OWASP TOP 10

Back to basics

REDGUARD
SECURING YOUR ASSETS

February 26th, 2020

whoami

Kevin Gasser

- Graduated as application developer
- Full-time security tester at Redguard since 2018
- Occasional Doughnut Santa around X-Mas

Intro

REDGUARD
SECURING YOUR ASSETS

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.

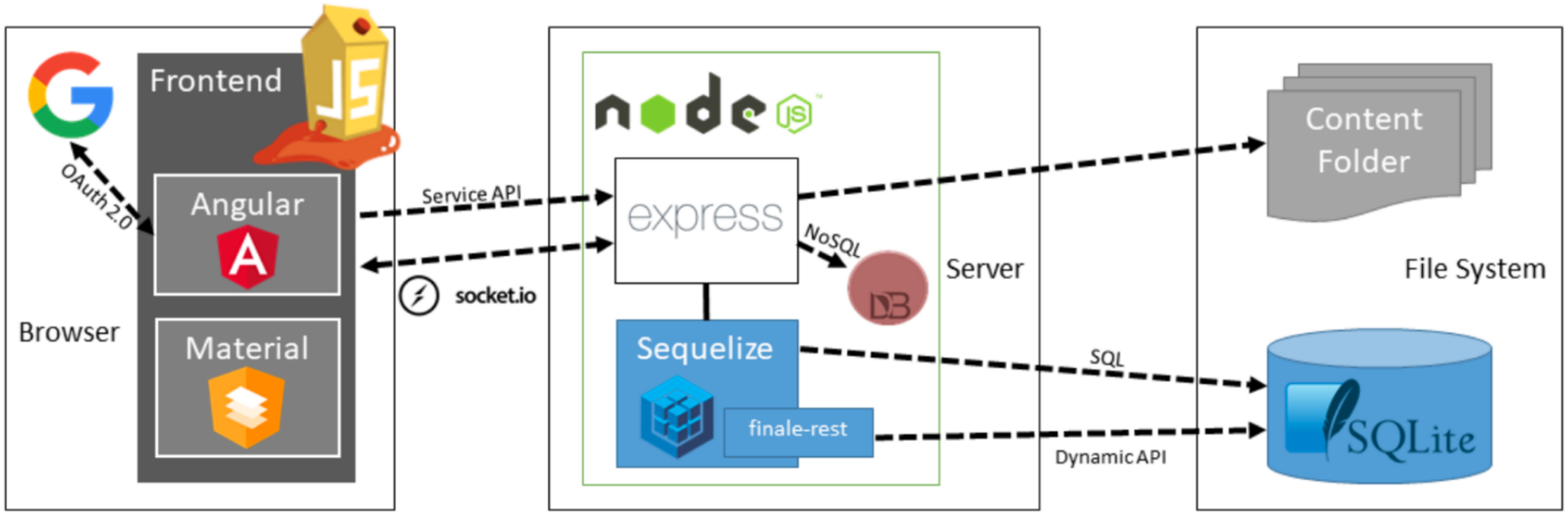
OWASP Top 10 - Data

- Open Survey
- Public Data Call
  - 40+ submissions
  - 23 contributors (used data out of submissions)
  - ~114,000 applications

REDGUARD
SECURING YOUR ASSETS

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	∪	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	∪	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Scenario - OWASP Juice Shop



http://owasp-juice.shop

A1 - Injection

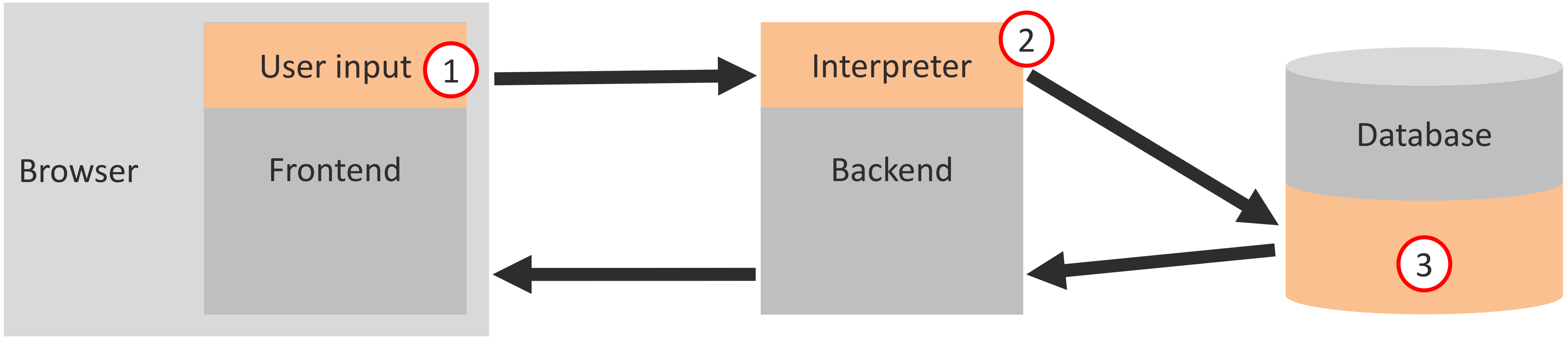
A1 - Injection

- Been number one for a while (~2010?)
- What is it?
  - Input is not properly sanitized and leads to unwanted interpreting of code.

- Examples:
  - SQL
  - NoSQL
  - OS commands
  - LDAP

- Our Example: The classic, SQL injection

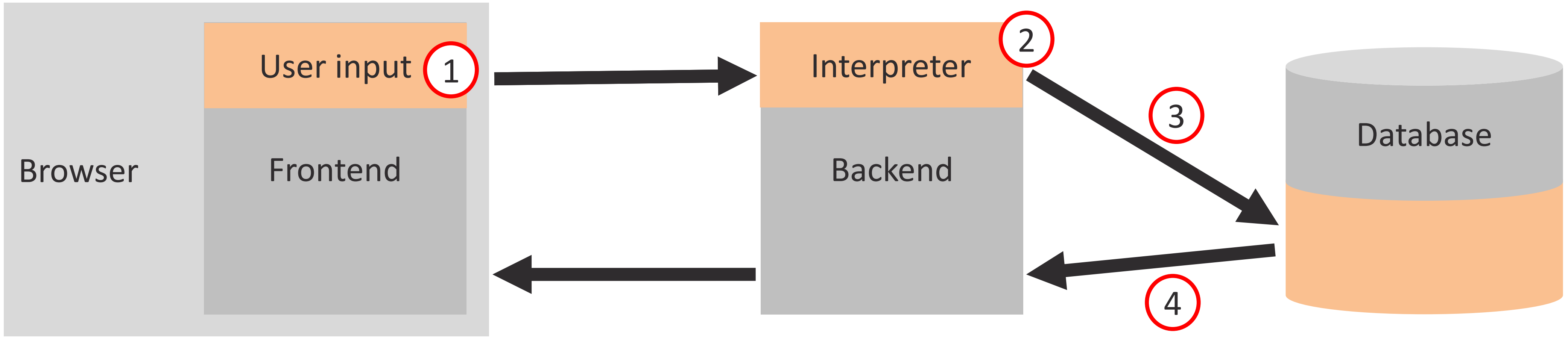
A1 - Scenario: SQL-Injection



What is required?

