

# Keep IT Secure

Apoppopsosäkok 4.0

# Vilka är vi?



\$ whoami

**Namn:** Mohammed "apple" Belcaid

**Yrke:** IT-Säkerhetskonsult

**Fokus:** Web/Mobile Appsec, Reverse Engineering

\$ whoami

**Namn:** Dennis "menac3" Dubrefjord

**Yrke:** IT-Säkerhetskonsult

**Fokus:** Web Appsec, CTF, Automation



# Vilka är ni?

- Fullstacksutvecklaren
- Arkitekten
- Den nitiske testaren



# Kursens TRE hörnstenar

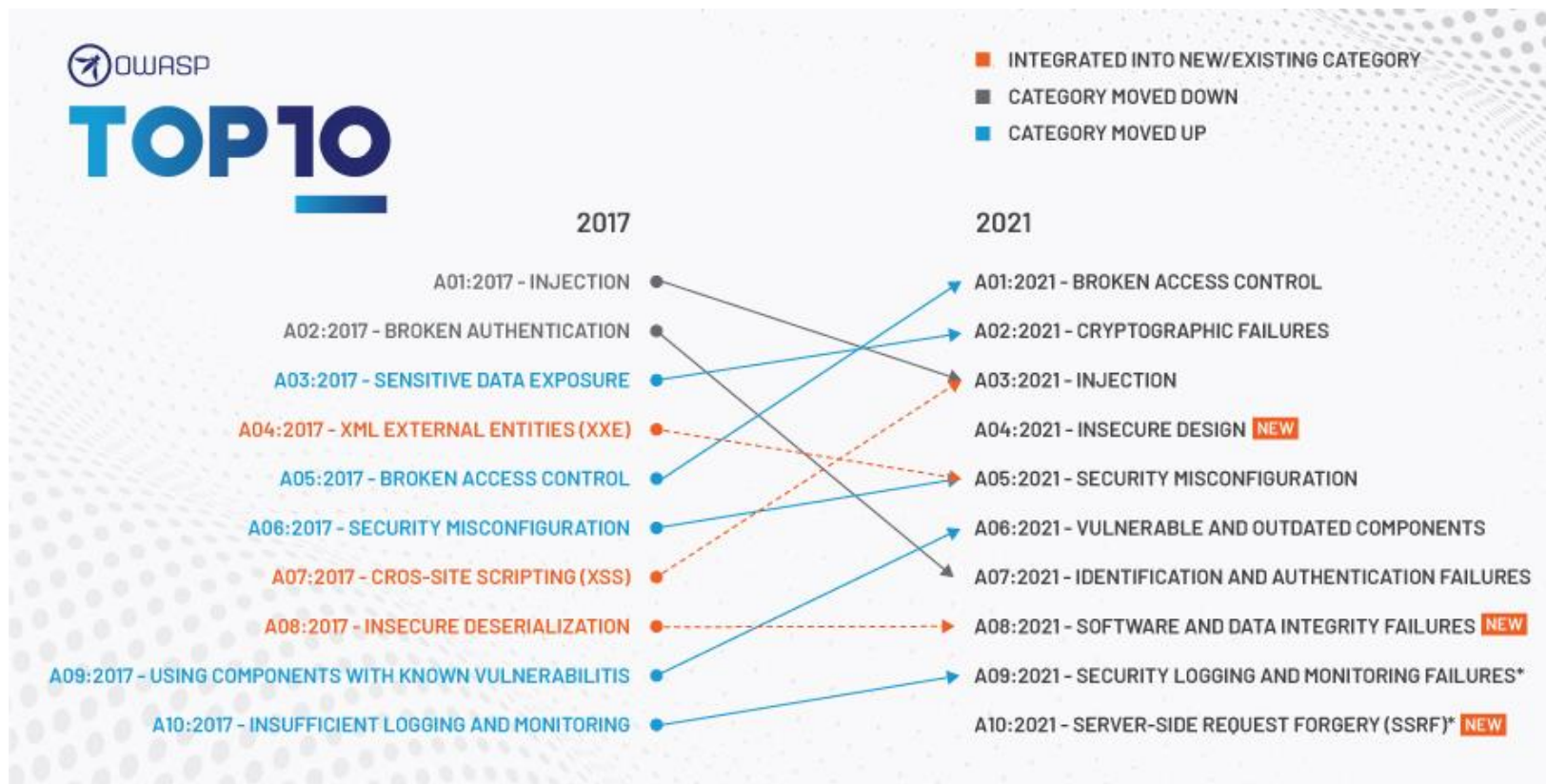


- 1 OWASP Top 10 2021
  - Vilka är de vanligaste sårbarheterna i webapplikationer
- 2 OWASP CheatSheet Series
  - Konkreta råd för appsäkerhet, riktat till utvecklare
- 3 Application Security Verification Standard 4.0
  - Hur kravställa och verifiera säkerhet

The Open Web Application Security Project (OWASP) is a worldwide free and open community focused on improving the security of application software.

# OWASP Top 10 2021 - Awareness

Lista på de 10 mest förekommande och kritiska sårbarheterna i web appar  
– Första lista 2003, 2021 ny lista



# Schema för dagen

Introduktion: 08:30

Intro (NU)

Spelplan

Förmiddag:

OWASP Web Top 10 08:45 – 10:00

[BREAK] 10:00 – 10:15

CSRF 10:15 – 11:10

[BREAK] 11:10 – 11:25

SSRF 11:25 – 12:00

Eftermiddag:

SQL Injektion 13:00 – 13:40

[BREAK] 13:40 – 13:55

XSS 13:55 – 14:45

[FIKA] 14:45 – 15:10

File Inclusion 15:10 – 15:40

File Upload 15:40 – 16:10

OWASP Cheatsheet & ASVS 16:10 – 16:20

Avslut 16:20 – 16:30

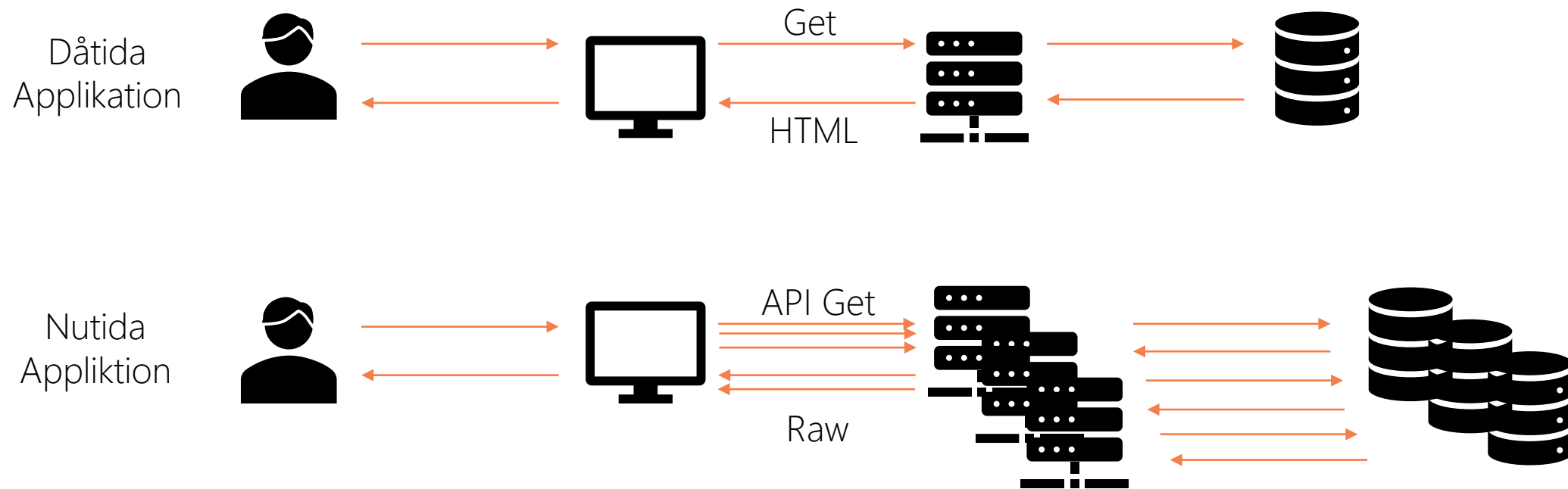
# Feedback

- Vilka förväntningar har ni?
- Feedback - vad kan vi förbättra?
- Vårt mål

# OWASP Web TOP 10



# Dåtid vs. Nutid

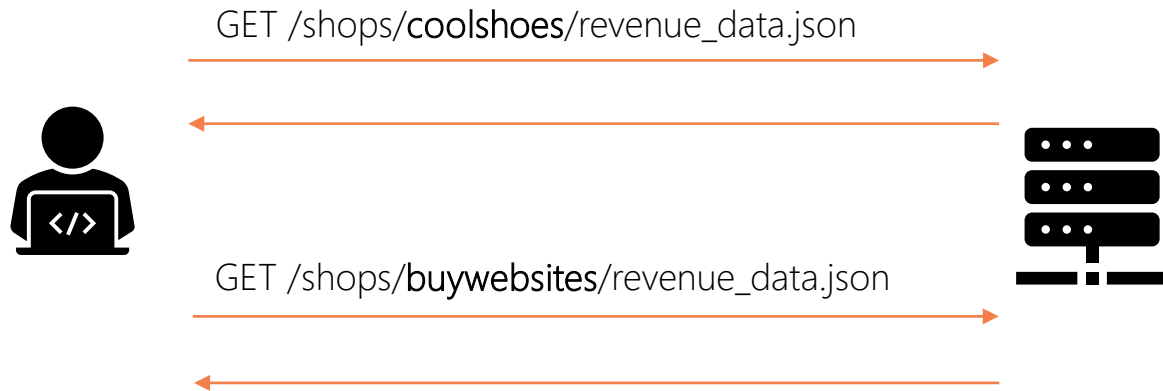


# OWASP Web Security TOP 10


- A1: Broken Access Control
- A2: Cryptographic Failure
- A3: Injection
- A4: Insecure Design
- A5: Security Misconfiguration
- A6: Vulnerable and Outdated Components
- A7: Identification and Authentication Failures
- A8: Software and Data Integrity Failures
- A9: Security Logging and Monitoring Failures
- A10: Server Side Request Forgery (SSRF)




# A1: Broken Access Control





- Användare har tillgång till saker som de inte borde
  - Admin-endpoints
  - Andra användares resurser

 Ayoub FATHI (ayoubfathi)


2500	-	5.06	89th	17.55	88th
Reputation	Rank	Signal	Percentile	Impact	Percentile

2  **IDOR on API endpoint leads to a leak of +12000 Shopify stores Sales and traffic data**

State  Resolved (Closed)

Severity  High (7 ~ 8.9)

Reported To [Shopify](#)

Participants 

# A1: Broken Access Control



## Encoded

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG91IiwiaWF0IjoxNTE2MzkwMjQ0fQ.SflKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

## Decoded

### HEADER:

```
{  "alg": "HS256",  "typ": "JWT"}
```

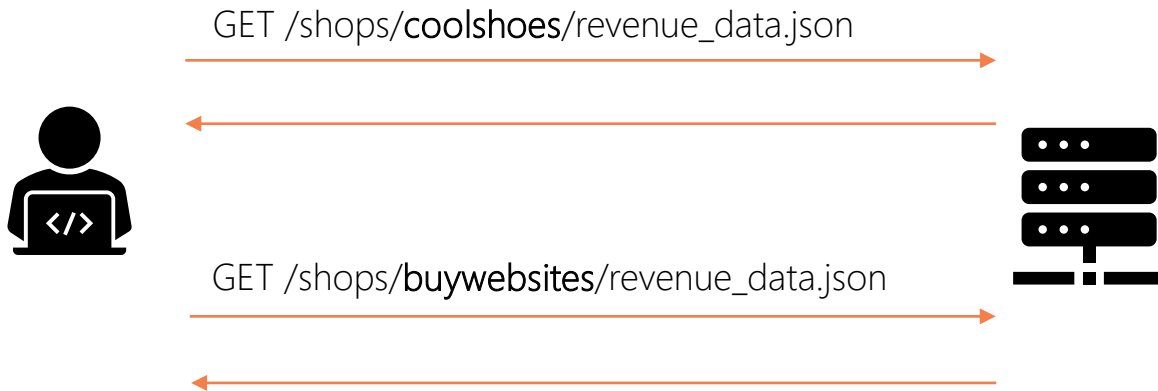
### PAYLOAD:


```
{  "sub": "1234567890",  "name": "John Doe",  "iat": 1516239022}
```

### VERIFY SIGNATURE


- Användare har tillgång till saker som de inte borde
  - Admin-endpoints
  - Andra användares resurser
- Användare kan manipulera sin roll
  - Gissa sessionsID
  - Manipulera session tokens


# A1: Broken Access Control




 Ayoub FATHI (ayoubfathi)


2500	-	5.06	89th	17.55	88th
Reputation	Rank	Signal	Percentile	Impact	Percentile

2  **IDOR on API endpoint leads to a leak of +12000 Shopify stores Sales and traffic data**

State  Resolved (Closed)

Reported To [Shopify](#)

Severity  High (7 ~ 8.9)

Participants 

## Mitigering

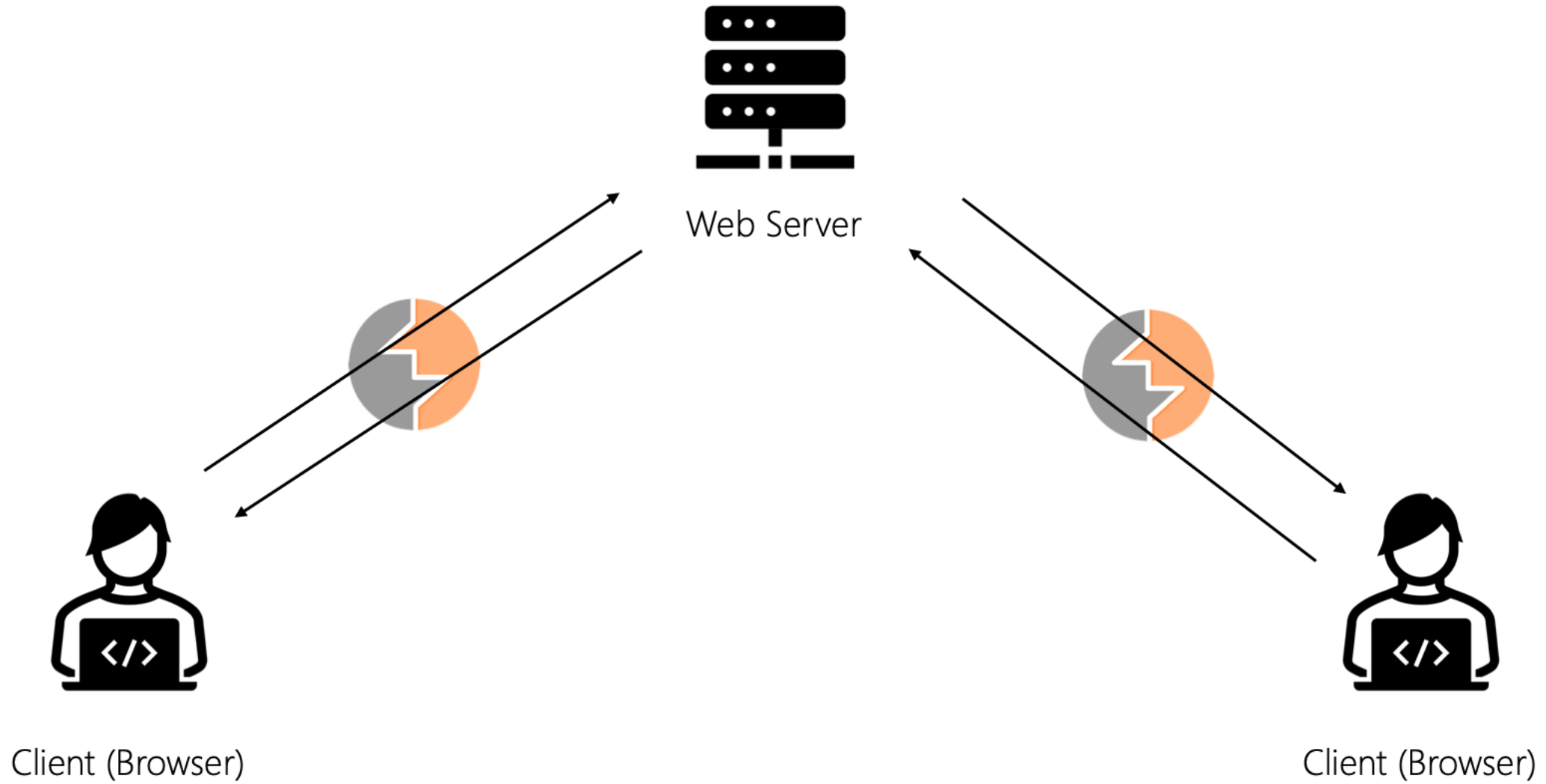
- Auktorisera användarens förfrågningar, neka by default
- Se till att auktoriseringskod går att granska, och granska den
- Använd slumpmässiga session ID som inte går att gissa

[cheatsheetseries.owasp.org/cheatsheets/...](https://cheatsheetseries.owasp.org/cheatsheets/...)

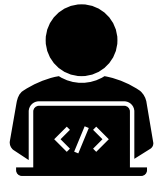
- Session\_Management\_Cheat\_Sheet.html
- JSON\_Web\_Token\_for\_Java\_Cheat\_Sheet.html

**DEMO**

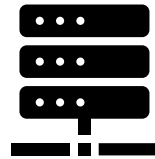
# Burp Suite



# A2: Cryptographic Failures



Pentesting/Code Review



```
public static final String DEFAULT_USER = "admin@website.com";  
public static final Hash.Algorithm MD5 DEFAULT_USER_PWD =  
    "0192023a7bbd73250516f069df18b500";  
public static final String PARTNER_ADMIN =  
    "partner+admin@website.com";  
public static final Hash.Algorithm MD5 PARTNER_ADMIN_PWD =  
    "482c811da5d5b4bc6d497ffa98491e38";
```

0192023a7bbd73250516f069df18b500 -> admin123

482c811da5d5b4bc6d497ffa98491e38 -> password123

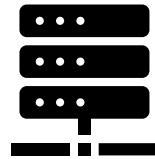
- Tidigare "Sensitive Data Exposure"
- Känslig data skickas eller lagras som klartext
- Utdaterade eller sårbara kryptofunktioner används
- Dåliga kryptonycklar används



# A2: Cryptographic Failures



Pentesting/Code Review



```
public static final String DEFAULT_USER = "admin@website.com";  
public static final Hash.Algorithm MD5 DEFAULT_USER_PWD =  
    "0192023a7bbd73250516f069df18b500";  
public static final String PARTNER_ADMIN =  
    "partner+admin@website.com";  
public static final Hash.Algorithm MD5 PARTNER_ADMIN_PWD =  
    "482c811da5d5b4bc6d497ffa98491e38";
```

0192023a7bbd73250516f069df18b500 -> admin123

482c811da5d5b4bc6d497ffa98491e38 -> password123

## Mitigering

- Kryptera känslig lagrad data, lagra bara det som behövs
- Lagra lösenord hashade med tex bcrypt, Argon2.
- Använd moderna kryptofunktioner med bra nycklar
- Kryptera data som skickas
  - Använd TLS 1.2 eller högre
  - Sätt response headern HSTS


[cheatsheetseries.owasp.org/cheatsheets/...](https://cheatsheetseries.owasp.org/cheatsheets/...)






- User\_Privacy\_Protection\_Cheat\_Sheet.html
- Transport\_Layer\_Protection\_Cheat\_Sheet.html
- Password\_Storage\_Cheat\_Sheet.html
- HTTP\_Strict\_Transport\_Security\_Cheat\_Sheet.html

# A3: Injection

Data tolkas som kod:

- SQL Injection
- NoSQL
- Cross-site scripting (XSS)
- OS

 spaceraccoon (spaceraccoon) 11621 Reputation 34th Rank 6.98 Signal 95th Percentile 19.75 Impact 91st Percentile

702 #531051 **SQL Injection Extracts Starbucks Enterprise Accounting, Financial, Payroll Database** Share:     

State ● Resolved (Closed)

Disclosed **August 6, 2019 7:51am +0200**




Reported To **Starbucks**

Asset Other non domain specific items (Other)

Weakness SQL Injection

Bounty **\$4,000**

Severity Critical (9.3)

Participants   

Visibility Disclosed (Limited)

[Collapse](#)

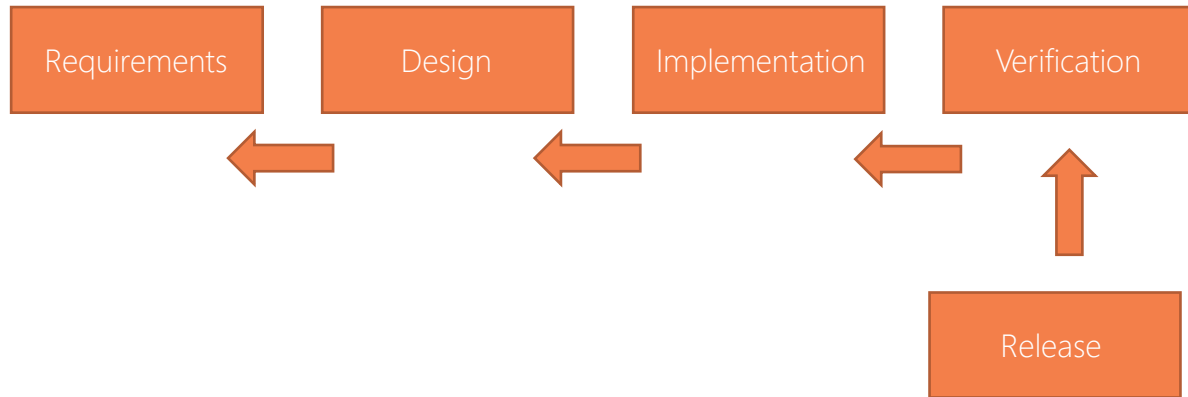
## Mitigering

- Validera indata
  - Definiera data typer och strängmönster som tillåts
- Använd ramverk som förhindrar SQL, NoSQL injektioner
- Sanitera indata
- Lita inte blint på s.k "betrodda" interna system

[cheatsheetseries.owasp.org/cheatsheets/...](https://cheatsheetseries.owasp.org/cheatsheets/...)

- [Injection\\_Prevention\\_Cheat\\_Sheet.html](#)

# A4: Insecure Design



- Ny kategori
- Förespråkar shift-left:
  - En osäker design kan inte fixas genom en bra implementation

## Hur gör vi det?

- Hotmodellering
- Secure Development Lifecycle
- *Threat modelling manifesto*

# A5: Security Misconfiguration

- Onödiga features aktiverade
- Interna resurser exponerade (.git, .bash\_history)
- Säkerhetsheaders saknas (CORS, CSP)
- Felmeddelanden som innehåller känslig information
- Applikation körs i debug mode

## Mitigering

- DevOps-process som sätter upp servrar och applikationer på ett strukturerat sätt
- Identifiera och verifiera att känsliga filer inte är exponerade i produktion (Kan automatiseras)
- Sätt säkerhetsheaders
- Returnera inte detaljerade felmeddelanden (stack traces)

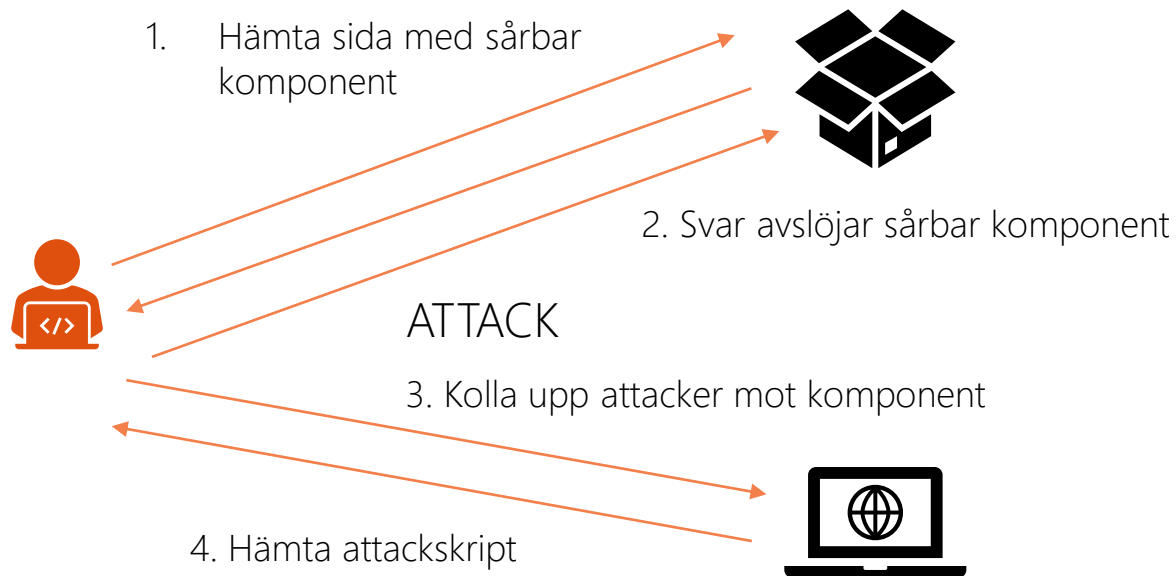
*[cheatsheetseries.owasp.org/cheatsheets/...](https://cheatsheetseries.owasp.org/cheatsheets/)*

- [HTTP-Headers-Cheat-Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/HTTP-Headers-Cheat-Sheet.html)
- [Content-Security-Policy-Cheat-Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Content-Security-Policy-Cheat-Sheet.html)

Owasp ASVS v14: Configuration

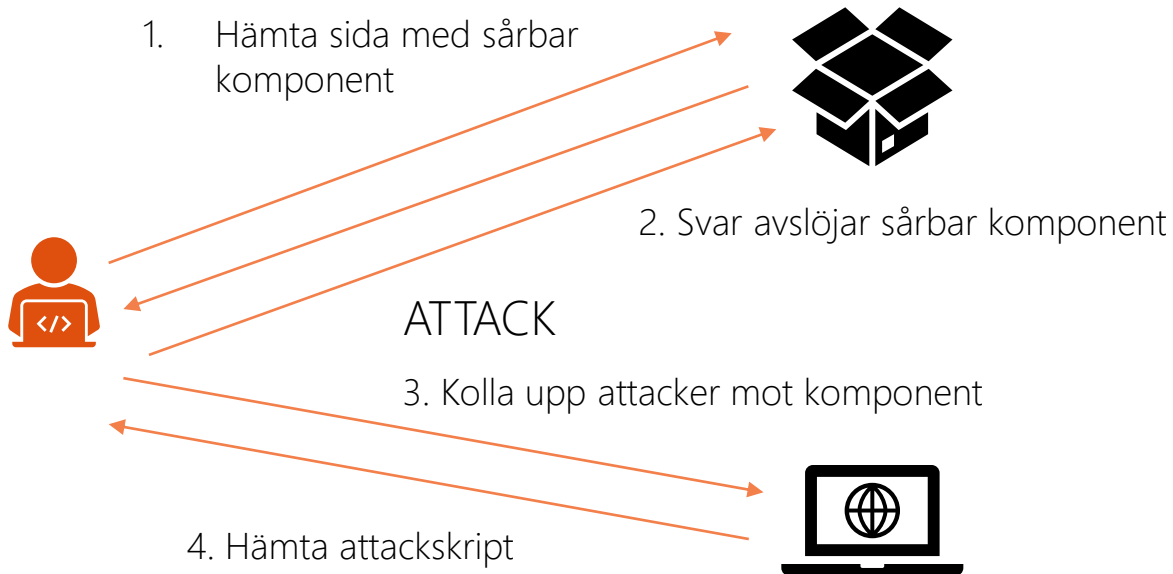
**DEMO**

# A6: Vulnerable and Outdated Components



- Mjukvarukomponenter kan vara sårbara eller utdaterade
  - Apache Log4j (Log4Shell) sårbarheter
  - Spring4Shell sårbarheter
- Både komponenter på frontend och backend kan vara sårbara

# A6: Vulnerable and Outdated Components



## Mitigering

- Ta bort dependencies som inte används
- Skapa en rutin för regelbunden patchning av mjukvara
- Använd Software Composition Analysis (SCA)-verktyg
  - OWASP Dependency-Check

[cheatsheetseries.owasp.org/cheatsheets/...](https://cheatsheetseries.owasp.org/cheatsheets/...)

