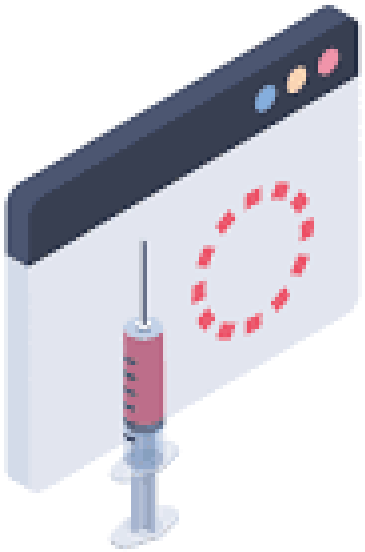
Bezbednost Aplikacija
Ranjivosti zasnovane na ubacivanju (injection) koda

Predmet: ZAŠTITA PODATAKA U KOMUNIKACIONIM MREŽAMA

Predavač: dr Dušan Stefanović

Ubacivanje koda

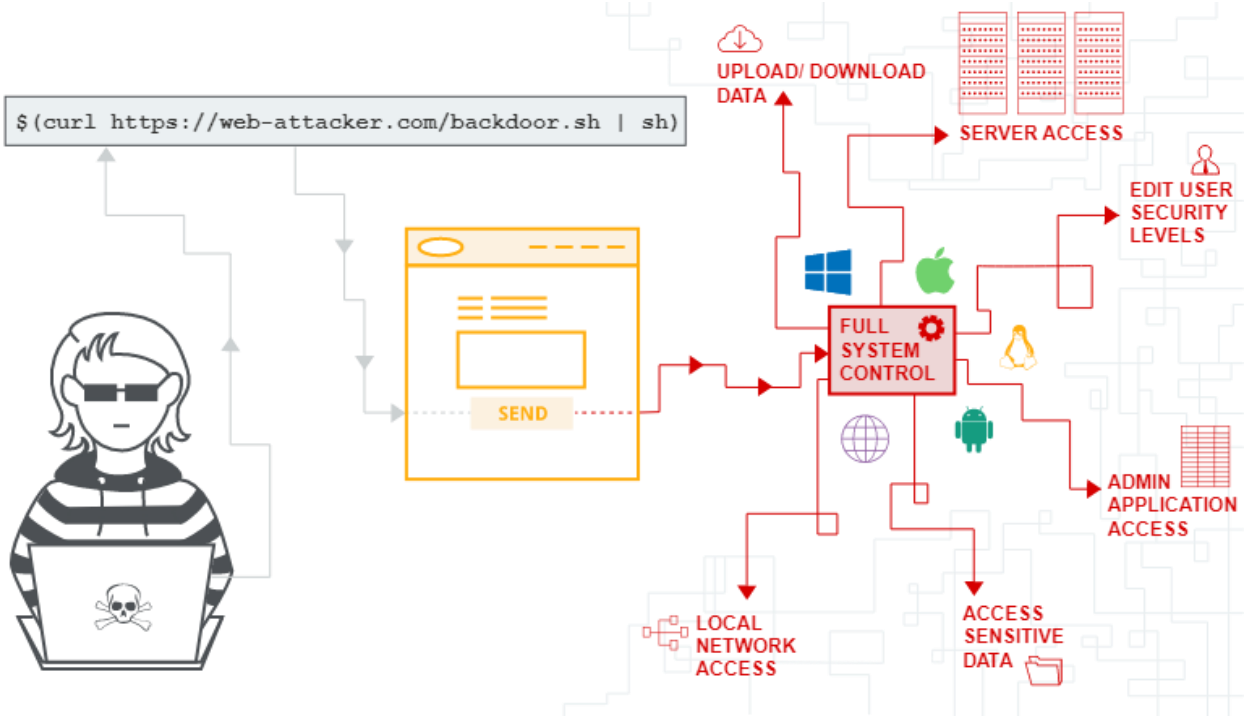


- Interaktivne web aplikacije koriste unos od korisnika, obrađuju ga i rezultat vraćaju korisniku
- Aplikacija koja je ranjiva na ubacivanje koda (injektiranje) prihvata unos od korisnika bez pravilne ili bilo kakve validacije i obrađuje ga
- Zlonamerni unos može da natera aplikaciju da izvrši zadatke za koje aplikacija nije programirana
- Ranjivost zasnovana na ubacivanju se koristi za dobijanje pristupa komponenti kojoj aplikacija šalje podatke za izvršenje nekog zadatka

Komponente	**Ranjivist ubacivanjem koda**
Operativni sistem	Injektiranje (ubacivanje) komande
Baza podataka	SQL/NoSQL injektiranje
Web pretraživač (browser)	Prenos malicioznog skripta preko sajta i njegovo izvršenje
LDAP direktorijum	LDAP injektiranje
XML	XPATH/XML External Entity injektiranje

Ubacivanje OS komande

- Interaktivne web aplikacije mogu da koriste skript jezike za pozivanje funkcionalnosti unutar operativnog sistema na web server za obradu unosa koji je primljen od korisnika
- Napadač može da pokuša da dobije obrađen unos u komandnoj liniji zaobilaženjem filtera za validaciju unosa.