1. User input
2. Interpreter
3. Something that does stuff (in our case SQL-DB)

A1 - Scenario: SQL-Injection



How does it work?

1. Untrusted data is sent to interpreter
2. Input data is not properly sanitized
3. Commands are executed on database
4. Results are returned to Backend and displayed in Browser

A1 - Scenario: SQL-Injection

Give access if the following statement returns more than 0 results:

```
SELECT * FROM user WHERE nickname='$nickname' AND password='$password';
 nickname and password
```

Fill expected input:

```
$nickname = alice
$password = s3cr3t

SELECT * FROM user WHERE nickname='alice' AND password='s3cr3t';
```

nickname	password
alice	s3cr3t
bob	banana123

A1 - Scenario: SQL-Injection

Give access if the following statement returns more than 0 results:

```
SELECT * FROM user WHERE nickname='$nickname' AND password='$password';
```

Fill unexpected input:

```
$nickname = alice
$password = x' or 1=1; --

SELECT * FROM user WHERE nickname='alice' AND password='x' or 1=1; -- ';
```

nickname	password
alice	s3cr3t
bob	banana123

A1 - Injection

What can be done?

- Don’t trust the client
  - Validate input
  - Escape special characters
- Keep data separate from commands and queries.

- Specifically DB command injection:
  - Query parameterization
  - Principle of least privilege

OWASP_Top_Ten_2017_A1-Injection
OWASP Injection_Prevention_Cheat_Sheet

A2 - Broken Authentication

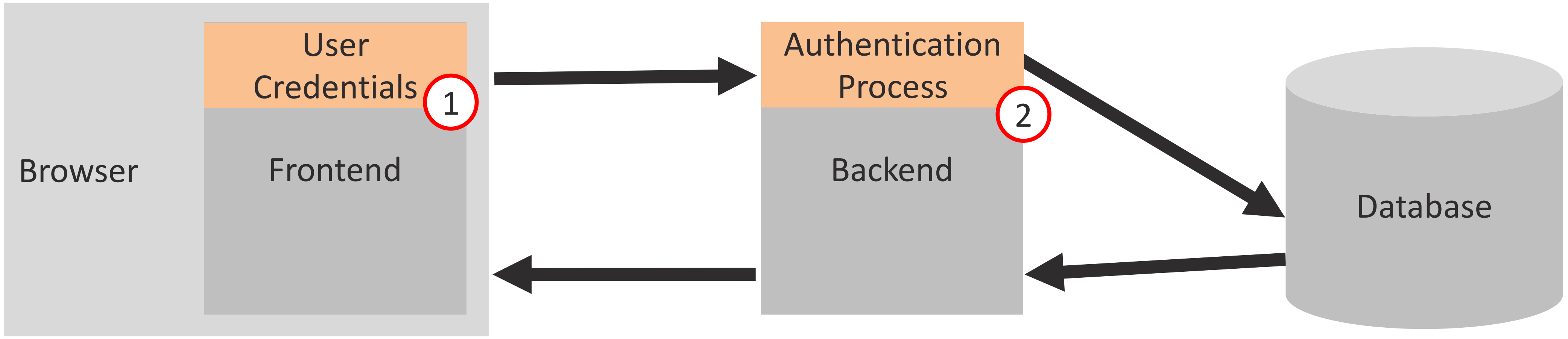
A2 - Broken Authentication

- What is it?
  - Vulnerabilities in authentication (login)
  - Authentication = Making sure that I am who I claim I am

- Examples:
  - Authentication bypass
  - Weak password requirements
  - Single factor authentication
  - Long session timeouts

- Our Example: Dictionary attack on login

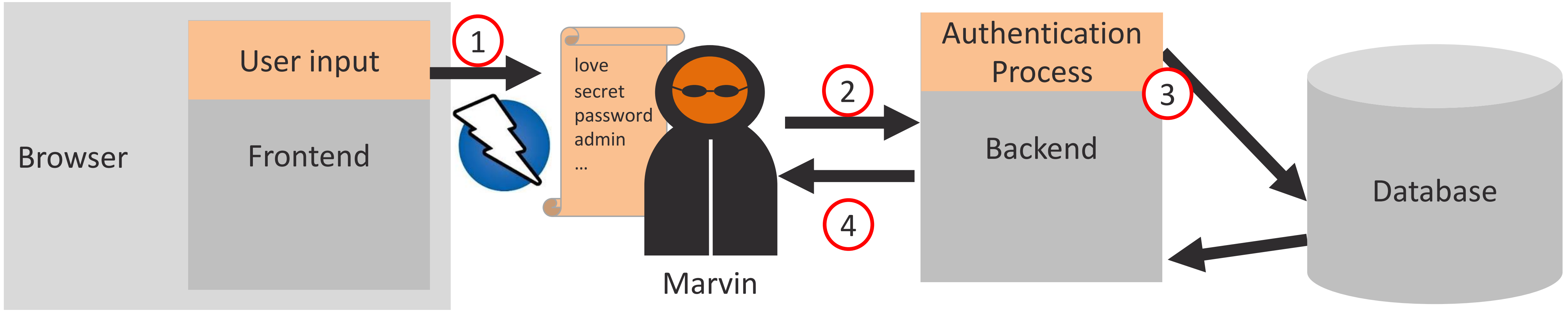
A2 - Broken Authentication



What is required?

1. User Credentials
2. Weak Authentication Process

A2 - Broken Authentication



How does it work?

1. Login attempt is captured
2. Request is filled with password from list
3. Credentials are checked
4. Result is returned

Repeat steps 2 to 4

A2 - Broken Authentication

What can be done?

