CS 6956 Software Security

Course Project

Abishek Krishnan, u1261980

4th Dec, 2019

Contents

About the project

Name : Study of Web Application Security (OWASP Top 10 - 2017)

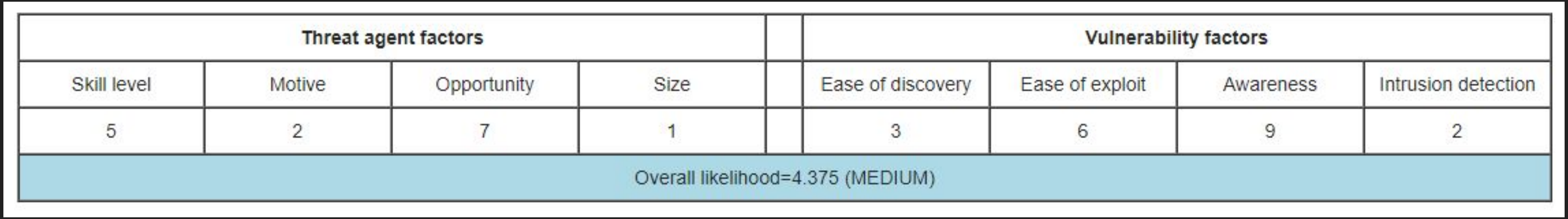
- Read the OWASP Top - 2017 paper (Open Web Application Security Project)
- Take an Online course on the topic
- Discover vulnerabilities in DVWA

How were the vulnerabilities ranked?

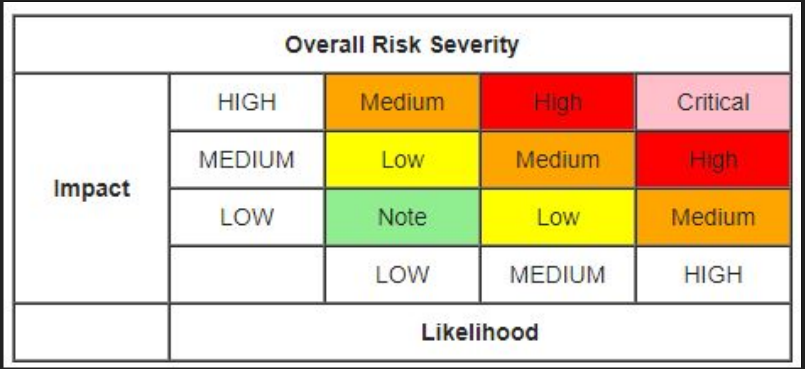
- 40+ data submissions from firms that specialize in application security

- Data spans vulnerabilities gathered from hundreds of organizations and over 100,000 real-world applications and APIs.

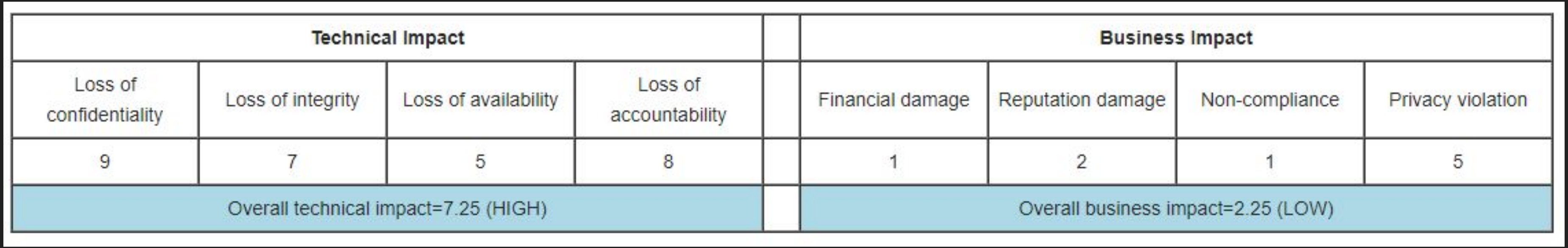
- The Top 10 items are selected and prioritized according to prevalence, in combination with consensus estimates of exploitability, detectability, and impact



Threat agent factors					Vulnerability factors			
Skill level	Motive	Opportunity	Size		Ease of discovery	Ease of exploit	Awareness	Intrusion detection
5	2	7	1		3	6	9	2
Overall likelihood=4.375 (MEDIUM)								



Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			



Technical Impact					Business Impact			
Loss of confidentiality	Loss of integrity	Loss of availability	Loss of accountability		Financial damage	Reputation damage	Non-compliance	Privacy violation
9	7	5	8		1	2	1	5
Overall technical impact=7.25 (HIGH)					Overall business impact=2.25 (LOW)			

List of vulnerabilities

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XXE)
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging & Monitoring

1. Injection

- Data being interpreted as command

Attacks

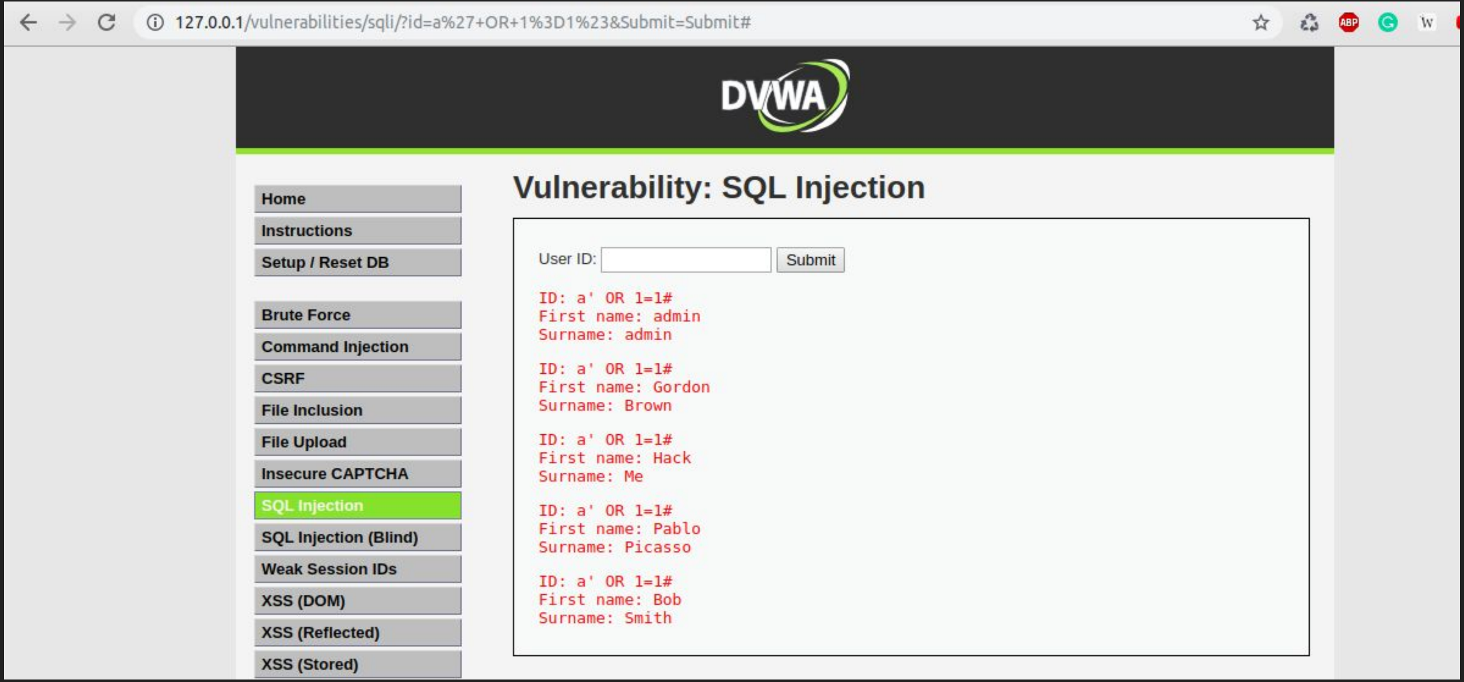
- 2013: Target - 110 million credit/debit-card Stolen
- 2016: Illinois Board of Elections, compromising up to 200,000 voter IDs

Prevention

- Input validation
- Parameterization - Prepared statements and stored procedures
- Principle of Least Privilege

User ID: a’ OR 1=1#

Represents any single numeric character (#)