Ubacivanje OS komande

Skripta koja je ranjiva na umetanje komande

Skripta prihvata IP adresu i šalje ICMP (ping) poruke na određenu adresu

U skripti ne postoji validacija unosa pre prosleđivanje ip parametra od strane korisnika

```
<?php
        $target = $_REQUEST[ 'IP' ];
        $cmd = shell_exec ( 'ping –c 3' . $target );
        $html .= '<pre>' .$cmd. '</pre>';
        echo $html;
?>
```

Ubacivanje OS komande

Zlonamerni korisnik može da pošalje zahtev:

http://test/info.php?**ip=127.0.0.1; uname -a**

Skripta koristi vrednost unosa korisnika i nadovezuje unos korisnika na ping –c 3

Finalna komanda koja se prosleđuje serveru na izvršenje prikazuje pored rezultata ping komande i verziju OS-a:

ping –c 127.0.0.1; uname -a

```
<?php
        $target = $_REQUEST[ 'IP' ];
        $cmd = shell_exec ( 'ping –c 3' . $target );
        $html .= '<pre>' .$cmd. '</pre>';
        echo $html;
?>
```

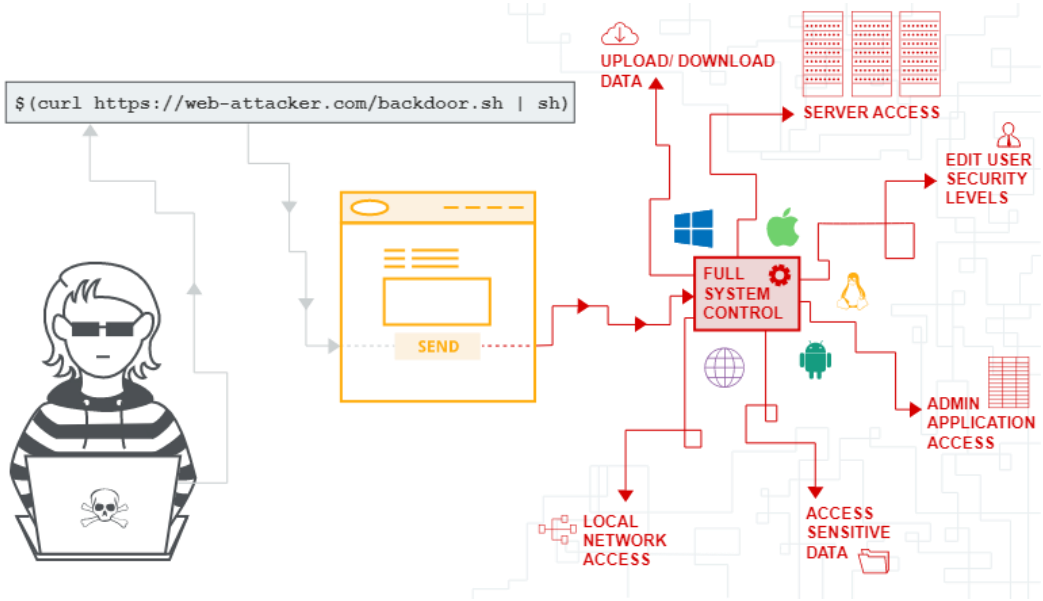
Ubacivanje OS komande

Umetnuta komanda će se pokrenuti koristeći privilegiju web servera

Web server treba da se pokrene sa ograničenim privilegijama ali i u tom slučaju napadač može da dođe do korisnih informacija

Injektiranje može da se upotrebi da bi server

- preuzeo i izvršio zlonamerne fajlove wget komanda
- dobio udaljenu komandnu liniju za pristup serveru



Ubacivanje OS komande - identifikovanje parametara za umetanje podataka

Prvi korak je potvrda da aplikacija dozvoljava umetanje koda tj. da interaguje sa komandnom linijom pozadinskog OS-a.

