**Table of Contents**

[**Introduction** 4](#_Toc157095754)

[**Environment and Tools** 5](#_Toc157095755)

[**Summary of Penetration Testing** 5](#_Toc157095756)

[**Steps to Emulate** 7](#_Toc157095795)

[Unprotected File 7](#_Toc157095798)

[Security Misconfiguration (Enabled Directory Listings) 8](#_Toc157095802)

[Privilege Escalation (Broken Authentication) 9](#_Toc157095806)

[Cross-site Scripting (XSS) 12](#_Toc157095810)

[**Analysis of Stored XSS** 12](#_Toc157095811)

[**Analysis of Persistent XSS** 14](#_Toc157095812)

[SQL Injection Attack 17](#_Toc157095815)

[Weak Password Mechanisms 20](#_Toc157095819)

[Conclusion 22](#_Toc157095822)

[**References** 23](#_Toc157095823)

# 

# **Introduction**

This report will detail the security weaknesses discovered during my penetration testing of badstore.net. I have successfully achieved the objectives of this exercise by detecting and leveraging typical website vulnerabilities such as unprotected file, security misconfiguration, authentication bypasses, Cross-Site Scripting (XSS), SQL injection and weak password mechanisms. Additionally, this report offers an in-depth analysis of each vulnerability, including its potential impact. It will also cover strategies for mitigating these vulnerabilities.

# **Environment and Tools**

I have used Oracle virtual box, badstore iso image and Linux OS to demonstrate the various web vulnerabilities for this assignment.

Used tools:

* Kali Linux OS
* Nmap
* Mozilla Firefox
* Burp Suite Community Addition
* SQLMap
* Hydra

**Summary of Penetration Testing**

## Summary of findings

BadStore.net is a deliberately flawed application designed for security training and testing purposes. It has been created to demonstrate typical vulnerabilities that can be found in numerous applications exposed to intranets, extranets, and the internet. The followings have been revealed throughout the test:

* Unprotected File
* Security Misconfiguration
* Privilege Escalation
* Cross-Site Scripting (XSS)
* SQL injection Attack
* Weal Password Mechanisms

## Severity Ratings, Ease of Exploitation and Affected URL

Considering the vulnerabilities that have been identified, I would assess the overall severity of BadStore.net as being significant. The broken authentication presents a risk of privilege escalation, resulting in potentially significant impacts. Furthermore, the stored XSS, session hijacking XSS, and SQL injection vulnerabilities could potentially enable an attacker to obtain unauthorized access to critical data or execute malicious code. This could have substantial consequences for the website owners.

|  |  |  |  |
| --- | --- | --- | --- |
| Attack Type | Severity | Ease of Exploitation | Affected URL |
| Unprotected File | LOW | EASY | <http://www.badstore.net/robots.txt> |
| Security Misconfiguration | MEDIUM | MEDIUM | <http://www.badstore.net/supplier/accounts> |
| Privilege Escalation | HIGH | HARD | <http://www.badstore.net/cgi-bin/badstore.cgi?action=loginregister> |
| XSS | CRITICAL | HARD | <http://www.badstore.net/cgi-bin/badstore.cgi?action=guestbook> |
| SQL Attack | CRITICAL | MEDIUM | <http://www.badstore.net/cgi-bin/badstore.cgi?searchquery=1000&action=search&x=10&y=8> |
| Weak Password Mechanisms | HIGH | MEDIUM | <http://www.badstore.net/cgi-bin/badstore.cgi?action=loginregister> |

In the above table, I have briefly stated each attack types with varying degrees of severity from low to critical and exploitation difficulty from easy to hard. Each attack type is associated with a specific URL where the vulnerability exists, with critical issues like XSS and SQL Injection being harder to exploit, indicating an urgent need for security measures to address these vulnerabilities. Privilege escalation can also be considered as one of the high risks’ vulnerabilities since malicious users could do many further attacks if they gained privileges. In Badstore.net, we can Easily take advantage as an admin role. The web has also implemented with weak password mechanisms without encryption system since it became the insecure website.