- Don’t trust the client
- 2FA
- No default users or passwords
- Don’t trust the client (Server side session mgmt)
- Resonable password complexity
- Limit login attempts

OWASP_Top_Ten_2017_A2-Broken_Authentication

OWASP_Authentication_Cheat_Sheet

A3 - Sensitive Data Exposure

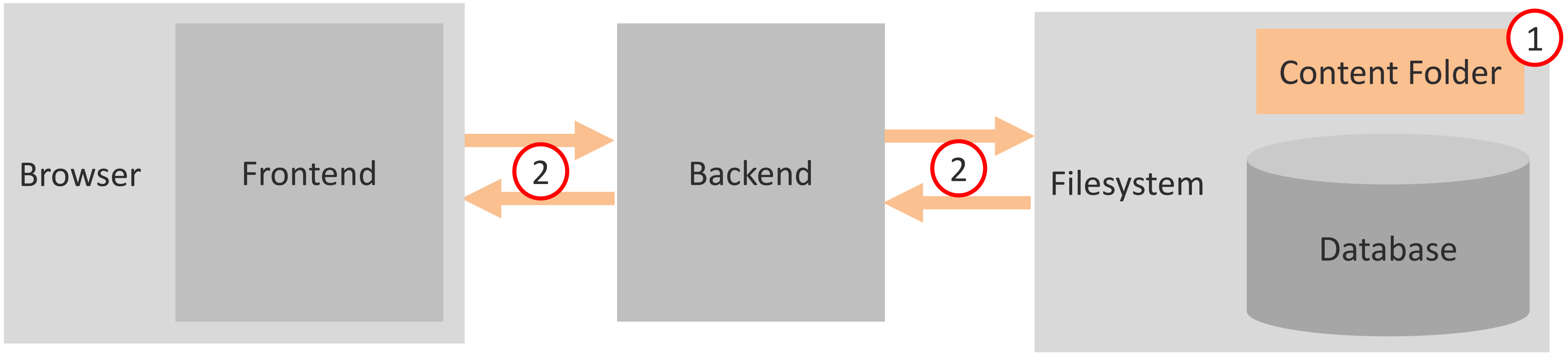
A3 - Sensitive Data Exposure

- What is it?
  - Insufficient protection of sensitive data (e.g. missing encryption)
  - In transit
  - At rest

- Examples:
  - HTTP (no TLS) login interfaces
  - Prod data in other environments
  - Saving passwords in plain text

- Our example: Access confidential documents

A3 - Sensitive Data Exposure

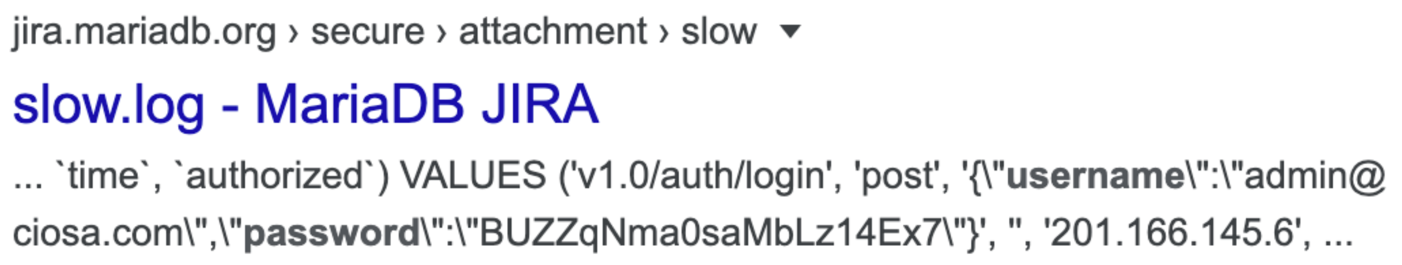


1. Insecure storage of sensitive data
2. Insecure transit of sensitive data

REDGUARD
SECURING YOUR ASSETS

A3 - Sensitive Data Exposure

- `intext:"username=" AND "password=" ext:log`

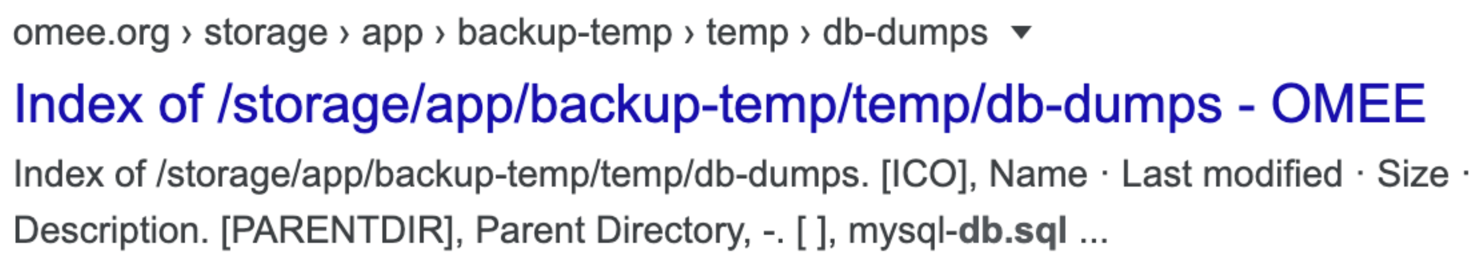


jira.mariadb.org › secure › attachment › slow ▾

slow.log - MariaDB JIRA

... `time`, `authorized`) VALUES ('v1.0/auth/login', 'post', '{\"**username**\":\"admin@ciosa.com\",\"**password**\":\"BUZZqNma0saMbLz14Ex7\"}', '', '201.166.145.6', ...