Vulnerability: SQL Injection

```
ID: a' OR 1=1#
First name: admin
Surname: admin

ID: a' OR 1=1#
First name: Gordon
Surname: Brown

ID: a' OR 1=1#
First name: Hack
Surname: Me

ID: a' OR 1=1#
First name: Pablo
Surname: Picasso

ID: a' OR 1=1#
First name: Bob
Surname: Smith
```

2. Broken Authentication

- Malicious user is able to login

Causes

- Phishing
- Credential Stuffing

Attacks

2016: Yahoo - 500 million user accounts exposed (https://haveibeenpwned.com)

Prevention

- Multi-factor authentication | Password Requirements | Encryption (AES)

3. Sensitive Data Exposure

Lack of classification - Public | Internal | Confidential | Restricted

Attacks

- 2013: Pennsylvania Health Care System - 4,500 patients data

Prevention

- Health Insurance Portability and Accountability Act
- Payment Card Industry (PCI) - Standards
- Reducing the scope according to classification

4. XML External Entities (XXE)

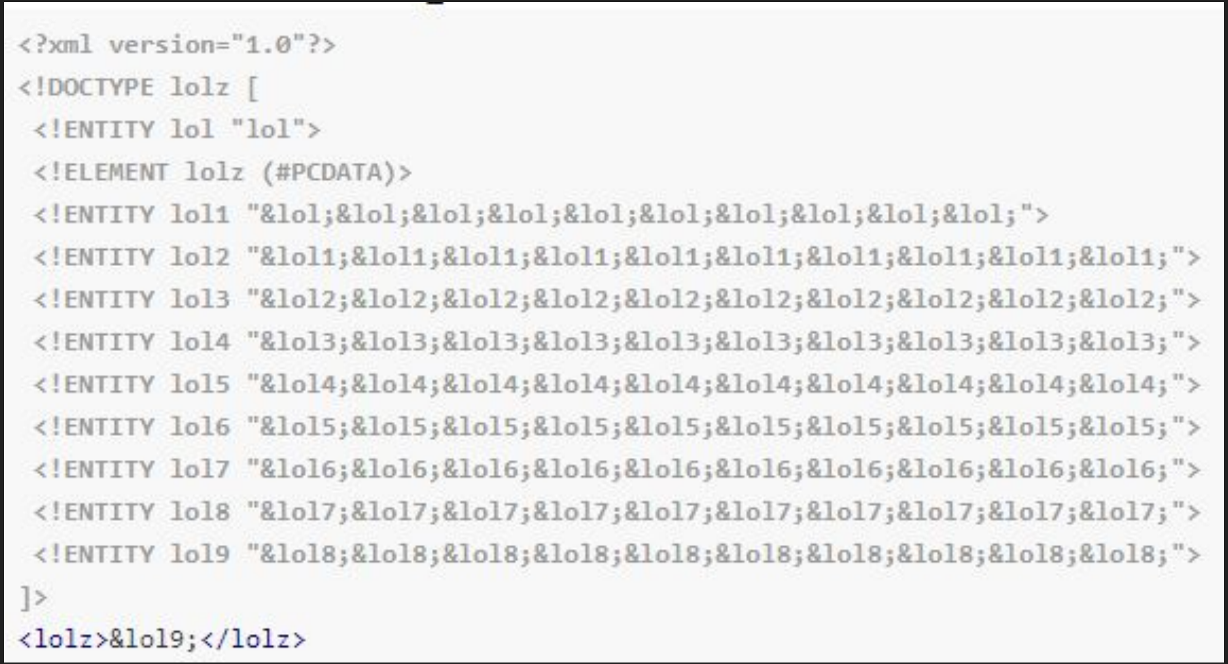
Subcategory of injection attacks (on XML)