Drugi korak je ispitivanje različitih parametara u aplikaciji i provera odgovora

Aplikacija može da koristi jedan od načina za prosleđivanje parametara ka web serveru

- **GET metoda** slanjem parametara kroz URL putanju
- **POST metoda** šalje parametre u HTTP telo
- **HTTP zaglavlje** se često koristi za identifikaciju krajnjih korisnika i prikaz prilagođenih informacija korisniku. Polja koja su bitna za proveru injektiranja komande su:

  Cookies

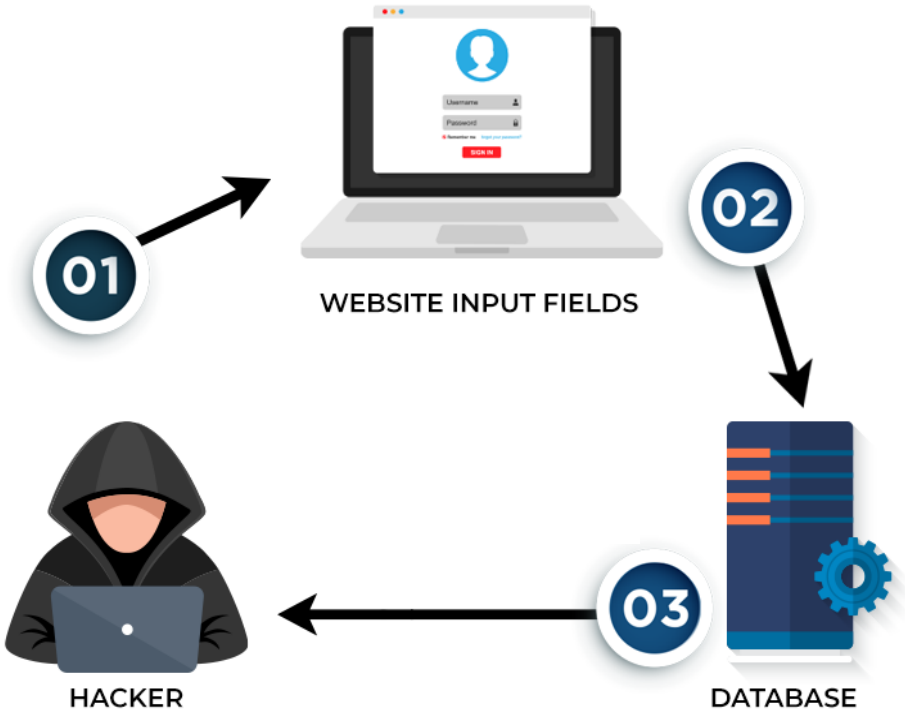
  User-Agent

  Referrer

  X Forwarded-For

SQL Injection

- SQL je jezik koji se koristi u radu sa podacima u relacionim bazama podataka
- SQL injection je vrsta napada na web aplikacije koja omogućuje napadaču da ubaci zlonamerni SQL kod u upit koji se izvršava na bazi podataka.
- To se obično postiže korišćenjem input polja na web stranici koje omogućuju korisniku da unese podatke koje aplikacija zatim koristi u SQL upitu.
- SQL injection je postupak umetanja SQL iskaza koji mogu da menjaju, brišu ili čitaju podatke iz baze podataka.



SQL Injection – vrste napada

Napadači koriste SQL injection ranjivost Web aplikacija za:

- krađu osetljivih podataka iz baze podataka
  - Napadač može izvršiti SELECT upit koji će vratiti osjetljive podatke kao što su korisnička imena i lozinke.
- menjanje ili uništavanje podataka
  - Napadač može izvršiti DELETE upit koji će obrisati podatke iz baze podataka.
  - Napadač može izvršiti UPDATE upit koji će izmijeniti podatke u bazi podataka.
- izvršavanje administratorskih komandi na bazi podataka
- preuzimanje kontrole Web aplikacije

SQL Injection – posledice napada

Primarne posledice SQL Injection napada su:

- Poverljivost jer obično SQL baze podataka sadrže osetljive podatke.
- Autentifikacija ako se korisnička imena i lozinke čuvaju kao čist tekst.
- Autorizacija ako se podaci o autorizaciji nalaze u SQL bazi podataka,
- Integritet podataka, ili čak izbrisati te podatke.

SQL Injection – primer napada 1

Primarne posledice SQL Injection napada su:

- Web stranica omogućuje korisnicima da pretražuju bazu podataka koja sadrži popis proizvoda.
- Na ovoj stranici postoji input polje za pretraživanje proizvoda.
- SQL upit koji se izvršava na bazi podataka

```
SELECT *
FROM products
WHERE name = 'input_value';
```

- Napadač može iskoristiti ovu ranjivost tako da umetne zlonamerni SQL kod u input polje.

SQL Injection – primer napada 1

Umesto da unese naziv proizvoda, napadač može uneti niz karaktera:

' OR 1=1;--

Kada se ovaj niz karaktera umetne u SQL upit, upit postaje:

```
SELECT *
FROM products
WHERE name = ' ' OR 1=1;--';
```

-- je oznaka za komentar, sve što sledi nakon ovog karaktera biće ignorisano.

Ovaj SQL upit će vratiti sve proizvode iz baze podataka, jer je uslov u WHERE klauzuli ispunjen (1=1 je uvek ispunjeno).

