

# General Topics

Insecure Design, Components with known Vulnerabilities, Integrity,  
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

# Design Flaw vs. Implementation Defect

**Design Flaw / Insecure Design**

**Implementation Defect**


# Design Flaw vs. Implementation Defect

Design Flaw / Insecure Design	Implementation Defect
Client-side security checks only	

# Design Flaw vs. Implementation Defect

Design Flaw / Insecure Design	Implementation Defect
Client-side security checks only	Error in a regex for input validation

# Design Flaw vs. Implementation Defect

## **Design Flaw / Insecure Design**

Client-side security checks only

Credential recovery solely relies on “security questions”

## **Implementation Defect**

Error in a regex for input validation

# Design Flaw vs. Implementation Defect

## Design Flaw / Insecure Design

Client-side security checks only

Credential recovery solely relies on “security questions”

## Implementation Defect

Error in a regex for input validation

Reset link is guessable

# Design Flaw vs. Implementation Defect

## Design Flaw / Insecure Design

Client-side security checks only

Credential recovery solely relies on “security questions”

Firmware update integrity isn't ensured

## Implementation Defect

Error in a regex for input validation

Reset link is guessable



# Design Flaw vs. Implementation Defect

## Design Flaw / Insecure Design

Client-side security checks only

Credential recovery solely relies on “security questions”

Firmware update integrity isn't ensured

## Implementation Defect

Error in a regex for input validation

Reset link is guessable

Digital signature uses outdated/vuln. algorithm

A secure design can still have implementation defects leading to vulnerabilities that may be exploited.

An insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks.

How can we prevent  
design flaws?

Remember our secure  
design principles?

Economy of Mechanism

---

Fail-safe Defaults

---

Complete Mediation

---

Least Privilege

---

Least Common Mechanism

---

Separation of Privilege

---

Open Design

---

Psychological Acceptability

- Saltzer and Schroeder, 1975 -

[http://web.cs.wpi.edu/~guttman/cs557\\_website/papers/saltzer1975.pdf](http://web.cs.wpi.edu/~guttman/cs557_website/papers/saltzer1975.pdf)

<https://adam.shostack.org/blog/the-security-principles-of-saltzer-and-schroeder/>

Earn or give, but never assume, trust.

---

Use an authentication mechanism that cannot be bypassed or tampered with.

---

Authorize after you authenticate

---

Strictly separate data and control instructions, and never process control instructions received from untrusted sources.

---

Define an approach that ensures all data are explicitly validated.

---

Use cryptography correctly.

---

Identify sensitive data and how they should be handled.

---

Always consider the user.

---

Understand how integrating external components changes your attack surface.

---

Be flexible when considering future changes to objects and actors.

- IEEE Center for Secure Design, 2014 -

<https://ieeecs-media.computer.org/media/technical-activities/CYBSI/docs/Top-10-Flaws.pdf>

# Secure Design Patterns

Utilize well proven public design patterns, e.g.

- <https://docs.microsoft.com/en-us/azure/architecture/patterns/>
  - <https://docs.microsoft.com/en-us/azure/architecture/patterns/federated-identity>
  - ...
- <https://docs.microsoft.com/en-us/azure/architecture/framework/security/>
- <https://www.opensecurityarchitecture.org/cms/library/patternlandscape>
- <https://www.ncsc.gov.uk/search?q=architectural%20pattern>
- <https://cheatsheetseries.owasp.org/>

Yes, a lot of them are actually for network / system design

But they are very useful anyway

# Secure Design Patterns

Create your own company-wide design patterns for e.g.

- Authentication
- Session Management
- Authorization
- Input- and Outputhandling
- Logging and Monitoring
- ...



Did anybody say  
Threat Modeling?

# Check out my Threat Modeling 101

<https://www.slideshare.net/SBA-Research/sba-live-academy-threat-modeling-101-eine-kurze-aber-praxisnahe-einfhrung-by-daniel-schwarz-senior-security-analyst-bei-der-condignum-gmbh>

# Threat Modeling in a nutshell

It's an engineering technique you can use to help you **identify threats, attacks, vulnerabilities, and countermeasures** that could affect your application.

You can use threat modeling to **shape your application's design, meet your company's security objectives, and reduce risk.**

- Microsoft -

<https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling>

# Threat Modeling in a nutshell

The 4 basic questions:

1. What are we working on?
2. What can go wrong?
3. What are we going to do about it?
4. Did we do a good enough job?

# Threat Modeling in a nutshell

Simplest approach:

just brainstorm in your team

think about worst-case szenarios  
and what you could do about it

# Threat Modeling in a nutshell

second simplest approach:

use some “tools” to structure your thoughts

# Threat Modeling in a nutshell

little example:

- We're creating a pizza order system
- Customers can save
  - their address
  - their personal pizza configurations
  - their credit card
- Payment is done via an external payment-provider

# Threat Modeling in a nutshell

The 4 basic questions:

1. What are we working on?
2. What can go wrong?
3. What are we going to do about it?
4. Did we do a good enough job?

<https://www.threatmodelingmanifesto.org/>

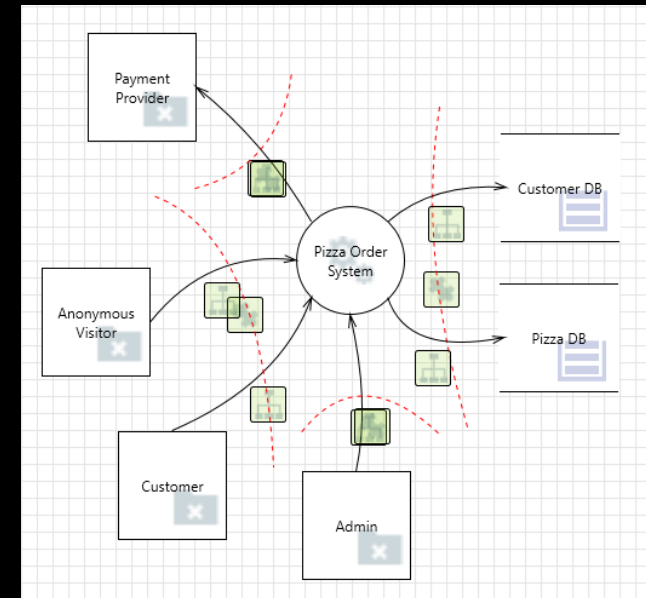


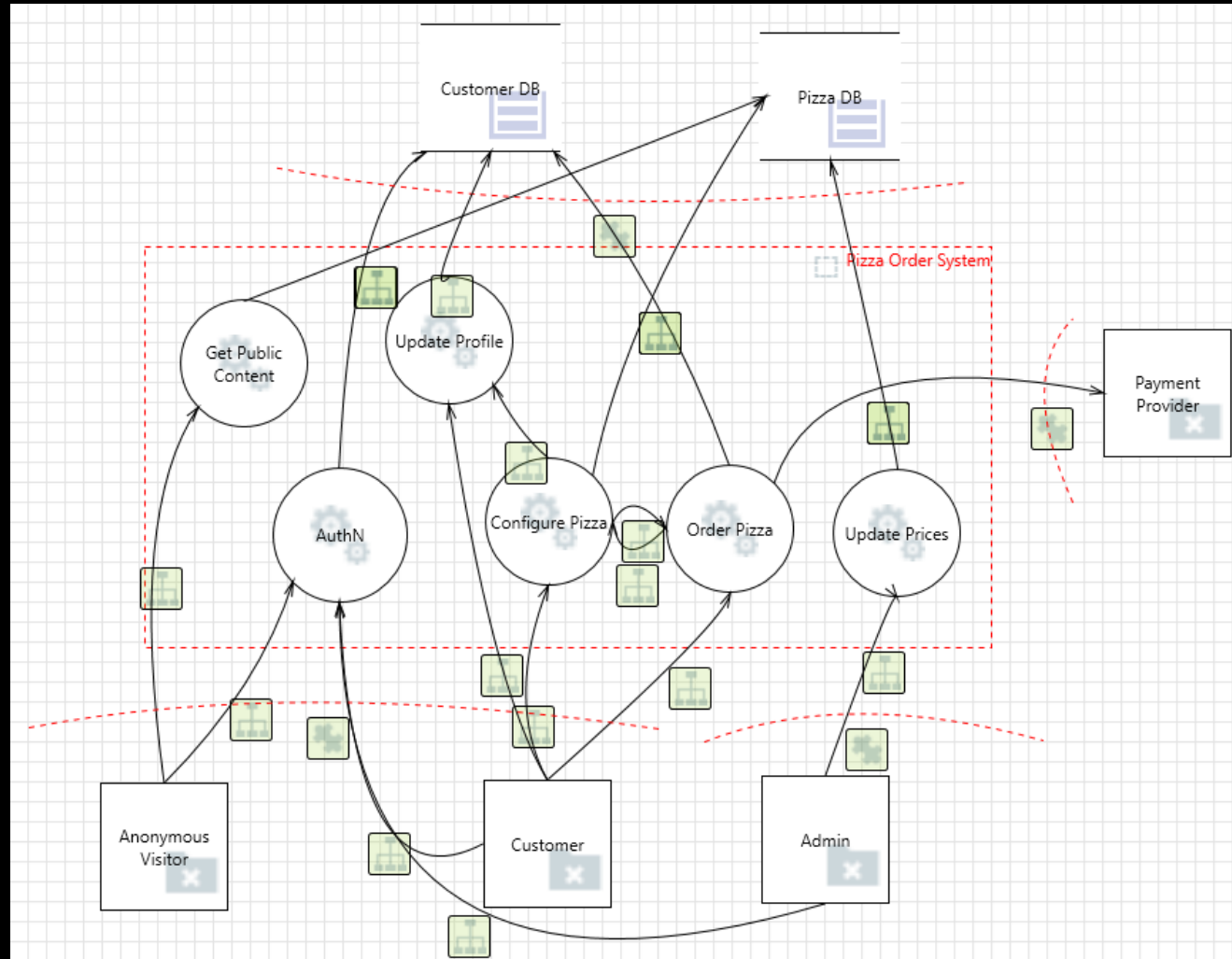
# Threat Modeling in a nutshell

## 1. What are we working on?

- list your assets, users and external entities
- draw a diagram
  - DFDs and the C4 model could be a good starting point

Assets	User	Ext. Entities
credit card data	anonymous	payment-provider
usernames	customer	
passwords	admin	
pizza prices		
pizza configurations		
...		