- [Vulnerable\\_Dependency\\_Management\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Vulnerable_Dependency_Management_Cheat_Sheet.html)

# A7: Identification and Authentication Failures



- Tidigare "Broken Authentication"
- Credential stuffing / Brute Force
- Multi-factor authentication saknas
- Svaga lösenord accepteras
- Ingen CAPTCHA / account logout
- Skickar känslig data som auth tokens i GET parametrar
- Osäker metod för glömt lösenord-funktion
- Samma session ID innan och efter inloggning

## Mitigering

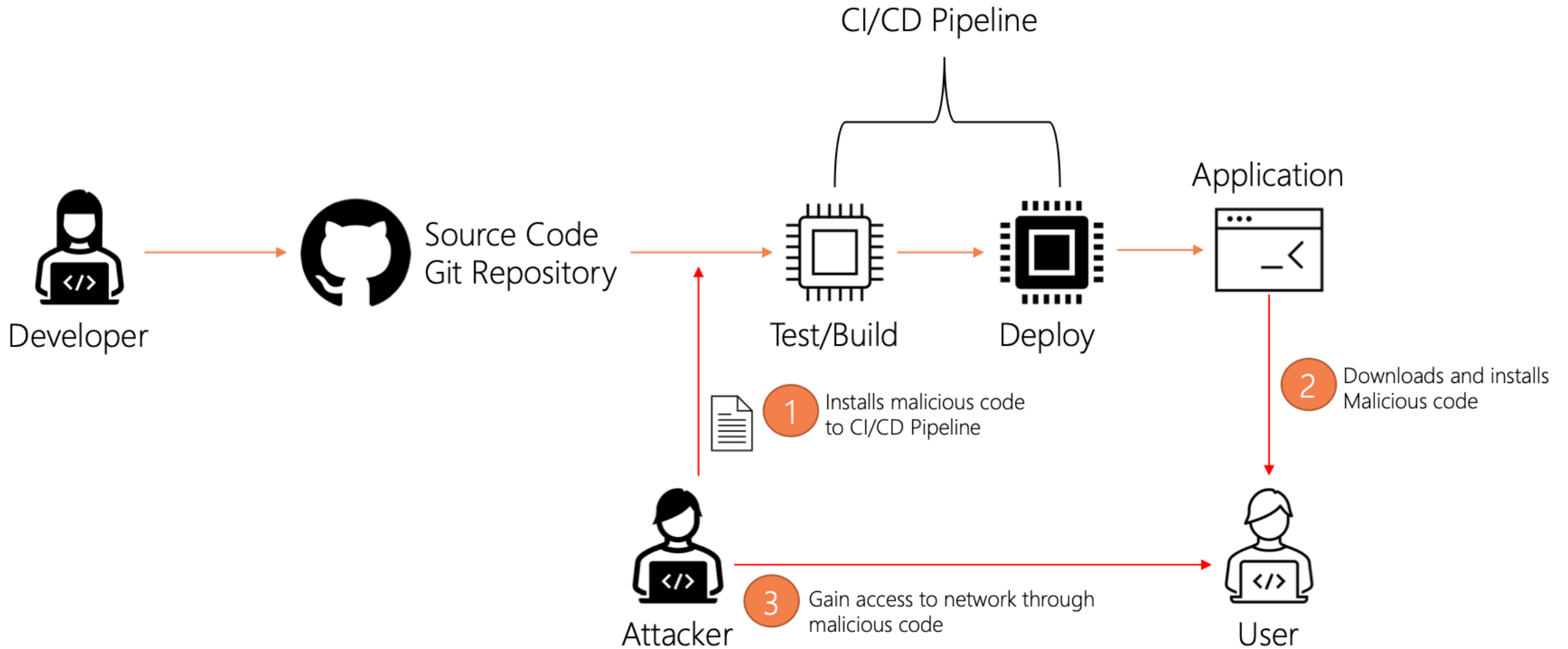
- Testa alla autentiseringsflöden
- Rate limiting och låsning av konto för att stoppa brute force
- Använd standard autentiseringsmekanismer, bygg ej egna
- 2FA / MFA
- Ändra defaultuppgifter för adminkonto
- Tvinga användare att välja komplexa lösenord

[cheatsheetseries.owasp.org/cheatsheets/...](https://cheatsheetseries.owasp.org/cheatsheets/...)

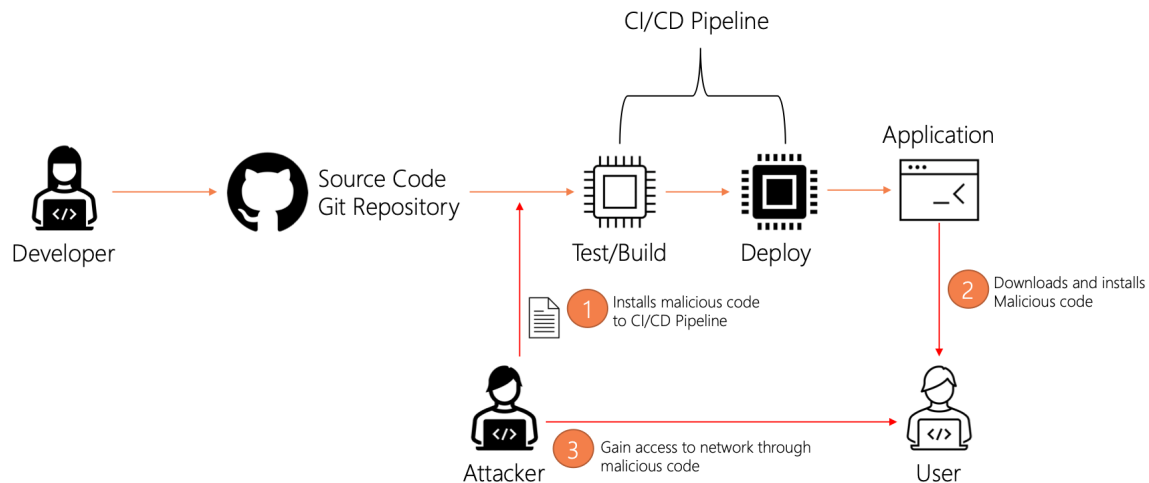
- Authentication\_Cheat\_Sheet.html



# A8: Software and Data Integrity Failures



# A8: Software and Data Integrity Failures



- Ny kategori
- Integritetsproblem som orsakar säkerhetshål
  - SolarWinds 2020 Attack
  - Object deserialization

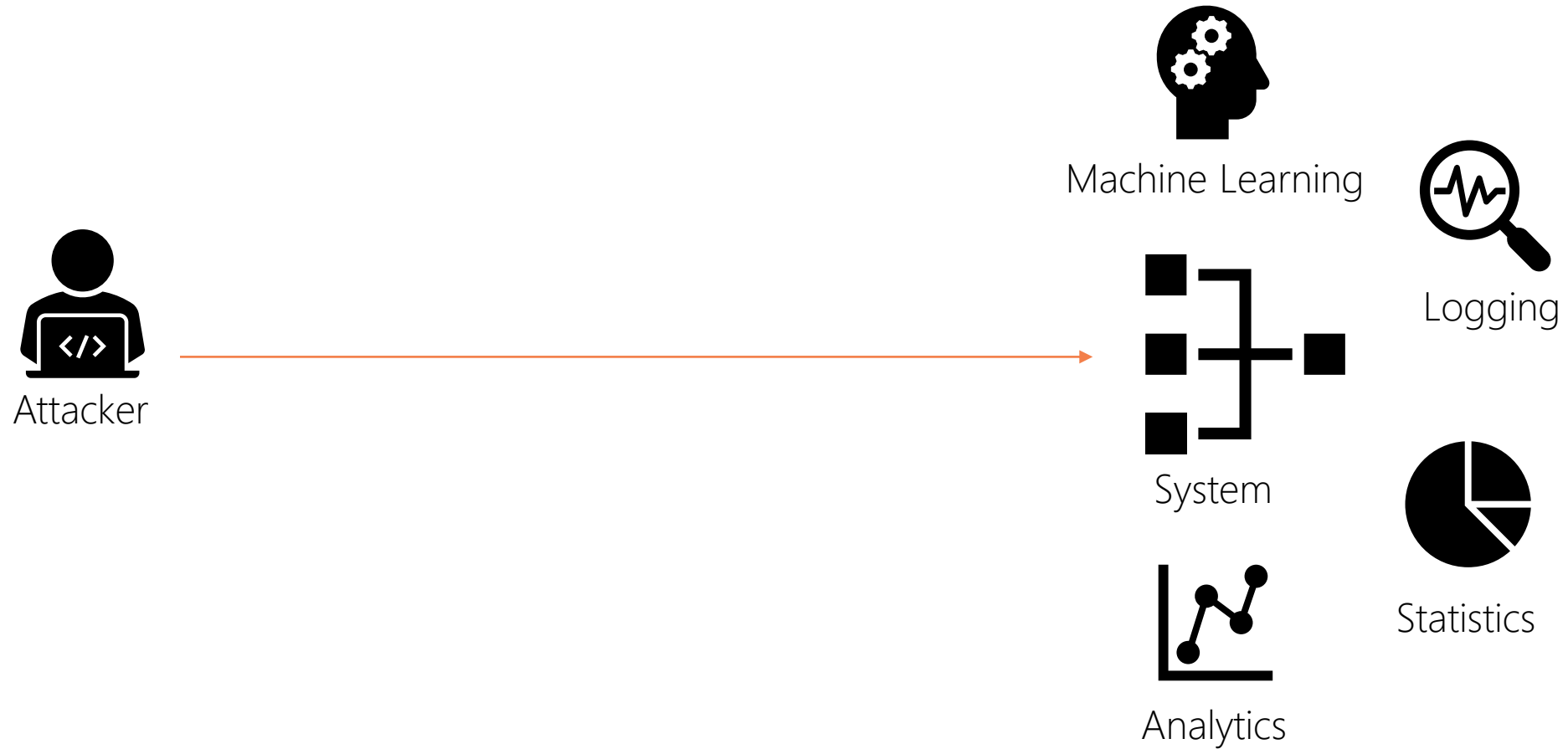
## Mitigering

- Använd digitala signaturer för att verifiera mjukvara
- Hämta kod från pålitliga källor
- Validera innehåll i serialiserade objekt innan de deserialiseras

[cheatsheetseries.owasp.org/cheatsheets/...](https://cheatsheetseries.owasp.org/cheatsheets/...)

- Infrastructure\_as\_Code\_Security\_Cheat\_Sheet.html
- Deserialization\_Cheat\_Sheet.html
- Software Supply Chain Security (Coming Soon)
- Secure build and deployment (Coming Soon)

# A9: Security Logging & Monitoring Failures



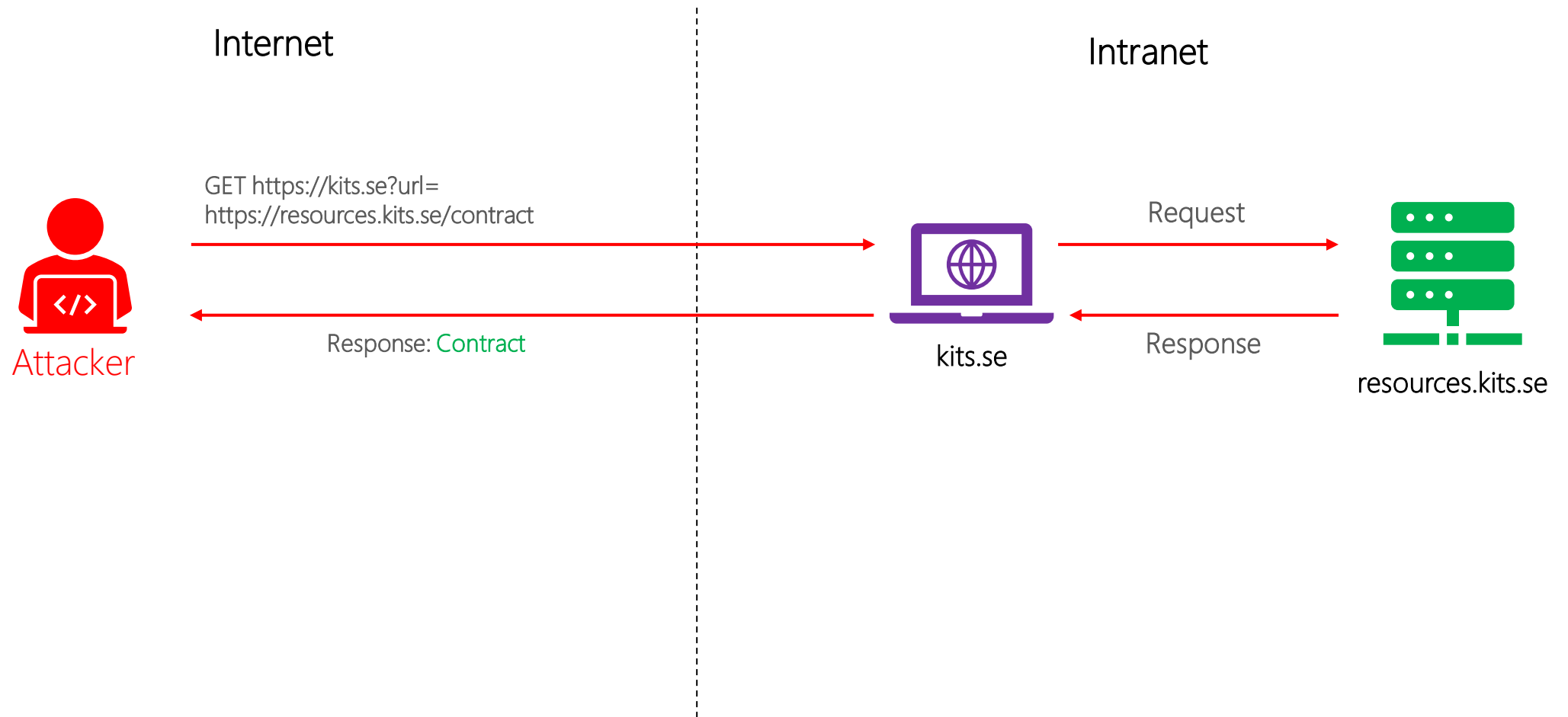
# A9: Security Logging & Monitoring Failures

- Logga misslyckade loginförsök, nekad åtkomst, misslyckad validering av indata, och säkerhetsmekanismer som misslyckas
- Se till att loggar är formaterade på ett sätt som är läsbart av andra verktyg
- Loggar ska anses som känslig data och skyddas därefter
- Inkludera information som skulle kunna spåra ett angrepp
- Undvik att inkludera känslig data i loggarna. Om det ej går, anonymisera
- Integrera med SIEM, monitorerings tjänster, visuallisera på en dashboard

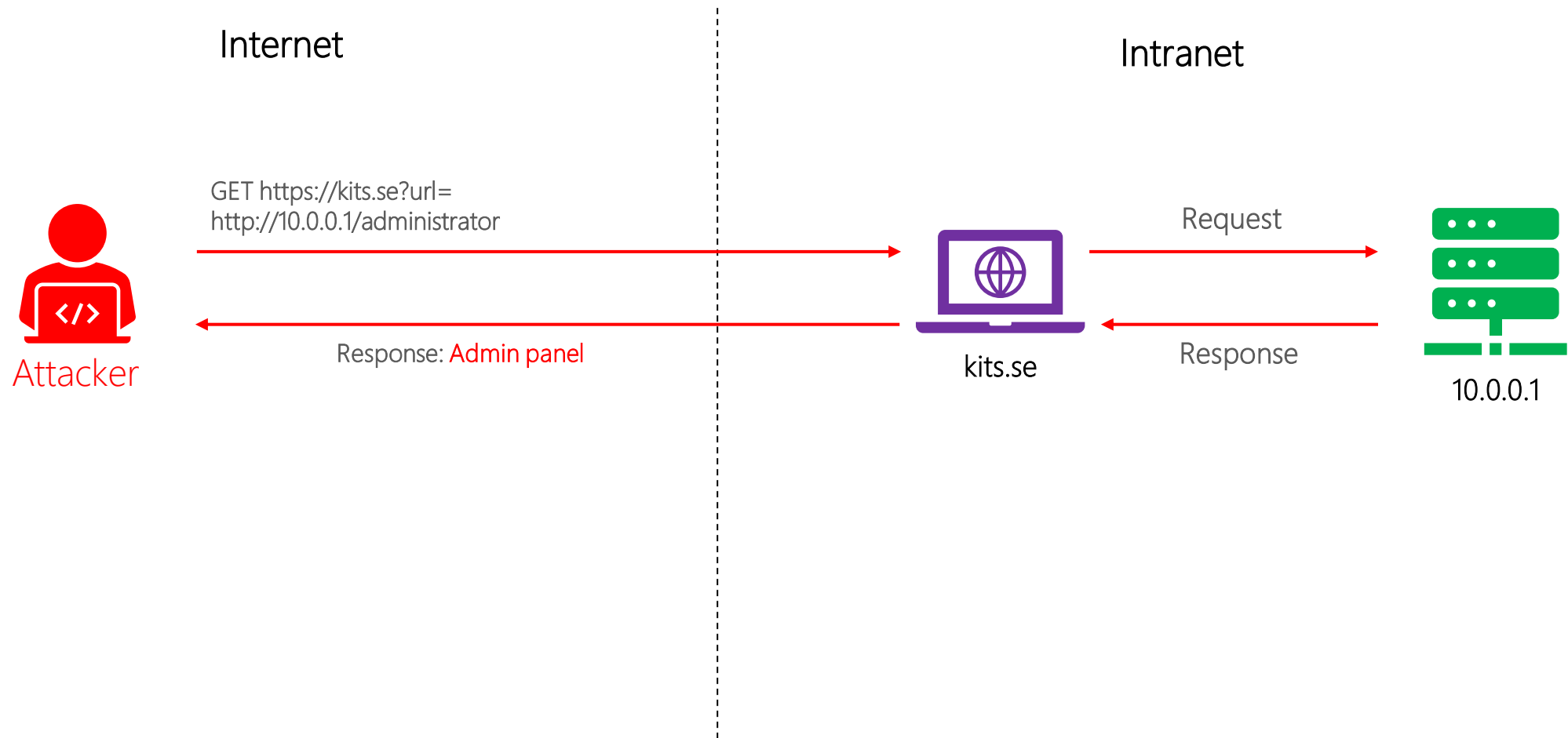
*[cheatsheetseries.owasp.org/cheatsheets/...](https://cheatsheetseries.owasp.org/cheatsheets/...)*

- [Logging\\_Cheat\\_Sheet.html](#)
- [Logging\\_Vocabulary\\_Cheat\\_Sheet.html](#)

# A10: Server Side Request Forgery (SSRF)



# A10: Server Side Request Forgery (SSRF)



The background is a solid olive green color. It features several white geometric shapes: a large vertical rectangle on the left, a horizontal rectangle in the top right, a vertical rectangle in the bottom right, and a parallelogram in the bottom center. A thin orange horizontal bar is at the very top, and a thin orange vertical bar is on the far left. The word "Kahoot!" is written in white text in the center of the olive green area.

Kahoot!

# Cross Site Request Forgery

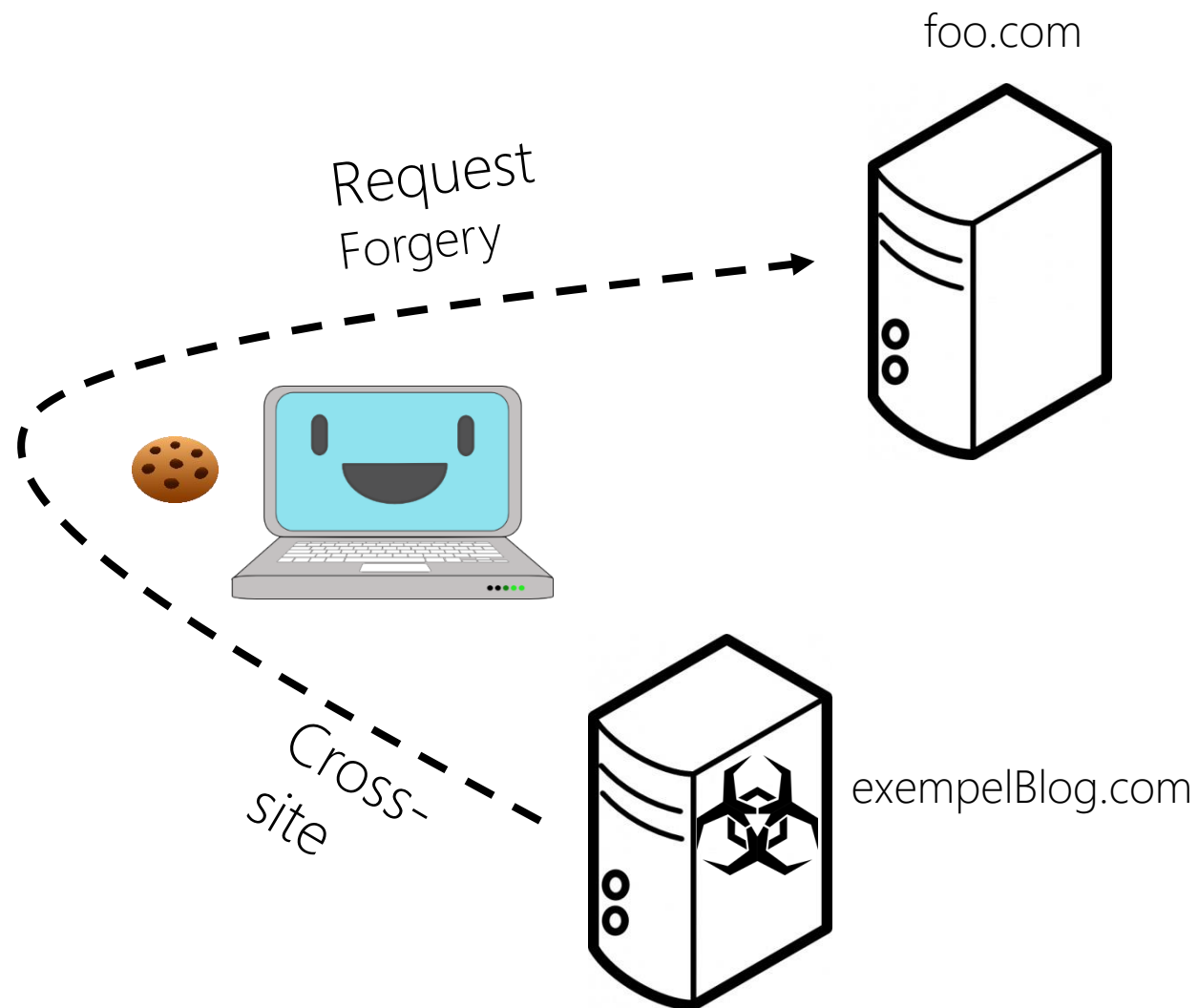


# CSRF

Cross-Site Request Forgery (CSRF) är en attack

... som lurar en slutanvändare att utföra oönskade åtgärder

... på en webbapplikation där de för närvarande är autentiserade.



The background is a solid teal color. There are three white rectangular shapes: a horizontal rectangle on the left, a vertical rectangle in the center, and a horizontal rectangle at the bottom. A thin orange horizontal bar is at the top, and a thin orange vertical bar is on the right.

SOP/CORS

# Same-Origin Policy

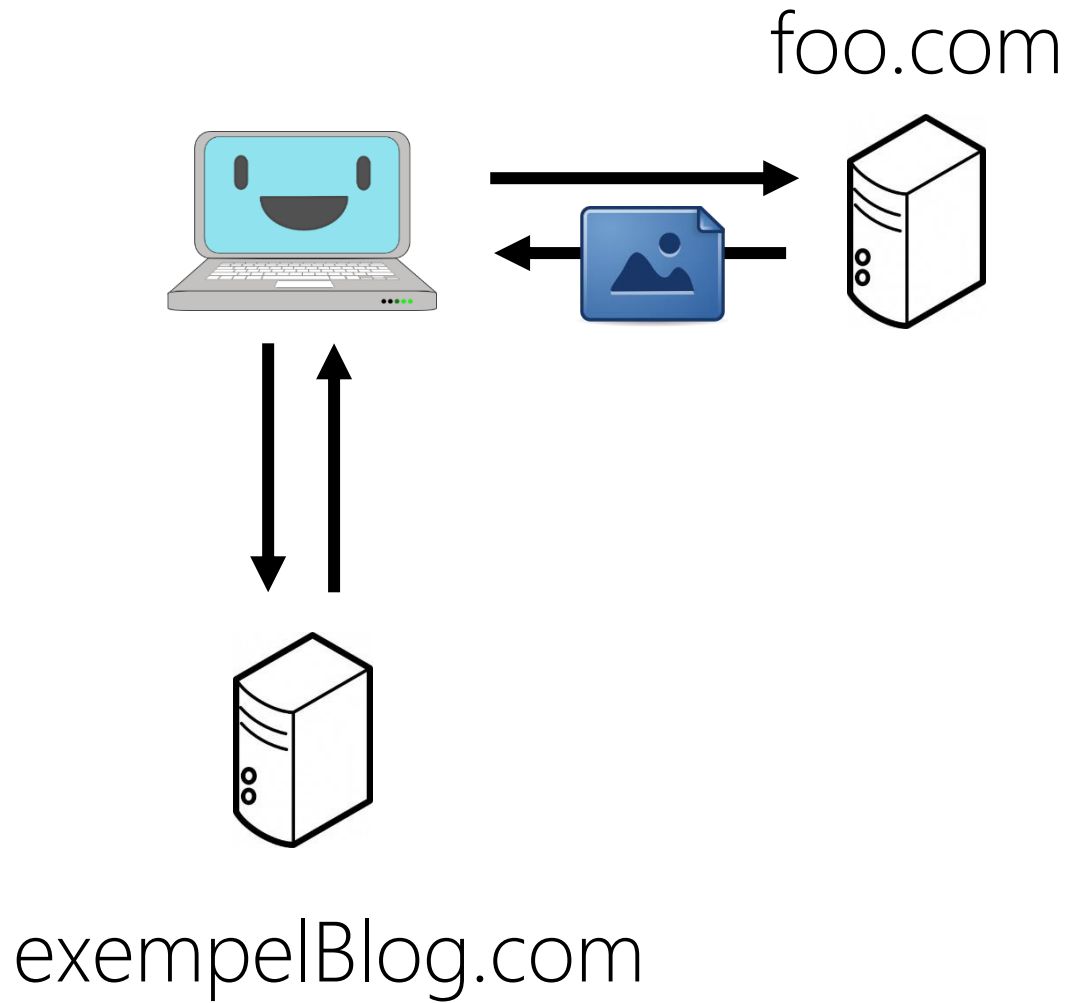
- Förhindra att data läcker mellan två olika domäner
- Implementeras i browsern
- Tillåter att man hämtar vissa resurser

[https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin\\_policy](https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy)

# Cross-Origin Resource Sharing

- Gör SOP mindre strikt
- Header: `Access-Control-Allow-Origin:` <https://example.com>

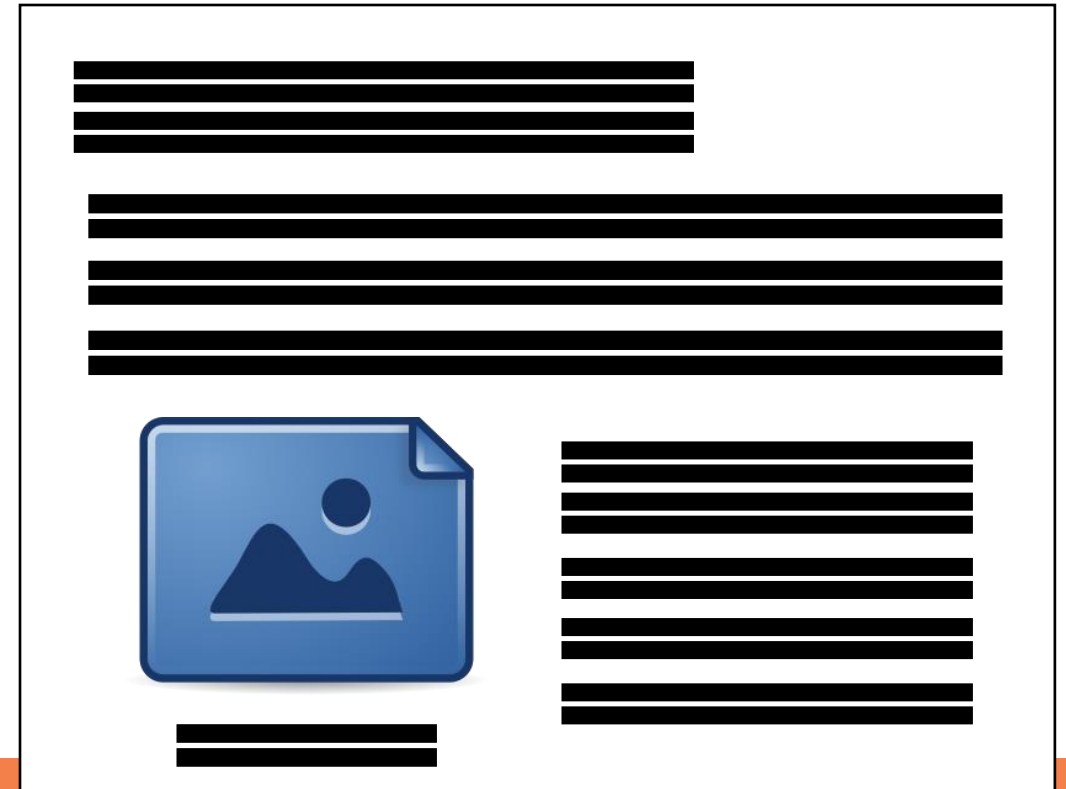
# SOP/CORS



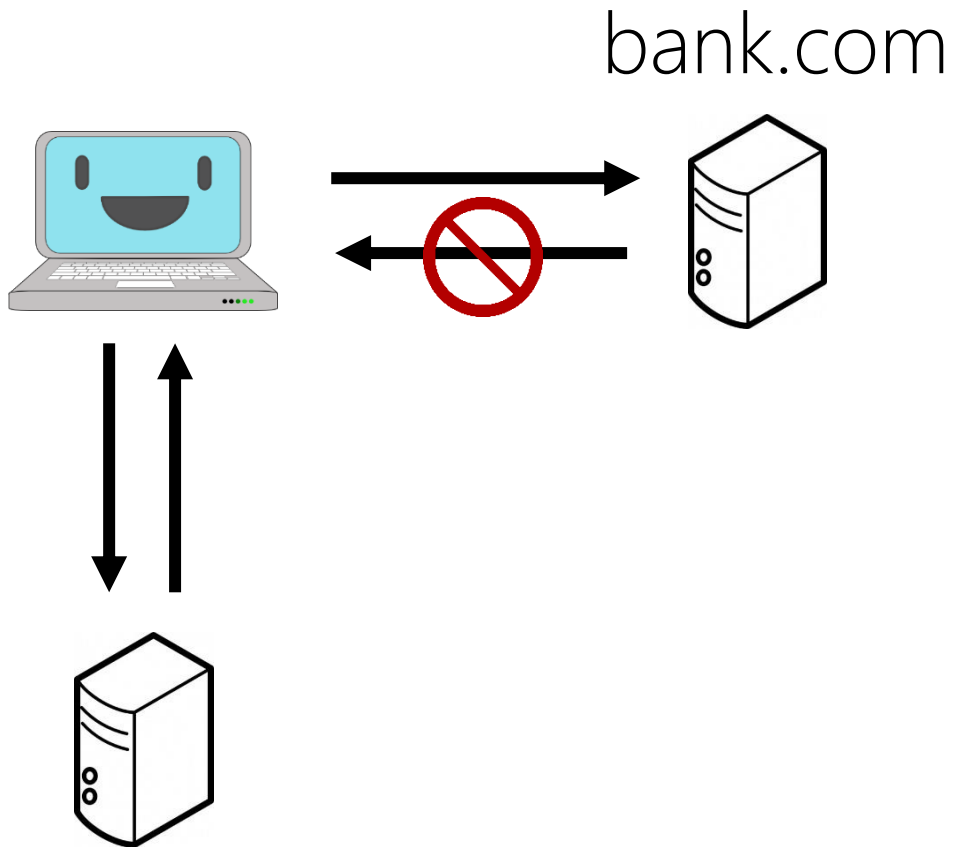
****

**<HTML>**

```
<html>
<title>HTML</title>
<body>
This is HTML!
</body>
</html>
```



# SOP/CORS



exempelBlog.com

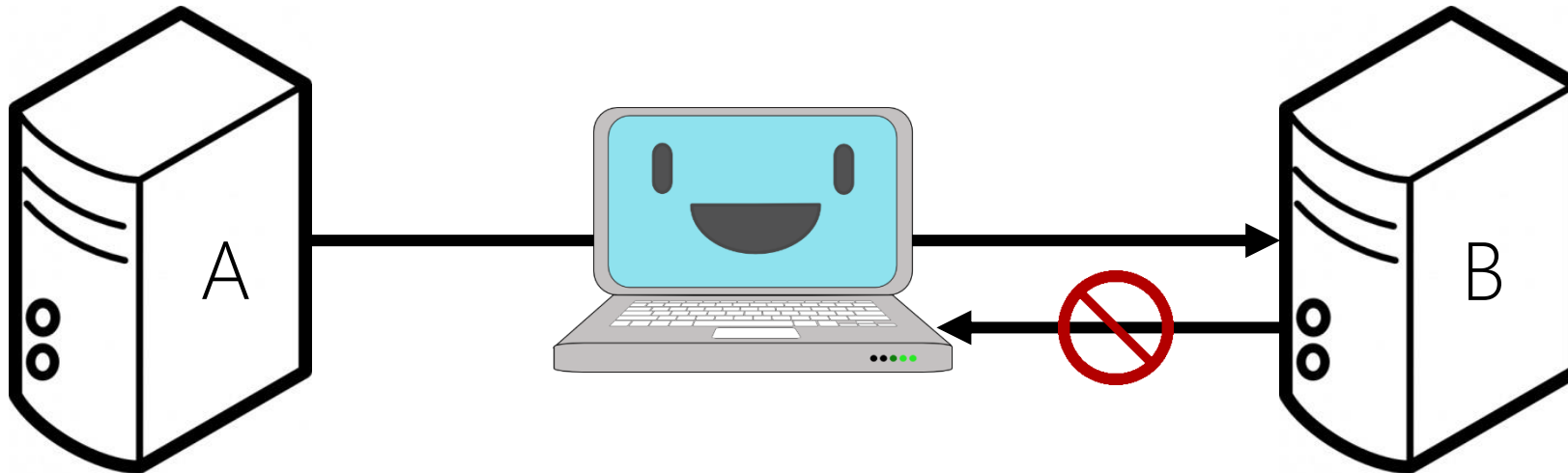
Kontosaldo: ?