**Steps to Emulate**

Badstore.net contains various threats and vulnerabilities that can harm our main operating system. It boots from CD-ROM and runs as an Apache server. It is not necessary for installation, and nothing is copied to the hard drive of our PC. The vulnerabilities present in BadStore.net would allow an attacker to access the hard drive on the host PC. As a result, it must only be installed on lab or test environment.

I have utilized Oracle Virtual box to test vulnerabilities which contain in Badstore.net. After setting up the BadStore.net lab environment in virtual box, it’s IP address need to connect with kali Linux by using host-only adapter network.

## Unprotected File

### **Analysis**

First and foremost, I tried to scan the url with nmap and I found Robots.txt file.

A screenshot of a computer

Description automatically generated

I searched the file by typing the url in below screenshot by using Mozilla Firefox.

A screenshot of a computer

Description automatically generated

Robots.txt file, is a major cause of information leakage, used used to instruct web robots (typically search engine crawlers) which pages on a website to crawl and index. Even a normal user can easily access it and find out the directories which should be protected from crawlers.

### **Impact**

Users can identify directories that are intriguing and not meant to be accessible by others, thus simplifying the task of navigating the web application. In this case, we can observe that /backup, /upload, /supplier, /cgi-bin, and /scanbot are all interesting directories that could potentially contain valuable information.

### **Mitigation**

Mitigating robots.txt vulnerability involves regularly reviewing and updating the file to prevent the inadvertent exposure of sensitive information, protect the file with .htaaccess file and implementing strong authentication.

## Security Misconfiguration (Enabled Directory Listings)

### **Analysis**

In BadStore.net, it is easy to view the directory listings which may expose hidden scripts, include files, and backup source files. Those files can contain sensitive information which can be used for further attacks and exploits. During the enumeration process, I found that the directory listing of supplier has accounts file with ID and credentials. Even they are encoded with base-64, the developer has stored them in a plaintext. Hence, I could easily decode, and exploit directories related to suppliers.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen shot of a computer program

Description automatically generated

### **Impact**

Exposing directory structure like this allows an attacker to know backups and secret files which is protected in hidden files and folders. It can also lead to privacy concerns if user data or personally identifiable information is exposed inadvertently.

### **Mitigation**

To mitigate these impacts, the developer must disable directory listings and implement proper access controls and security measures to protect sensitive information and enhance user experience.

## Privilege Escalation (Broken Authentication)

### **Analysis**

When I registered as a new user, I used Burp Suite Community Edition to intercept the source page. While intercepting, I noticed that during the new user registration process, there was a hidden field called 'role' which had values (U for user, A for admin, and S for supplier). By manipulating this field through a proxy, I was able to register a user as an admin or supplier without proper validation and authorization. This vulnerability allowed anyone to become an admin. Consequently, I changed the 'role' to 'A' and gained access to the page as an admin. Inside the administration portal, I could view numerous databases containing backup files, sales report, user email addresses, passwords, names, and roles.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

### **Impact**

This critical vulnerability can allow malicious users to register as administrators, granting them full privileges over the web application. This leads to a complete compromise of the web app's security.

### **Mitigation**

The user registration process should initially assign the lowest privilege level, such as 'User,' to new registrants. Any potential elevation of privileges to 'Administrator' should be left for subsequent actions by an authorized administrator. This indicates a flaw in the user registration module itself.

## Cross-site Scripting (XSS)

### **Analysis of Stored XSS**

In BadStore.net, XSS vulnerabilities can be found in signing guestbook page. After injecting some html malicious scripts, it changed the appearance of that section. This is more severe than the reflective XSS.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

### **Analysis of Persistent XSS**

There is also an another XSS vulnerability present in Guest book section of Badstore.net which is persistent cross-site scripting. It allows me to inject a malicious script into the into the web application's database. Unlike stored XSS, in persistent XSS, the injected script is stored on the server which is my kali machine. This means the script becomes a permanent part of the web page. Every time a user accesses the affected page, the malicious script is executed in the client’s browser. The script can steal the user’s single sign-on ID (SSOID). In this case, the session hijacking was done by making my virtual machine as the server and my host machine as a client. Moreover, I have used netcat listener to hijack the ID.