- `intitle:"index of/" "db.sql”`

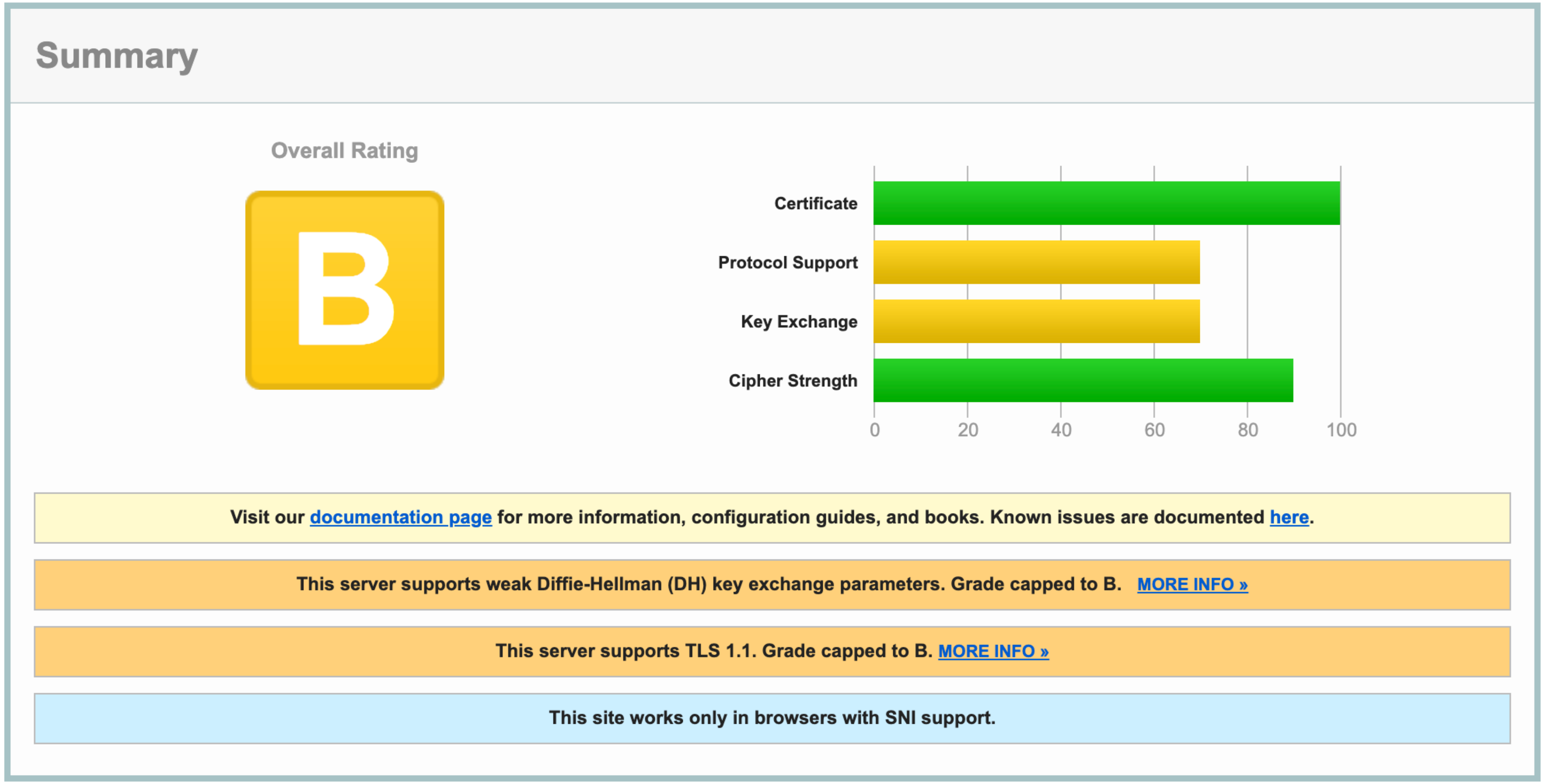


omee.org › storage › app › backup-temp › temp › db-dumps ▾

Index of /storage/app/backup-temp/temp/db-dumps - OMEE

Index of /storage/app/backup-temp/temp/db-dumps. [ICO], Name · Last modified · Size · Description. [PARENTDIR], Parent Directory, -. [ ], mysql-**db.sql** ...

A3 - Sensitive Data Exposure



Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This server supports weak Diffie-Hellman (DH) key exchange parameters. Grade capped to B. MORE INFO »

This server supports TLS 1.1. Grade capped to B. MORE INFO »

This site works only in browsers with SNI support.

https://www.ssllabs.com/ssltest/ or https://testssl.sh/

A3 - Sensitive Data Exposure

- What can be done?
  - Classify data
    - Know what data you have and where
    - Is storage of the data necessary? Now&later
  - Encrypt your data at rest and in transit
    - Use strong ciphers
    - Don’t make your own implementations of encryption. Use known frameworks.

OWASP_Top_Ten_2017_A3-Sensitive_Data_Exposure

OWASP_Password_Storage_Cheat_Sheet

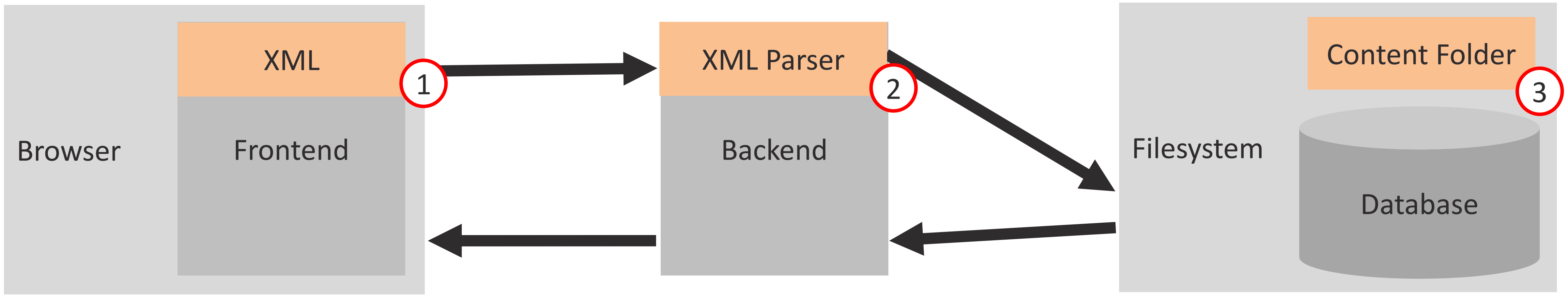
A4 - XML External Entities (XXE)

A4 - XML External Entities (XXE)