`$.get("bank.com/saldo")`



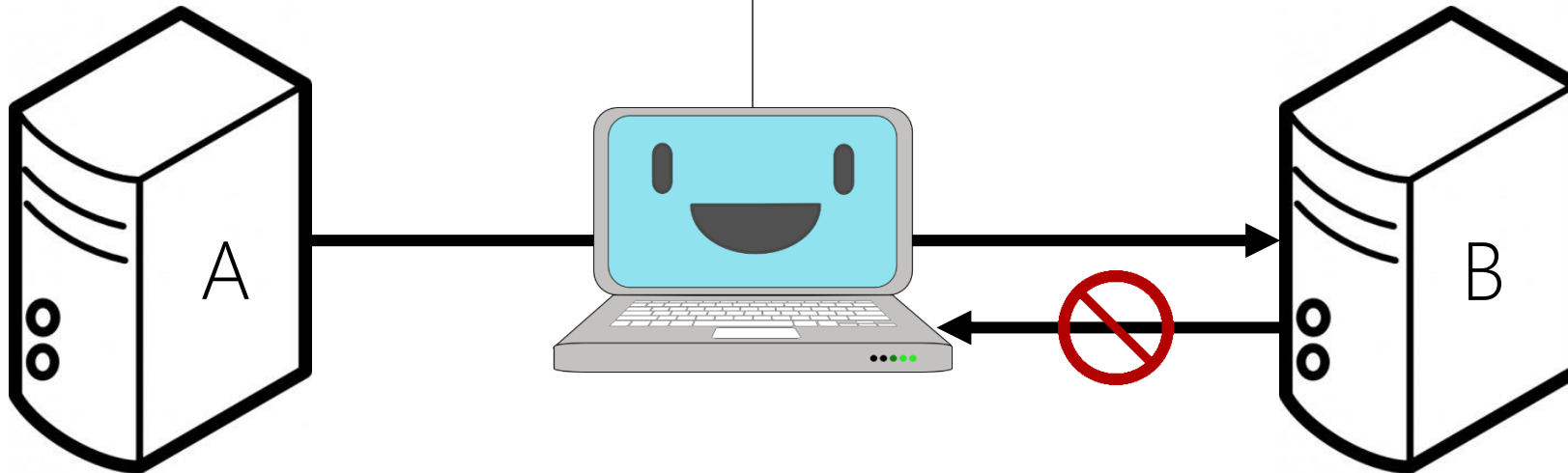
# Vad säger browsern?

XMLHttpRequest cannot load http://bank.com/saldo/. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'https://exempelBlog.com' is therefore not allowed access.



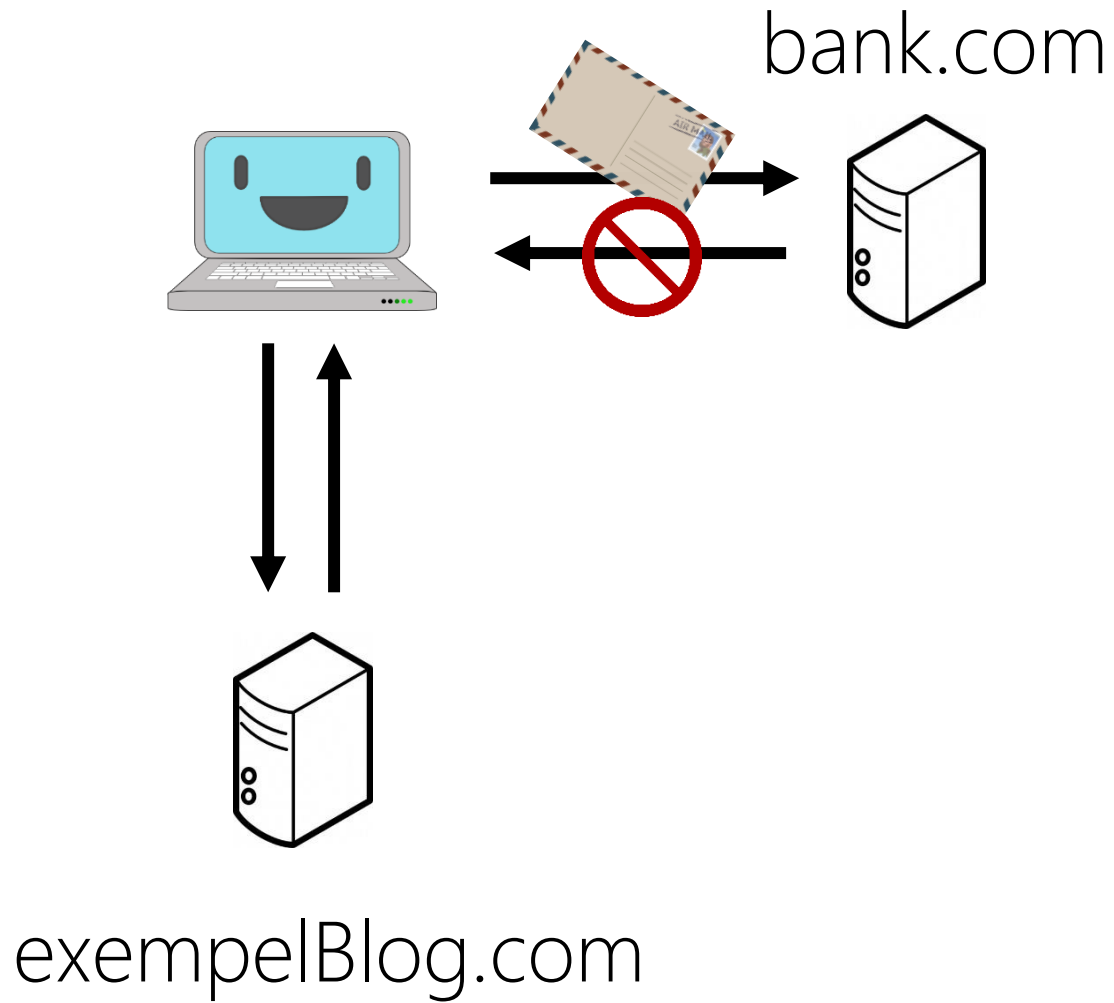
## Vad säger browsern?

XMLHttpRequest cannot load http://B/some\_resource/. No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'https://A' is therefore not allowed access.

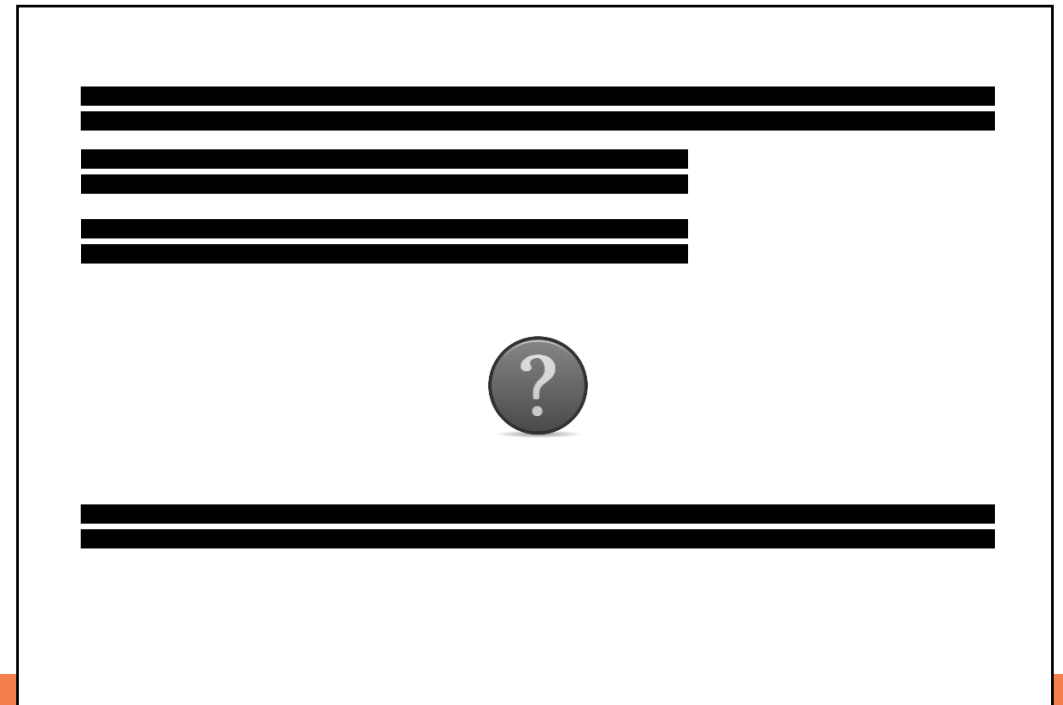




# SOP/CORS



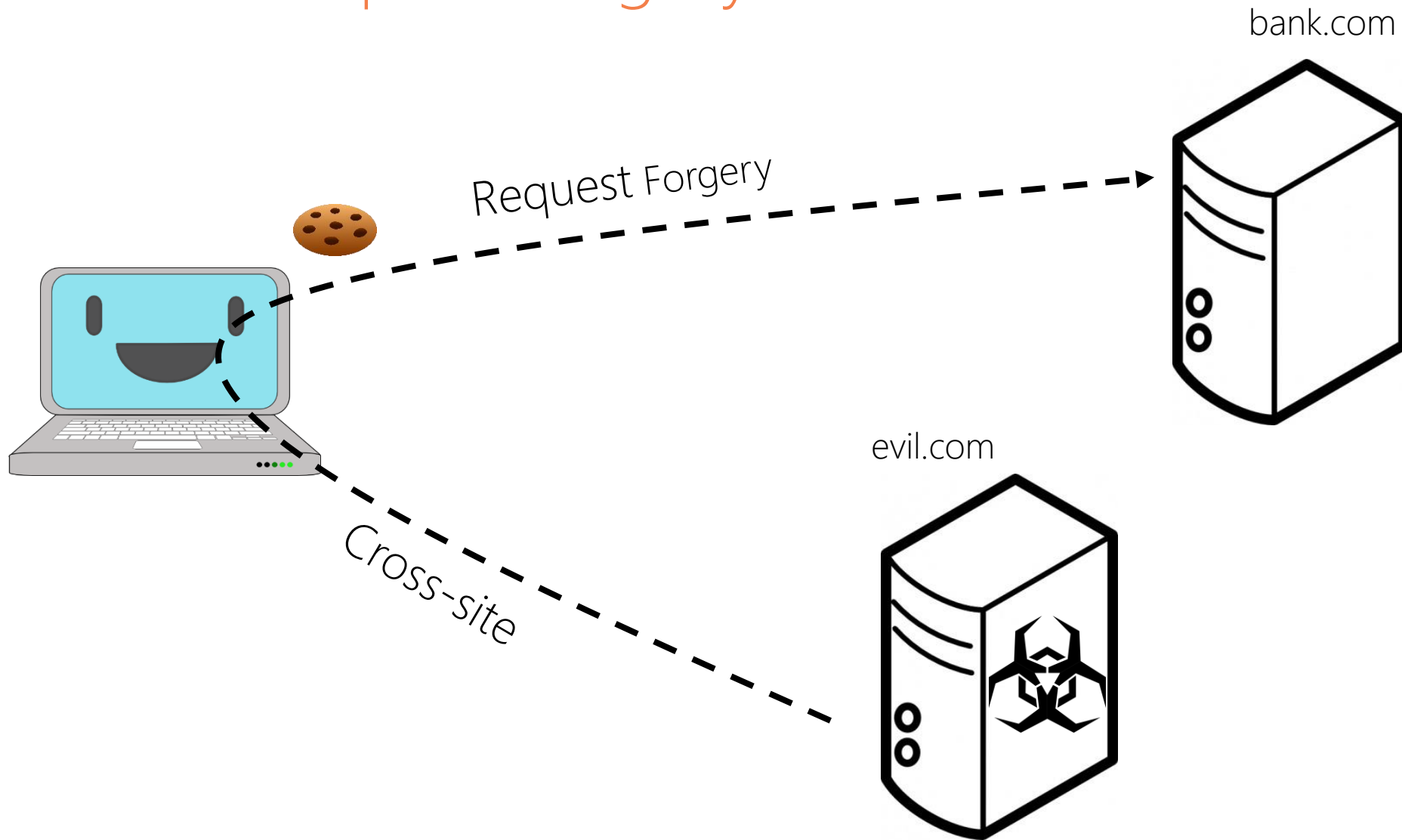
`$.put("bank.com/transfer")`



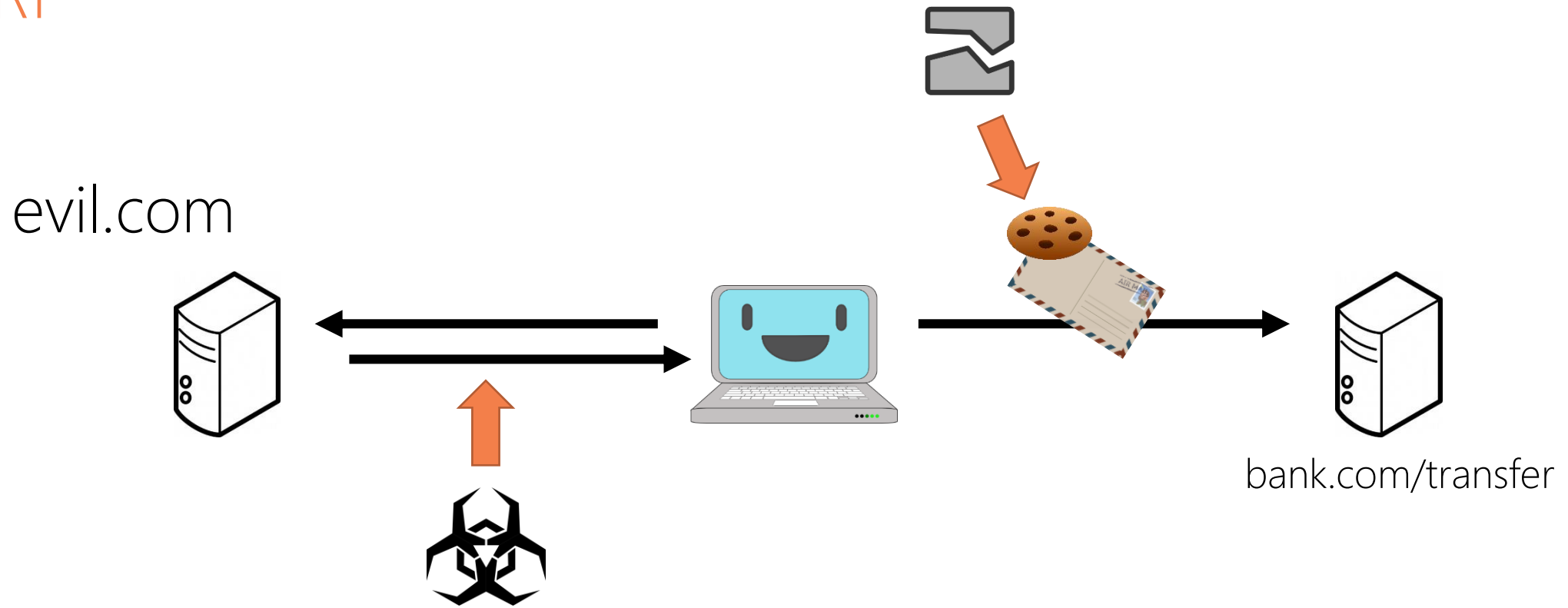
The background is a solid olive green color. It features several white geometric shapes: a large vertical rectangle on the left, a horizontal rectangle in the top right, a trapezoid in the bottom center, and a vertical rectangle on the bottom right. A thin orange horizontal bar is at the very top, and a thin orange vertical bar is on the far left. The text 'CSRF' is centered in the olive green area.

CSRF

# Cross-Site Request Forgery



# CSRF



# CSRF

```
<html>
<body>



</body>
</html>
```

```
<iframe style="width:0; height:0; border:0;
border:none">
<form action="https://bank.com/api/Transfer"
method="POST" id="form1">
<p>Account Number: <input type="text"
name="accNumber" value="532-58962"/></p>
<p>Transfer Amt: <input type="text" name="amount"
value="1000000"/></p>
<p><input type="submit" value="Submit"></form>
<script>document.getElementById("form1").submit();
</script>
</iframe>
```

The image features a solid olive green background. It is framed by a thin orange border at the top and bottom. On the left side, there is a large, vertical white rectangle. To its right, the word "Demo" is written in a white, sans-serif font. Further to the right, there are two white rectangular blocks stacked vertically. At the bottom center, there is a white parallelogram shape that appears to be a folded corner or a stylized base. The overall composition is minimalist and geometric.

Demo

# Mitigering

*[cheatsheetseries.owasp.org/cheatsheets/...](https://cheatsheetseries.owasp.org/cheatsheets/...)*

- [Cross-Site\\_Request\\_Forgery\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html)

# Vad kan man göra för att skydda sig?

Mitigering

CSRF-token  
SameSite Cookies

Minska attackytan

Kontrollera headers  
– Räcker inte



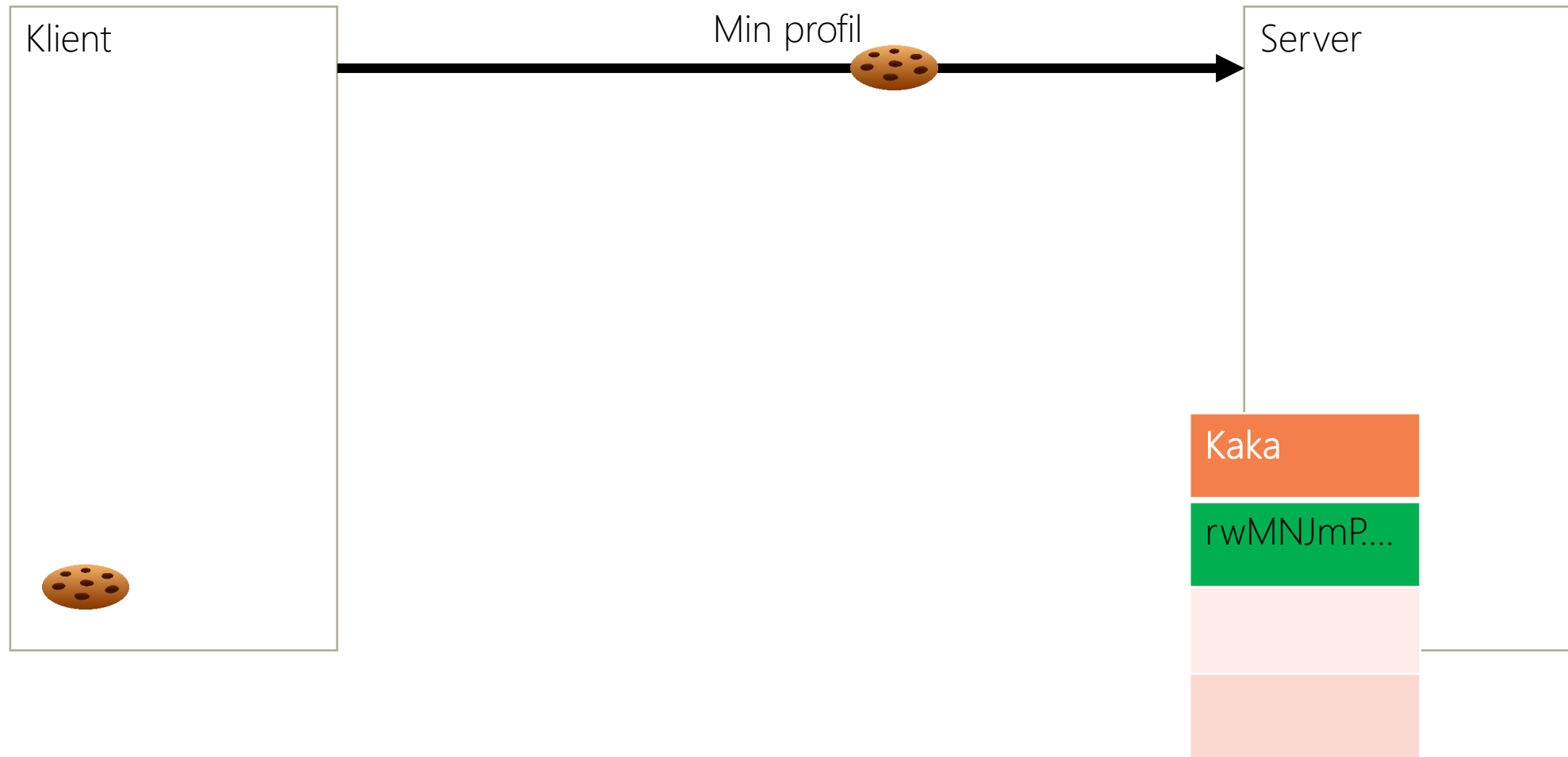
# Vad kan man göra för att skydda sig?

- CSRF-token
  - Nyckel associerad med sessionen och per formulär
  - Finns färdigt i flera ramverk
    - OWASP CSRFGuard
    - Spring Security, .Net AntiForgeryToken
- SameSite Cookies
  - Enkelt och bra men inte helt vattentätt