I injected the code shown in below from server side which is my virtual machine.

A screenshot of a computer

Description automatically generated

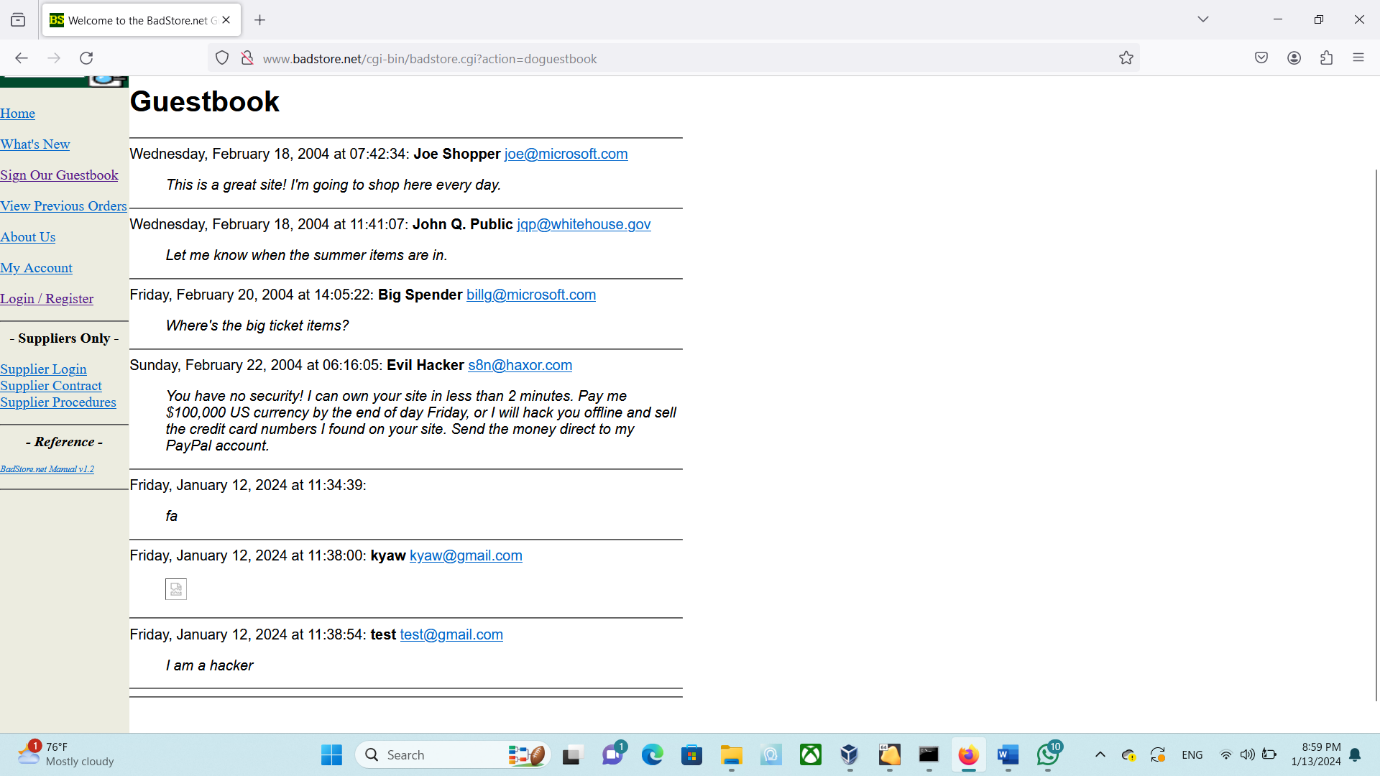
A screenshot of a computer

Description automatically generated

When the client which is my host machine commented in the Guestbook, the SSOID have been stolen from the server.