# Threat Modeling in a nutshell

## 2. What can go wrong?

- Design Principles, OWASP Top 10 ...
- STRIDE
  - Spoofing
  - Tampering
  - Repudiation
  - Information Disclosure
  - Denial of Service
  - Elevation of Privileges

<https://www.microsoft.com/security/blog/2007/09/11/stride-chart/>

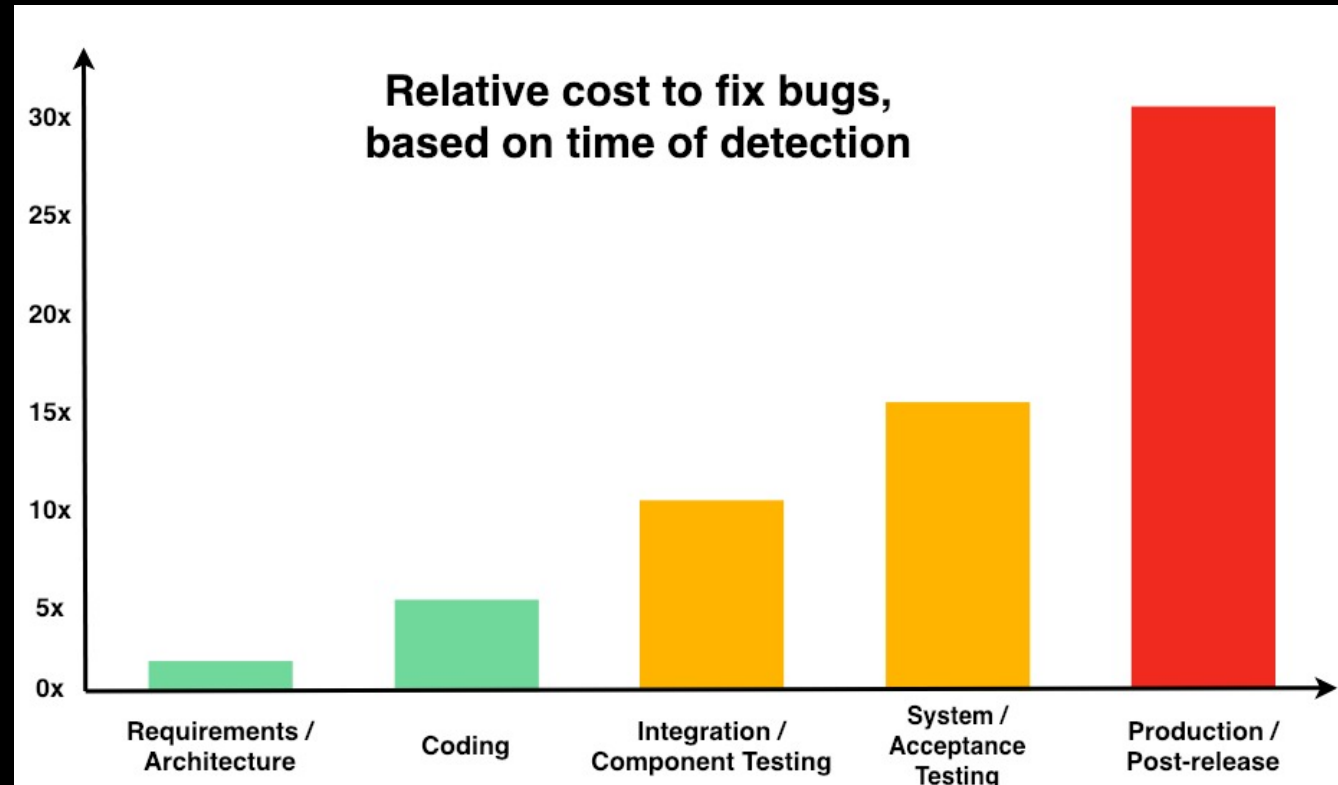
# Threat Modeling in a nutshell

## 3. What are we going to do about it?

- Think about “how” the scenarios can happen and how to prevent it
- Use resources like OWASP ASVS

<https://owasp.org/www-project-application-security-verification-standard/>

# It's even cheaper to invest in a secure design



<https://deepsources.io/blog/exponential-cost-of-fixing-bugs/>

Design vs. Architecture

WTF is the difference?

# Design vs. Architecture

An application consists of multiple building blocks

## Design

- Every decision to combine these building blocks in a specific way

## Architecture

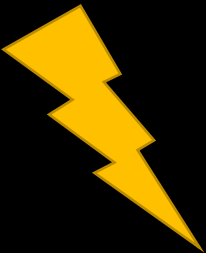
- The most significant design decisions (high cost of change)

All architecture is design, but not all design is architecture.

- Grady Booch

<https://static.architectis.jp/software-architecture-for-developers.pdf>

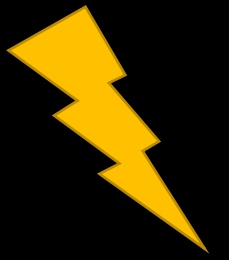
# Insecure Design



Goal	Exploit design flaws to do all kinds of bad stuff
How	
Solution	
OWASP Top 10	
(Primary) Violated Principle	

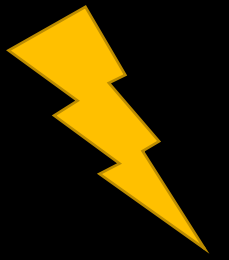


# Insecure Design



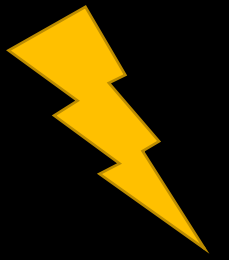
Goal	Exploit design flaws to do all kinds of bad stuff
How	Understand the internal structure and workflows of an application
Solution	
OWASP Top 10	
(Primary) Violated Principle	

# Insecure Design



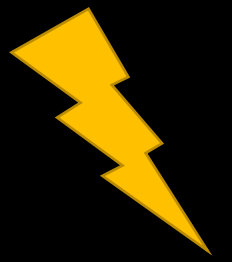
Goal	Exploit design flaws to do all kinds of bad stuff
How	Understand the internal structure and workflows of an application
Solution	Stick to the secure design principles
OWASP Top 10	
(Primary) Violated Principle	