SQL Injection – primer napada 2

Primarne posledice SQL Injection napada su:

- Web stranica koja omogućuje korisnicima da se prijave na svoj nalog.
- Na web stranici postoji input polje za unos korisničkog imena i lozinke.
- SQL upit koji se izvršava u bazi podataka izgleda

```
SELECT *
FROM users
WHERE username = 'input_username' AND password = 'input_password';
```

- Napadač može iskoristiti ovu ranjivost tako da umetne zlonamerni SQL kod u input polje.

SQL Injection – primer napada 2

Umesto da unese svoje korisničko ime i lozinku, napadač može uneti sledeći niz karaktera kao korisničko ime:

```
' OR 1=1;--
```

Kada se ovaj niz karaktera umetne u SQL upit, upit postaje:

```
SELECT *
FROM users
WHERE username = '' OR 1=1;--' AND password = 'input_password';
```

-- je oznaka za komentar, sve što sledi nakon ovog karaktera biće ignorisano.

Ovaj SQL upit će vratiti zapise u tabeli korisnika iz baze podataka, jer je uslov u WHERE klauzuli ispunjen (1=1 je uvek ispunjeno).

Napadač sada može pristupiti bilo kojem korisničkom nalogu na web stranici bez poznavanja stvarne lozinke.

SQL Injection – primer napada 3

- Pretpostavimo da postoji web stranica koja omogućuje administratorima da brišu naloge korisnika iz baze podataka.
- Na ovoj stranici postoji input polje za unos korisničkog ID-a koji se briše iz baze podataka.
- SQL upit koji se izvršava u bazi podataka izgleda

DELETE FROM users WHERE id = 'input_id';

- Napadač može iskoristiti ovu ranjivost tako da umetne zlonamerni SQL kod u input polje.

SQL Injection – primer napada 3

Umesto da unese ispravan ID korisnika, napadač može uneti sledeći niz karaktera:

'; DROP TABLE users;--

Kada se ovaj niz karaktera umetne u SQL upit, upit postaje:

DELETE FROM users

WHERE id = ''; DROP TABLE users;--';

Ovaj upit će prvo izbrisati red sa ID-em praznog stringa (jer prva WHERE klauzula nije zadovoljena)

Zatim će izbrisati kompletnu tabelu korisnika (jer druga naredba nije ograničena ni jednom WHERE klauzulom)

Napadač može izbrisati celu tabelu korisnika iz baze podataka.

Odbrana od SQL Injection napada

1. Upotreba parametarizovanih upita
    - Umesto da se koristite stringovi za formiranje SQL upita, koristite se parametrizovani upiti koji omogućuju razdvajanje SQL koda od korisničkih unosa.
    - Na taj način će se korisnički unosi tretirati kao podaci, a ne kao deo SQL koda.
2. Provera korisničkog unosa
    - Validacija korisničkog unosa, filtriranje posebnih karaktera i ograničavanje dozvoljenih karaktera.
3. Ograničena prava pristupa
    - Koristite se odvojeni korisnički nalozi sa različitim privilegijama pristupa bazi podataka.
    - Nalog koji se koristi za čitanje podataka iz baze podataka ne bi trebao imati pristup funkcijama koje omogućuju brisanje i ažuriranje podataka.

Odbrana od SQL Injection napada

4. Ažuriranje i održavanje softvera
5. Primena bezbednih biblioteke i frameworka
    - Mnogi programski jezici imaju biblioteke i framework za zaštitu od SQL injection napada.

Parametarizovani upiti

Parametrizovani upit je način pisanja SQL upita koji koristi parametre umesto konkretnih vrednosti.

Parametri su mesto gde će se kasnije umetnuti vrednosti.

Umesto da se vrednosti ubacuju direktno u SQL upit, koristi se poseban znak koji predstavlja parametar, često "?", a zatim se pre izvršavanja upita vezuju konkretni parametri za ta mesta.

Na taj način, korisnički unosi su tretirani kao podaci, a ne kao deo SQL koda, što smanjuje rizik od SQL injection napada

```php
<?php
// uspostavljanje konekcije sa bazom podataka
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

$conn = new mysqli($servername, $username, $password, $dbname);

// priprema parametarizovanog upita
$stmt = $conn->prepare("SELECT * FROM users WHERE username = ? AND password = ?");
$stmt->bind_param("ss", $username, $password);

// postavljanje parametara
$username = "john_doe";
$password = "my_password";

// izvršavanje upita
$stmt->execute();

// čuvanje rezultata upita
$result = $stmt->get_result();

// ispis rezultata
while ($row = $result->fetch_assoc()) {
  echo "ID: " . $row["id"] . " - Name: " . $row["name"] . " - Email: " . $row["email"] . "<br>";
}

// zatvaranje konekcije
$stmt->close();
$conn->close();
?>
```