Security Testing Course

W: www.sangbui.com

T: @sangsecurity

E: sangbuicom@gmail.com

OWASP
Open Web Application
Security Project

Agenda

- Introducing to OWASP top 10
- Cross-Site Scripting (XSS)
- Sensitive Data Exposure
- SQL Injection
- Hands on Labs
- Q/A

OWASP Top 10

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.]

Cross-Site Scripting

X

- XSS = Cross Site Scripting
- XSS vulnerabilities arise when an application fails to validate the user's input, allowing an attacker to inject malicious code into web pages & viewed by other users.

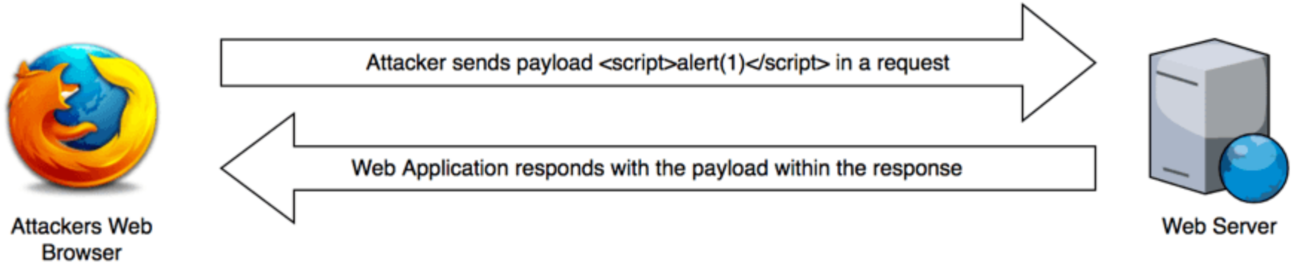
Cross-Site Scripting

- Common web vulnerabilities
- Difficult to identify and remove
- XSS can steal personal information / cookies
- JavaScript it is possible to log all key strokes that a user enters
- Web site defacement

Cross-Site Scripting

- Non-persistent (Reflected XSS)
- Persistent (Stored XSS)
- DOM Based XSS

Cross-Site Scripting


Attacker sends payload <script>alert(1)</script> in a request
Web Application responds with the payload within the response
Attackers Web Browser
Web Server

Non-persistent (Reflected XSS)

Cross-Site Scripting

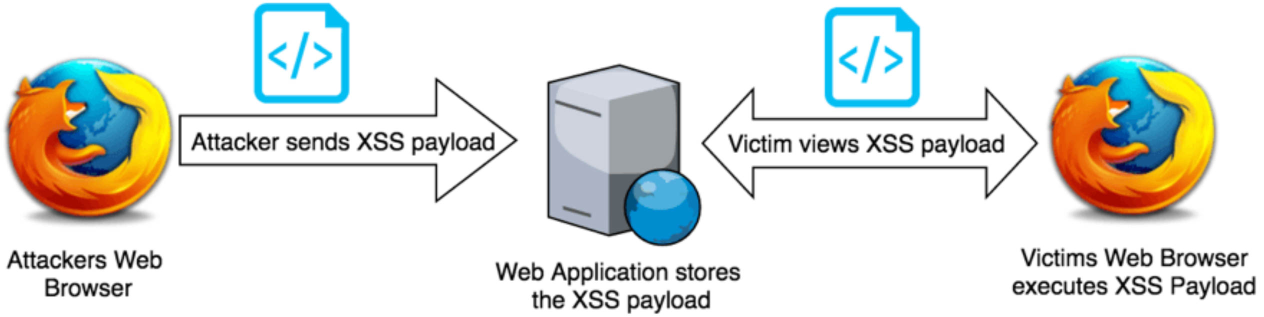
```
http://somepage.com/users.php?register=<script>alert('hacked')</script>
```