# Insecure Design



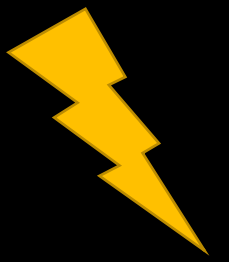
Goal	Exploit design flaws to do all kinds of bad stuff
How	Understand the internal structure and workflows of an application
Solution	Stick to the secure design principles
	Utilize proven secure design patterns
OWASP Top 10	
(Primary) Violated Principle	

# Insecure Design



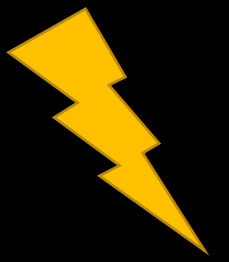
Goal	Exploit design flaws to do all kinds of bad stuff
How	Understand the internal structure and workflows of an application
Solution	Stick to the secure design principles
	Utilize proven secure design patterns
	Include Threat Modeling in your SSDL (design phase)
OWASP Top 10	
(Primary) Violated Principle	

# Insecure Design



Goal	Exploit design flaws to do all kinds of bad stuff
How	Understand the internal structure and workflows of an application
Solution	Stick to the secure design principles
	Utilize proven secure design patterns
	Include Threat Modeling in your SSDL (design phase)
OWASP Top 10	A04:2021-Insecure Design
(Primary) Violated Principle	

# Insecure Design



Goal	Exploit design flaws to do all kinds of bad stuff
How	Understand the internal structure and workflows of an application
Solution	Stick to the secure design principles
	Utilize proven secure design patterns
	Include Threat Modeling in your SSDL (design phase)
OWASP Top 10	A04:2021-Insecure Design
(Primary) Violated Principle	„Earn or give, but never assume, trust.“

# 3<sup>rd</sup> party components

It's ok to use 3<sup>rd</sup> party components

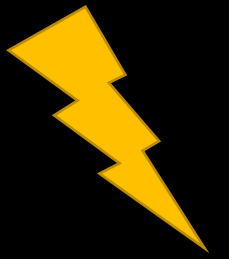
- libraries
- frameworks
- etc.

Just be aware you also include their problems

- e.g. Commons Collection in 2015

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

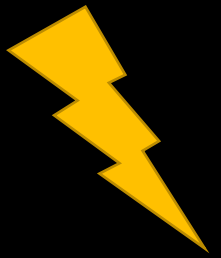
Solution

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

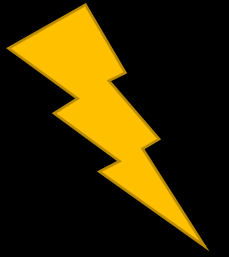
Fingerprint application  
Search the web for corresponding vulnerabilities and exploits  
e.g. <https://www.cvedetails.com/>, <https://www.exploit-db.com/>, etc.

Solution

OWASP Top 10

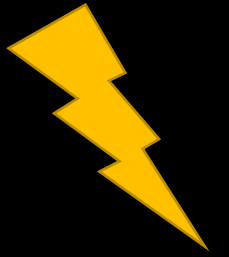
(Primary)  
Violated Principle

# Vulnerabilities in 3rd Party Components



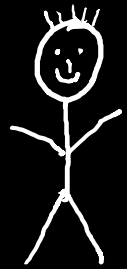
Goal	Compromising an application by exploiting a publicly known vulnerability in one of it's included components (libraries, frameworks etc.)
How	Fingerprint application Search the web for corresponding vulnerabilities and exploits e.g. <a href="https://www.cvedetails.com/">https://www.cvedetails.com/</a> , <a href="https://www.exploit-db.com/">https://www.exploit-db.com/</a> , 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. <a href="https://www.cvedetails.com/">https://www.cvedetails.com/</a> , <a href="https://www.exploit-db.com/">https://www.exploit-db.com/</a> , 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



IDE

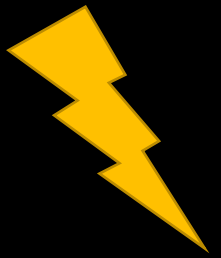
3<sup>rd</sup> party  
check



CI/CD

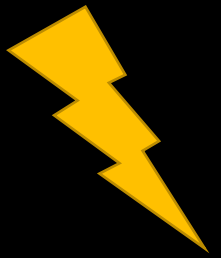
App Server

# Vulnerabilities in 3rd Party Components



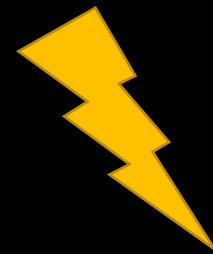
Goal	Compromising an application by exploiting a publicly known vulnerability in one of it's included components (libraries, frameworks etc.)
How	Fingerprint application Search the web for corresponding vulnerabilities and exploits e.g. <a href="https://www.cvedetails.com/">https://www.cvedetails.com/</a> , <a href="https://www.exploit-db.com/">https://www.exploit-db.com/</a> , 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



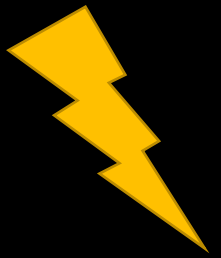
Goal	Compromising an application by exploiting a publicly known vulnerability in one of it's included components (libraries, frameworks etc.)
How	Fingerprint application Search the web for corresponding vulnerabilities and exploits e.g. <a href="https://www.cvedetails.com/">https://www.cvedetails.com/</a> , <a href="https://www.exploit-db.com/">https://www.exploit-db.com/</a> , 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. <a href="https://www.cvedetails.com/">https://www.cvedetails.com/</a> , <a href="https://www.exploit-db.com/">https://www.exploit-db.com/</a> , 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	

