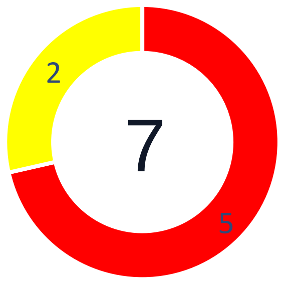


- 🟥 Bug
- 🟨 Vulnerability *
- 🟦 Security Hotspots
- 🟩 Code smell

Vulnerabilities and security hotspots in details

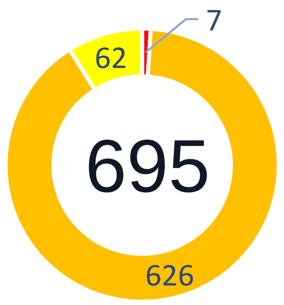
Type	Severity	Vulnerability	Number of issues	Number of instances
Vulnerability	Blocker	Disable access to external entities in XML parsing.	5	5
	Critical	Verify the origin of the received message.	2	1
		Specify a target origin for this message.		1
Hotspot	High	Authentication	7	5
		SQL injection		2
	Medium	Code injection (RCE)	626	35
		Denial of service (DoS)		574
		Weak cryptography		17
	Low	Encryption of Sensitive Data	62	3
		Insecure configuration		38
		Others		21

Vulnerabilities by severity



- 🟥 Blocker
- 🟨 Critical

Hotspots by severity



- 🟥 High
- 🟧 Medium
- 🟨 Low

Top 5 vulnerabilities and security hotspots

- ✓ Disable access to external entities in XML parsing.
- ✓ Verify the origin of the received message.
- ✓ Specify a target origin for this message.
- ✓ Authentication
- ✓ SQL injection

*Vulnerabilities are not shown in the report **2022-10-22-TestVuln-analysis-report.docx** due to bug in CNES reports plugin. Plugin is not compatible with SonarQube Community Edition

Source Code scanning - Methodologies

sonarqube

SAST	Detect security issues in code review with Static Application Security Testing (SAST)
Hotspots	Security Hotspots are uses of security-sensitive code. They might be okay, but human review is required to know for sure. As developers code and interact with Security Hotspots, they learn to evaluate security risks while learning more about secure coding practices.
Vulnerabilities	Security Vulnerabilities require immediate action. SonarQube provides detailed issue descriptions and code highlights that explain why your code is at risk. Just follow the guidance, check in a fix and secure your application.

Top 5 vulnerabilities – 1/2

Severity	Alert	Number of instances
Blocker	Disable access to external entities in XML parsing.	5

Description

XML standard allows the use of entities, declared in the DOCTYPE of the document, which can be internal or external.

When parsing the XML file, the content of the external entities is retrieved from an external storage such as the file system or network, which may lead, if no restrictions are put in place, to arbitrary file disclosures or server-side request forgery (SSRF) vulnerabilities.

Solution

It's recommended to limit resolution of external entities by using one of these solutions:
If DOCTYPE is not necessary, completely disable all DOCTYPE declarations.
If external entities are not necessary, completely disable their declarations.
If external entities are necessary then:
Use XML processor features, if available, to authorize only required protocols (eg: https).
And use an entity resolver (and optionally an XML Catalog) to resolve only trusted entities.

Risk Level	Alert	Number of instances
Critical	Verify the origin of the received message.	1

Description

Browsers allow message exchanges between Window objects of different origins.
Because any window can send or receive messages from another window, it is important to verify the sender's/receiver's identity.

Solution

When receiving a message with a message event, the sender's identity should be verified using the origin and possibly source properties.

Risk Level	Alert	Number of instances
Critical	Specify a target origin for this message.	1

Description

Browsers allow message exchanges between Window objects of different origins.
Because any window can send or receive messages from another window, it is important to verify the sender's/receiver's identity.

Solution

When sending a message with the postMessage method, the identity's receiver should be defined (the wildcard keyword (*) should not be used).

Top 5 vulnerabilities – 2/2

Risk Level	Alert	Number of instances
High	Authentication	5

Description

Because it is easy to extract strings from an application source code or binary, passwords should not be hard-coded. This is particularly true for applications that are distributed or that are open-source.
In the past, it has led to the following vulnerabilities:
CVE-2019-13466
CVE-2018-15389
Passwords should be stored outside of the code in a configuration file, a database, or a password management service.
This rule flags instances of hard-coded passwords used in database and LDAP connections. It looks for hard-coded passwords in connection strings, and for variable names that match any of the patterns from the provided list.

Solution

Recommended Secure Coding Practices
Store the credentials in a configuration file that is not pushed to the code repository.
Store the credentials in a database.
Use your cloud provider’s service for managing secrets.
If a password has been disclosed through the source code: change it.

Risk Level	Alert	Number of instances
High	SQL injection	2

Description

Formatted SQL queries can be difficult to maintain, debug and can increase the risk of SQL injection when concatenating untrusted values into the query. However, this rule doesn’t detect SQL injections (unlike rule S3649), the goal is only to highlight complex/formatted queries.

Solution

Recommended Secure Coding Practices
Use parameterized queries, prepared statements, or stored procedures and bind variables to SQL query parameters.
Consider using ORM frameworks if there is a need to have an abstract layer to access data.