- What is it?
  - XML parser processes a a reference to an external entity

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [ <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/shadow" >]>
<creds>
  <user>&xxe;</user>
  <pass>s3cr3t</pass>
</creds>
```

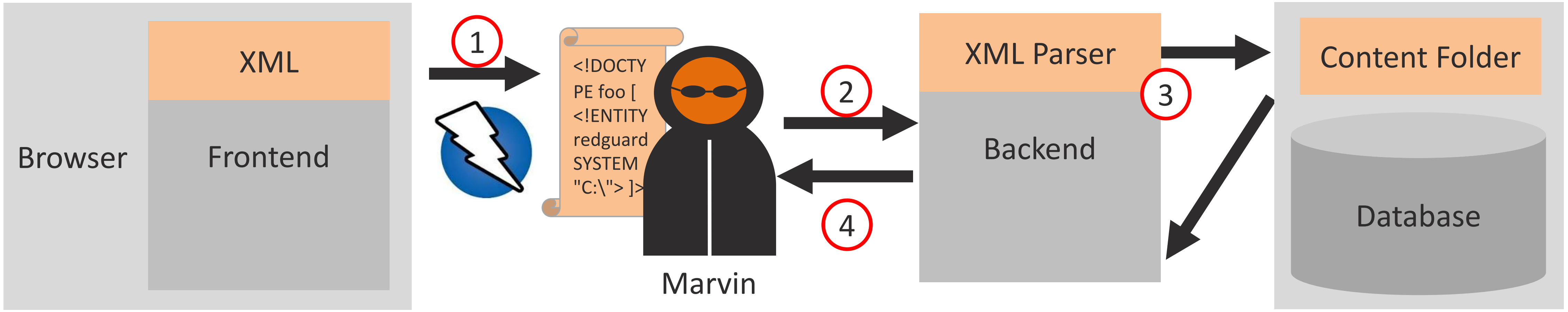
A4 - XML External Entities (XXE)



- What is required?

1. XML POST request
2. XML parser
3. Sensitive content

A4 - XML External Entities (XXE)



How does it work?

1. XML request is captured
2. URI replaced
3. XML is parsed
4. Resource is returned

A4 - XML External Entities (XXE)

- What can be done?
  - Patch and upgrade XML processors and libraries
  - Disable XXE processing
  - Input validation
  - Developer training
  - Don’t trust the client
  - WAF, API security gateway

OWASP_Top_Ten_2017_A4-XML_External_Entities_(XXE)
OWASP_XML_External_Entity_(XXE)_Processing

A5 - Broken Access Control

A5 - Broken Access Control

What is it?

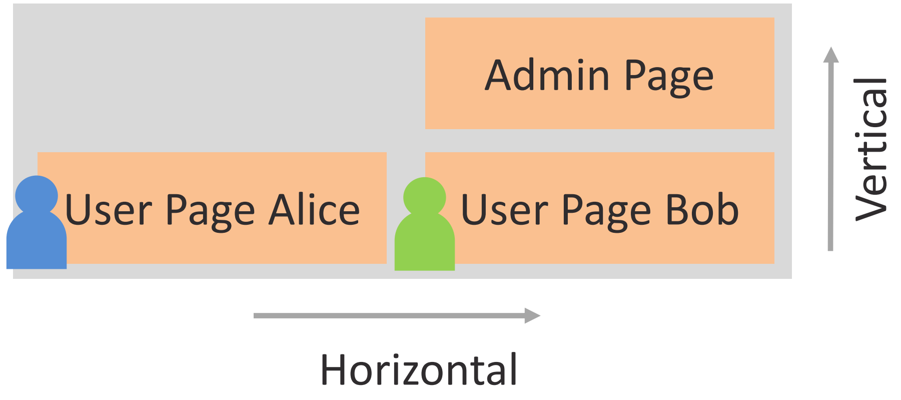
- Attacker gains access to more than he should be able to
- Access Control =/= Authentication
- Horizontal privilege escalation
- Vertical privilege escalation

Examples:

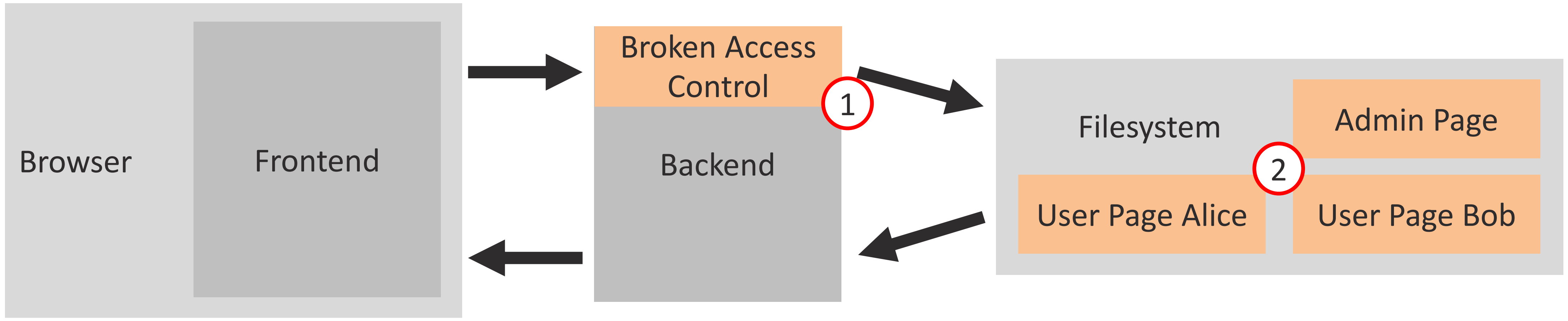
- Missing Access control on server side
- Access control bypass
- Missing access controls for POST, PUT, DELETE, …

A5 - Broken Access Control

Horizontal vs vertical privilege escalation



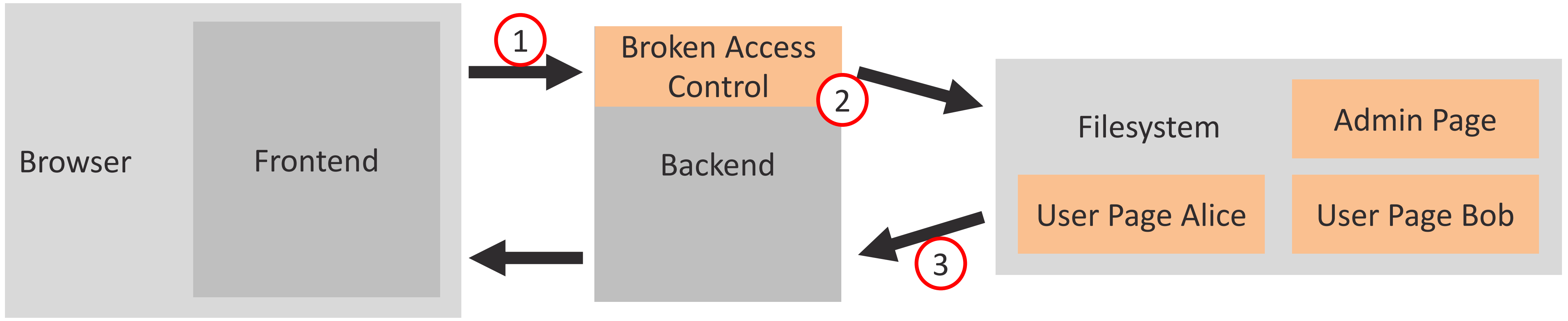