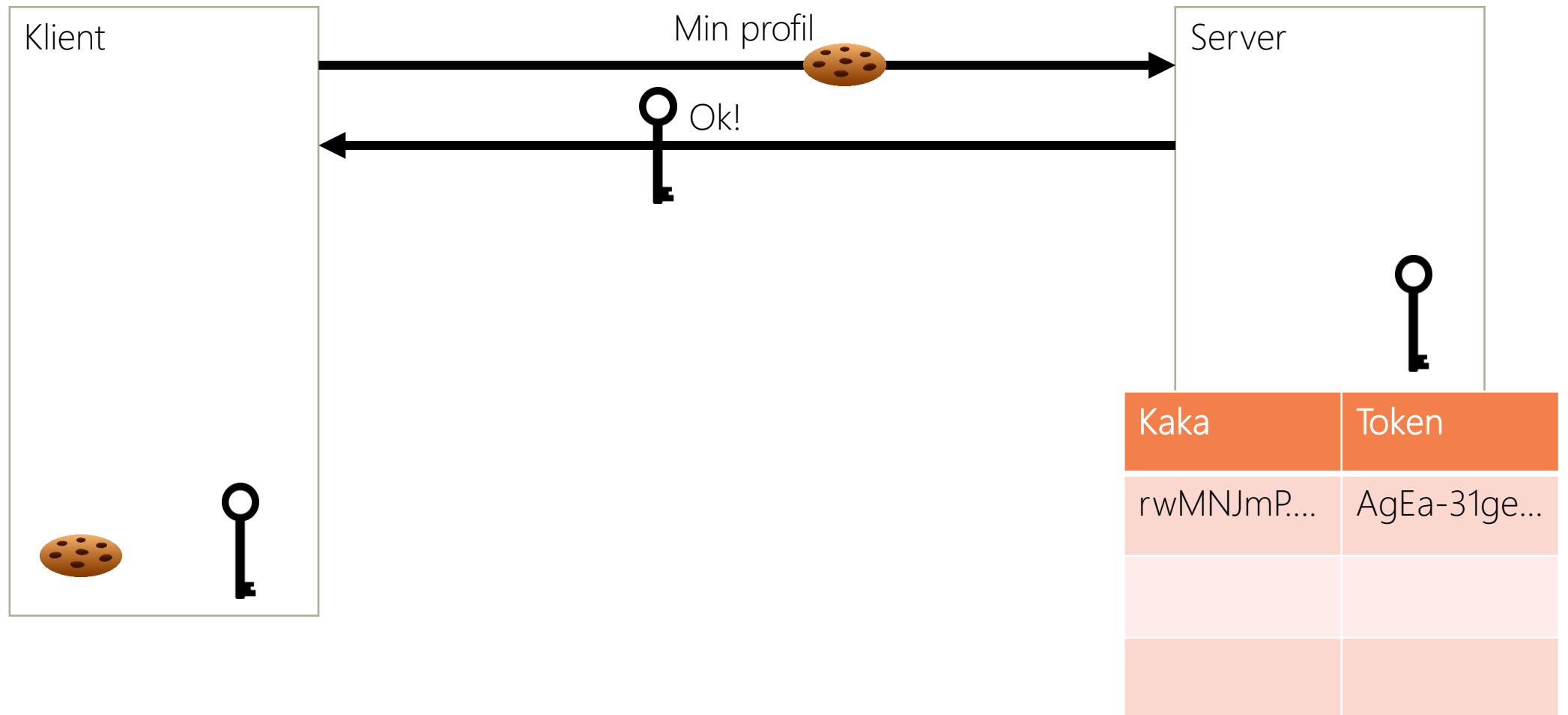
# CSRF Token



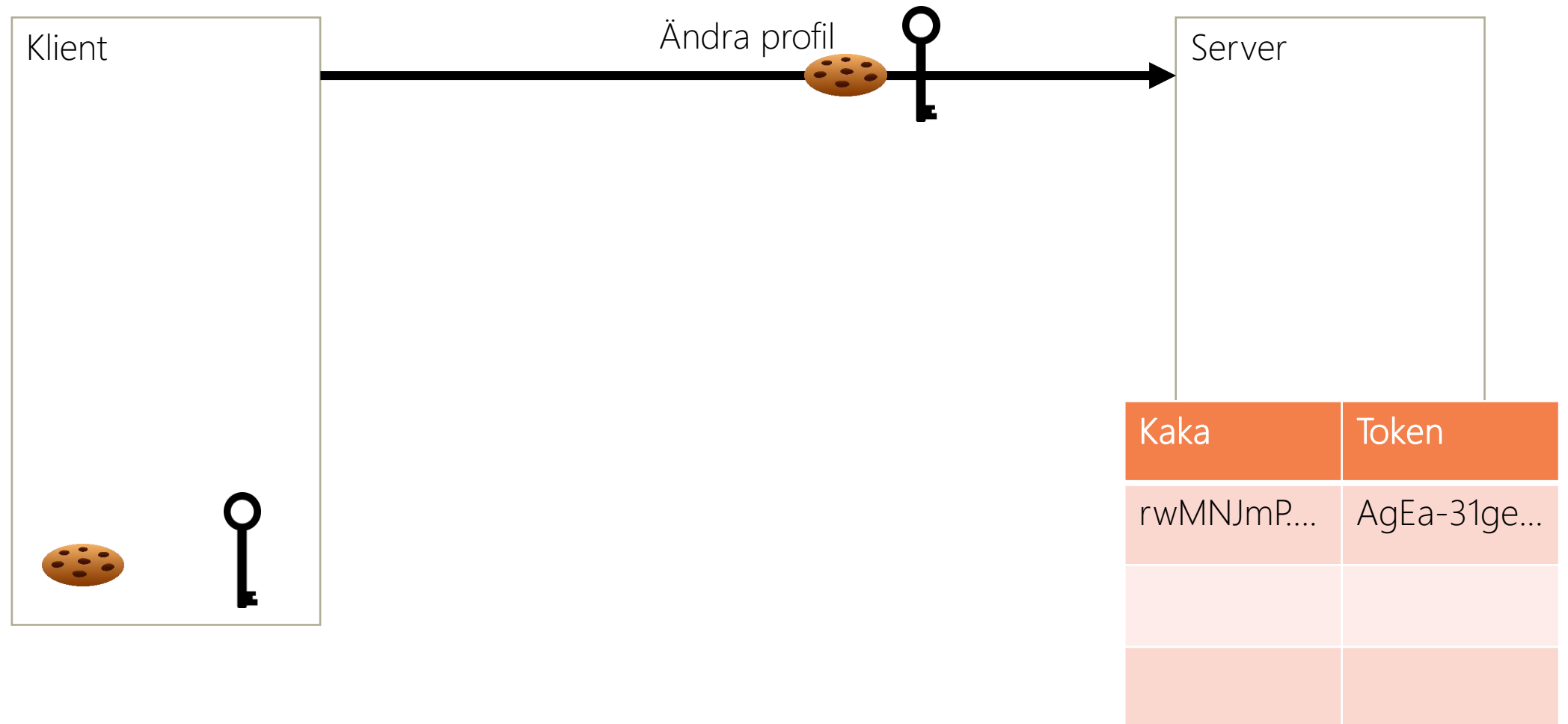
# CSRF Token



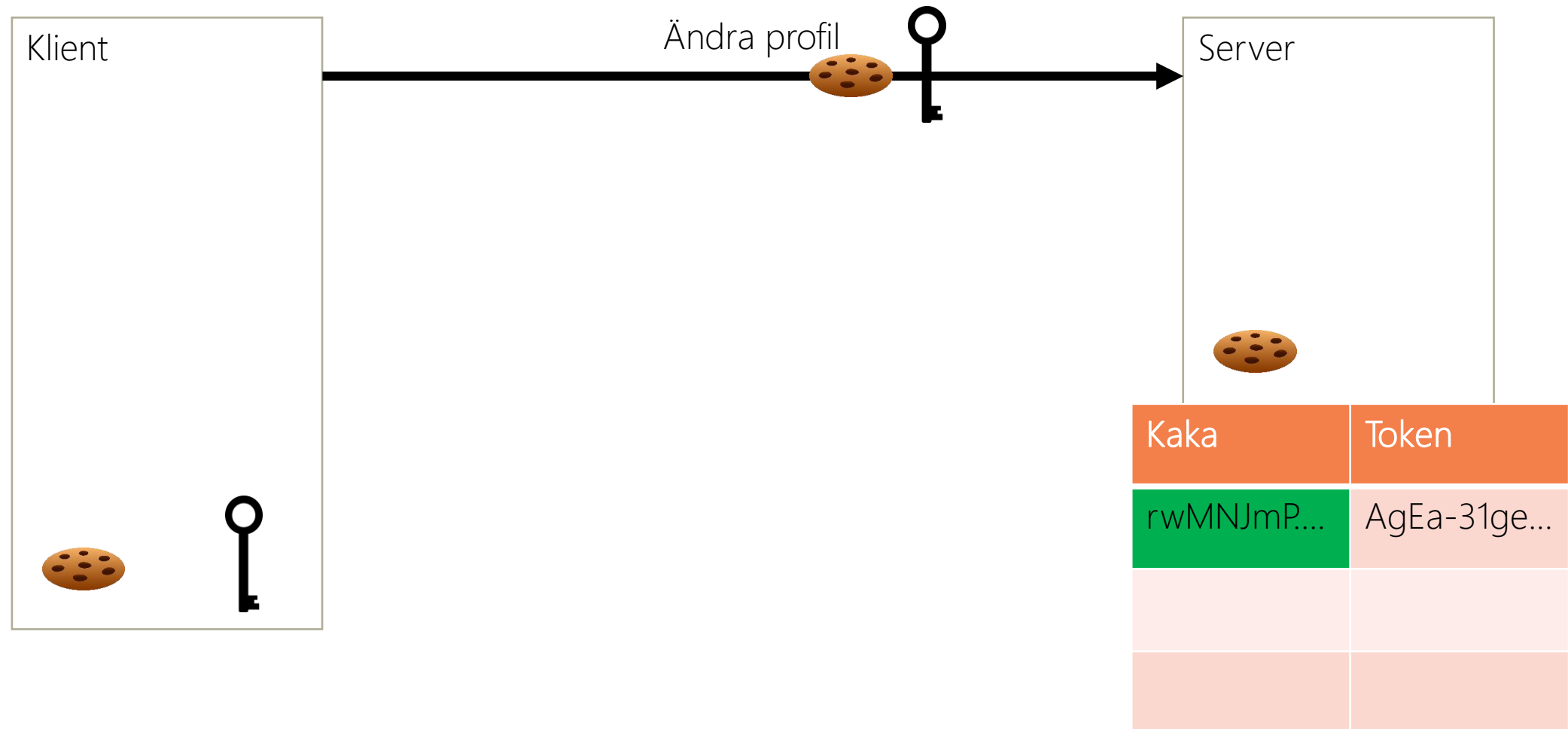
# CSRF Token



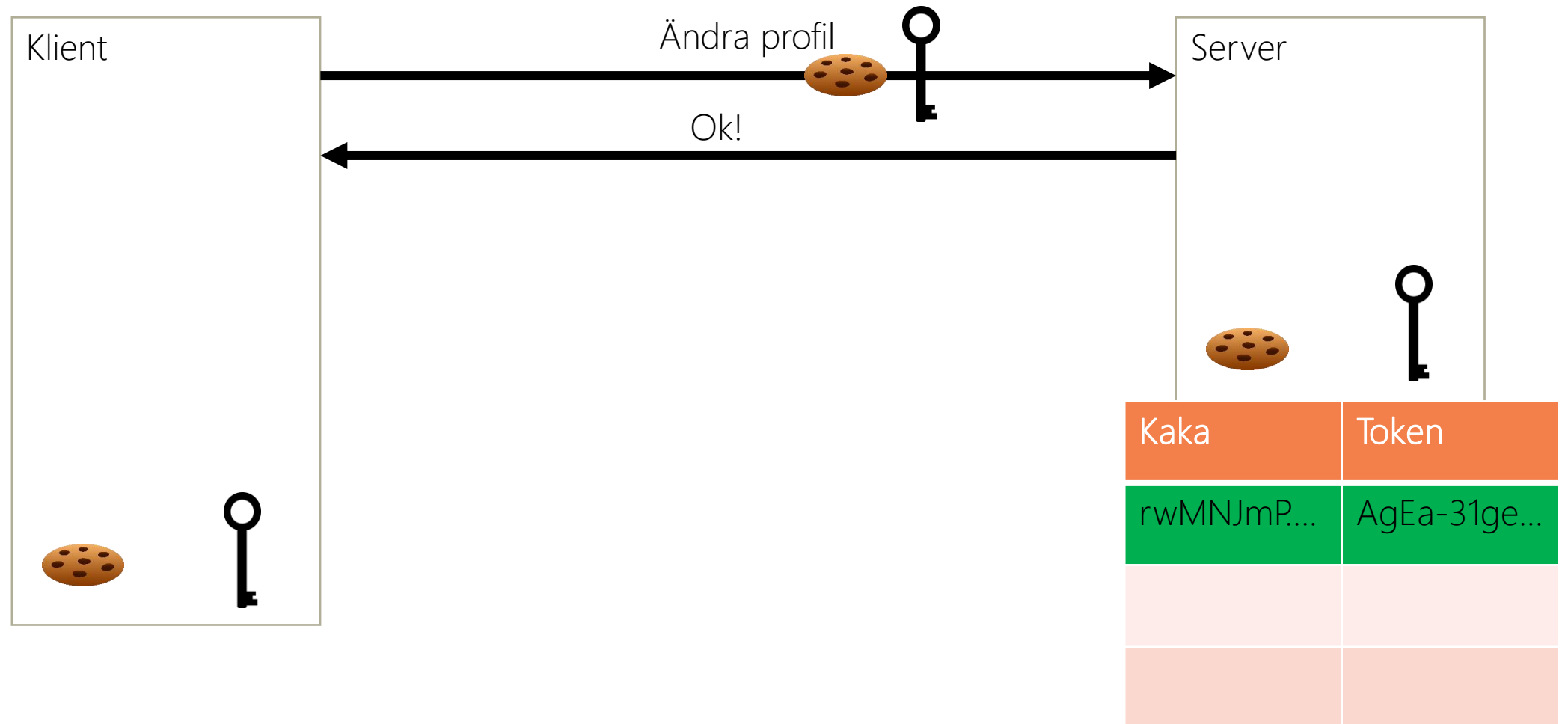
# CSRF Token



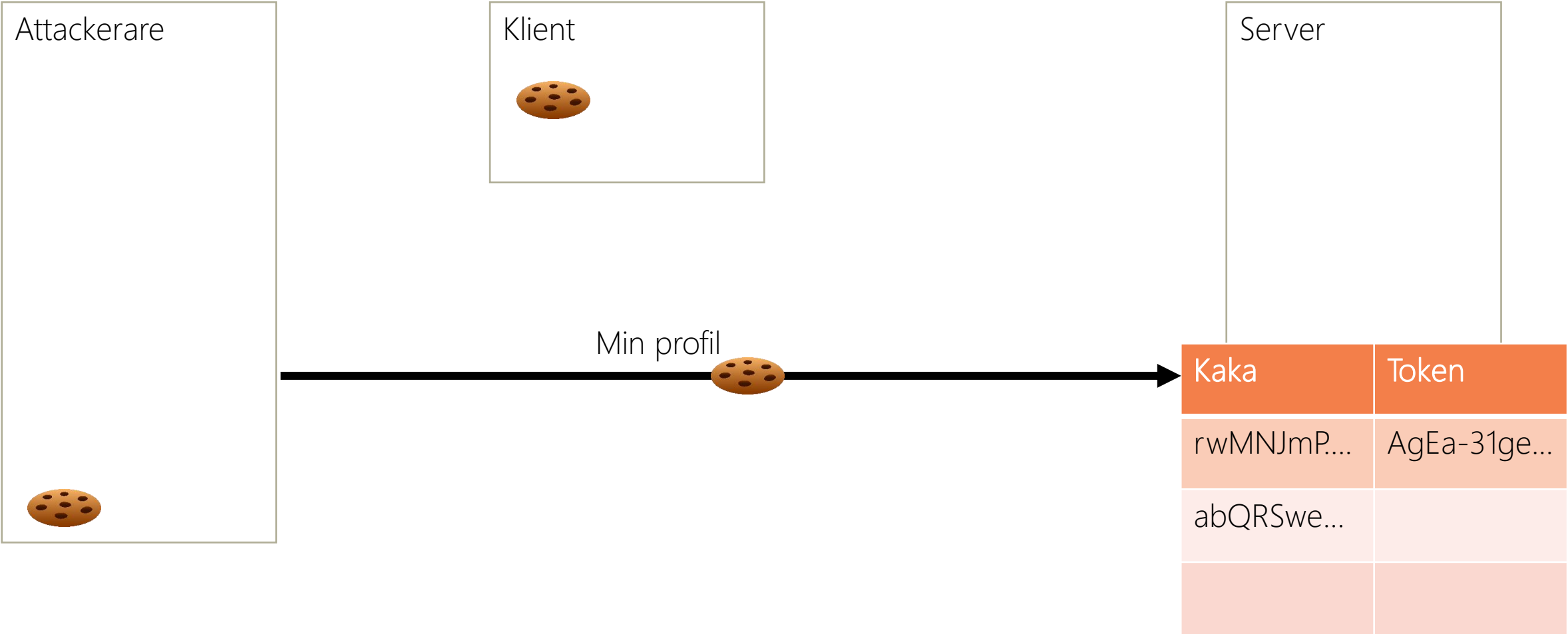
# CSRF Token



# CSRF Token

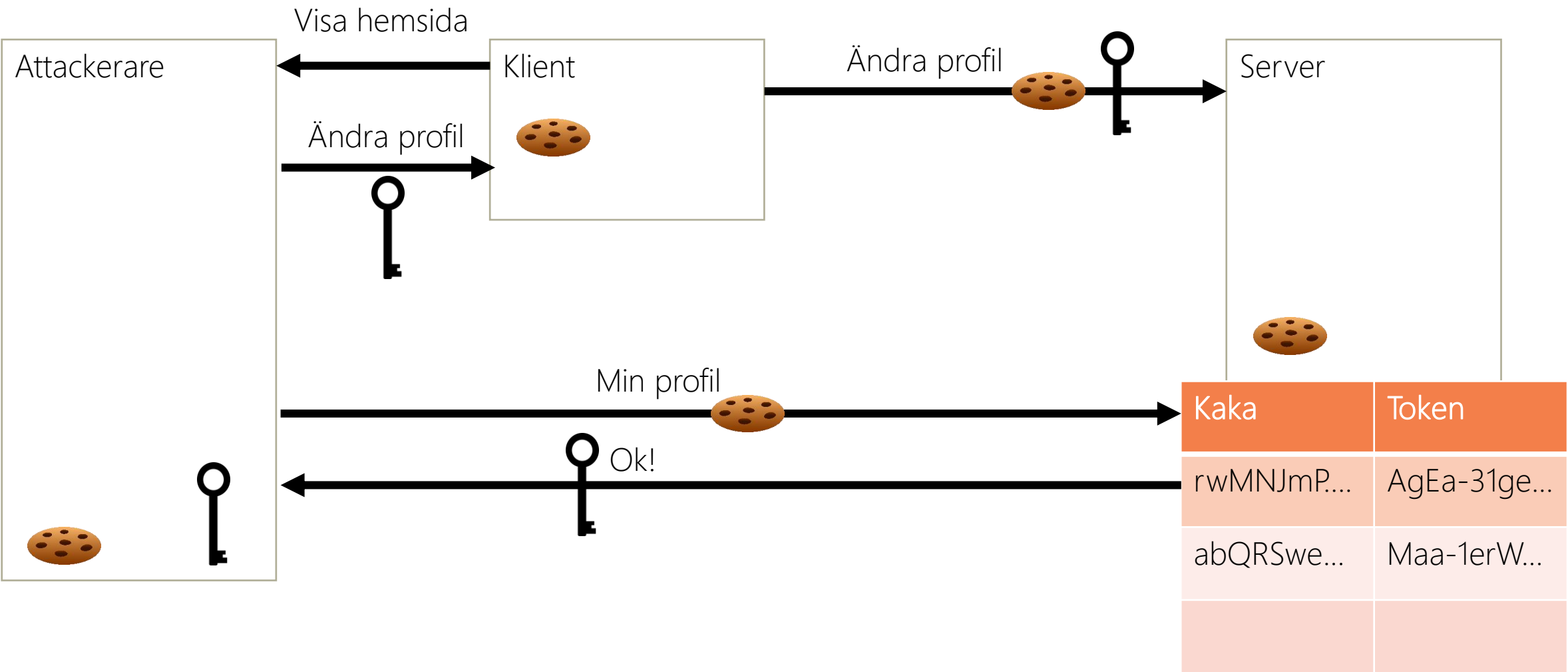


# CSRF Token

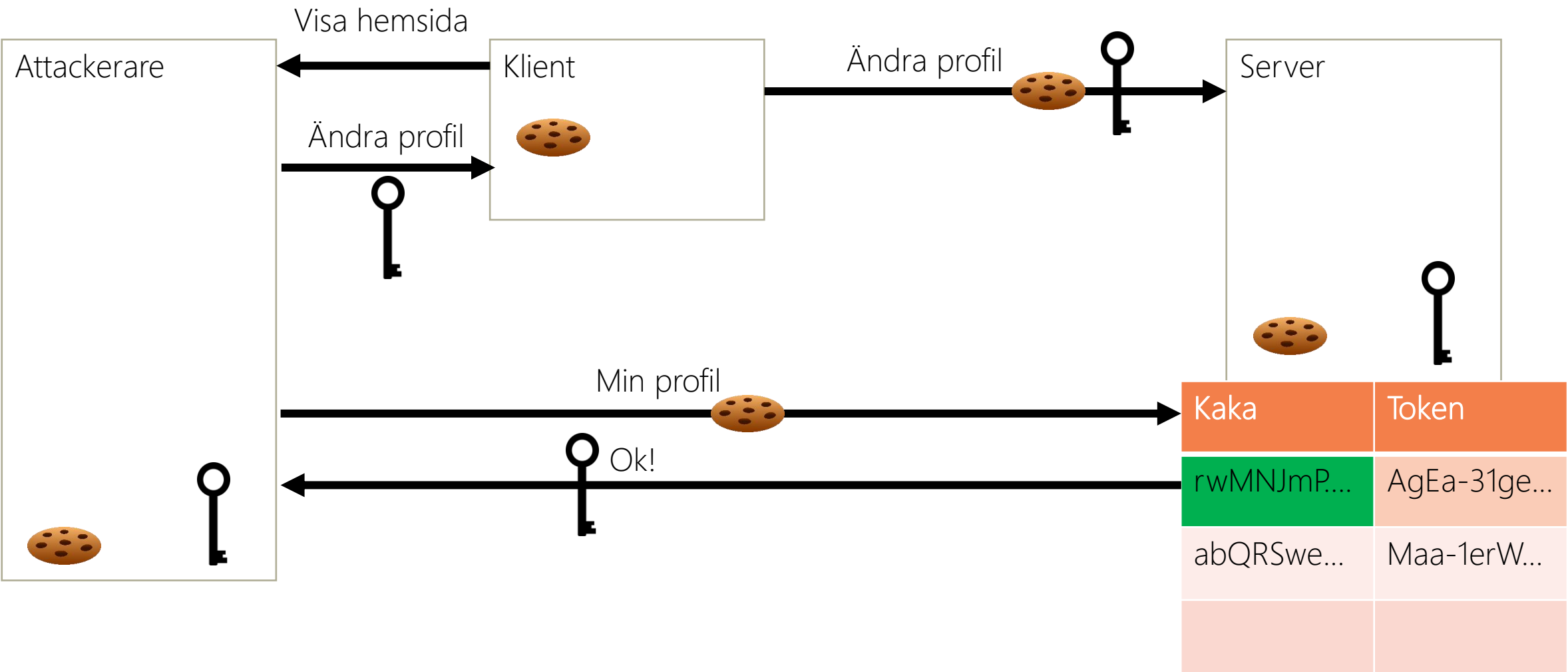




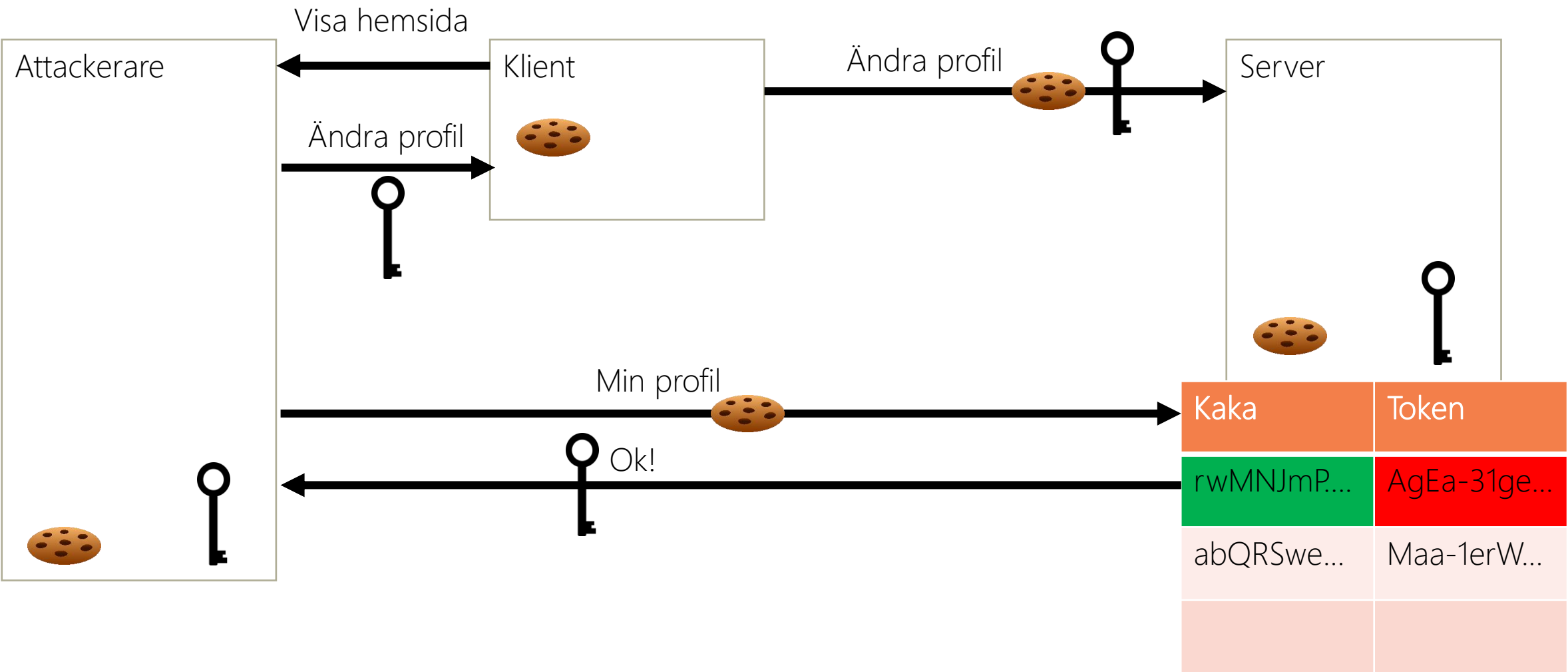
# CSRF Token



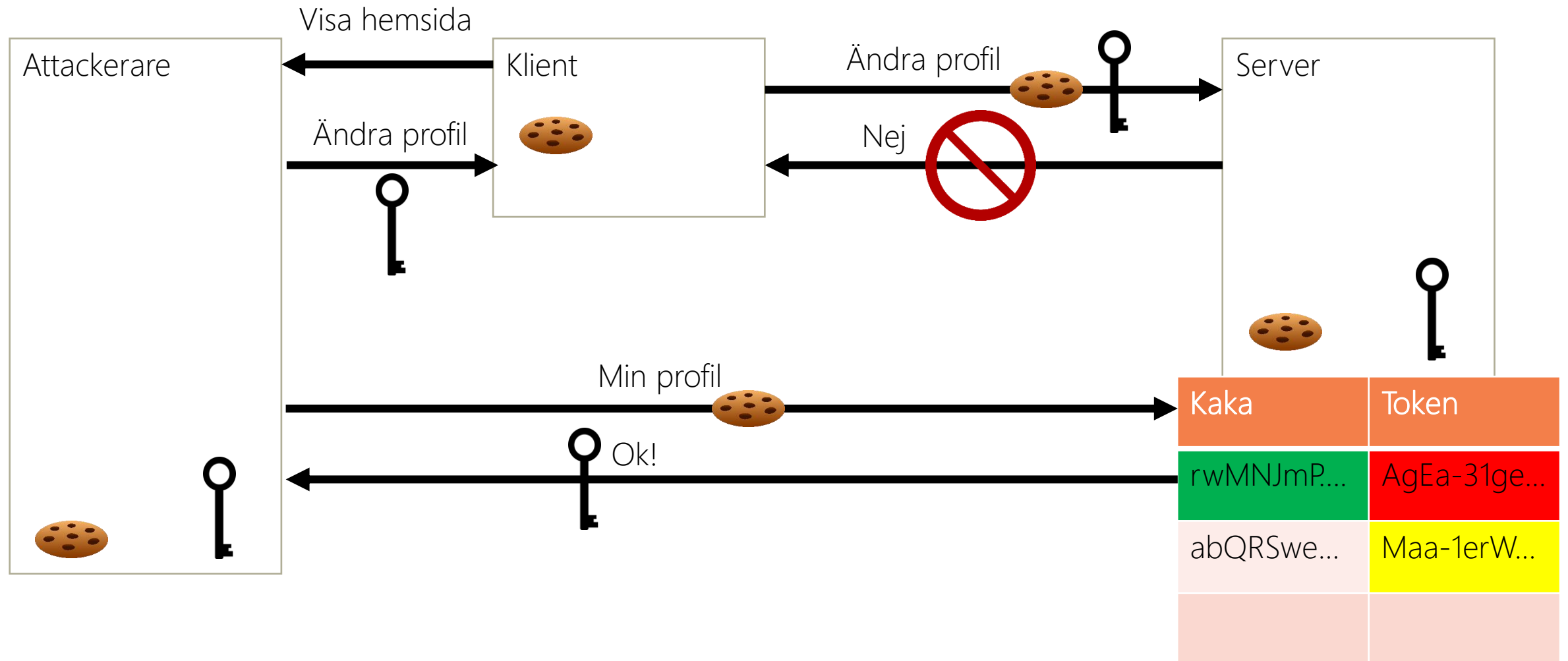
# CSRF Token



# CSRF Token



# CSRF Token



# CSRF Mitigering (SameSite Cookies)



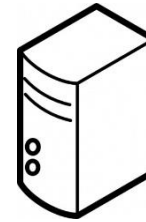
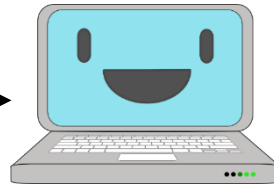
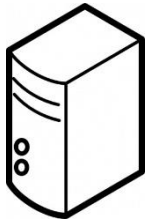
# CSRF Mitigering (SameSite Cookies)



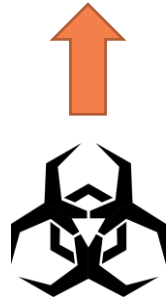
# CSRF Mitigering (SameSite Cookies)



exempelBlog.com



bank.com/db.php?db=prod&drop=true



Cookien har nu attributet  
SameSite=Lax

# CSRF Mitigering (SameSite Cookies)

- *SameSite-ness* baseras på den registrerade domänen (*example.com*, *kits.se*)
- *SameSite* attributet stödjer *strict* och *lax* mode
  - *strict* mode: Browsern lägger aldrig till cookien på cross-site requests
  - *lax* mode: Cookien inkluderas bara i requests som navigerar browsern till en ny sida
- *SameSite* med lax mode hindrar de flesta CSRF attacker
  - Undantaget är om attacken kan utföras med en GET-request, vilket inte borde kunna ske (GET-requests bör inte vara state changing)



# CSRF Mitigering (SameSite Cookies i moderna browsers)

- Moderna browsers sätter numera *SameSite=Lax* som default för cookies
  - Förändringen påverkar framförallt tracking, Single Sign On, betalningar.
- Defaultinställningen kan frångås genom att sätta SameSite=None
  - None gör att cookien skickas oavsett om det är cross site.
  - Moderna browsers accepterar SameSite=None endast om cookien har *Secure*-flaggan satt.
- *SameSite=None* innebär öppen dörr för CSRF, se till att skydda oss på andra sätt!
- Lita inte på att användaren har en modern browser!

# SameSite cookies inkluderas fortfarande i requests mellan subdomäner!

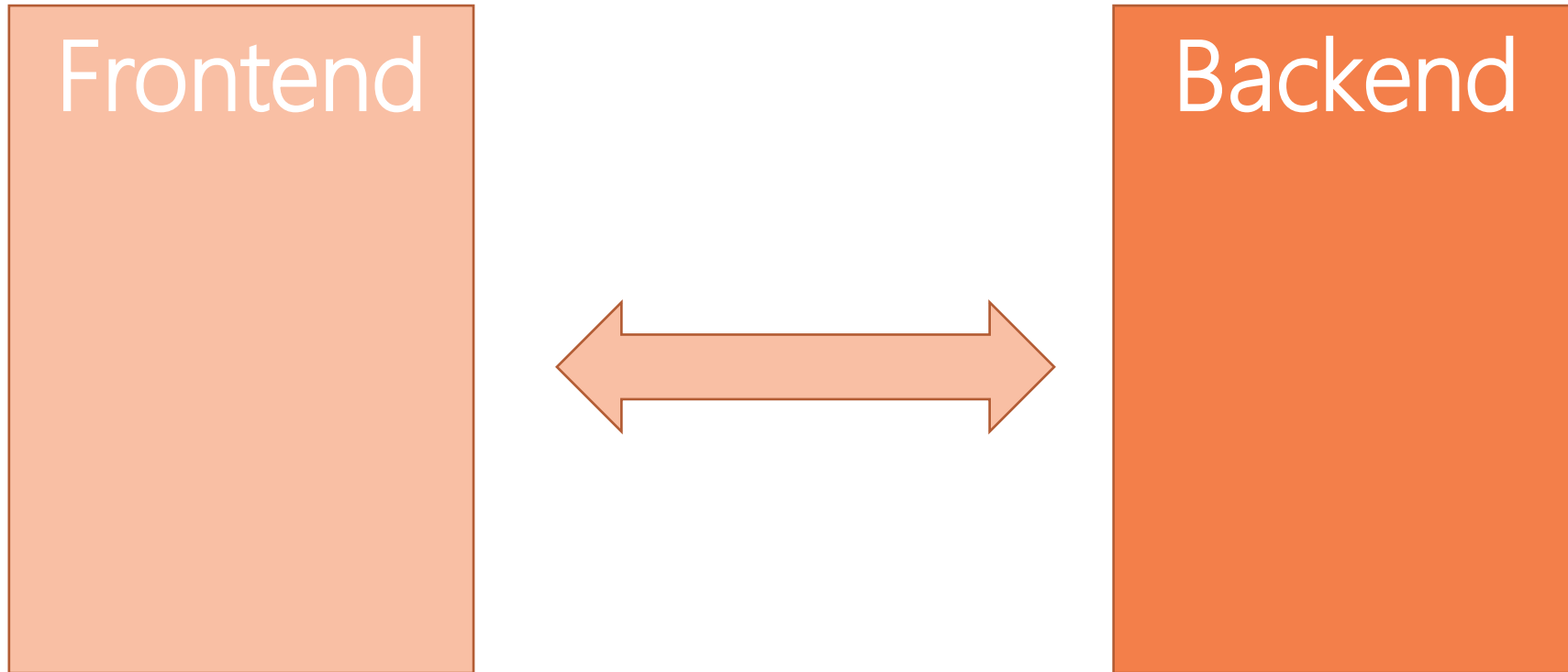


Om vi ska vara petiga så är en request mellan subdomäner inte cross-site, attacken borde kallas "Same Site but Cross Origin Request Forgery".