Link

Malicious code

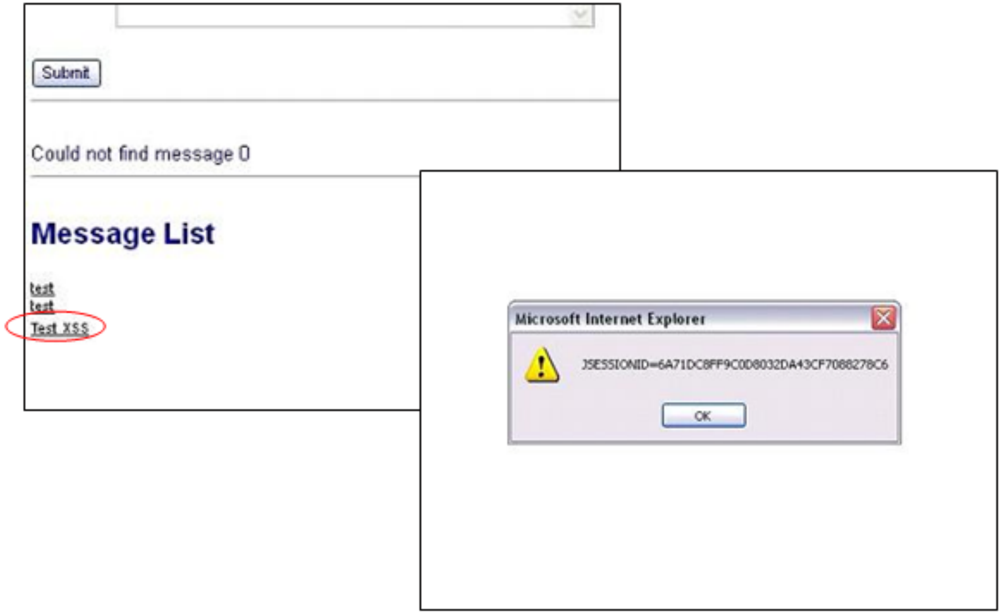
Non-persistent (Reflected XSS)

Cross-Site Scripting


Attacker sends XSS payload
Victim views XSS payload
Attackers Web Browser
Web Application stores the XSS payload
Victims Web Browser executes XSS Payload

Persistent (Stored XSS)

Cross-Site Scripting



Could not find message 0

Message List

test
test
Test XSS

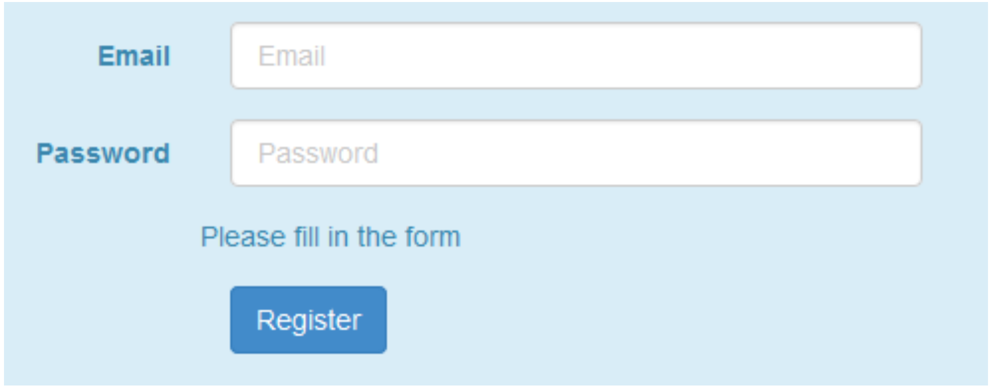
Microsoft Internet Explorer

JSESSIONID=6A71DC8FF9C0D8032DA43CF7088278C6

Persistent (Stored XSS)