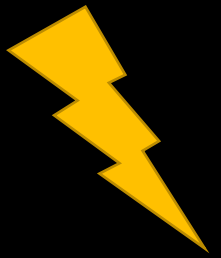
# Vulnerabilities in 3rd Party Components



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. <a href="https://www.cvedetails.com/">https://www.cvedetails.com/</a> , <a href="https://www.exploit-db.com/">https://www.exploit-db.com/</a> , 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	A06:2021-Vulnerable and Outdated Components
(Primary) Violated Principle	

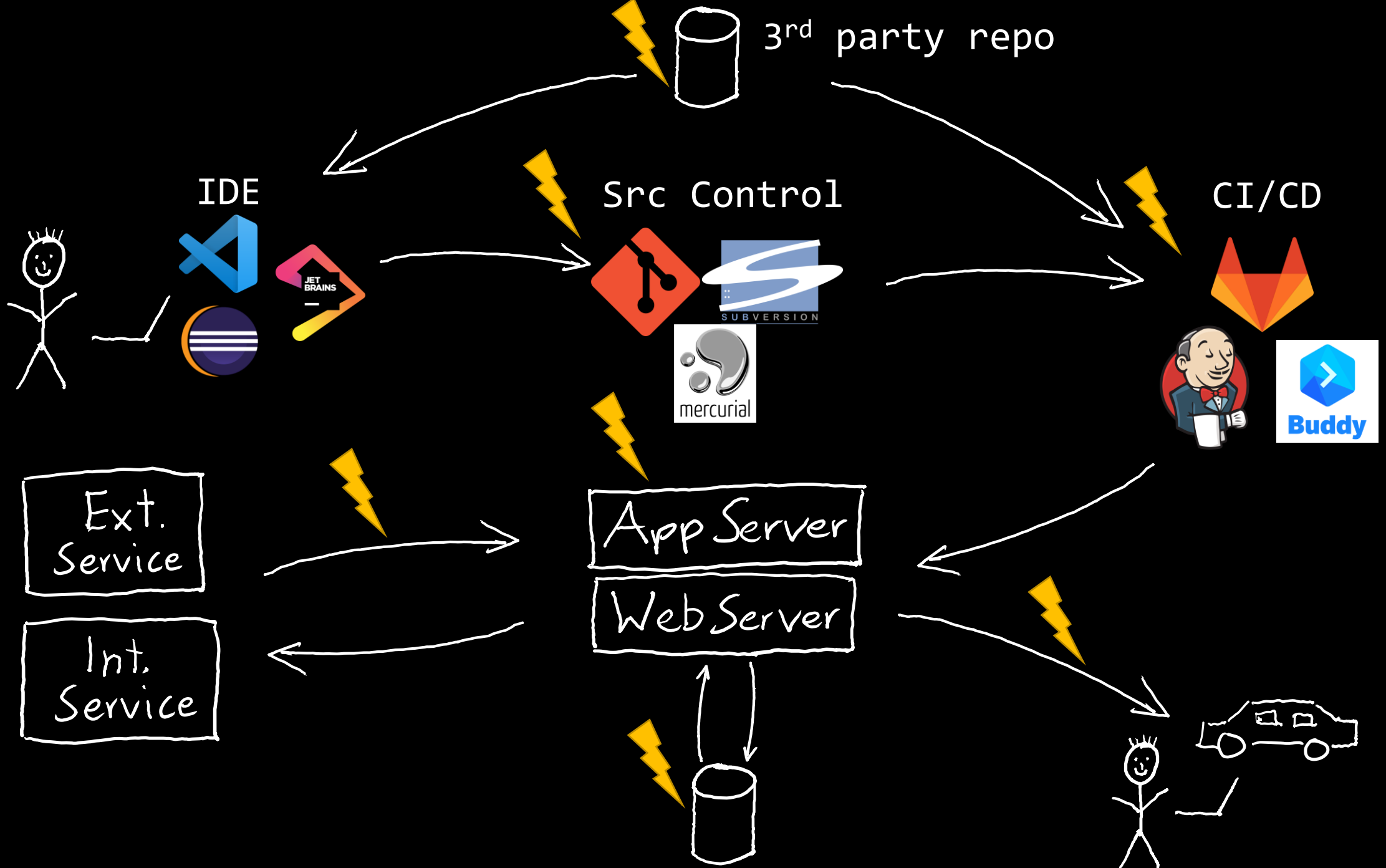


# Vulnerabilities in 3rd Party Components

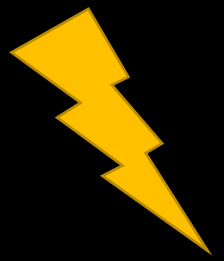


Goal	Compromising an application by exploiting a publicly known vulnerability in one of it's included components (libraries, frameworks etc.)
How	Fingerprint application Search the web for corresponding vulnerabilities and exploits e.g. <a href="https://www.cvedetails.com/">https://www.cvedetails.com/</a> , <a href="https://www.exploit-db.com/">https://www.exploit-db.com/</a> , 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	A06:2021-Vulnerable and Outdated Components
(Primary) Violated Principle	„Understand how integrating external components changes your attack surface“

let's talk about  
integrity

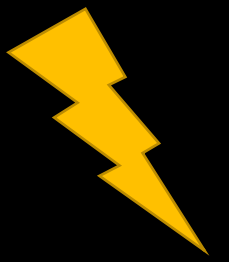


# Software and Data Integrity Failures



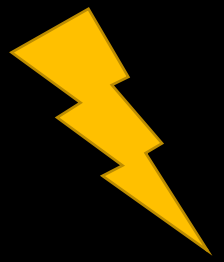
Goal	Manipulate the application itself or the application's data.
How	
Solution	
OWASP Top 10	
(Primary) Violated Principle	