Attacks

- 2002: Billion Laughs Attacks (code on next slide) - Leads to DoS

Prevention

- Disable XML External Entities (OWASP - Cheat Sheet)
- Input Validation & Whitelisting
- Patch and upgrade XML processor's



```
<?xml version="1.0"?>
<!DOCTYPE lolz [
 <!ENTITY lol "lol">
 <!ELEMENT lolz (#PCDATA)>
 <!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
 <!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">
 <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
 <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
 <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
 <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
 <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
 <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
 <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```

- Root Element - "lolz", that contains the text "&lol9;"
- "&lol9;" is a defined entity that expands to a string containing ten "&lol8;" strings and so on
- <1 KB XML Block results in ~ 3 GB of Memory

5. Broken Access Control

Improper access controls | Insecure direct object reference

Attacks

- 2018: Facebook’s data breach (50M users) - View as feature vulnerability
- 2016: Worldpay - 1.5M credit card information

Prevention

- Principle of least privilege
- Logging and alerts
- Manual testing of access controls

6. Security Misconfiguration

Sensitive data leak while error handling | Default Security Configuration

Attacks

- 2017: SVR Tracking - 0.5M driver and track info - unsecured Amazon S3 buckets
- 2018: RNC - 198M voter information - left on Amazon’s servers

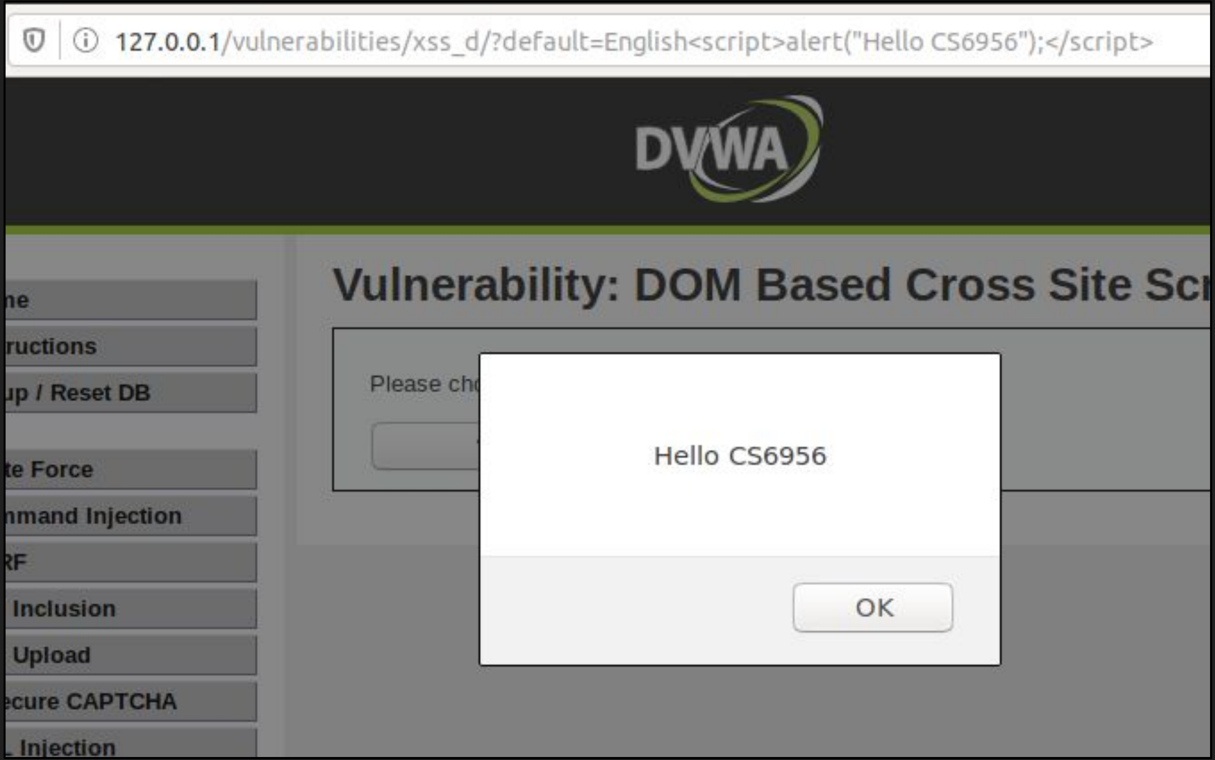
Prevention

- Register to NVD of NIST
- Test & scan s/w regularly

7. Cross-site scripting

- DOM - JavaScript frameworks, single-page applications, and APIs that dynamically include attacker-controllable data to a page
- Reflected - The application or API includes unvalidated and unescaped user input as part of HTML output
- Stored - The application or API stores unsanitized user input that is viewed at a later time by another user or an administrator

DOM XSS (included in the URL)

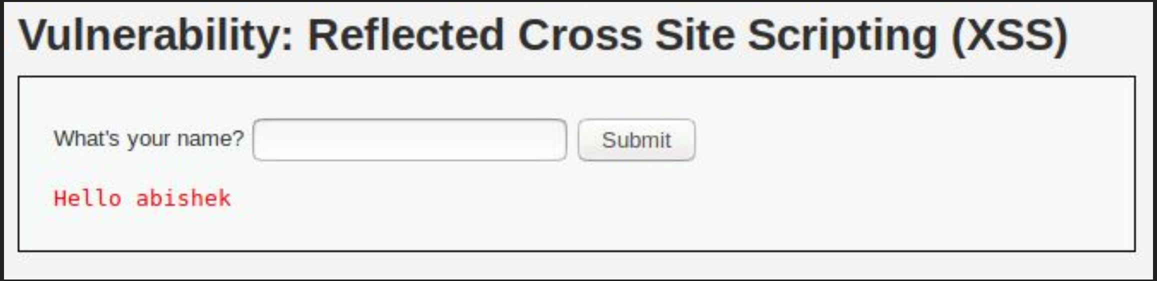