Cross-Site Scripting

?message=Please fill in the form

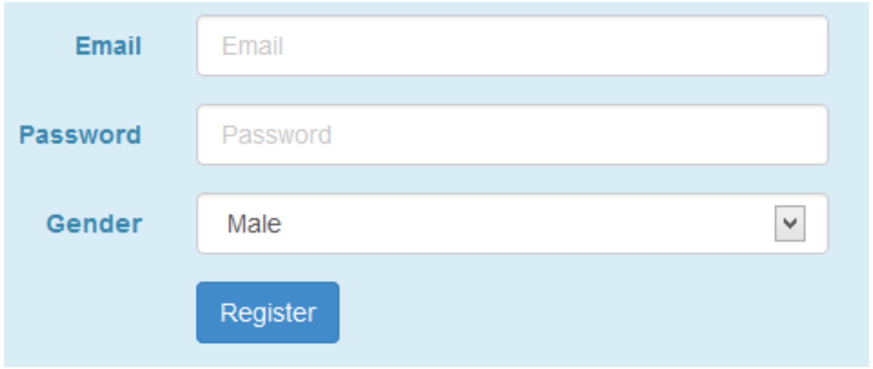


Please fill in the form

DOM Based XSS

Cross-Site Scripting

message=<label>Gender</label> <div class="col-sm-4"> <select class = "form-control" onchange="java_script_:show()"> <option value="Male">Male</option> <option value="Female">Female</option> </select> </div> <script>function show(){alert();}</script>



DOM Based XSS

Cross-Site Scripting

- Filter input parameters
- Filter output based on input parameters
- Encode output

  <script> gets converted to &lt;script&gt;
- Update Web Browser
- Self-defense

Sensitive Data Exposure

Sensitive Data Exposure occurs when an application does not protect the sensitive information as well as it should.

Sensitive Data Exposure

- Information used in authentication
  - Credentials
  - PINs
  - Session identifiers
  - Tokens, Cookies
- Information protected by laws, regulations or specific organizational policy.
  - Credit Cards
  - Customers data

Sensitive Data Exposure

The danger lies in the data being exposed, and the potential impact reflects the data's sensitivity.

Sensitive Data Exposure

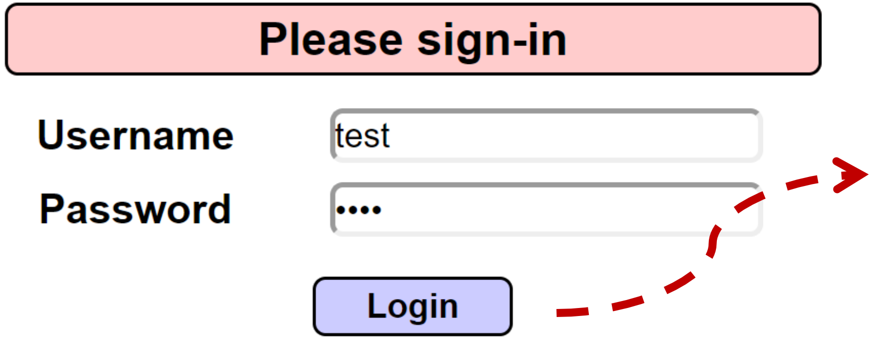
Most vulnerabilities within this category cannot be scanned for due to two main reasons:

1. To determine risk, it must be decided what information is considered sensitive, hard task to carry out automatically.
2. An external Pentester cannot know whether internal data is encrypted or not as that is not exposed.

SQL Injection

- SQL injection attacks are a type of injection attack.
- Insertion (or "injection") of a SQL query via the input data from the client to the application - effect the execution of predefined SQL commands.

SQL Injection


Please sign-in

Username test

Password ••••

SELECT * FROM users
WHERE name= 'test'
and password= '1234'

SQL Injection

```
1' or 1=1 --
```

```
SELECT * FROM users WHERE name= 'test' and password= '1234'
```

random

SQL Injection

```
SELECT * FROM users
WHERE name='1' or 1=1 --'
and password='abc'
```

```
SELECT * FROM users
WHERE name='1' or 1=1
```

SQL Injection

```
SELECT * FROM users
WHERE name='1' or 1=1 --'
and password='abc'
```

```
SELECT * FROM users
WHERE name='1' or 1=1
```

TRUE

SQL Injection

```
admin' or 1=1
admin' or 1=1--
admin' or 1=1#
admin' or 1=1/*
admin') or ('1'='1
admin') or ('1'='1'--
```

```
SELECT * FROM users
WHERE name='1' or 1=1 --'
and password='abc’
```

SQL Injection

A successful SQL injection exploit leading to:

- Read sensitive data from the database
- Modify database data
- Execute administration operations on the database

Q/A!

W: www.sangbui.com

T: @sangsecurity

E: sangbuicom@gmail.com