# Implementering

# Vad gör man på backend?



# Java EE



You are on your own

## Vad kan man göra?

- OWASP CSRFGuard
- Tomcat-filter
- GenericCSRFFilter
- CSRFPProtection
- MvcContext
- Andra ramverk

# Spring Security

- CSRF-token som standard
- Validerar att token finns med i alla formulär
- Automatiskt ibland
  - `<form:form>` Ok!
  - `<form>` Måste lägga till själv
- SameSite Attribute



# Express Integration

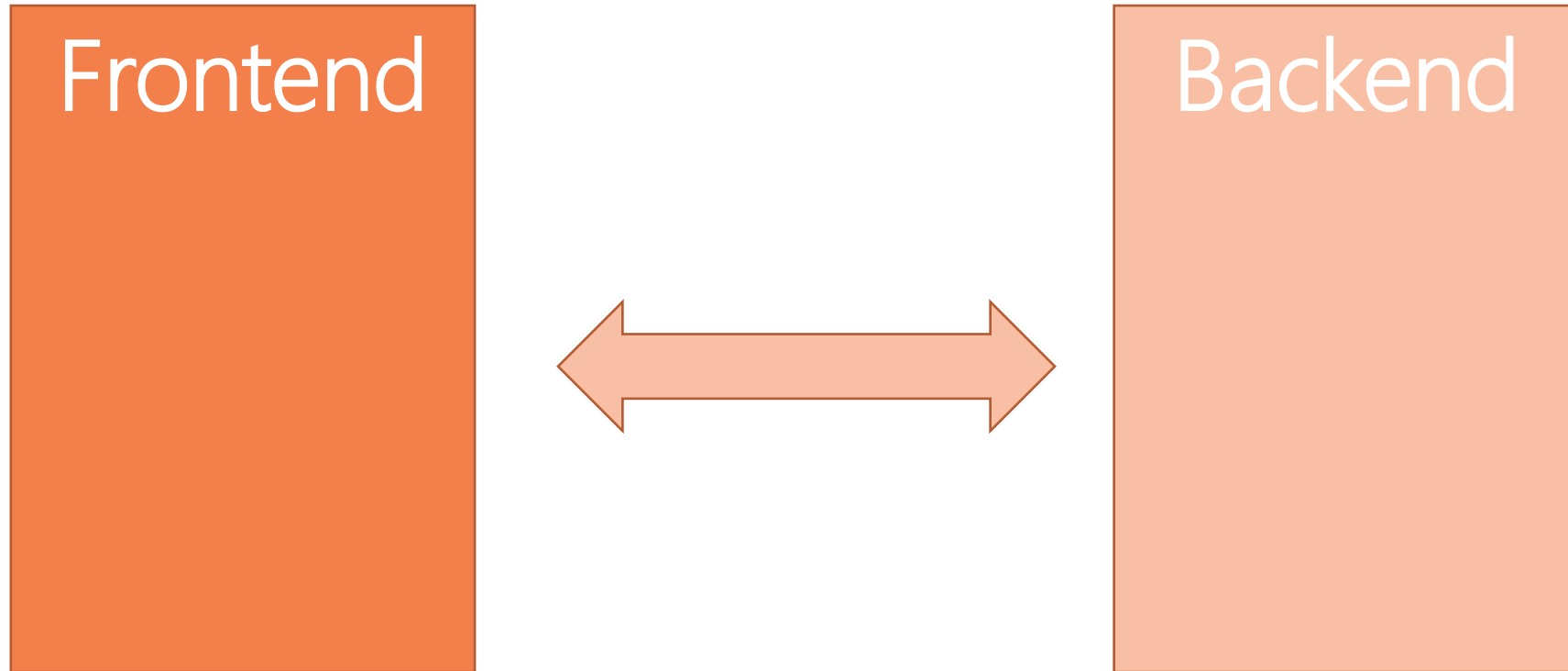
```
var express = require('express');
var router = express.Router();
var { randomBytes } = require('crypto');

/* GET home page. */
router.get('/', function(req, res, next) {
  if (req.session.csrf === undefined) {
    req.session.csrf =
      randomBytes(100).toString('base64'); //
      convert random data to a string
  }

  res.render('index', { title: 'Express' });
});

module.exports = router;
```

Vad gör man på frontend?





# Angular

This page is intentionally left blank

# Angular XSRF Skydd

## Server skickar ut

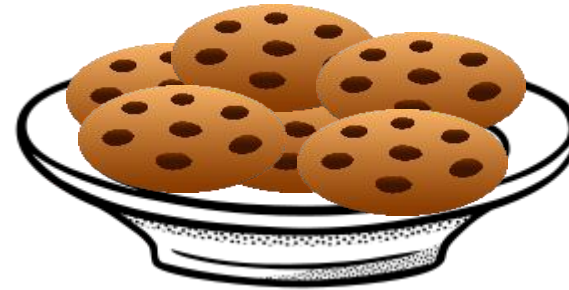
```
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64)
           AppleWebKit/537.36 (KHTML, like Gecko)
           Chrome/53.0.2785.143 Safari/537.36
authorization: Basic YWRtaW46YWRtaW4=
Referer: http://localhost:8080/
Accept-Encoding: gzip, deflate, sdch
Accept-Language: sv-SE,sv;q=0.8,en-US;q=0.6,en;q=0.4
Cookie: PHPSESSID=8tol5b9iv34849ue2jluu886b3;
        XSRF-TOKEN=5e09e681-4bbe-406b-88c1-1e559b51c5ce
```

## Klienten svarar

```
GET /user HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Accept: application/json, text/plain, */*
X-XSRF-TOKEN: 5e09e681-4bbe-406b-88c1-1e559b51c5ce
X-Requested-With: XMLHttpRequest
```

# Måste man ha kakor?

- Nej!
- Använd Authorization header för att skicka användardata istället.
  - Browsers skickar inte automatiskt med dessa i requests, ingen CSRF!



The background is a solid olive green color. It features several white geometric shapes: a large vertical rectangle on the left, a horizontal rectangle in the top right, a trapezoid in the bottom center, and a vertical rectangle on the right side. A thin orange horizontal bar is at the top, and a thin orange vertical bar is on the left. The word "Kahoot!" is written in white text in the center of the olive green area.

Kahoot!

# Server Side Request Forgery

(SSRF)

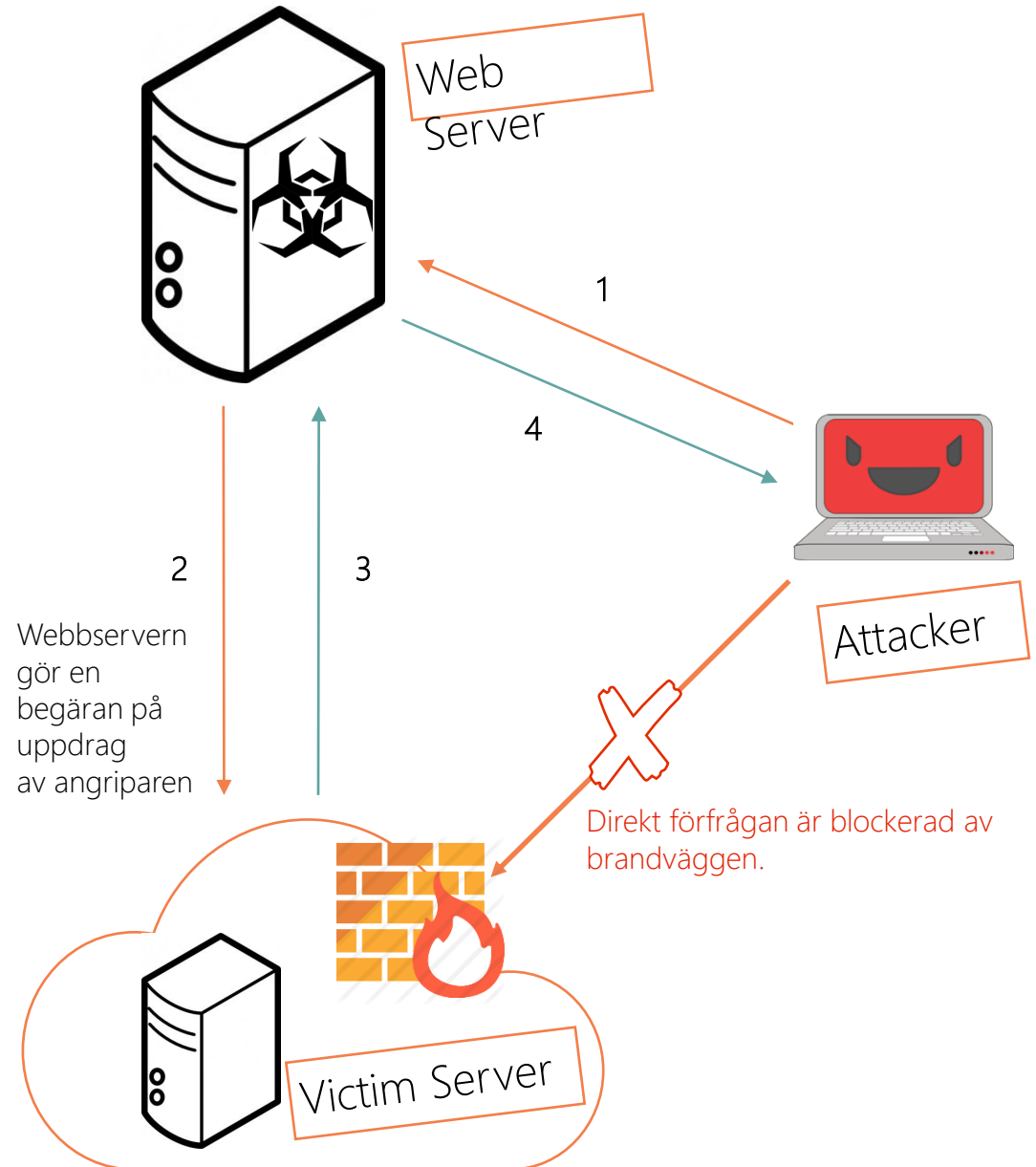
# Vad är SSRF?

- En sårbarhet i en applikation som lurar en webserver att göra requests:
- Angriparen kan skapa interna eller externa requests som utförs av webservern. Exempel på förfrågningar:

<https://kits.se?url=https://resources.kits.se/contract>

<http://127.0.0.1/admin/reset/cache>

<http://127.0.0.1/admin/adduser?u=hack&pw=123>



# Skillnad mellan CSRF och SSRF

	CSRF	SSRF
Attackmål	Användare	Server
Attack Syfte	Lurar en slutanvändare att utföra oönskade åtgärder på en webbapplikation där de för närvarande är autentiserade, som tex ändra lösenord	Få tillgång till känslig data, interna nätverk och applikationer
Upptäckt	Båda typerna har samma problem, dvs, att utföra oönskade åtgärder skapade av en angripare	

SSRF in action!



# A hacker gained access to 100 million Capital One credit card applications and accounts



## Capital One fined \$80 million for 2019 hack of 100 million credit card applications



By [Devlin Barrett](#)


August 6, 2020 at 11:12 a.m. EDT



## Capital One to pay \$190M settlement in data breach linked to Seattle woman

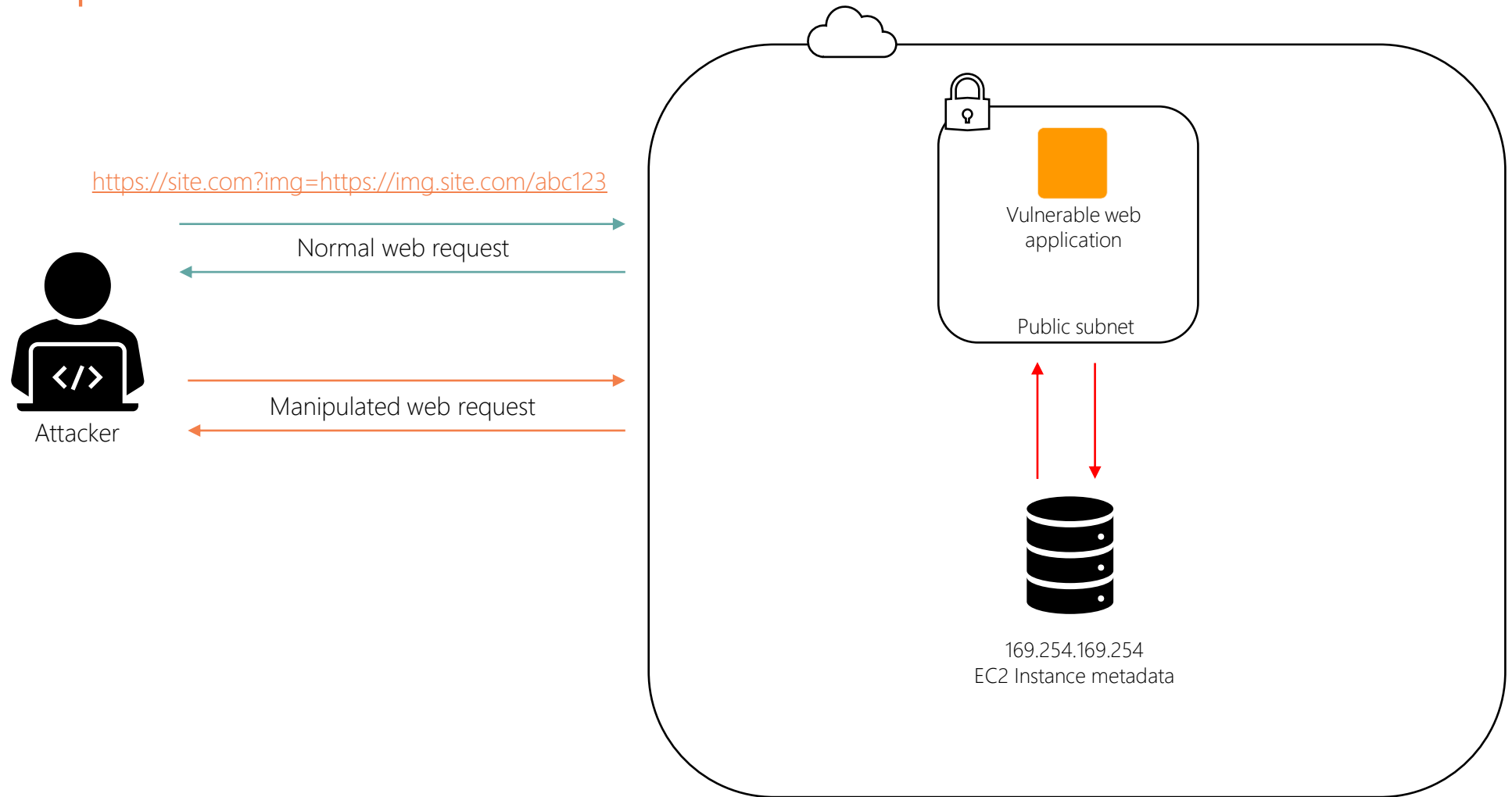
Dec. 23, 2021 at 2:16 pm | Updated Dec. 23, 2021 at 7:48 pm



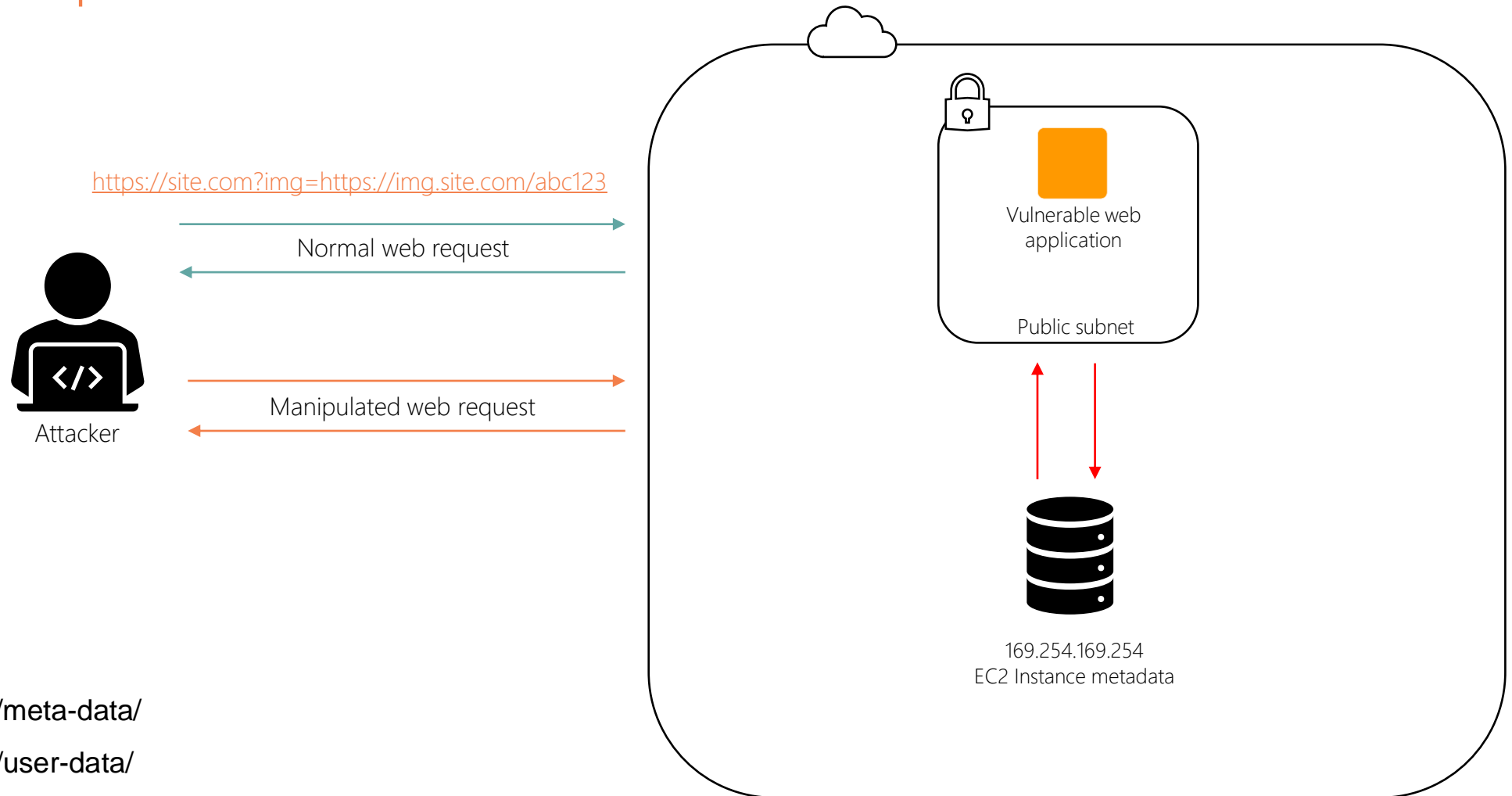
The background is a solid teal color. There are several white geometric shapes: a horizontal rectangle on the left, a vertical rectangle in the center, and a small horizontal rectangle at the bottom right.

SSRF exempel  
(AWS)

# SSRF Attack på AWS



# SSRF Attack på AWS



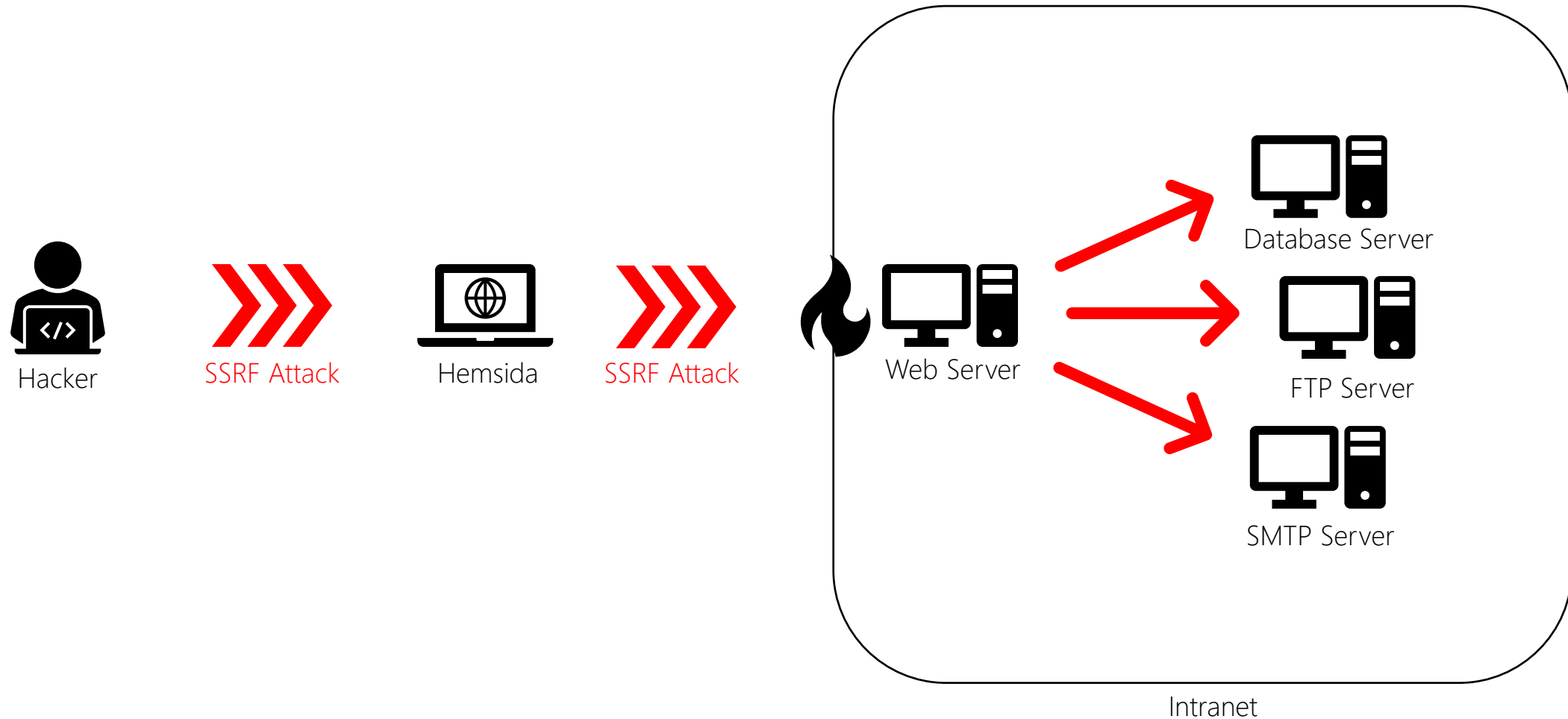
<http://169.254.169.254/latest/meta-data/>

<http://169.254.169.254/latest/user-data/>

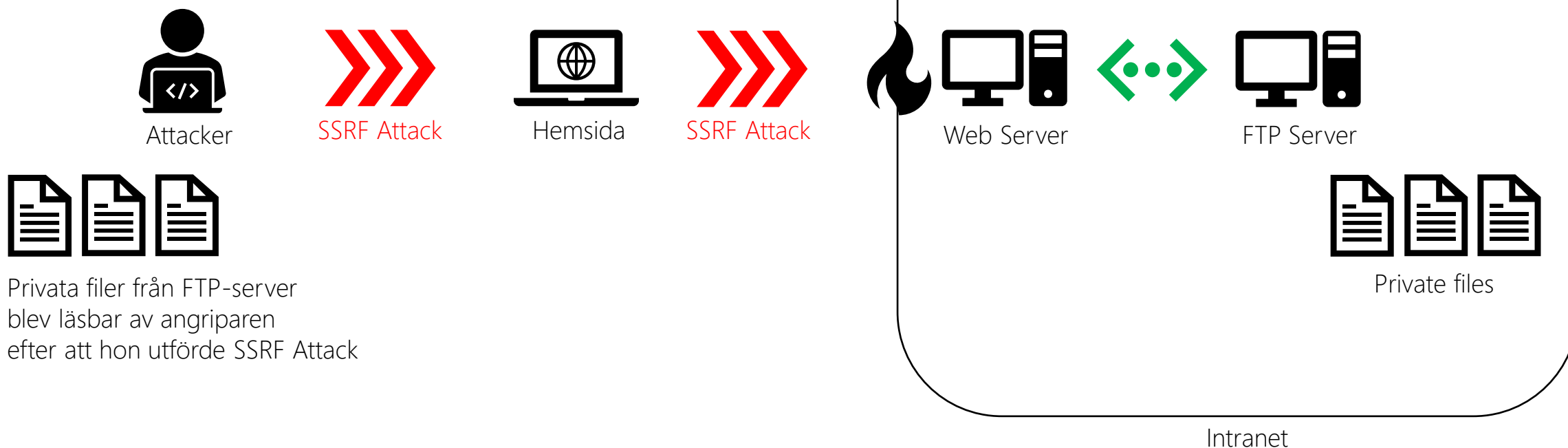
[http://169.254.169.254/latest/meta-data/iam/security-credentials/IAM\\_USER\\_ROLE\\_HERE](http://169.254.169.254/latest/meta-data/iam/security-credentials/IAM_USER_ROLE_HERE)

<http://169.254.169.254/latest/meta-data/iam/security-credentials/PhotonInstance>

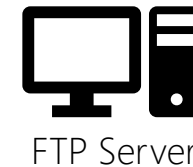
# Missbruka tillitsförhållandet mellan den sårbara servern och andra system



# Läs resurser som inte är tillgängliga för allmänheten



# Skanna intranätverket som servern är ansluten till



Attacker

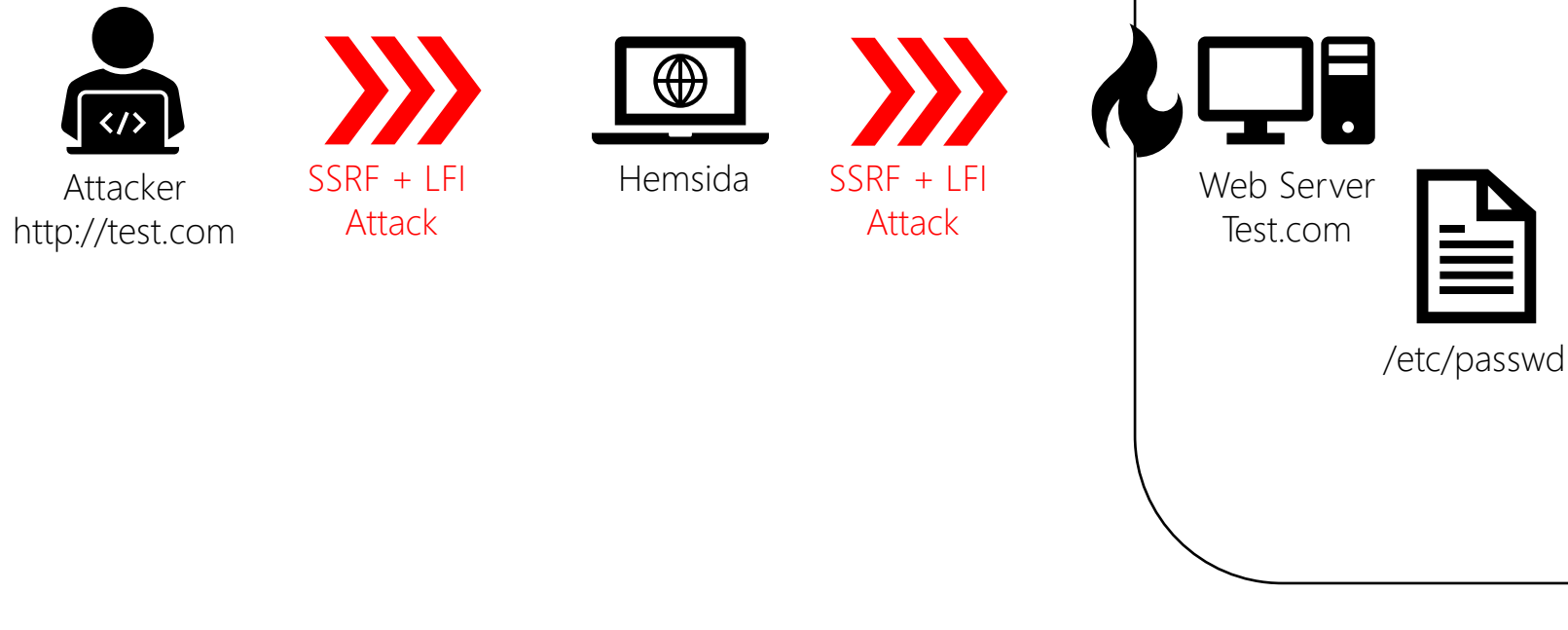


Angriparen skannar nätverket för att hitta  
Vilken tjänst är ansluten.