```
127.0.0.1/vulnerabilities/xss_d/?default=English<script>alert("Hello CS6956");</script>
```

Vulnerability: DOM Based Cross Site Scr

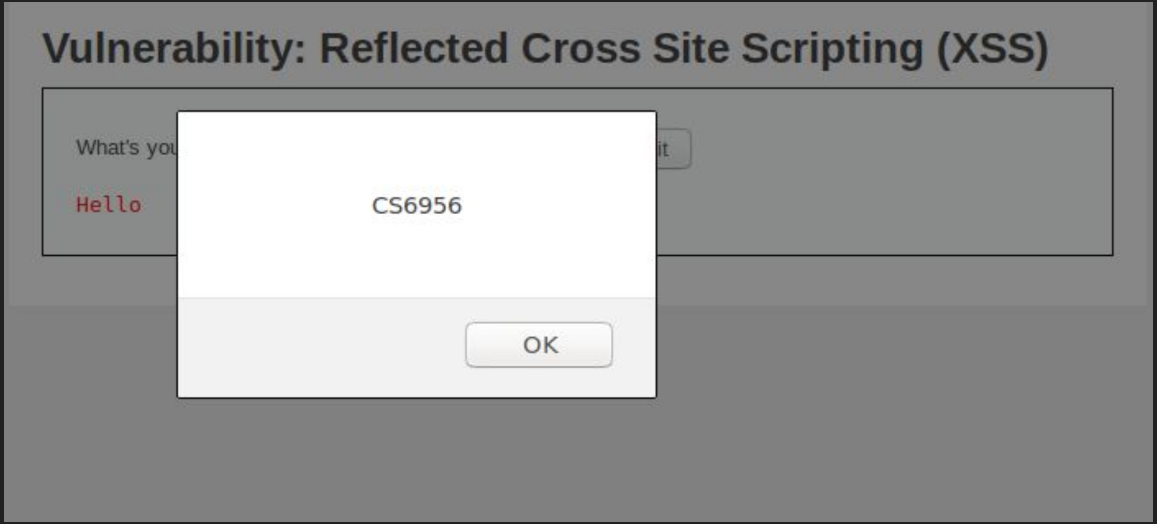
Hello CS6956

Reflected XSS



Vulnerability: Reflected Cross Site Scripting (XSS)

Hello abishek

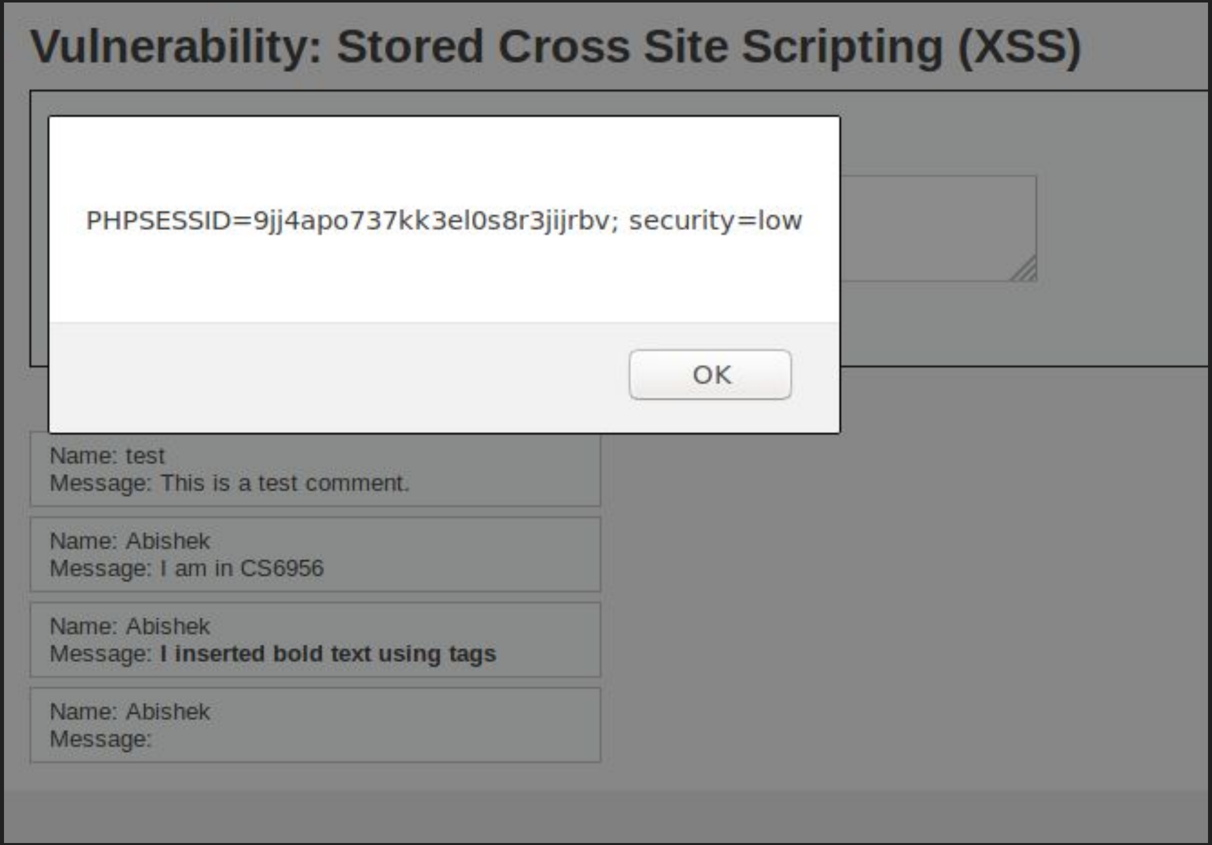


Vulnerability: Reflected Cross Site Scripting (XSS)

Hello

CS6956

Stored XSS



Vulnerability: Stored Cross Site Scripting (XSS)