A5 - Broken Access Control



What is required?

1. Broken Access Control
2. Sensitive Data or Functions

A5 - Broken Access Control



How does it work?

1. Resource or function is requested
2. Access control fails
3. Sensitive data is returned or function executed

A5 - Broken Access Control

What can be done?

- Developer training
- Don’t trust the client

- SAST / DAST
  – Can identify if access control is in place, but not how effective
- Deny by default
- Proper documentation of user roles
- Implement Access controls once, test and reuse
- Validate on server side
- Log failures
- Least privilege

A6 - Security Misconfiguration

A6 - Security Misconfiguration

- What is it?
  - Insecure configuration
- Examples
  - Default login credentials
  - Missing network separation
  - DEV config in PROD
  - Vulnerable sample applications
  - Directory listing
  - Error messages

A6 - Security Misconfiguration

- What can be done?
  - Hardening guidelines
  - Minimal functionality
  - Regular updates of software
  - Automated process to verify configurations

OWASP_Top_Ten_2017_A6-Security_Misconfiguration

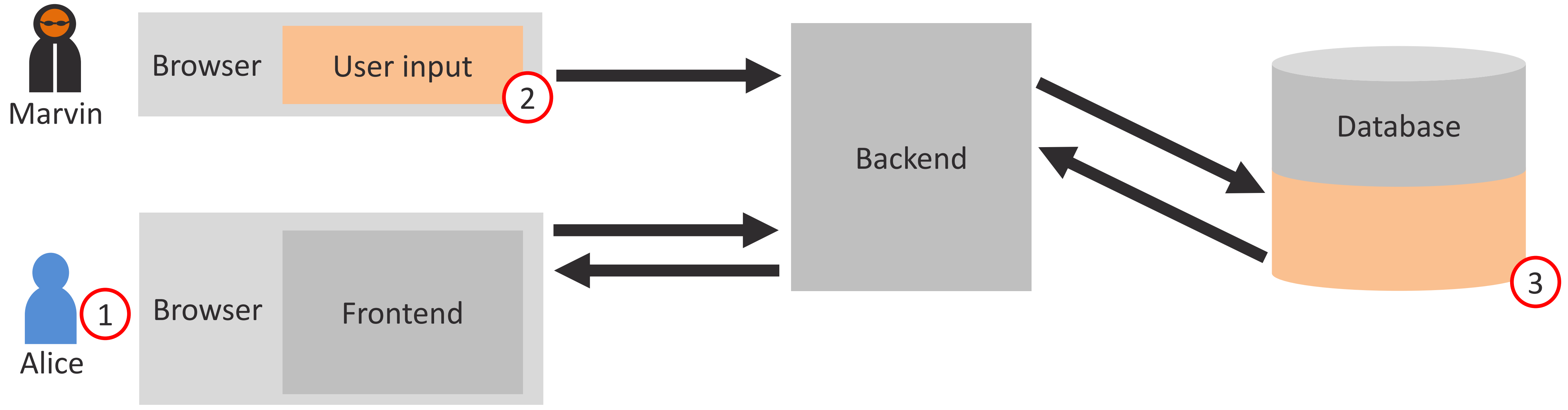
A7 - Cross-Site Scripting (XSS)

A7 - Cross-Site Scripting (XSS)

- What is it?
  - Specific kind of injection attack
  - Client side code injection
- Went down quite a lot in the list

- Various kinds of XSS
  - Persistent
  - Reflected
  - DOM-based

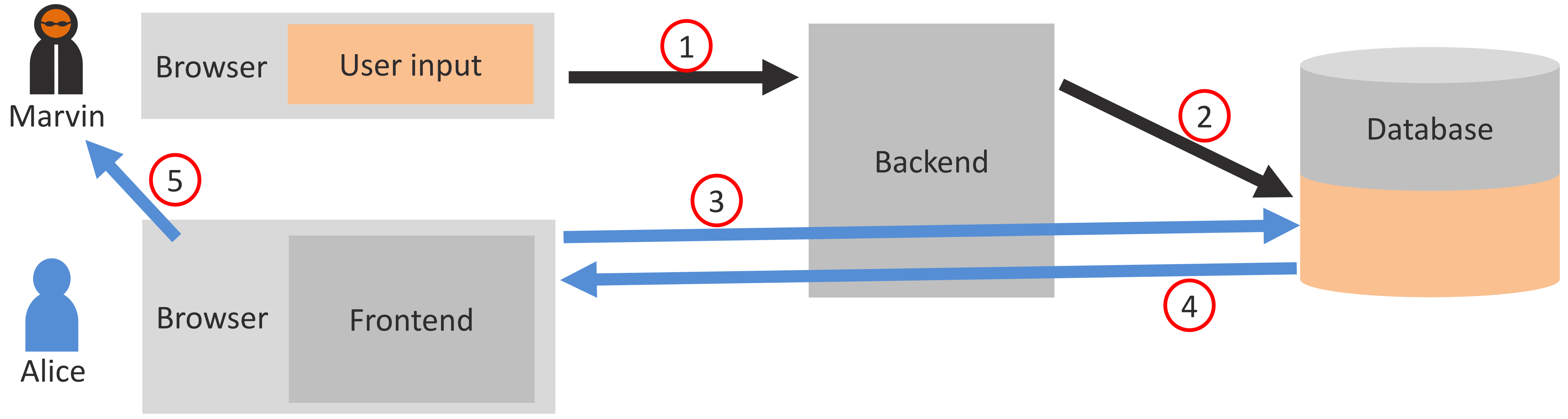