# Software and Data Integrity Failures



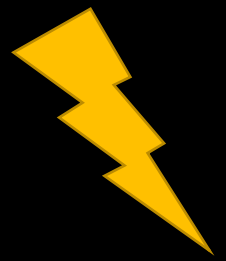
Goal	Manipulate the application itself or the application's data.
How	Diverse manipulation options along the whole application lifecycle
Solution	
OWASP Top 10	
(Primary) Violated Principle	

# Software and Data Integrity Failures



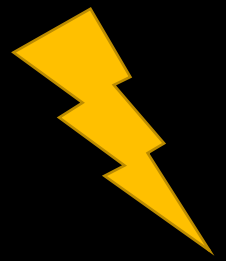
Goal	Manipulate the application itself or the application's data.
How	Diverse manipulation options along the whole application lifecycle
Solution	Review process (e.g. 4 eyes principle) for all code changes
OWASP Top 10	
(Primary) Violated Principle	

# Software and Data Integrity Failures



Goal	Manipulate the application itself or the application's data.
How	Diverse manipulation options along the whole application lifecycle
Solution	Review process (e.g. 4 eyes principle) for all code changes
	Strict access control for CI/CD pipeline and servers / DBs
OWASP Top 10	
(Primary) Violated Principle	

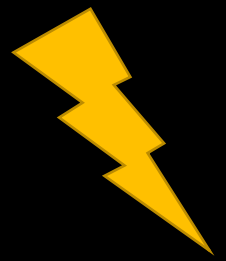
# Software and Data Integrity Failures



Goal	Manipulate the application itself or the application's data.
How	Diverse manipulation options along the whole application lifecycle
Solution	Review process (e.g. 4 eyes principle) for all code changes
	Strict access control for CI/CD pipeline and servers / DBs
	Obtain 3 <sup>rd</sup> party components from trusted sources only
OWASP Top 10	
(Primary) Violated Principle	

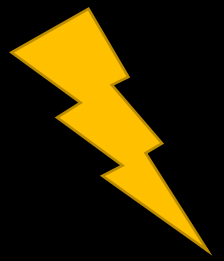


# Software and Data Integrity Failures



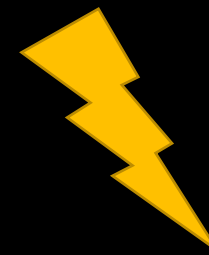
Goal	Manipulate the application itself or the application's data.
How	Diverse manipulation options along the whole application lifecycle
Solution	Review process (e.g. 4 eyes principle) for all code changes
	Strict access control for CI/CD pipeline and servers / DBs
	Obtain 3 <sup>rd</sup> party components from trusted sources only
	Use TLS for all communication
OWASP Top 10	
(Primary) Violated Principle	

# Software and Data Integrity Failures



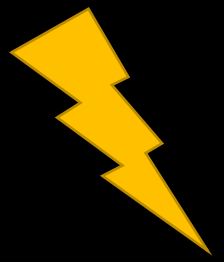
Goal	Manipulate the application itself or the application's data.
How	Diverse manipulation options along the whole application lifecycle
Solution	Review process (e.g. 4 eyes principle) for all code changes
	Strict access control for CI/CD pipeline and servers / DBs
	Obtain 3 <sup>rd</sup> party components from trusted sources only
	Use TLS for all communication
	Explicitly ensure integrity (e.g. via digital signatures) for updates / software packages critical data serialized data streams
OWASP Top 10	
(Primary) Violated Principle	

# Software and Data Integrity Failures



Goal	Manipulate the application itself or the application's data.
How	Diverse manipulation options along the whole application lifecycle
Solution	Review process (e.g. 4 eyes principle) for all code changes
	Strict access control for CI/CD pipeline and servers / DBs
	Obtain 3 <sup>rd</sup> party components from trusted sources only
	Use TLS for all communication
	Explicitly ensure integrity (e.g. via digital signatures) for updates / software packages critical data serialized data streams
OWASP Top 10	A08:2021-Software and Data Integrity Failures
(Primary) Violated Principle	