```
PHPSESSID=9jj4apo737kk3el0s8r3jijrbv; security=low
```

Name: test
Message: This is a test comment.

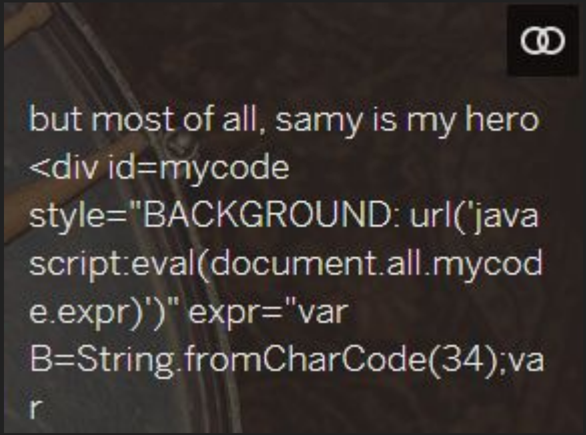
Name: Abishek
Message: I am in CS6956

Name: Abishek
Message: **I inserted bold text using tags**

Name: Abishek
Message:

Attacks

- 2005: Infamous Samy Computer Worm on MySpace



```
but most of all, samy is my hero
<div id=mycode
style="BACKGROUND: url('java
script:eval(document.all.mycod
e.expr)')" expr="var
B=String.fromCharCode(34);va
r
```

Prevention

- Use of Content Secure Policy
- Input Filtering
- Encode output (which contain user-controllable data)

8. Insecure Serialization

New entrant to the OWASP Top 10's

- AppSecCali 2015: Marshalling Pickles - how deserializing objects will ruin your day (Chris Frohoff & Gabriel Lawrence)

Attacks

- 2017: Paypal's 1.6 million customers data breach

Prevention

- https://github.com/frohoff/ysoserial - A proof-of-concept tool for generating payloads that exploit unsafe Java object deserialization

9. Using Components with Known Vulnerabilities

Due to heavy use of open-source and third party software

Attacks

- 2017: Equifax - personal information of 147 million people - didn't patch Apache Struts update on time

Prevention

- OWASP's dependency check
- Web Application Firewall (WAF) - attack mitigation is usually part of a suite of tools which together create a holistic defense against a range of attack vectors

10. Insufficient Logging & Monitoring

https://www.ponemon.org/ - Ponemon Institute

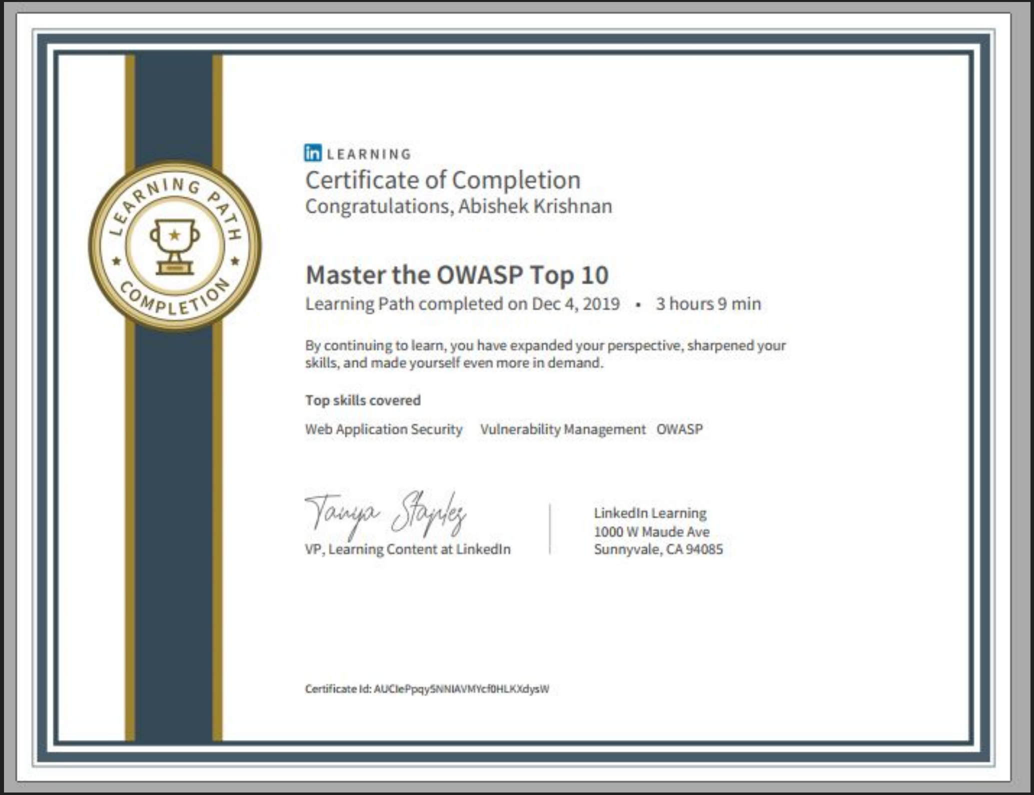
Attacks

- 2019: A data breach at Georgia Tech has exposed personal information of up to 1.3 million people | Took ~9 years to realize

Prevention

- Log “Who” performs a particular operation
- NIST Guide on reporting

Certificate



LEARNING

Certificate of Completion

Congratulations, Abishek Krishnan

Master the OWASP Top 10

Learning Path completed on Dec 4, 2019 • 3 hours 9 min

By continuing to learn, you have expanded your perspective, sharpened your skills, and made yourself even more in demand.

Top skills covered

Web Application Security Vulnerability Management OWASP

Tanya Staples

VP, Learning Content at LinkedIn

LinkedIn Learning
1000 W Maude Ave
Sunnyvale, CA 94085

Certificate Id: AUClePpqy5NNIAVMYcf0HLKXdysW

Questions?

Thank You!