A7 - Cross-Site Scripting (XSS)



What is required?

1. A victim and his browser
2. Vulnerable input
3. Place to store XSS code

A7 - Cross-Site Scripting (XSS)



How does it work?

1. POST request with XSS by Marvin
2. XSS is stored
3. GET request by Alice
4. Script is returned to Alice
5. Script is executed in browser and session cookie sent to Marvin

A7 - Cross-Site Scripting (XSS)

What can be done?

- Developer training
- Don’t trust the client
  - Validate input and escape/encode output
- Use frameworks that escape XSS payloads correctly by design
- WAF

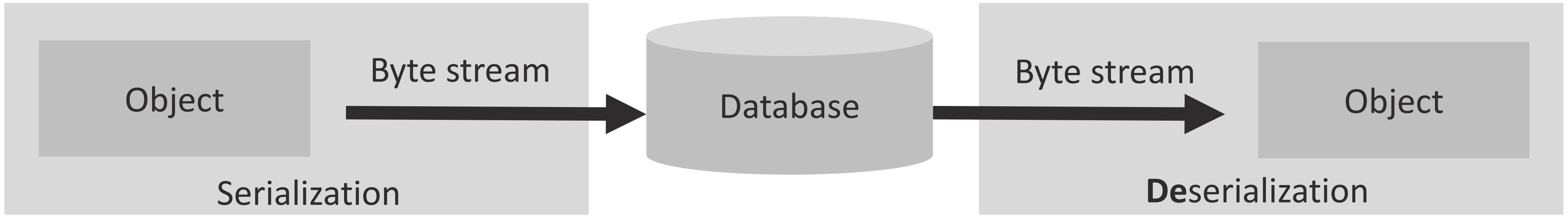
- OWASP Java Encoder Project
- OWASP Java HTML Sanitizer Project
- Microsoft Encoder and AntiXSS Library

https://excess-xss.com/
OWASP_Top_Ten_2017_A7-Cross-Site_Scripting_(XSS)
Cross_Site_Scripting_Prevention_Cheat_Sheet

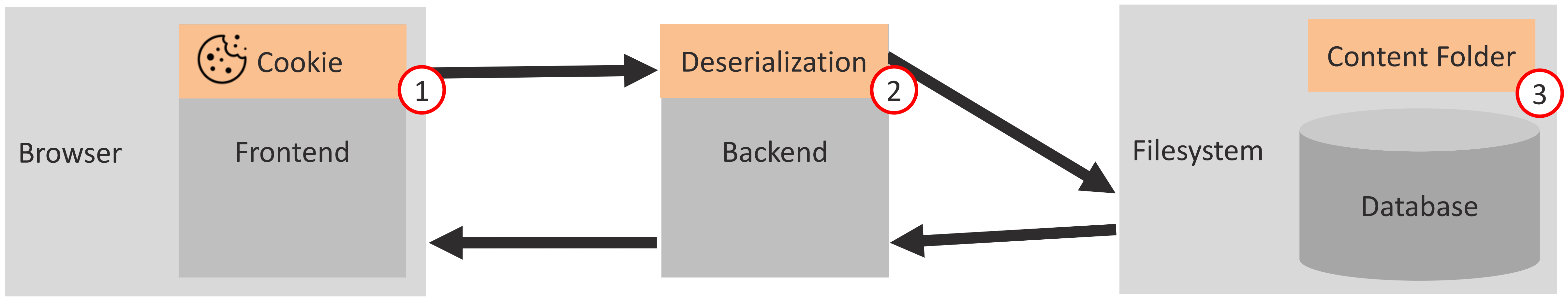
A8 - Insecure Deserialization

A8 - Insecure Deserialization

- What is it?
  - Serialization
  - Deserialization



A8 - Insecure Deserialization



What is required?

1. E.g. a serialized cookie (userID, role, pwhash, state data)
2. Deserialization step to create an object
3. Action performed on the new object

REDGUARD
SECURING YOUR ASSETS

A8 - Insecure Deserialization

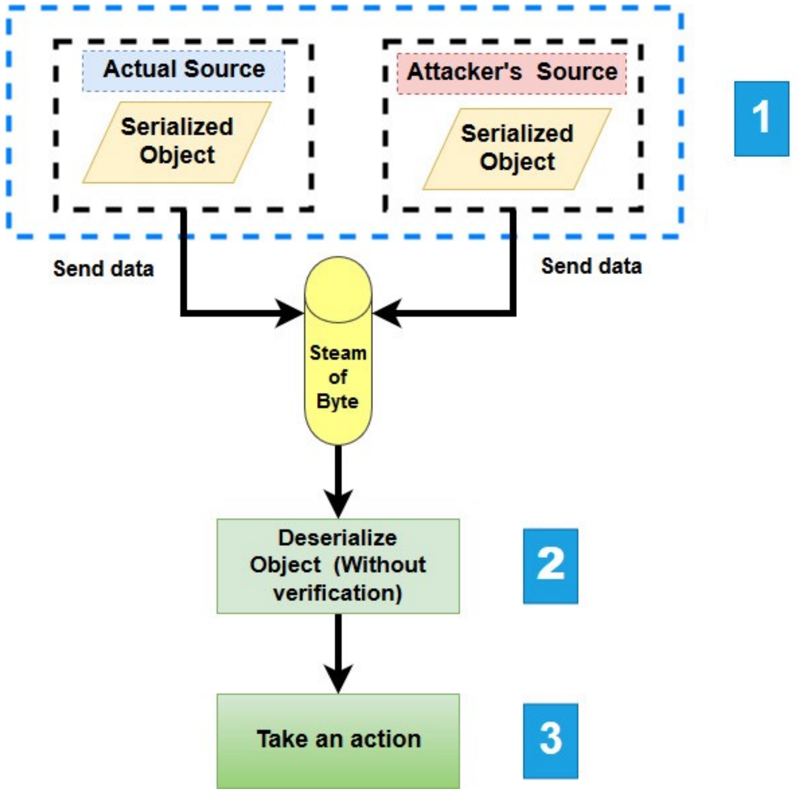
- How does it work?

A forum uses object serialization to save a “super” cookie, containing the user’s user ID, role, password hash, and other state:

```
a:4:{i:0;i:132;i:1;s:7:"Alice";i:2;s:4:"user";
i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```

An attacker changes the serialized object to give themselves admin privileges:

```
a:4:{i:0;i:1;i:1;s:5:"Marvin";i:2;s:5:"admin";
i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```



A8 - Insecure Deserialization

- What can be done?
  - Implementing integrity checks
  - Log deserialization exceptions and errors
  - Monitor deserialization
  - Enforce strict type constraints

OWASP_Top_Ten_2017_A8-Insecure_Deserialization

Deserialization_Cheat_Sheet

A9 - Using Components With Known Vulnerabilities

A9 Using Components With Known Vulnerablilties

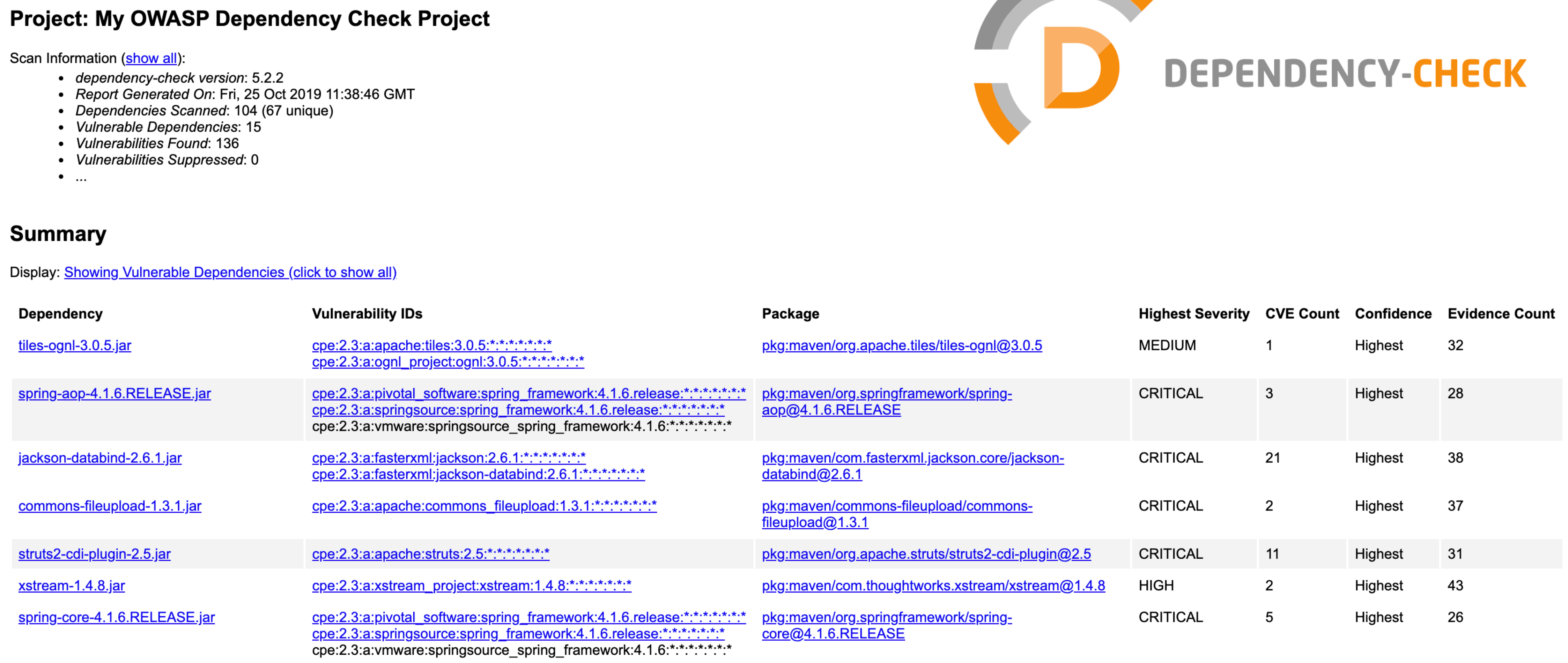
What is it?

- What is says on the tin

Examples

- Libraries
- Applications
- Operating systems

A9 Using Components With Known Vulnerablilties



Project: My OWASP Dependency Check Project

Scan Information (show all):

- *dependency-check version*: 5.2.2
- *Report Generated On*: Fri, 25 Oct 2019 11:38:46 GMT
- *Dependencies Scanned*: 104 (67 unique)
- *Vulnerable Dependencies*: 15
- *Vulnerabilities Found*: 136
- *Vulnerabilities Suppressed*: 0
- ...

Summary

Display: Showing Vulnerable Dependencies (click to show all)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
tiles-ognl-3.0.5.jar	cpe:2.3:a:apache:tiles:3.0.5:*:*:*:*:*:*:* cpe:2.3:a:ognl_project:ognl:3.0.5:*:*:*:*:*:*:*	pkg:maven/org.apache.tiles/tiles-ognl@3.0.5	MEDIUM	1	Highest	32
spring-aop-4.1.6.RELEASE.jar	cpe:2.3:a:pivotal_software:spring_framework:4.1.6.release:*:*:*:*:*:*:* cpe:2.3:a:springsource:spring_framework:4.1.6.release:*:*:*:*:*:*:* cpe:2.3:a:vmware:springsource_spring_framework:4.1.6:*:*:*:*:*:*:*	pkg:maven/org.springframework/spring-aop@4.1.6.RELEASE	CRITICAL	3	Highest	28
jackson-databind-2.6.1.jar	cpe:2.3:a:fasterxml:jackson:2.6.1:*:*:*:*:*:*:* cpe:2.3:a:fasterxml:jackson-databind:2.6.1:*:*:*:*:*:*:*	pkg:maven/com.fasterxml.jackson.core/jackson-databind@2.6.1	CRITICAL	21	Highest	38
commons-fileupload-1.3.1.jar	cpe:2.3:a:apache:commons_fileupload:1.3.1:*:*:*:*:*:*:*	pkg:maven/commons-fileupload/commons-fileupload@1.3.1	CRITICAL	2	Highest	37
struts2-cdi-plugin-2.5.jar	cpe:2.3:a:apache:struts:2.5:*:*:*:*:*:*:*	pkg:maven/org.apache.struts/struts2-cdi-plugin@2.5	CRITICAL	11	Highest	31
xstream-1.4.8.jar	cpe:2.3:a:xstream_project:xstream:1.4.8:*:*:*:*:*:*:*	pkg:maven/com.thoughtworks.xstream/xstream@1.4.8	HIGH	2	Highest	43
spring-core-4.1.6.RELEASE.jar	cpe:2.3:a:pivotal_software:spring_framework:4.1.6.release:*:*:*:*:*:*:* cpe:2.3:a:springsource:spring_framework:4.1.6.release:*:*:*:*:*:*:* cpe:2.3:a:vmware:springsource_spring_framework:4.1.6:*:*:*:*:*:*:*	pkg:maven/org.springframework/spring-core@4.1.6.RELEASE	CRITICAL	5	Highest	26

A9 Using Components With Known Vulnerablilties

What can be done?

- Defense in depth
- Processes for:
  - Regular vulnerability scans
  - Patch cycles
- Vulnerability Management
- CVE & NVE

OWASP_Top_Ten_2017_A9-Using_Components_with_Known_Vulnerabilities

A10 - Insufficient Logging & Monitoring

A10 - Insufficient Logging & Monitoring

- What is the difference between “Logging” and “Monitoring”?

- In 2019, identifying a data breach took an average of 206 days and another 73 to contain the breach (source: IBM)

A10 - Insufficient Logging & Monitoring

- What can be done?
  - Log security related events (negative and positive)
  - Centralize and standardize log collection
  - Be sure logs can easily be processed
  - Verify that monitoring and alerting works
  - Ensure data integrity
  - Establish incident response and recovery plans

OWASP_Top_Ten_2017_A10-Insufficient_Logging_and_Monitoring

Logging_Cheat_Sheet

Summary

Summary

- This is not all there is
- Educate your developers
- Don’t trust the user - sanitize user input
- Don't just do security at the end
- Use layered defense

Thanks!

BERN
Redguard AG
Eigerstrasse 60
CH-3007 Bern

ZÜRICH
Redguard AG
Thurgauerstrasse 36/38
CH-8050 Zürich

Phone: +41 (0)31 511 37 50
contact@redguard.ch
www.redguard.ch