# Software and Data Integrity Failures



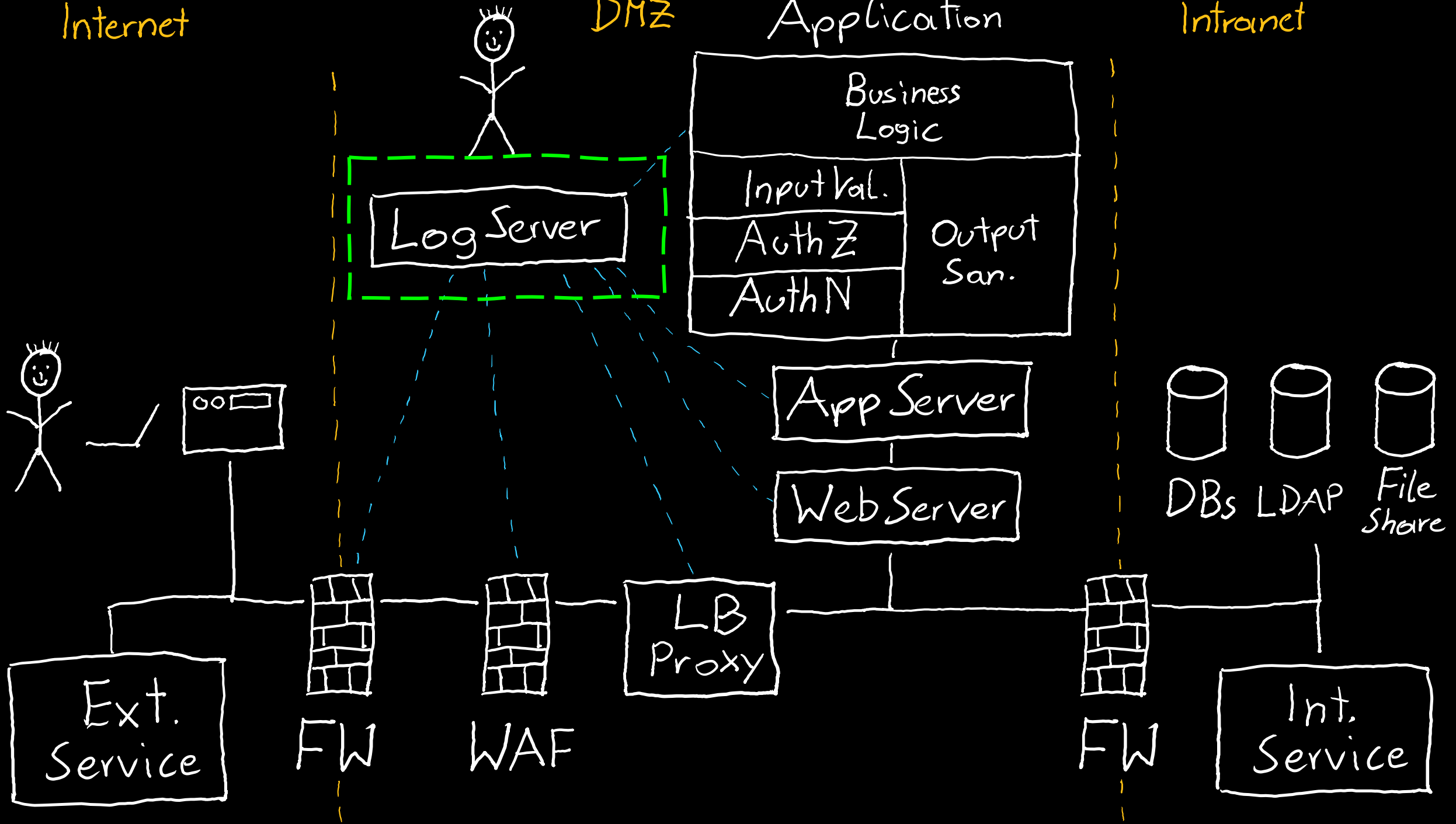
Goal	Manipulate the application itself or the application's data.
How	Diverse manipulation options along the whole application lifecycle
Solution	Review process (e.g. 4 eyes principle) for all code changes
	Strict access control for CI/CD pipeline and servers / DBs
	Obtain 3 <sup>rd</sup> party components from trusted sources only
	Use TLS for all communication
	Explicitly ensure integrity (e.g. via digital signatures) for updates / software packages critical data serialized data streams
OWASP Top 10	A08:2021-Software and Data Integrity Failures
(Primary) Violated Principle	„Define an approach that ensures all data are explicitly validated.“

Internet

DMZ

Application

Intranet



# Security Logging and Monitoring Failures



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

# Security Logging and Monitoring Failures



Goal

Hide attacks and go unnoticed.

How

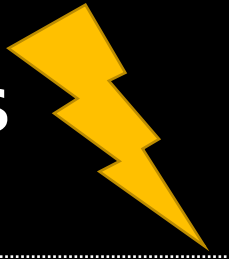
Security relevant events are not logged appropriately

Solution

OWASP Top 10

(Primary)  
Violated Principle

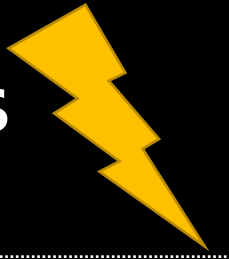
# Security Logging and Monitoring Failures



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	

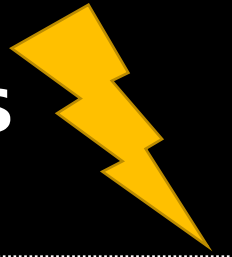


# Security Logging and Monitoring Failures



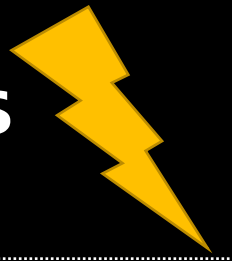
Goal	Hide attacks and go unnoticed.
How	Security relevant events are not logged appropriately Logs are not monitored regularly No appropriate alert thresholds are in place
Solution	
OWASP Top 10	
(Primary) Violated Principle	

# Security Logging and Monitoring Failures



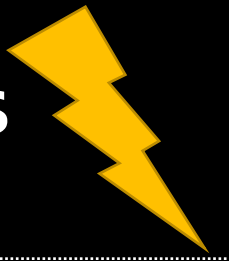
Goal	Hide attacks and go unnoticed.
How	Security relevant events are not logged appropriately Logs are not monitored regularly No appropriate alert thresholds are in place No suitable response process is in place
Solution	
OWASP Top 10	
(Primary) Violated Principle	

# Security Logging and Monitoring Failures



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	

# Security Logging and Monitoring Failures



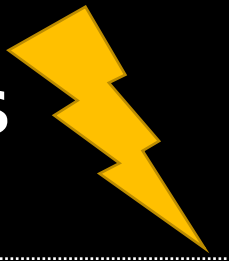
Goal	Hide attacks and go unnoticed.
How	Security relevant events are not logged appropriately Logs are not monitored regularly No appropriate alert thresholds are in place No suitable response process is in place
Solution	Define which events are security relevant and therefore should be logged e.g. failed 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	