Intranet

# Läs filer från webbservern

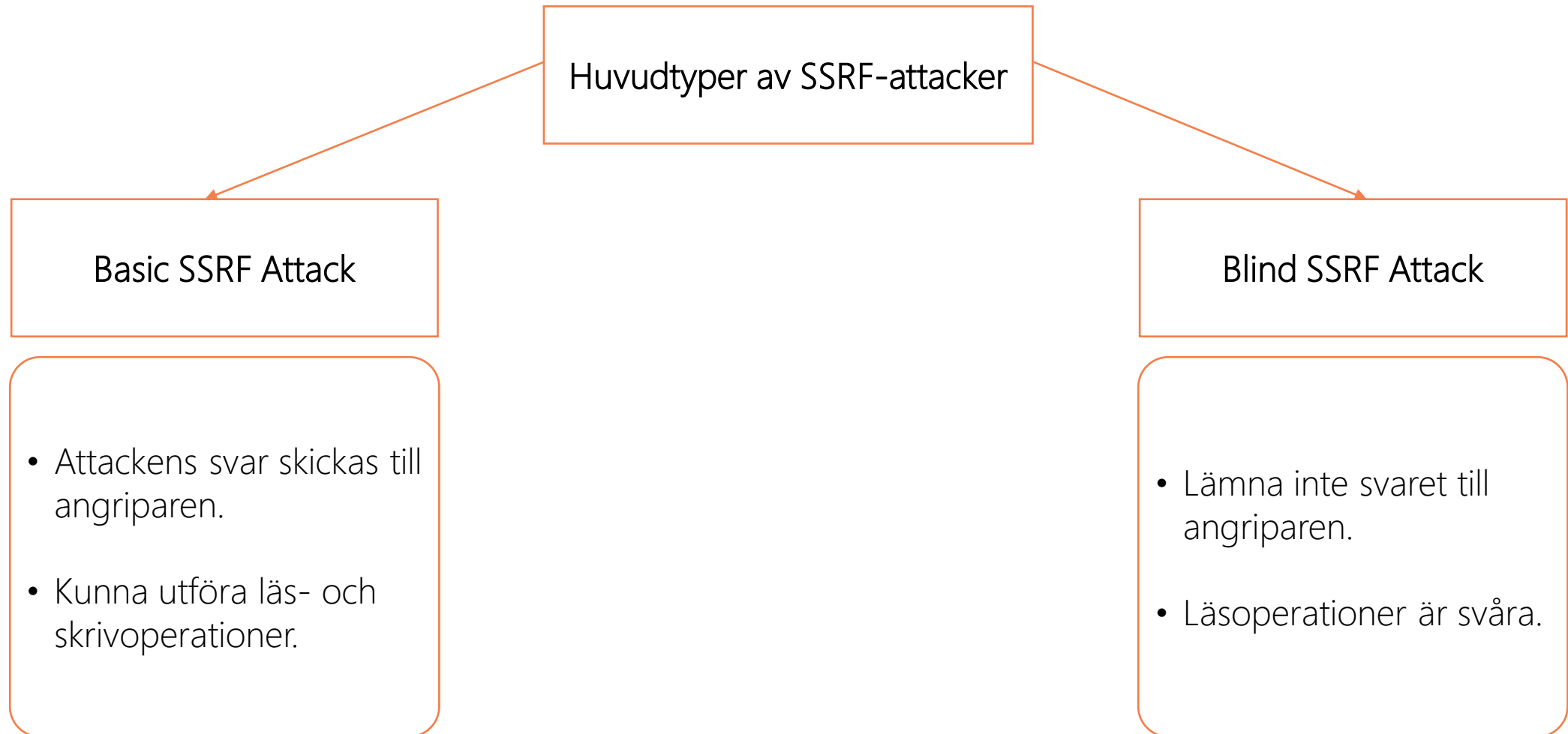


<https://test.com/icon?url=file://../../../../../../../../etc/passwd>



```
root:!:0:0:::/usr/bin/ksh
daemon:!:1:1::/etc:
bin:!:2:2::/bin:
sys:!:3:3::/usr/sys:
...
```

# Typer av SSRF-attacker





The background is a solid teal color. On the left side, there are two white geometric shapes: a horizontal rectangle and a vertical rectangle. The vertical rectangle is positioned to the right of the horizontal one, and they overlap slightly. Below these, there are more white shapes, including a long horizontal bar and a smaller vertical bar, creating a stepped or architectural look.

SSRF in the cloud

# SSRF Attack på AWS



- <https://test.com/icon?url=http://169.254.169.254/latest/meta-data/>
- <https://test.com/icon?url=http://169.254.169.254/latest/user-data/>
- <https://test.com/icon?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/ISRM-WAF-Role>

```
{
  "Code" : "Success",
  "LastUpdated" : "2019-08-03T20:42:03Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIA5A6IYGGDLBWIFH5UQ",
  "SecretAccessKey" : "sMX7//Ni2tu2hJua/fOXGfrapiq9PbyakBcJunpyR",
  "Token" :
    "AgoJb3JpZ2ZuX2VjEH0aCXVzLWVhc3QtMSJHMEUCIQDFoFMUFs+lth0JM2IEddR/8LRHwdB4HiT1MBpEg8d+EAlgCKqMjkjdET/XjgYGDf9/eoNh1+5Xo/tnmDXeDE+3eKlq4wMI9v/////////
    /ARAAGgw4OTUzODQ4MTU4MzAiDEF3/SQw0vAVzHKrgCq3A84uZvhGAswagrFjgrWAVlj4cJd6eI5Gcje09FyFRPmALKJymfQgpTQN9TtC/sBhlylCfni8JJvGesQZGi9c0ZFIWqdlmM/2rdZ6Ga
    qcZY9V+0LspbwiDK0FUjrRcquBVswSlxWs8Tr0Uhpka20mUQOBhovmVyXNzyTQUQnBE9qgFLbYY+t86yUXmXMXxGPd4sWuLgkoCF2iPlMkgUwZq8hZvoiVf7TVQU32sgstKN7ozJiJcgTBpa6
    /batScGBtNpck4LOvHzNwwYv/FuVkpC70bPhqNXVxMEcpwt4s7RkHHowdFINpnPpm57dfAYwZwoklWJdvtqFQ0tZHusZ65vJqyk5cZ8f3P/Cf7UlzoZPslarWcgfiDvkQliU9fY6Brt7jyjrF5h7oJ
    bW/LUS4R9SDp+qKMtUY2JmLZRovsW4GfhfLJWv7wrW81QZVC8rBKLzWFRTLrkhITFsS7A5JscuKoORyDxGQq/pGRsE30effdS9G1xNmzKwn45/V0XsilhTE7pOJGGopuLfBo5KD46hVS9v1iB
    uvxrVxsHFz7mnD/GKiwi1hbFAKEvypagZ28qEJaarNvAdi2QOowjuOX6gU6tAFrFFVBb6ZTI4btIjHNNNoT0TFW5iYD0dkD+csqC4nTVpnAG/FFBk+CAHdy5Gh/aBISO7OQF9xKJSXkd+Syf62pg5X
    iMseL3n2+2+IWdDgKwhZYxeVlMbX88QYX3P9sX+OWHWidAVgtQhZw3xJ+VBV33EKgJ4b8Bk6mgo0kiB1hnoN0KX8RXr1axpYnJv2GHb8h/det89iwpky77+8YcEvRc+DGTlicUlxDoirgck9bp
    P3EBXfs=",
  "Expiration" : "2019-08-04T03:16:50Z"
}
```

# SSRF Attack på Google Cloud



Google Cloud

<http://169.254.169.254/computeMetadata/v1/>

<http://metadata.google.internal/computeMetadata/v1/>

<http://metadata/computeMetadata/v1/>

<http://metadata.google.internal/computeMetadata/v1/instance/hostname>

<http://metadata.google.internal/computeMetadata/v1/instance/id>

<http://metadata.google.internal/computeMetadata/v1/project/project-id>

# SSRF Attack på Digital Ocean

<http://169.254.169.254/metadata/v1.json>

<http://169.254.169.254/metadata/v1/>

<http://169.254.169.254/metadata/v1/id>

<http://169.254.169.254/metadata/v1/user-data>

<http://169.254.169.254/metadata/v1/hostname>

<http://169.254.169.254/metadata/v1/region>

<http://169.254.169.254/metadata/v1/interfaces/public/0/ipv6/address>



# SSRF Attack på Microsoft Azure

<http://169.254.169.254/metadata/v1/maintenance>



# Kringgå URL filter – IPs kan vara luriga!

## # Bypass using HTTPS:

http://127.0.0.1/  
http://localhost/

## # Bypass using localhost with [::]:

http://[::]:80/  
http://[::]:25/ SMTP  
http://[::]:22/ SSH  
http://[::]:3128/ Squid

## # Bypass using a decimal IP location:

http://0177.0.0.1/  
http://2130706433/ = http://127.0.0.1  
http://3232235521/ = http://192.168.0.1  
http://3232235777/ = http://192.168.1.1

## # Bypass using IPv6/IPv4 Address Embedding:

http://[0:0:0:0:0:ffff:127.0.0.1]

## # Bypass against a weak parser by Orange Tsay:

http://127.1.1.1:80\@127.2.2.2:80/  
http://127.1.1.1:80\@@127.2.2.2:80/  
http://127.1.1.1:80:\@@127.2.2.2:80/  
http://127.1.1.1:80#\@127.2.2.2:80/

## # Abusing Enclosed Alphanumerics

http://169。254。169。254/  
http://169。254。169。254/  
http://(16)(9)。(2)(5)(4)。(16)(9)。(2)(5)(4)/  
http://(0)(x)(a)(9)。(0)(x)(f)(e)。(0)(x)(a)(9)。(0)(x)(f)(e):80/  
http://(0)(x)(a)(9)(f)(e)(a)(9)(f)(e):80/

**DEMO**

# Mitigering

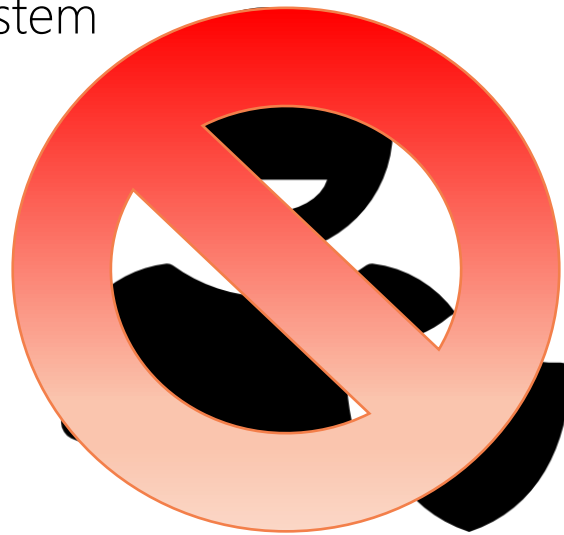
*[cheatsheetseries.owasp.org/cheatsheets/...](https://cheatsheetseries.owasp.org/cheatsheets/...)*

- [Server\\_Side\\_Request\\_Forgery\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html)



# Hur kan man mildra mot SSRF?

- Allowlistning av DNS och IP till system
- Inaktivera oönskade protokoll och oanvända URL-scheman
- Validera respons
- Använd autentisering på interna system



Kahoot!

**LUNCH!**

# Schema för dagen

Introduktion: 08:30

Intro

Spelplan

Förmiddag:

OWASP Web Top 10 08:45 – 10:00

[BREAK] 10:00 – 10:15

CSRF 10:15 – 11:10

[BREAK] 11:10 – 11:25

SSRF 11:25 – 12:00

Eftermiddag:

SQL Injektion 13:00 – 13:40

[BREAK] 13:40 – 13:55

XSS 13:55 – 14:45

[FIKA] 14:45 – 15:10

File Inclusion 15:10 – 15:40

File Upload 15:40 – 16:10

OWASP Cheatsheet & ASVS 16:10 – 16:20

Avslut 16:20 – 16:30

The background is a solid teal color. On the left side, there are several white geometric shapes: a horizontal rectangle, a vertical rectangle, and a small square at the bottom left. These shapes are arranged in a way that they appear to be part of a larger, abstract design.

# SQL Injection!









# Databasen är guldgruvan

- Läsa, modifiera, radera data
- Varje arbetande person i Bulgarien fick sin data stulen



# Datal

- Läs
- Val

▲ 15		<b>SQL Injection through /include/findusers.php</b> By <a href="#">egix</a> to <a href="#">ImpressCMS</a>   ● Resolved   ● Critical	disclosed 2 months ago
▲ 1		<b>REST API gets `query` as parameter and executes it</b> By <a href="#">paulocsanz</a> to <a href="#">Rocket.Chat</a>   ● Resolved   ● Medium	disclosed 2 months ago
▲ 4		<b>Regex account takeover</b> By <a href="#">ghaem51</a> to <a href="#">Rocket.Chat</a>   ● Resolved   ● Critical	disclosed 2 months ago
▲ 19		<b>Blind User-Agent SQL Injection to Blind Remote OS Command Execution at [REDACTED]</b> By <a href="#">echidonut</a> to <a href="#">Sony</a>   ● Resolved   ● Critical	disclosed 5 months ago
▲ 31		<b>sql injection via https://setup.p2p.ihost.com/</b> By <a href="#">exploitmsf</a> to <a href="#">IBM</a>   ● Resolved   ● Critical	disclosed 5 months ago
▲ 7		<b>SQL injexion via vulnerable doctrine/dbal version</b> By <a href="#">nickvergessen</a> to <a href="#">Nextcloud</a>   ● Resolved   ● High	disclosed 7 months ago
▲ 28		<b>SQL injection in URL path processing on www.ibm.com</b> By <a href="#">asterite</a> to <a href="#">IBM</a>   ● Resolved   ● Critical	disclosed 7 months ago
▲ 100		<b>SQL Injection at https://files.palantir.com/ due to CVE-2021-38159</b> By <a href="#">haxor31337</a> to <a href="#">Palantir Public</a>   ● Resolved   ● High	disclosed 8 months ago

Pricing

try

```
SELECT *  
FROM users  
WHERE user='...'  
AND  
password = '...'
```

User: *admin*

User	Password
admin	p4\$\$w0rd
pontus	znibb
code	a



```
SELECT *  
FROM users  
WHERE user='admin'  
AND  
password = '...'
```

User: *admin*

Password: *test*

User	Password
admin	p4\$\$w0rd
pontus	znibb
code	a

```
SELECT *  
FROM users  
WHERE user='admin'  
AND  
password = 'test'
```

User: *admin*

Password: *test*

User	Password
admin	p4\$\$w0rd
pontus	znibb
code	a

```
SELECT *  
FROM users  
WHERE user='admin'  
AND  
password = 'test'
```

User: *admin*

Password: *test*

User	Password
admin	p4\$\$w0rd
pontus	znibb
code	a

```
SELECT *  
FROM users  
WHERE user='admin'  
AND  
password = 'test'
```

User: *admin*

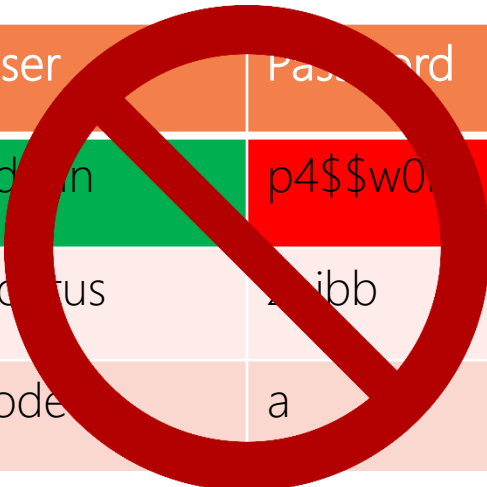
Password: *test*

User	Password
admin	p4\$\$w0rd
pontus	znibb
code	a

```
SELECT *  
FROM users  
WHERE user='admin'  
AND  
password = 'test'
```

User: *admin*

Password: *test*



User	Password
admin	p4\$\$w0rd
postgres	postgres
code	a

```
SELECT *  
FROM users  
WHERE user='admin'  
AND  
password = '...'
```

User: *admin*

Password: ' or 1=1 --

User	Password
admin	p4\$\$w0rd
pontus	znibb
code	a

```
SELECT *  
FROM users  
WHERE user='admin'  
AND  
password = ' ' or 1=1 -- '
```

User: *admin*

Password: ' or 1=1 --

User	Password
admin	p4\$\$w0rd
pontus	znibb
code	a

```
SELECT *  
FROM users  
WHERE user='admin'  
AND  
password = ''  
  
OR  
  
1 = 1-- '
```

User: *admin*

Password: ' *or 1=1 --*

User	Password
admin	p4\$\$w0rd
pontus	znibb
code	a



```
SELECT *  
FROM users  
WHERE user='admin'  
AND  
password = ''  
  
OR  
  
1 = 1-- '
```

User: *admin*

Password: ' *or 1=1 --*

User	Password
admin	p4\$\$w0rd
pontus	znibb
code	a

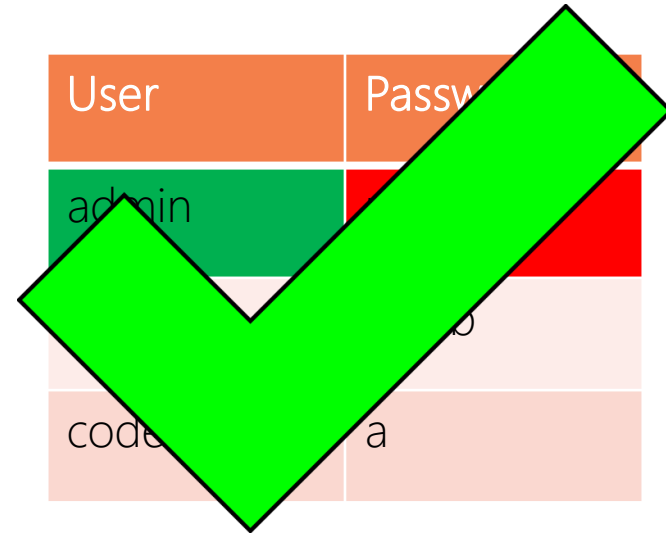
```
SELECT *  
FROM users  
WHERE user='admin'  
AND  
password = ''
```

OR

1 = 1 -- '

User: *admin*

Password: ' *or 1=1 --*



# Olika typer av SQL Injections

- Simple SQLi
- Error based SQLi
- Blind SQLi
  - Boolean SQLi
  - Time based SQLi

# Olika typer av SQL Injections

## Error based SQLi (MySQL)

- ?id=1 and substring(version(),1,1)=5
- ?id=1 and ascii(lower(substr(Version(),1,1)))=51

## Time based SQLi

- ?id=1 AND IF(ASCII(SUBSTRING((SELECT USER()),1,1)))>=100,1,BENCHMARK(2000000,MD5(NOW())) –
- ?id=1 AND IF(ASCII(SUBSTRING((SELECT USER()), 1, 1)))>=100, 1, SLEEP(3)) --

**DEMO**

# Mitigering

*[cheatsheetseries.owasp.org/cheatsheets/...](https://cheatsheetseries.owasp.org/cheatsheets/...)*

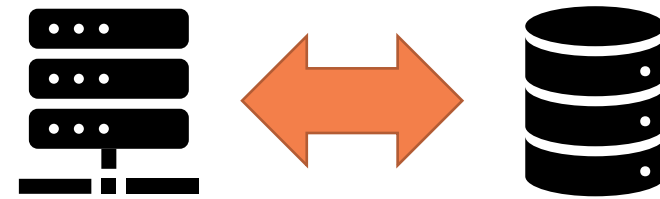
- [SQL\\_Injection\\_Prevention\\_Cheat\\_Sheet.html](#)

# Identifiera ditt problem

Vad är problemet?

Data tolkas som kod

Var finns problemet?



# Lösningsförslag!

## Mitigering

- Sätt rätt kontext
  - "Prepared statement"
  - SpringData

## Minska inverkan

- Minsta privilegium
  - Får inte köra databasfrågor som ändrar data
  - Använd inte standard super admin användaren i DB



# SQL-Injection Exempel

```
Session session = HibernateUtil.getSessionFactory().openSession();  
String sql = "SELECT * FROM User WHERE user_id=" + id;  
Query query = session.createQuery(sql);  
List result = query.list();
```

Sätter vi `id` till `1' OR '1' = '1`

Så blir queryn:

```
SELECT * FROM User WHERE user_id = '1' OR '1' = '1';
```

Vilket ger oss alla användare i databasen

# Prepared statement

```
Session session = HibernateUtil.getSessionFactory().openSession();  
String sql = "SELECT * FROM User WHERE user_id=:id";  
Query query = session.createQuery(sql);  
query.setString("id", id);  
List result = query.list();
```

# SpringData

**CustomerRepository cr**

**cr.findOne(1);**

- Repository
  - Primärnyckel (ID)
  - Returobjekt (Customer)
- Customer (Entitet)
  - ID
  - Namn

# Anpassad hämtning

```
List<Customer> findByLastName(String lastName);
```

# Tester

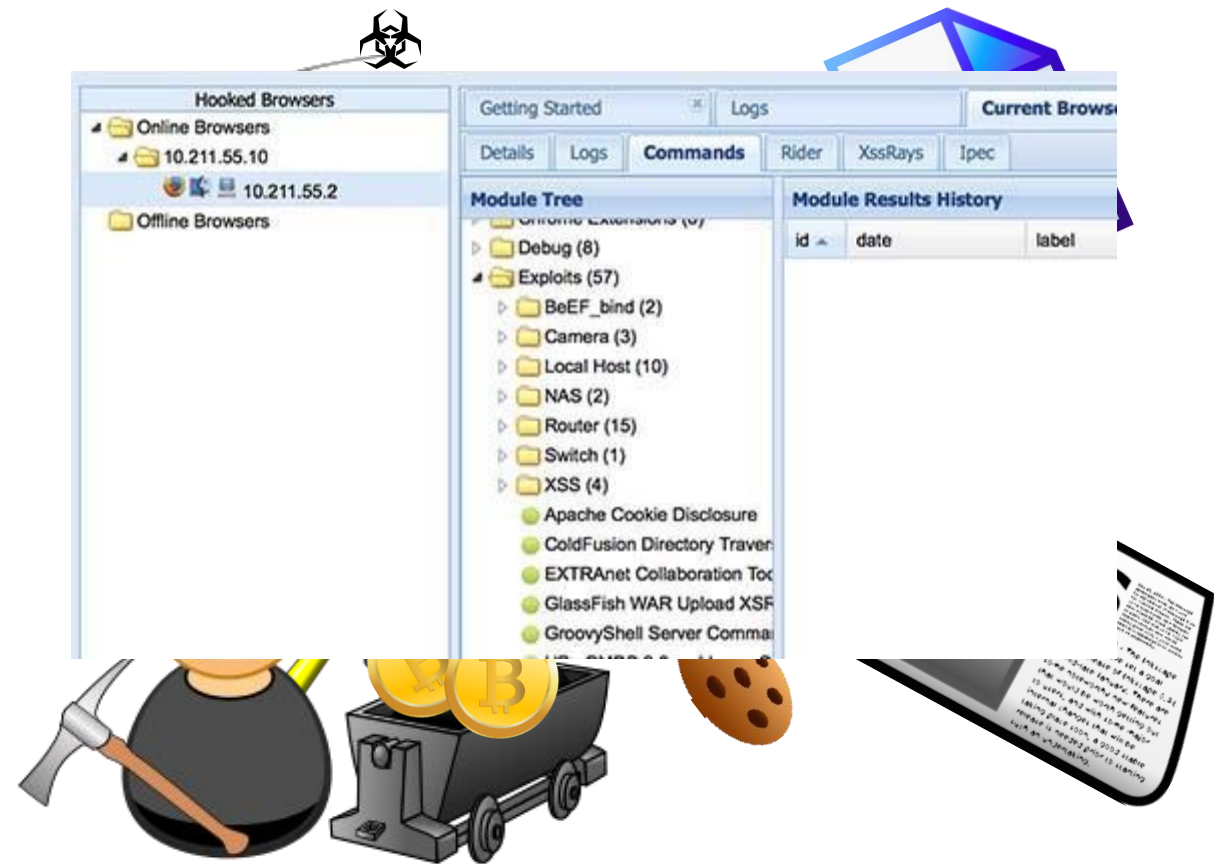
- Vad ska vi testa?
  - Var ska vi testa det?
  - Hur testar vi det?
- Gränssnitt till DB
  - FuzzDB, OWASP Listor, Sqlmap
    - Expertkunskap
  - Kodgranskning

Kahoot!

# XSS

Cross Site Scripting

# Vad kan XSS leda till?







# Olika nyanser av XSS

# Lagrad XSS exempel

## HTML

```
<tr>  
  <td> <b>Message:</b> </td>  
  <td>Text</td>  
</tr>
```

Message: Text

Submit

Message: Text

# Lagrad XSS exempel

## HTML

<tr>

<td> <b>Message:</b> </td>

<td>Mer text

<script>alert('xss')</script> </td>

</tr>

Meddelande: Mer Text<script>alert('xss')</script>

Submit

Message: Mer text

localhost:8080 säger:

xss

☐ Förhindra att den här sidan öppnar ytterligare dialogrutor.

OK

# Reflekterad XSS exempel

## HTML

```
<div id="result">
```

Results for: search term

```
</div>
```

site.com/search?q=search%20term

Search:

# Reflekterad XSS exempel

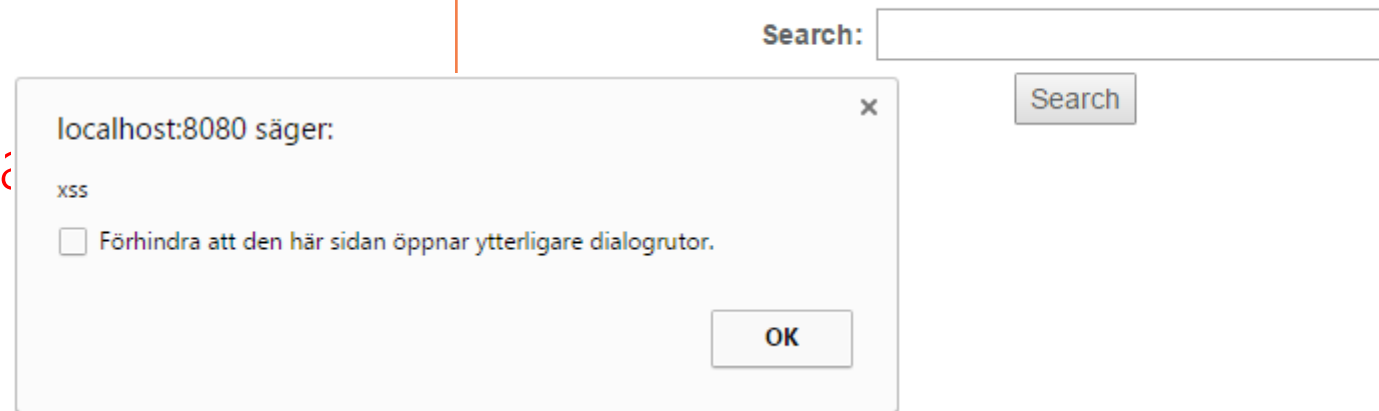
<div id="result">

Result for:

<img onmouseover="alert('xss')"  
src=x>

</div>

site.com/search?q=%3Cimg%20onmouseover%3D%22  
alert('xss')%22%20src%3Dx%3E



# Potentiella XSS vektorer

a  
a2  
abbr  
acronym  
address  
animate  
animatemotion  
animatetransform  
applet  
area  
article  
aside  
audio  
audio2  
b  
base  
basefont  
bdi  
bdo  
bgsound  
big  
blink  
blockquote  
body  
br  
button  
canvas  
caption  
center  
cite

code  
col  
colgroup  
command  
content  
custom tags  
data  
datalist  
dd  
del  
details  
dfn  
dialog  
dir  
discard  
div  
dl  
dt  
element  
em  
embed  
fieldset  
figcaption  
figure  
font  
footer  
form  
frame  
frameset  
h1  
head  
header

hgroup  
hr  
html  
i  
iframe  
iframe2  
image  
image2  
image3  
img  
img2  
input  
input2  
input3  
input4  
ins  
isindex  
kbd  
keygen  
label  
legend  
li  
link  
listing  
main  
map  
mark  
marquee  
menu  
menuitem  
meta  
meter  
multicol  
nav  
nextid  
nobr

noembed  
noframes  
noscript  
object  
ol  
optgroup  
option  
output  
p  
param  
picture  
plaintext  
pre  
progress  
q  
rb  
rp  
rt  
rtc  
ruby  
s  
samp  
script  
section  
select  
set  
shadow  
slot  
small  
source  
spacer  
span  
strike  
strong  
style  
sub

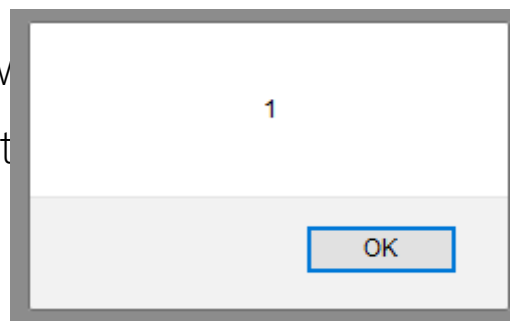
summary  
sup  
svg  
table  
tbody  
td  
template  
textarea  
tfoot  
th  
thead  
time  
title  
tr  
track  
tt  
u  
ul  
var  
video  
video2  
wbr  
xmp

# DOM XSS

Körs i klientens DOM (Document Object Model)

```
1 Select your language:
2
3 <select><script>
4
5 document.write("<OPTION value=1>" + document.location.href.substring(document.location.href.indexOf("default=") + 8) + "</OPTION>");
6 document.write("<OPTION value=2>English</OPTION>");
7
8 </script></select>
```

https://www  
https://www.some.site



...ault=Swedish

...ipt>alert(1)</script>

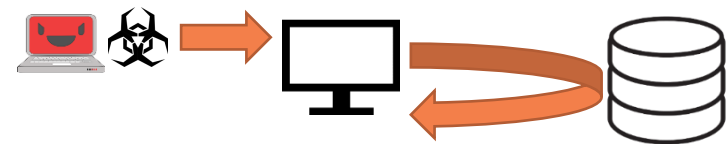
# Reflekterad XSS, Lagrad XSS, DOM XSS

## Reflekterad XSS

Datan skickas till servern och sedan skickas tillbaka till klienten, med en XSS svaret.

XSS sparas inte på servern

Påverkar *oftast* bara 1 klient



## Lagrad XSS

Datan skickas och sparas på servern, exekveras nästa gång som klienten ber om att få läsa datan.

XSS sparas på servern

Kan påverka många klienter



## DOM XSS

Datan behöver inte skickas till servern utan kan injiceras direkt i klientens DOM

XSS sparas inte på servern

Påverkar bara 1 klient





**DEMO**

# Mitigering

*[cheatsheetseries.owasp.org/cheatsheets/...](https://cheatsheetseries.owasp.org/cheatsheets/)*

- [Cross\\_Site\\_Scripting\\_Prevention\\_Cheat\\_Sheet.html](#)
- [Content\\_Security\\_Policy\\_Cheat\\_Sheet.html](#)

# Vad kan man göra för att skydda sig?

- Mitigering

Output Encoding  
Sanitization

- Minska attackytan

Content Security Policy  
Sätt httpOnly på cookies

# Output Encoding

- Om indatan inte ska tolkas som HTML så används output encoding
- Encoding översätter farliga karaktärer till ofarliga
- Hur vet vi vilka karaktärer som är farliga?

Input: `x onerror=alert(document.cookie)`

```
<tr>
  <td><b>Message:</b></td>
  <td><%= input %></td>
</tr>
```

```
<tr>
  <img src=<%= input %> alt="profilbild">
</tr>
```

&	&amp;
<	&lt;
>	&gt;
"	&quot;
'	&#x27;

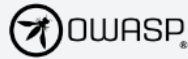
# Output Encoding

- Vi behöver kontextberoende encoding!
- OWASP Java Encoder
- Olika ramverks egna encoding

```
<tr>
```

```
  <img src= <%= Encode.forHtmlAttribute(input)  
  %> alt="profilbild">
```

```
</tr>
```

OWASP®

PROJECTS CHAPTERS EVENTS ABOUT

Search OWASP

**OWASP Java Encoder**[Main](#) [How to Use](#) [Numbers](#) [Grave Accent](#) [Template Literals](#) [Deploy](#) [News](#)**About**

The OWASP Java Encoder is a Java 1.5+ simple-to-use drop-in high-performance encoder class with no dependencies and little baggage. This project will help Java web developers defend against Cross Site Scripting!

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts (primarily JavaScript) are injected into otherwise trusted web sites. One of the primary defenses to stop Cross Site Scripting is a technique called Contextual Output Encoding. WARNING: Please note that XSS prevention requires other defensive strategies besides encoding! For more information, please read the [Cross Site Scripting prevention cheatsheet](#).

We actively track project issues and seek to remediate any issues that arise. The project owners feel this project is stable and ready for production use and are seeking project status promotion.

Happy Encoding!

# Ramverks inbyggda skydd

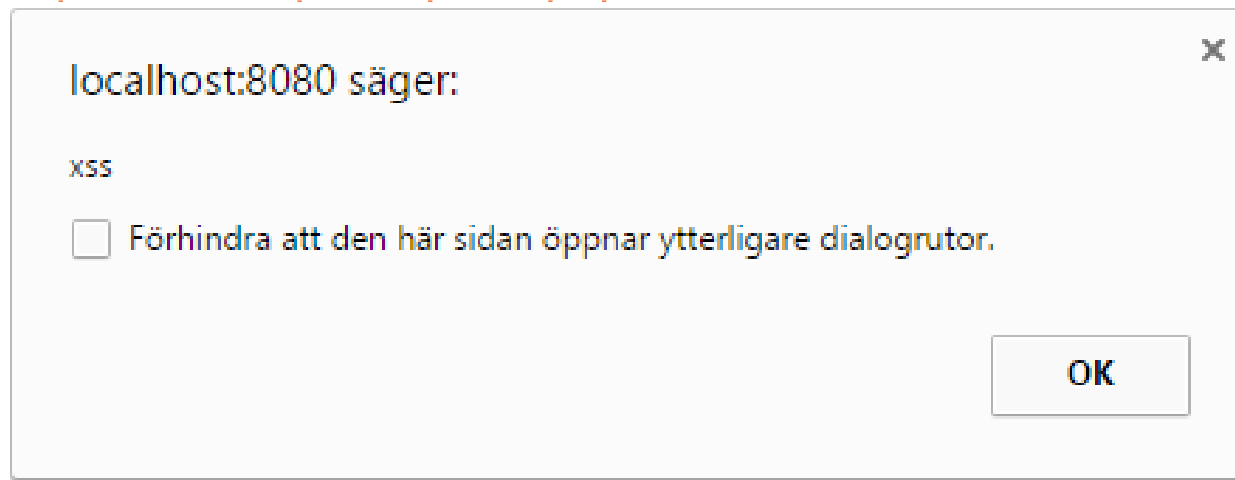
`<p th:text="\${name}">Test</p>`

`<p>{name}</p>`

`name= <script>alert(1)</script>`

`&lt;script&gt;alert(1)&lt;/script&gt;`

# Ramverks i



`<p th:utext="{name}">Test</p>`

`<script>alert("xss")</script>`

`<p dangerouslySetInnerHTML={name} />`

# DOMPurify

bower package 2.0.12 npm package 2.0.12 Build and Test passing downloads 2M/month gzip size 6.84 kB install size 503 kB



<https://github.com/cure53/DOMPurify>

DOMPurify is a DOM-only, super-fast, uber-tolerant XSS sanitizer for HTML, MathML and SVG.

DOMPurify is written in JavaScript and works in all modern browsers (Safari, Opera (15+), Internet Explorer (10+), Edge, Firefox and Chrome - as well as almost anything else using Blink or WebKit).

DOMPurify is written by security people who have vast background in web attacks and XSS. Fear not. For more details please also read about our [Security Goals & Threat Model](#). Please, read it. Like, really.

`DOMPurify.sanitize('<img src=x onerror=alert(1)//>'); // becomes `

`DOMPurify.sanitize('<svg> <g/onload=alert(2)//><p>'); // becomes <svg> <g> </g> </svg>`

`DOMPurify.sanitize('<p>abc<iframe//src=jAva&Tab;script:alert(3)>def</p>'); // becomes <p>abcdef</p>`

`DOMPurify.sanitize('<math> <mi//xlink:href="data:x,<script>alert(4)</script>">'); // becomes <math> <mi> </mi> </math>`

`DOMPurify.sanitize('<TABLE> <tr> <td>HELLO</tr> </TABL>'); // becomes <table> <tbody> <tr> <td>HELLO</td> </tr> </tbody> </table>`

`DOMPurify.sanitize('<UL> <li> <A HREF=//google.com>click</UL>'); // becomes <ul> <li> <a href="//google.com">click</a> </li> </ul>`



# HttpOnly och Content Security Policy

- HttpOnly: cookies kan inte kommas åt av javascript

Set-Cookie: \_\_Host session=...; Secure; HttpOnly;

Content-Security-Policy: script-src https://example.com/

<script src="https://not-example.com/js/library.js"> </script>

<script>  
 var inline = 1;  
</script>

Content-Security-Policy: script-src 'nonce-2726c7f26c'

<script nonce="2726c7f26c">  
 var inline = 1;  
</script>

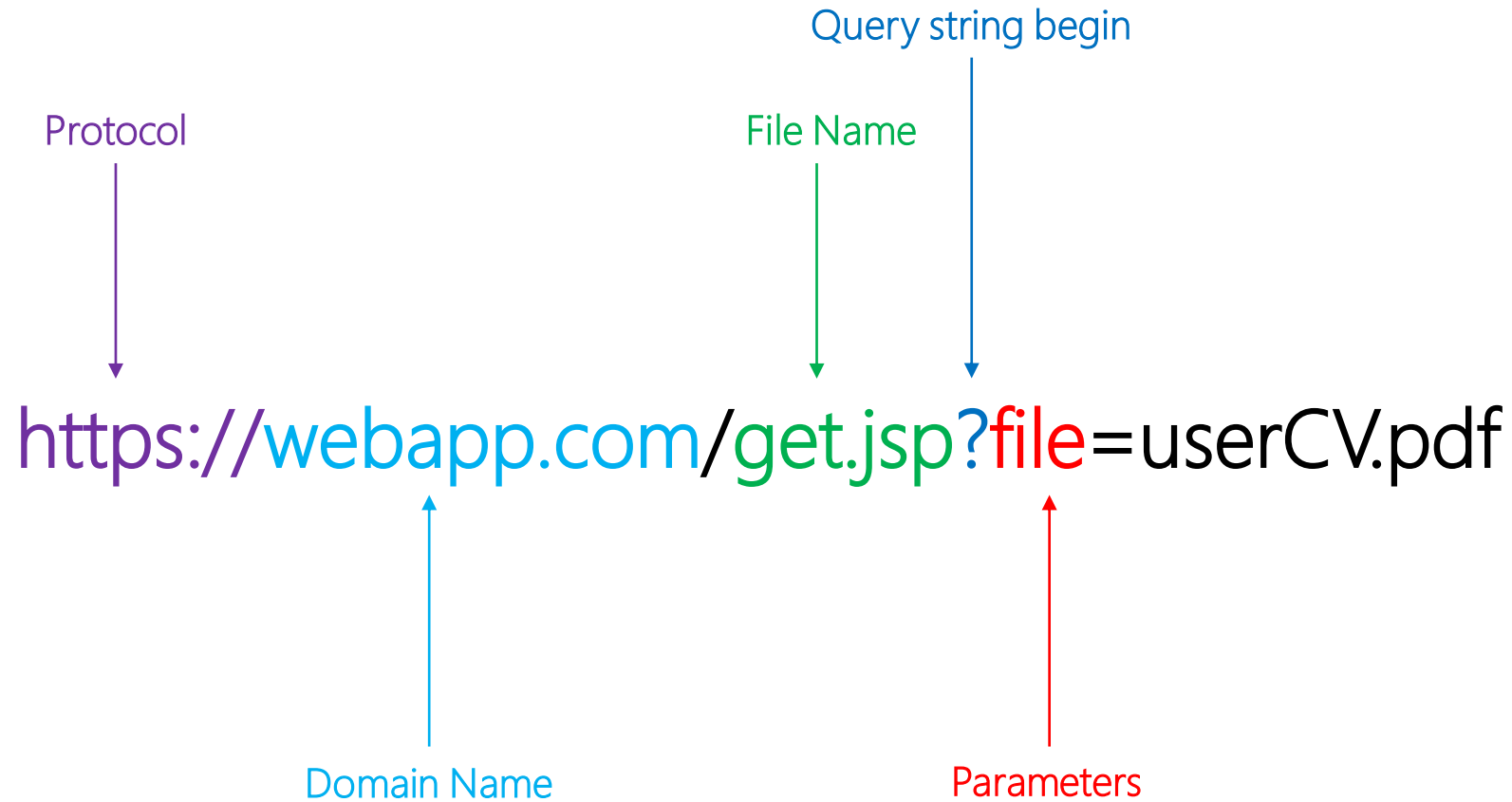
# Tester

- Vad ska vi testa?
  - Vart ska vi testa det?
  - Hur testar vi det?
- XSS
  - Från källa till sänka, in i DOM:en
  - FuzzDB, OWASP Listor

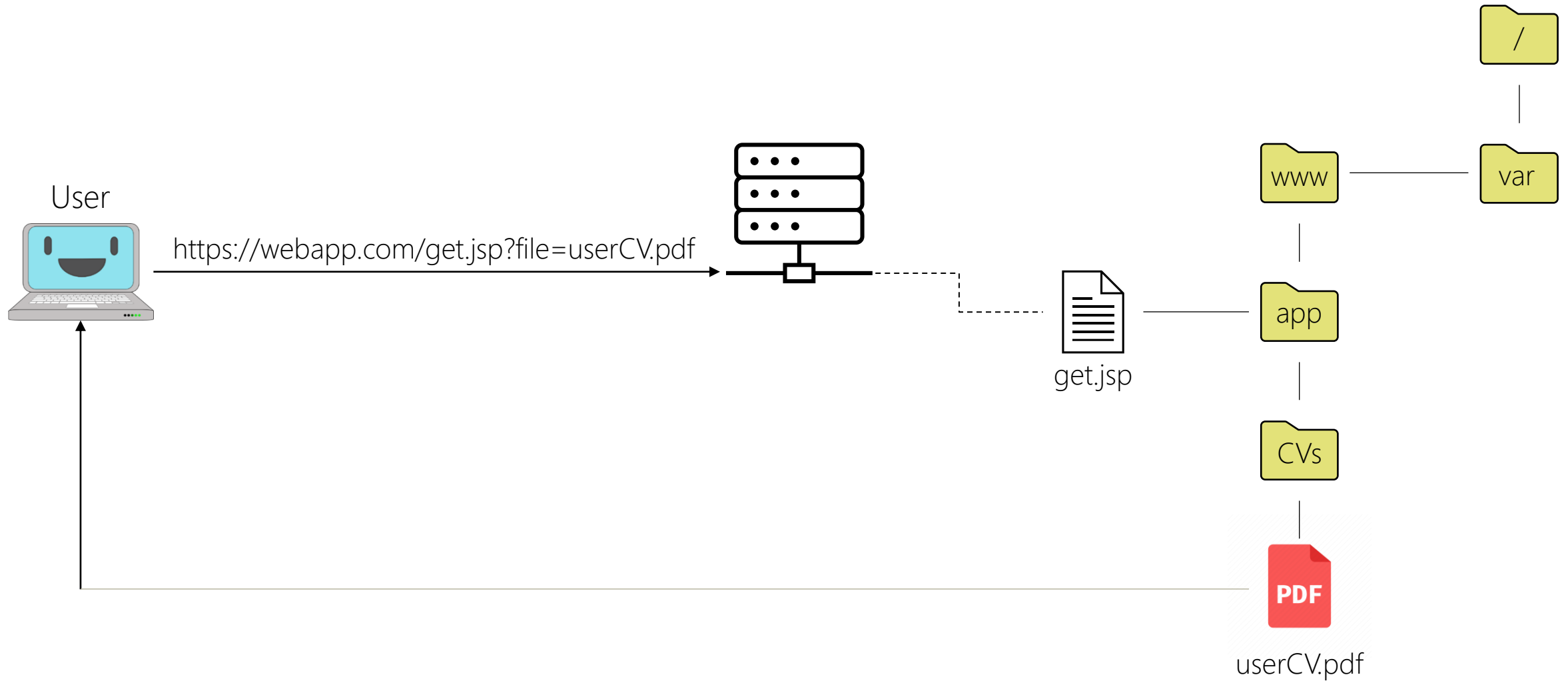
Kahoot!

# File Inclusion

# Vad är en File Inclusion?



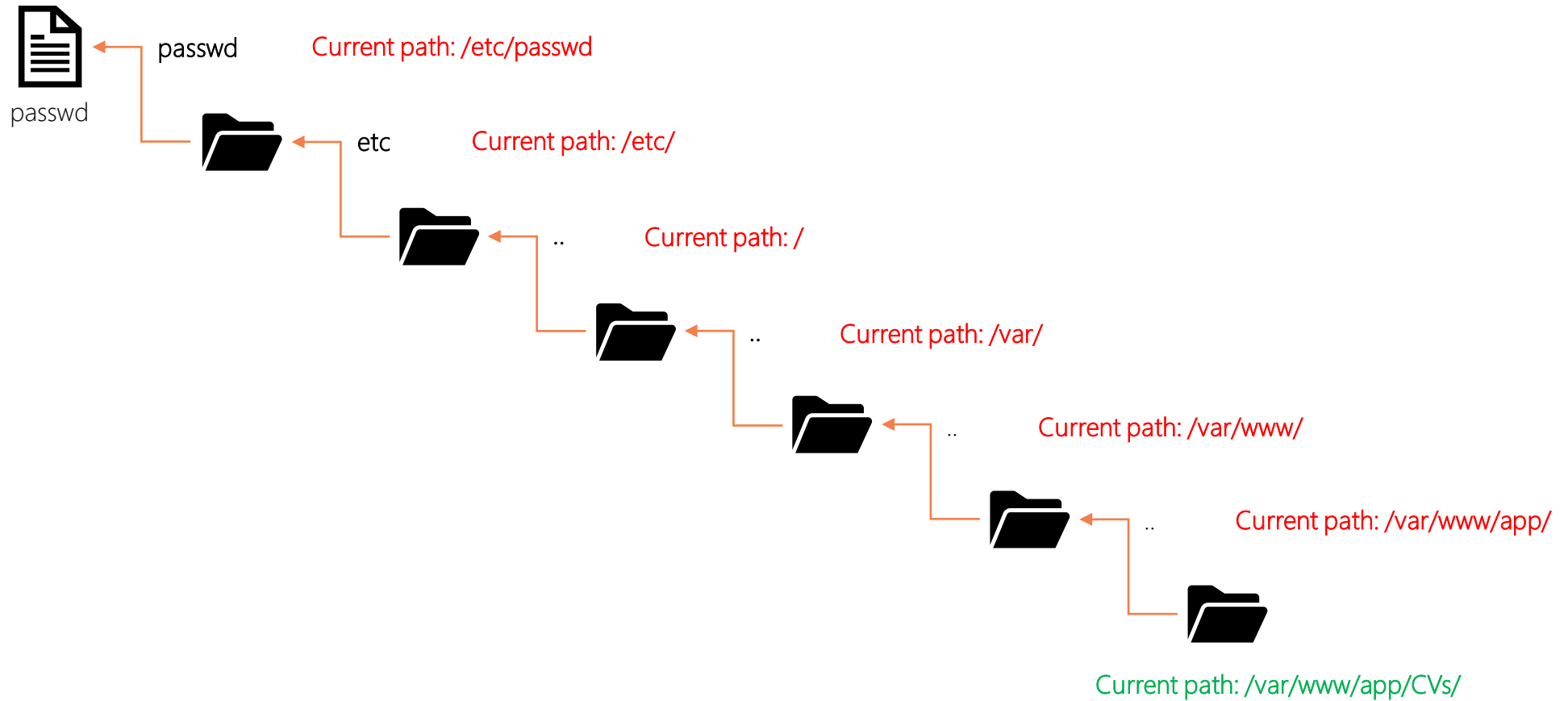
# Vad är File Inclusion?



# File Inclusion sårbarheter

- Path Traversal
- Local File Inclusion (LFI)
- Remote File Inclusion (RFI)

# Vad är Path Traversal?





# Vad är Path Traversal?

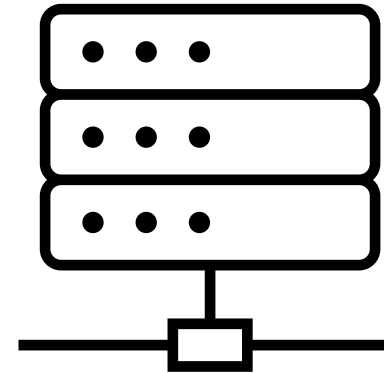
<https://webapp.thm/get.jsp?file=../../../../windows/win.ini>

<https://webapp.thm/get.jsp?file=../../../../boot.ini>

<https://webapp.com/get.jsp?file=../../../../etc/passwd>



root:x:0:0:root:/root:/bin/bash  
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
bin:x:2:2:bin:/bin:/usr/sbin/nologin  
sys:x:3:3:sys:/dev:/usr/sbin/nologin  
sync:x:4:65534:sync:/bin:/bin/sync  
games:x:5:60:games:/usr/games:/usr/sbin/nologin  
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin  
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin  
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin  
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin  
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin  
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin  
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin  
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin



# JSP Path Traversal

```
File file = new File(BASE_DIRECTORY, userInput);
```

```
If (file.getCanonicalPath().startsWith(BASE_DIRECTORY)) {
```

```
    // process file
```

```
}
```

# Path Traversal: OS-filer att testa med

- /etc/issue
- /etc/profile
- /etc/version
- /etc/passwd
- /etc/shadow
- /root/.bash\_history
- /var/log/dmmessage
- /var/mail/root
- /root/.ssh/id\_rsa
- /var/log/apache2/access.log
- C:\boot.ini

## Vad är Local File Inclusion (LFI)

<https://webapp.com/index.php?lang=EN>



Warning: include(languages/EN.php): failed to open stream:  
No such file or directory in /var/www/html/THM-4/index.php on line 12

# Vad är Local File Inclusion (LFI)

`https://webapp.com/index.jsp?lang=../../../../etc/passwd`



Warning: `include(languages/etc/passwd)`: **failed** to open stream:  
No such file or directory in `/var/www/html/THM-5/index.php` on line 15



`Include(languages/etc/passwd)`

# Vad är Local File Inclusion (LFI)

```
<%  
    String p = request.getParameter("lang");  
    @include file="<%= "languages/" + p + ".jsp"%>"  
%>
```

---

```
<?PHP  
    include($_GET["lang"]);  
?>
```

# JavaScript Local File Inclusion

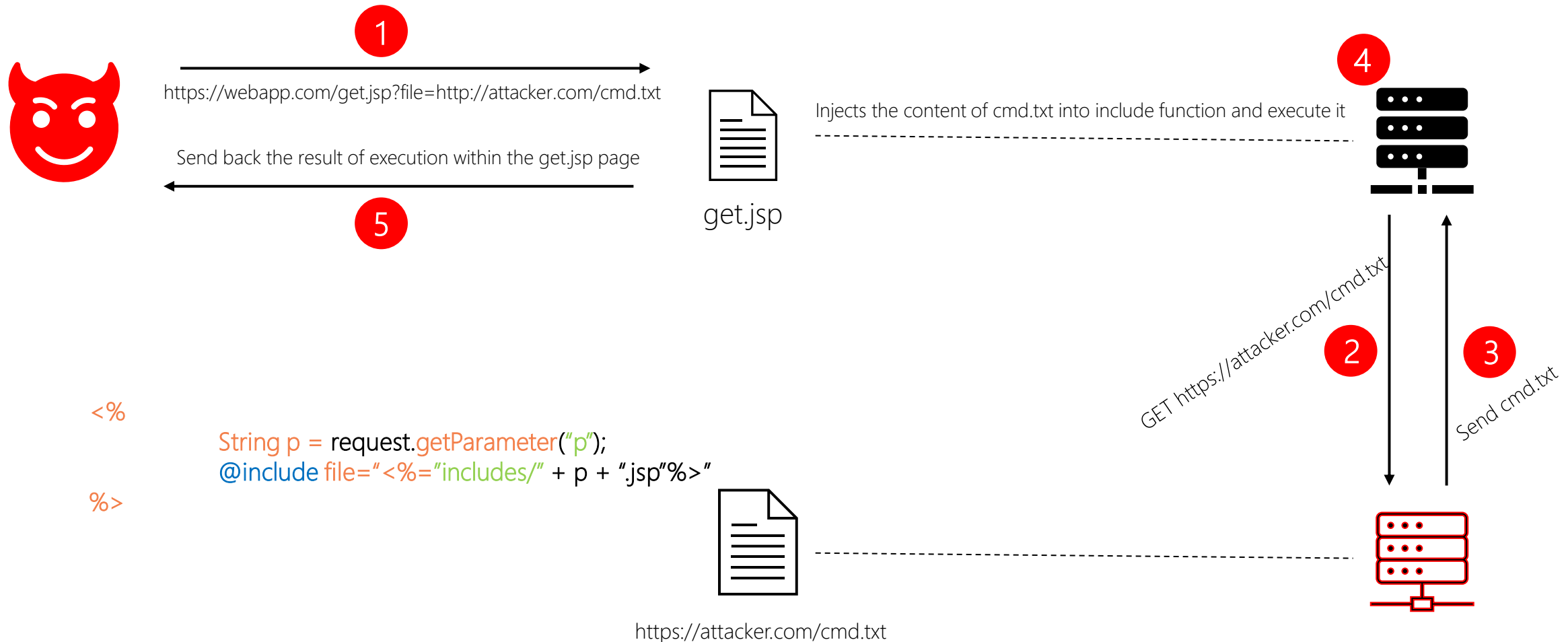
```
const basePath = "public/html"  
const inputPath = req.query.path // user input  
  
const finalPath = path.join(basePath, inputPath)
```



inputPath = "../.env"

# Vad är Remote File Inclusion (RFI)

Allow\_url\_fopen: ON





**DEMO**

# Mitigering

*[cheatsheetseries.owasp.org/cheatsheets/...](https://cheatsheetseries.owasp.org/cheatsheets/...)*

- [Input\\_Validation\\_Cheat\\_Sheet.html](#)

# Vad kan man göra för att skydda sig?

- Lägg paths till filer i en databas och låt användaren skicka in id istället, eller whitelist specific filer
- Lita inte på användaren, validera indata
- Håll ramverk uppdaterade så att kända sårbarheter inte kan utnyttjas

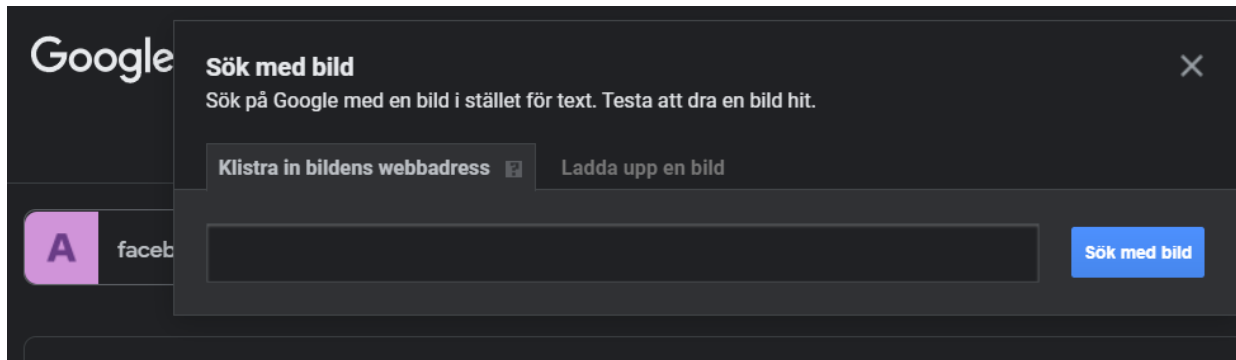
The background is a solid teal color. There are several white geometric shapes: a horizontal rectangle on the left, a vertical rectangle in the center, and a small horizontal rectangle at the bottom center. A thin orange horizontal bar is at the top, and a thin orange vertical bar is on the right.

Kahoot!

# Filuppladdningssårbarheter

# Filuppladdningar finns överallt!

- Profilbild
- Dokumentuppladdning
- Kundtjänst
- ...



## Ladda upp filer till supporten

Här kan du ladda upp en eller flera filer till vår support.

Du kan bara ladda upp filer om du kommit överens med oss om det och fått ett ärendenummer.



• Ärendenummer:

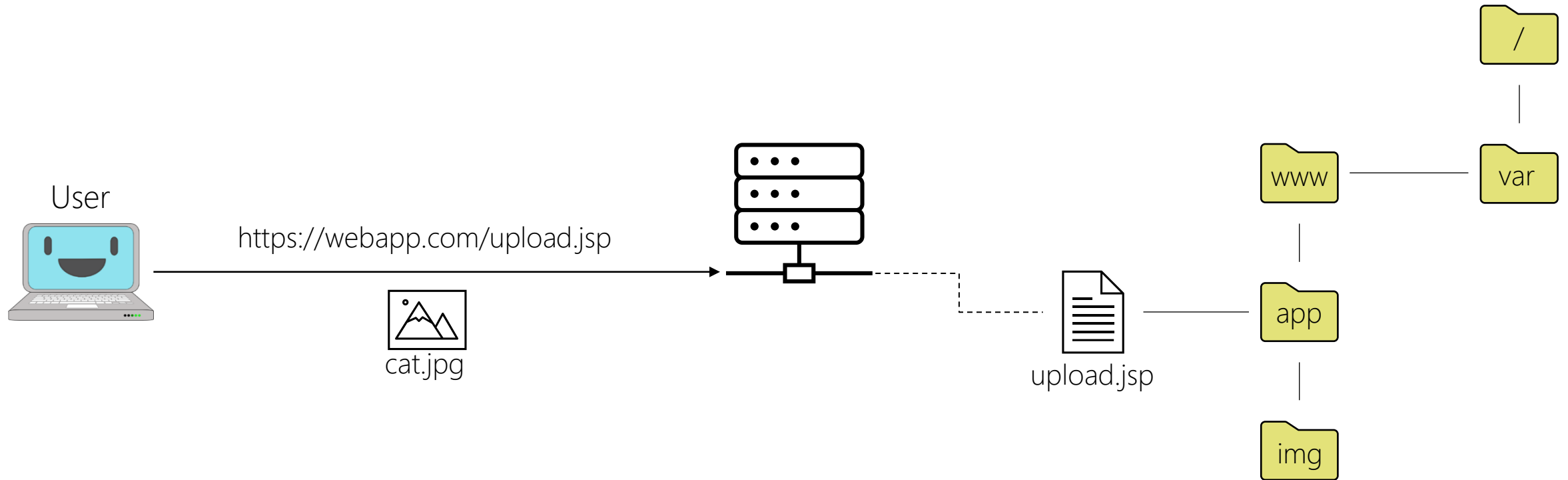
•



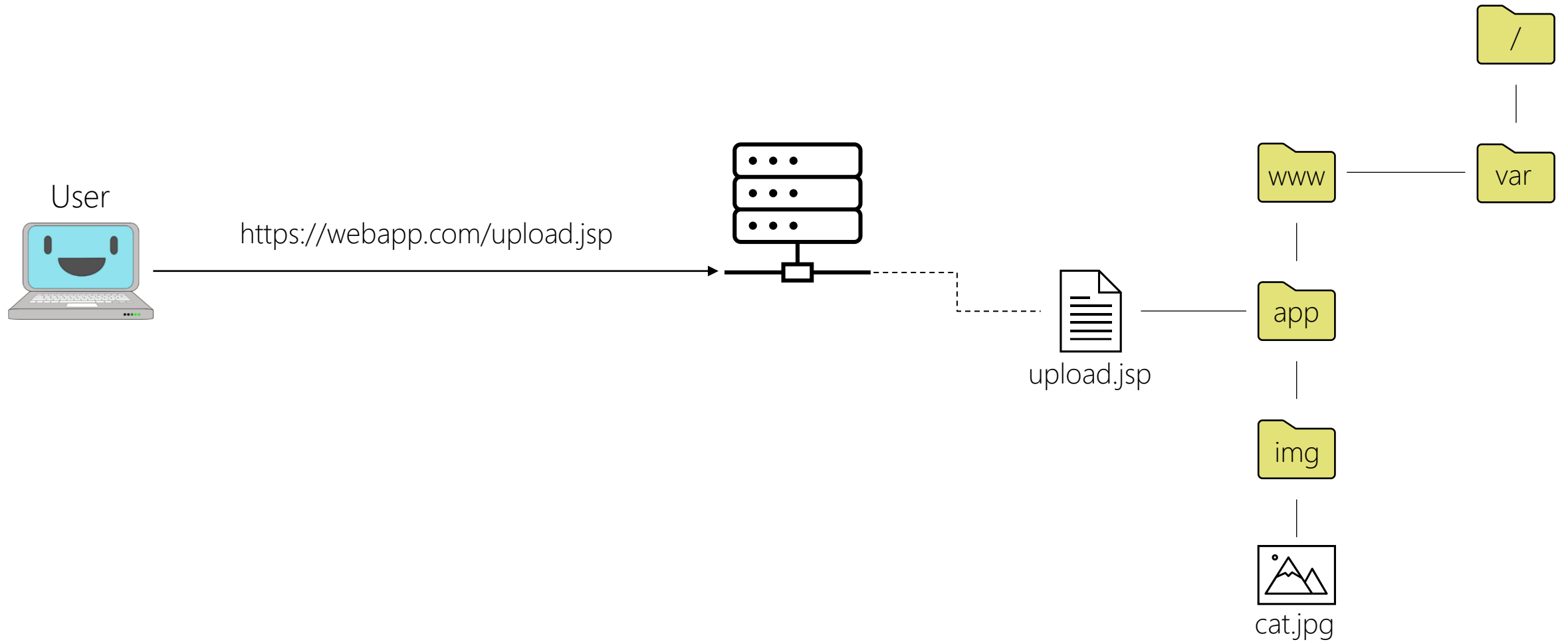
(Empty list)