# Security Logging and Monitoring Failures



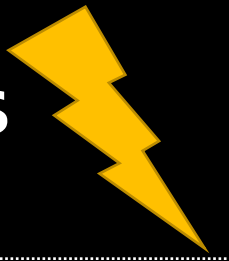
Goal	Hide attacks and go unnoticed.
How	Security relevant events are not logged appropriately Logs are not monitored regularly No appropriate alert thresholds are in place No suitable response process is in place
Solution	Define which events are security relevant and therefore should be logged e.g. failed 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	

# Security Logging and Monitoring Failures



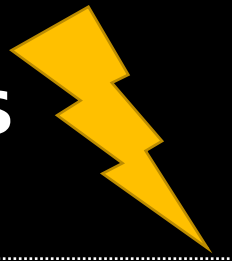
Goal	Hide attacks and go unnoticed.
How	Security relevant events are not logged appropriately Logs are not monitored regularly No appropriate alert thresholds are in place No suitable response process is in place
Solution	Define which events are security relevant and therefore should be logged e.g. failed 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	

# Security Logging and Monitoring Failures



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	A09:2021-Security Logging and Monitoring Failures
(Primary) Violated Principle	

# Security Logging and Monitoring Failures



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	A09:2021-Security Logging and Monitoring Failures
(Primary) Violated Principle	„Earn or give, but never assume, trust.“

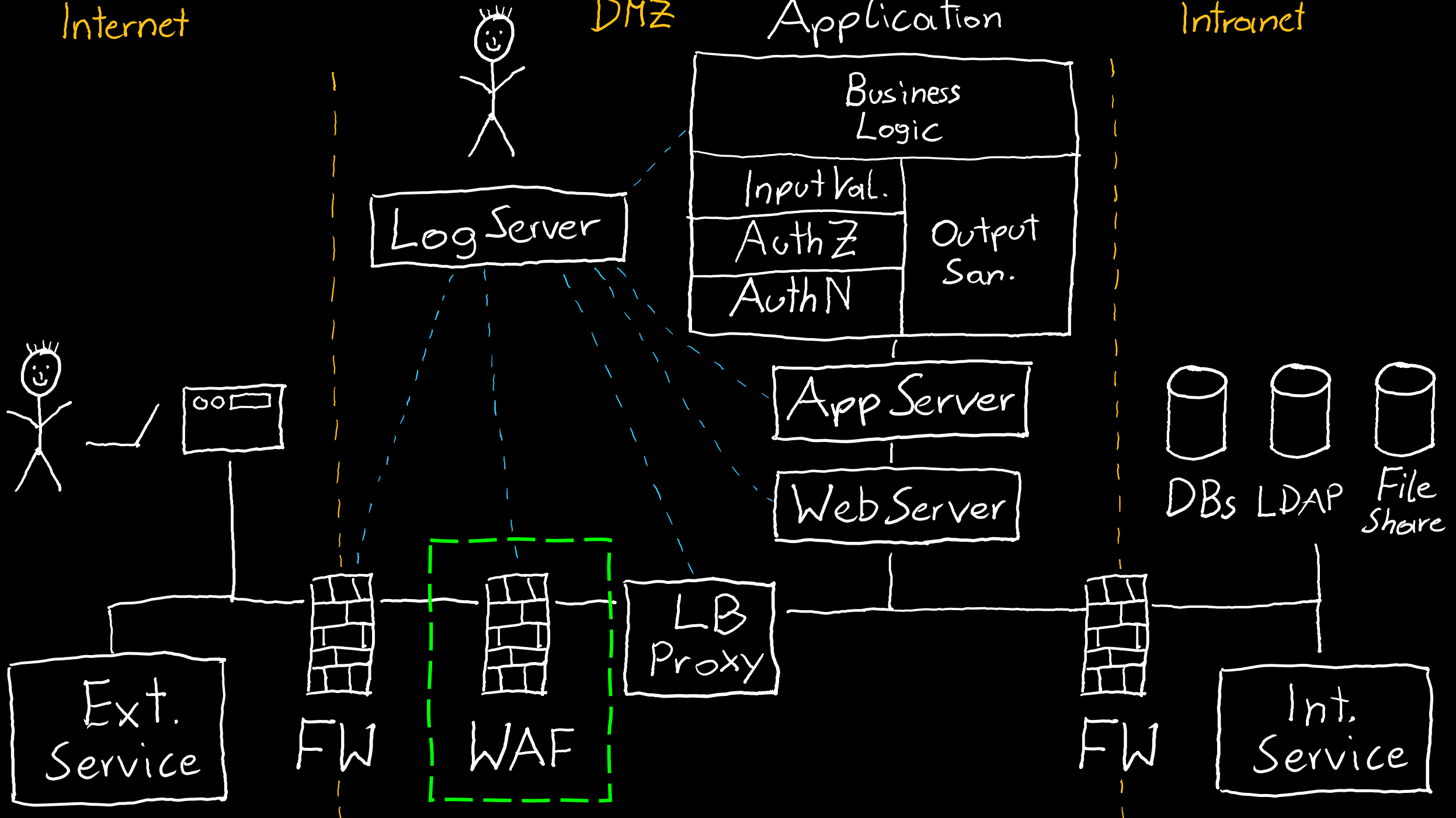


Internet

DMZ

Application

Intranet



# 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 (2<sup>nd</sup> 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

- A secure design is worth the money
- Explicitly ensure the integrity of your software (components) and your critical data
- 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