(Upload files)

# Vanlig filuppladdning



# Vanlig filuppladdning





# Filuppladdning attackvektorer

- Ta upp resurser (Denial of Service)
  - Stora filer
  - Zipbomb
- Skriva över filer
- Remote Code Execution (RCE)

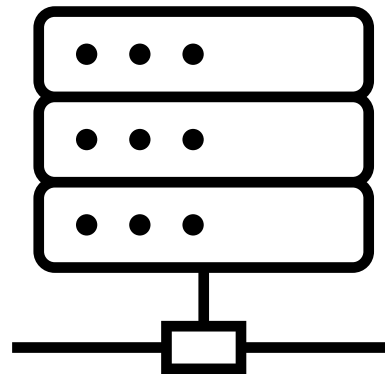
# Uppladdning för att skriva över en fil



cat.jpg (Hacker version)

Ladda upp fil med samma namn som en befintlig fil

GET /?submit=success



cat.jpg  
(Server version)

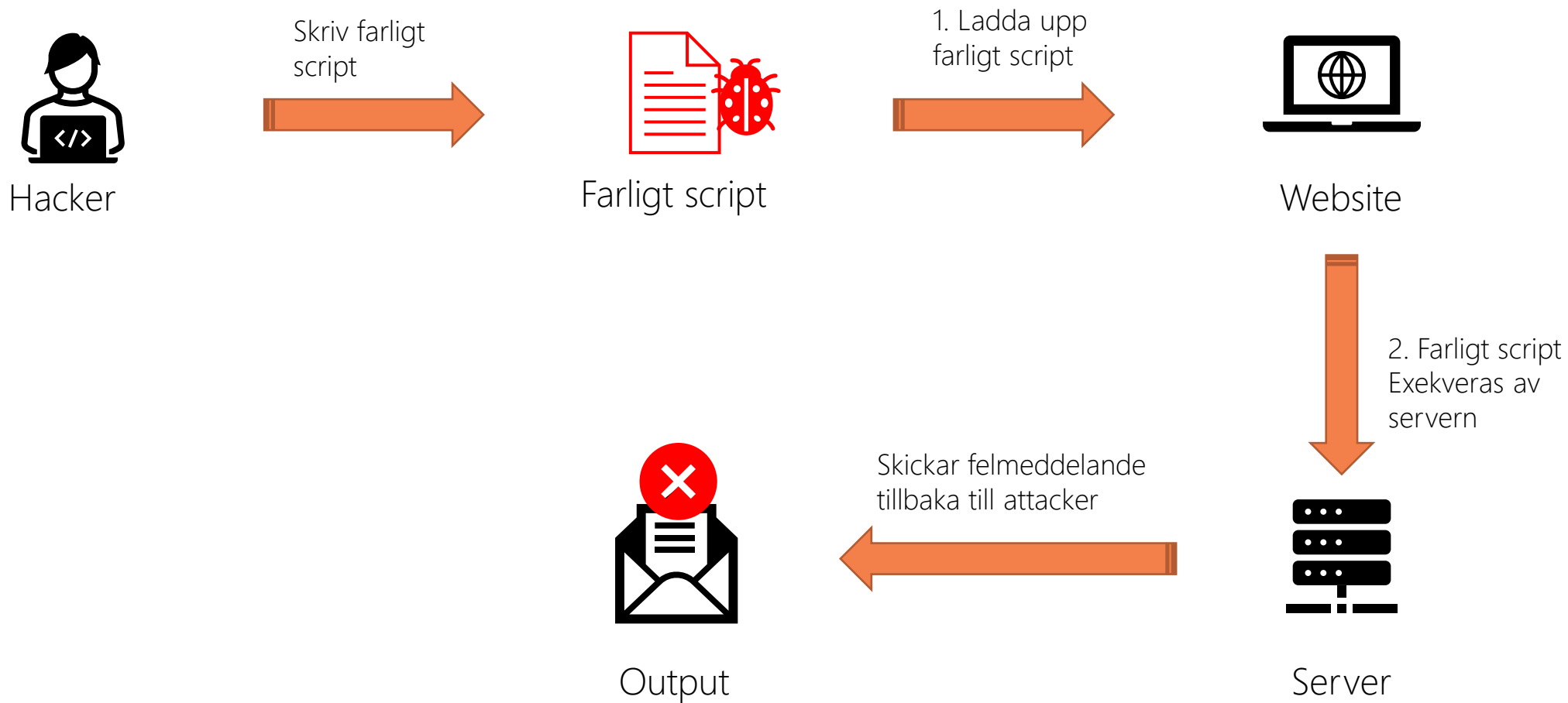


cat.jpg  
(Hacker version)

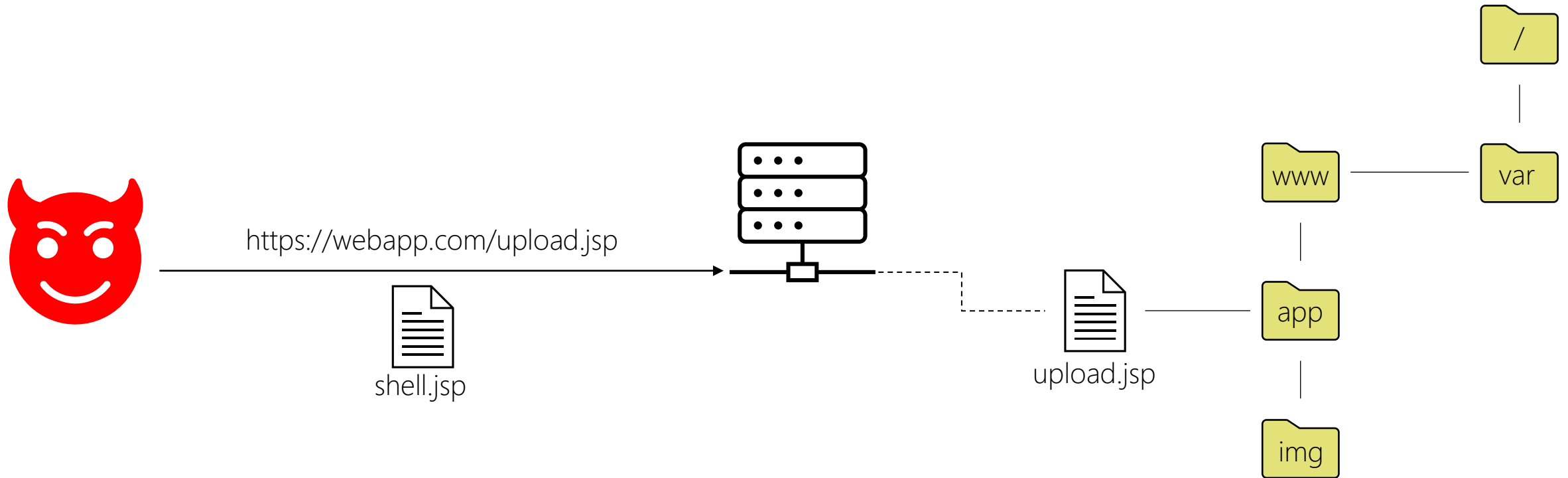
# Uppladdning för att skriva över en fil



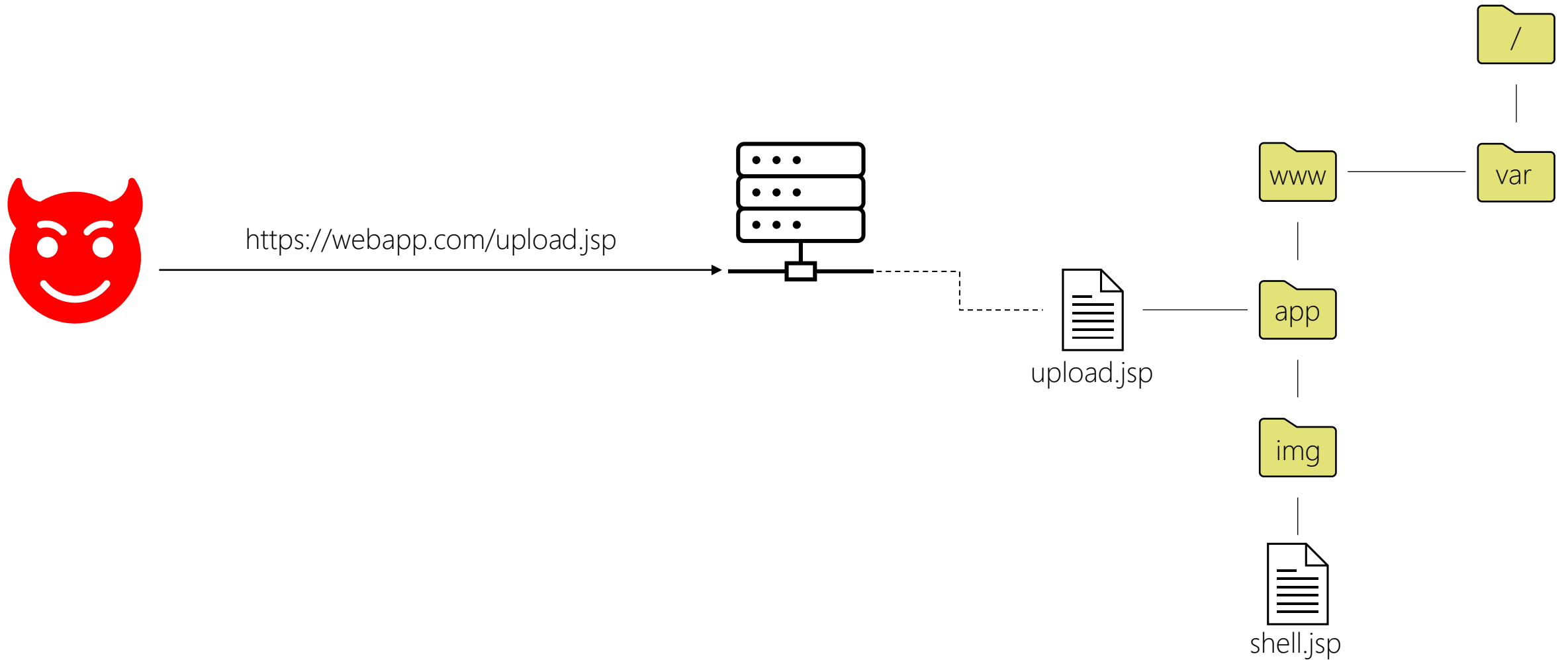
# Remote Code Execution



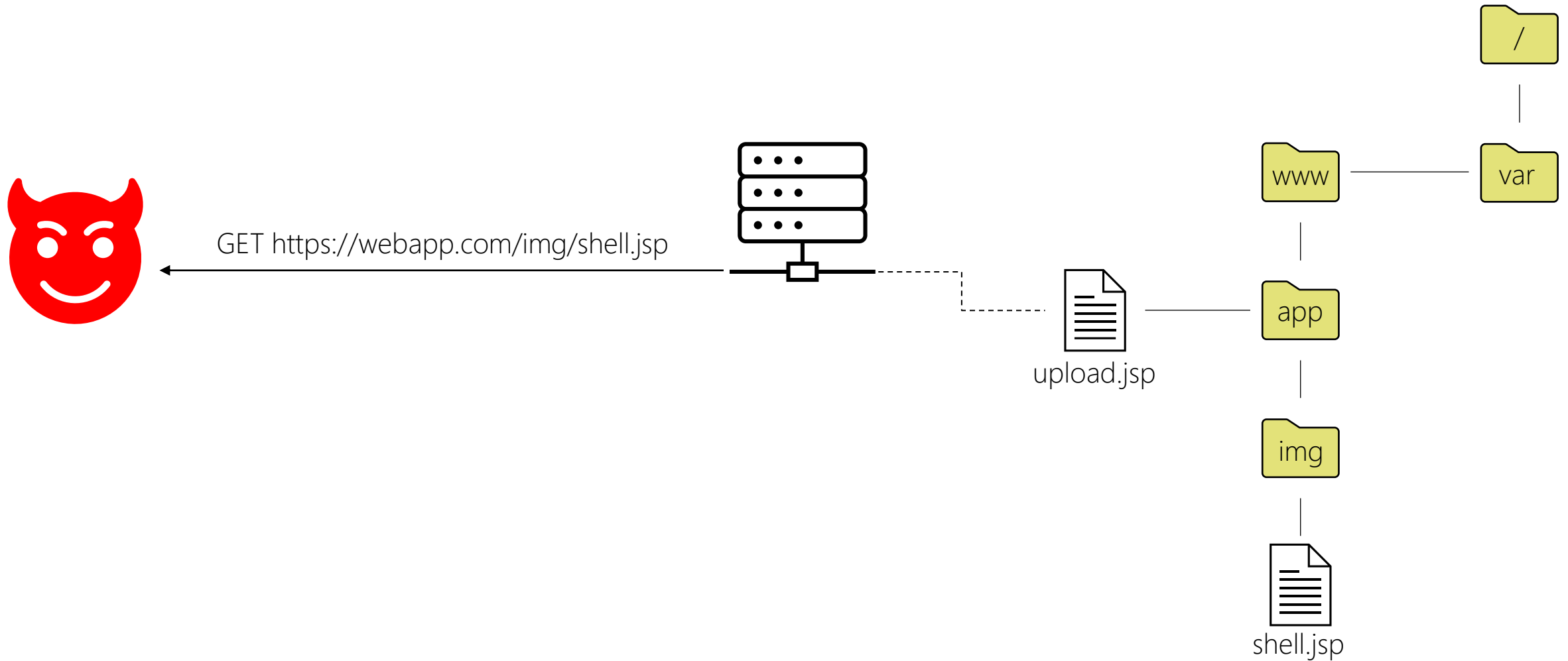
# Filuppladdning RCE



# Filuppladdning RCE



# Filuppladdning RCE



# Filuppladdning RCE: ImageTragick

- Farlig bild laddas upp
- Bildramverket ImageMagick ska skala om bilden
  - Bilden parsas av ImageMagick
  - Bildfilens innehåll triggar en bug i parser → RCE





**DEMO**

# Mitigering

*[cheatsheetseries.owasp.org/cheatsheets/...](https://cheatsheetseries.owasp.org/cheatsheets/...)*

- [File\\_Upload\\_Cheat\\_Sheet.html](#)

# Filuppladdning mitigering

- Ta upp resurser (Denial of Service)
  - Unzippa inte filer som laddas upp
  - Begränsa och validera filens storlek
  - Begränsa antalet uppladdningar per användare och tid, inga oautentiserade uppladdningar!
- Skriva över filer
  - Låt inte användaren styra filnamnet, ersätt det med ett eget slumpat filnamn
- Remote Code Execution (RCE)
  - Validera filinnehåll
    - Filändelse
    - Magic number
    - Antivirusscan
  - Sandboxad hantering kan begränsa skadan av en farlig fil

# Ersätta filnamn

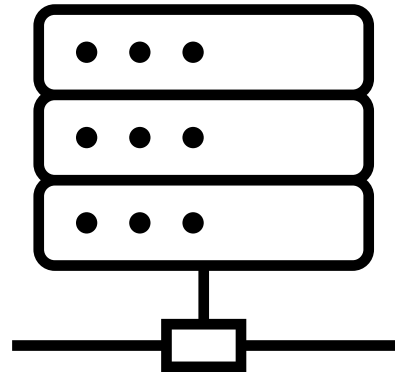
NOT FILTERED



File.jpg (Hacker version)

Upload similar file name as file name hosted in the server

GET /?submit=success



File.jpg  
(Server version)



File.jpg  
(Hacker version)

# Ersätta filnamn

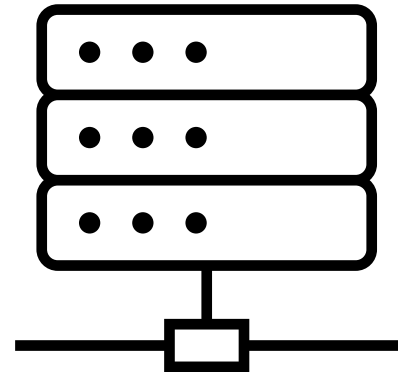
FILTERED



File.jpg (Hacker version)

Upload similar file name as file name hosted in the server

GET /?submit=success



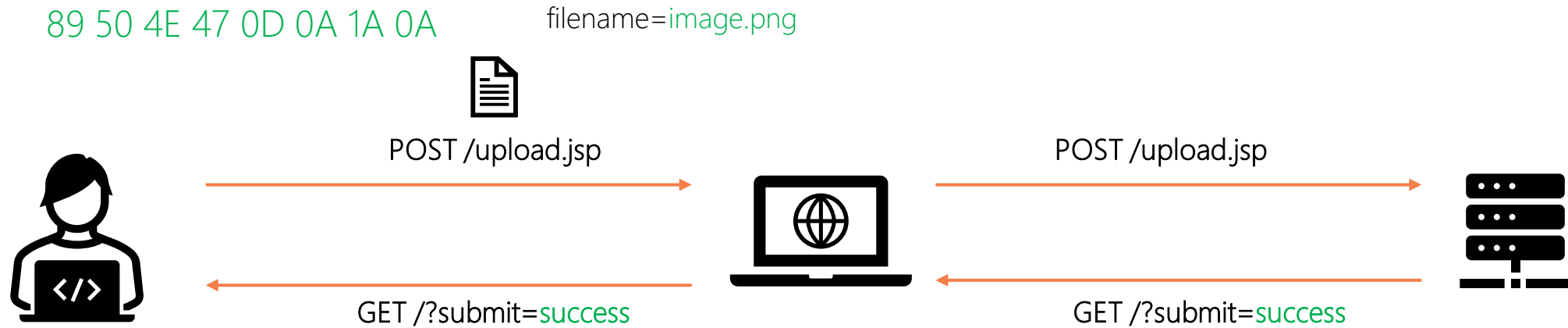
5901b04885e9d051788a8dd77695ee6e.jpg  
(random filename)



File.jpg  
(Hacker version)

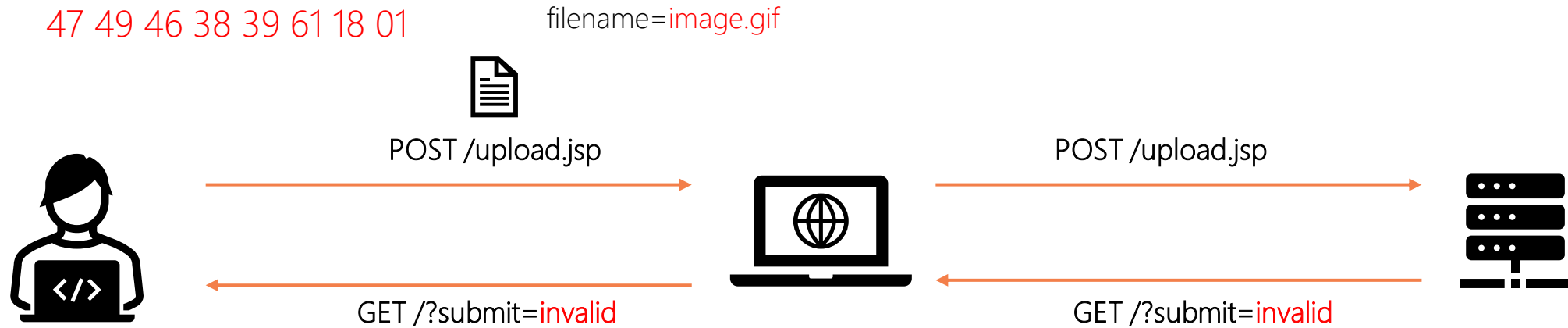


# Filtrera på filtyp (magic number validering)



```
MagicNumberFilter javaClassFileFilter = MagicNumberFileFilter(new byte[] {(byte) 0x47, (byte) 0x49, (byte) 0x46, (byte) 0x38, (byte) 0x39, (byte) 0x61, (byte) 0x18, (byte) 0x01});
```

# Filtrera på filtyp (magic number validering)



```
MagicNumberFilter javaClassFileFilter = MagicNumberFileFilter(new byte[] {(byte) 0x47, (byte) 0x49, (byte) 0x46, (byte) 0x38, (byte) 0x39, (byte) 0x61, (byte) 0x18, (byte) 0x01});
```

The background is a solid teal color. There are three white rectangular shapes: a horizontal rectangle on the left, a vertical rectangle in the center, and a horizontal rectangle at the bottom. A thin orange horizontal bar is at the top, and a thin orange vertical bar is on the right.

Kahoot!



# OWASP CheatSheets + OWASP Application Security Verification Standard



# OWASP

CHEAT SHEET  
SERIES PROJECT

Life is too short • AppSec is tough • Cheat!

# OWASP Application Security Verification Standard

Ge *utvecklare och kravställare* en gemensam plattform att utgå ifrån när man ska designa och bygga säker mjukvara

# Vad är OWASP ASVS?

ASVS is a list of application security requirements or tests that can be used by architects, developers, testers, security professionals, tool vendors, and consumers to define, build, test and verify secure applications. **Provides developers with a list of requirements for secure development.**



## Nivå 1

Minimum nivå för att uppnå en god säkerhetsnivå som skyddar bl.a mot OWASP Top 10. Hanterar inte känslig data. Första steget in i säkerhetsarbetet. Ej kvalificerade hot.



## Nivå 2

För applikationer som hanterar känslig data som t.ex hälsouppgifter eller industrier där integritet är viktigt för att skydda företaget. Hoten är oftast kvalificerade.

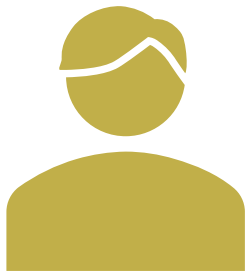


## Nivå 3

För industrier som militär, vård, banker och industriella system. Kräver djupare analys och testning av systemet. Säkerhet finns på flera nivåer, infrastruktur, design, kod, skalbarhet osv. Högsta nivå enligt CIA + AAA modellen. Hoten är avancerade och väldigt kvalificerade.

# Krav

Projektledare



Vi ska bygga ett nytt inloggningsflöde, användare ska kunna logga in och ut.

Välj det som är lämpligt

Google

🔍 Vad är en hash algorithm?



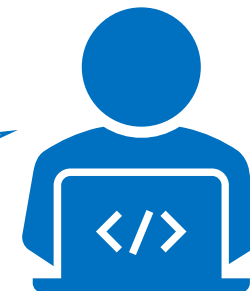
Sök på Google

Jag har tur

Ok, vad är det för krav på lösenord och hash-algoritm?

Alright, det fixar vi!

Utvecklare



Google

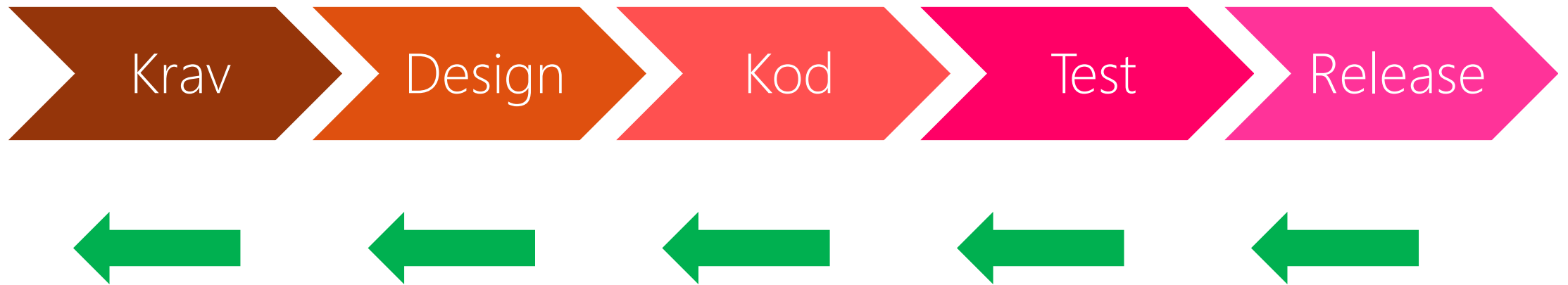
🔍 Vilken hash algorithm bör man använda?



Sök på Google

Jag har tur

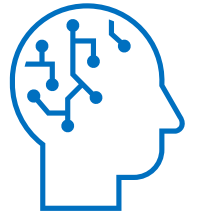
# SDLC - Push it to the left!



# Områden som ASVS täcker



- V1: Architecture, Design and Threat Modeling Requirements
- V2: Authentication Verification Requirements
- V3: Session Management Verification Requirements
- V4: Access Control Verification Requirements
- V5: Validation, Sanitization and Encoding Verification Requirements
- V6: Stored Cryptography Verification Requirements
- V7: Error Handling and Logging Verification Requirements
- V8: Data Protection Verification Requirements
- V9: Communications Verification Requirements
- V10: Malicious Code Verification Requirements
- V11: Business Logic Verification Requirements
- V12: File and Resources Verification Requirements
- V13: API and Web Service Verification Requirements
- V14: Configuration Verification Requirements



# ASVS V2.1 Password Security Requirements

#	Description	L1	L2	L3	CWE	NIST §	#	Description	L1	L2	L3	CWE	NIST §
2.1.1	Verify that user set passwords are at least 12 characters in length. <a href="#">(C6)</a>	✓	✓	✓	521	5.1.1.2		breached, the application must require the user to set a new non-breached password. <a href="#">(C6)</a>					
2.1.2	Verify that passwords 64 characters or longer are permitted. <a href="#">(C6)</a>	✓	✓	✓	521	5.1.1.2	2.1.8	Verify that a password strength meter is provided to help users set a stronger password.	✓	✓	✓	521	5.1.1.2
2.1.3	Verify that passwords can contain spaces and truncation is not performed. Consecutive multiple spaces MAY optionally be coalesced. <a href="#">(C6)</a>	✓	✓	✓	521	5.1.1.2	2.1.9	Verify that there are no password composition rules limiting the type of characters permitted. There should be no requirement for upper or lower case or numbers or special characters. <a href="#">(C6)</a>	✓	✓	✓	521	5.1.1.2
2.1.4	Verify that Unicode characters are permitted in passwords. A single Unicode code point is considered a character, so 12 emoji or 64 kanji characters should be valid and permitted.	✓	✓	✓	521	5.1.1.2	2.1.10	Verify that there are no periodic credential rotation or password history requirements.	✓	✓	✓	263	5.1.1.2
2.1.5	Verify users can change their password.	✓	✓	✓	620	5.1.1.2	2.1.11	Verify that "paste" functionality, browser password helpers, and external password managers are permitted.	✓	✓	✓	521	5.1.1.2
2.1.6	Verify that password change functionality requires the user's current and new password.	✓	✓	✓	620	5.1.1.2	2.1.12	Verify that the user can choose to either temporarily view the entire masked password, or temporarily view the last typed character of the password on platforms that do not have this as native functionality.	✓	✓	✓	521	5.1.1.2
2.1.7	Verify that passwords submitted during account registration, login, and password change are checked against a set of breached passwords either locally (such as the top 1,000 or 10,000 most common passwords which match the system's password policy) or using an external API. If using an API a zero knowledge proof or other mechanism should be used to ensure that the plain text password is not sent or used in verifying the breach status of the password. If the password is	✓	✓	✓	521	5.1.1.2							



# Vad kan ASVS hjälpa till med?

## Utvecklare

- Guideline för hur man bygger säkra applikationer
- Kan utgå direkt från kraven när man skriver kod
- Automatisera säkerhetstester (enhets- och integrations tester)
- Konkreta åtgärder
- Inte lika bra som CheatSheets

## Projektägare

- Ställa krav på säkerhetskontroller för känsliga funktioner
- Dokumentera exakta skyddsåtgärder som implementerats
- Acceptanstester

# Hur ser testning ut idag?

Test	Förväntat Resultat	Godkänt
Skicka ett POST request till /api/v1/login med följande data: {"user": "admin", "pw": "123"}	Du blir inloggad och får: {"accesstoken": "12345asdf"}	Ja
Skicka ett POST request till /api/v1/logout med följande data: {"accesstoken": "12345asdf"}	Du blir utloggad och får status kod 200 tillbaka	Ja
Skicka ett PUT request till /api/v1/changepw med följande data: {"newpw": "asdf", "verifypw": "asdf"}	Du får HTTP status kod 200 tillbaka	Ja

## Vad har vi testat?

- Att man kan logga in
- Att man kan logga ut
- Att man kan byta lösenord

## Vad har vi *inte* testat/verifierat?

- Lösenord komplexitet
- Säkerhetskrav för byte av lösenord
- V2.1 Password Security Requirements

# Hur kan tester se ut i framtiden?

Test	Förväntat Resultat	Godkänt
[...]	[...]	[...]
Följer lösenordshantering V2.1 Password Security Requirements?	Ja	Ja
Är REST API:et byggt enligt V13.2 RESTful Web Service Verification Requirements	Ja	Ja

## Vad har vi uppnått?

- Kravställare kan inkludera konkreta säkerhetskontroller som ska implementeras
- Utvecklare får guidelines på vad som behövs för att bygga säkra applikationer
- Acceptanstester tydliggör vilka säkerhetskontroller som förväntas
- Beslutsfattare kan enkelt få en överblick vilka kontroller som finns
  - Ger en bild av den generella säkerheten kring en applikation

# Tack för oss!

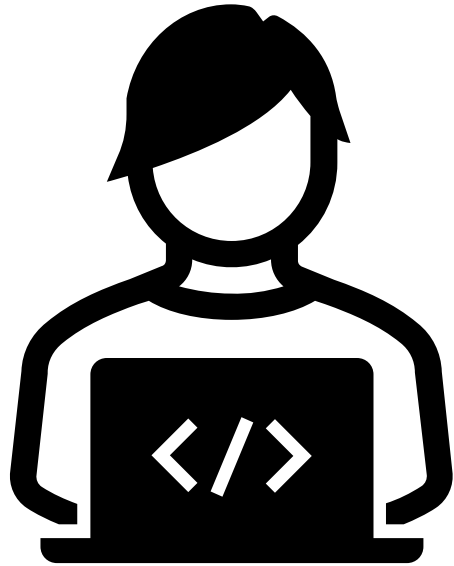
Fortsatta studier?

- Referenslista
  - PortSwigger Academy
  - Cheatsheets
  - ASVS



<https://tinyurl.com/3877bv5c>

# Feedbackformulär



<https://forms.gle/A9gksLXzTDBjYiis6>