A screenshot of a computer

Description automatically generated



A screen shot of a computer

Description automatically generated

### **Impact**

Effectively exploiting an XSS vulnerability can lead to the compromise of a user's session, potentially granting unauthorized access to sensitive information or actions within a web application. In more severe cases, sophisticated attackers may employ specialized XSS frameworks such as Beef (Browser Exploitation Framework) to escalate their attacks. These advanced tools can facilitate the compromise not only of user sessions but also pose a significant risk to the security of the entire system, potentially allowing attackers to gain control over various aspects of the system's functionality and data.

### **Mitigation**

The recommendation for mitigation of this vulnerability is to exercise whitelisting method which specifies the actions that are only permitted. Utilizing of anti XSS libraries can also mitigate this vulnerability and use tools like regular expression (regex) to sanitize user inputs.

## SQL Injection Attack

### **Analysis**

For ease of penetration testing, I used automation tool like SQLMap to detect SQL injection vulnerabilities present in Badstore.net. By using this tool, I have exploited the item and user databases which contains sensitive information such as login email addresses and passwords of users. In item database, I found the item numbers, prices, and quantities. First, I injected manual query like ‘1’ and ‘1=1’, however, later I used SQLMap for injection. SQLMap has provided me with users’ email addresses, plaintexts of passwords and hash of passwords with MD5 checksum.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A black background with yellow dots

Description automatically generated

**A screen shot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**A screenshot of a computer program

Description automatically generated

# 

### **Impact**

Effectively exploiting vulnerabilities as demonstrated above can lead to a complete compromise of the entire database. In such scenarios, even email addresses and passwords become exposed, attackers can also bypass the authentication mechanisms with relative ease. This is a significant security risk, potentially granting unauthorized access to sensitive data and compromising the overall integrity.

**Mitigation**

To mitigate the impact described above, it's crucial to implement strong security measures. This includes conducting regular security audits and vulnerability assessments to identify and patch potential weaknesses. Employ robust input validation and parameterized queries to prevent SQL injection attacks. Additionally, use encryption to safeguard sensitive data.

## Weak Password Mechanisms

### **Analysis**

In BadStore.net, I found that it has been implemented with a very weak password mechanism. The more critical issue here is that it has used MD5 hashing algorithms which is known to have a lot of vulnerabilities. Since it has been designed for speed and efficiency, it is susceptible to brute force and dictionary attacks.

First, I used hash-identifier in kali and crackstation on my host machine to identify the hash of admin’s password and then I utilized Hydra in kali to crack the password. To get the http post form, I intercepted the BadStore.net’s login form by using Burpsuite community edition. I have cracked passwords for my own account on badstore and admin account.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A black background with white text

Description automatically generated

A black screen with white text

Description automatically generated

### **Impact**

Weak password mechanisms can often result in unauthorized access and data breaches. These weak passwords are vulnerable to attacks like brute force, dictionary, or social engineering, posing serious risks. For individuals, this can mean compromised personal and financial details, leading to identity theft and financial damage. Organizations face confidential data loss, legal issues, diminished customer trust, and hefty costs for breach management and reparations. Additionally, weak passwords can allow attackers deeper network access, causing extensive system damage, data breaches, and operational interruptions.

**Mitigation**

To reduce the risks from weak password practices, it's essential to enforce advanced password policies and additional security measures. Strong password policies should mandate complex, hard-to-crack passwords, combining different characters and enforcing regular updates. Educating users about creating strong passwords and avoiding common mistakes is also crucial. Additionally, multi-factor authentication (MFA) significantly bolsters security by requiring multiple proofs of identity, greatly minimizing unauthorized access risks.

## Conclusion

In this report, I have evaluated the security vulnerabilities identified during the penetration testing of BadStore.net, highlighting a range of issues from unprotected files and security misconfigurations to serious threats like Cross-Site Scripting (XSS) and SQL injection. Utilizing tools such as Oracle Virtual Box and Kali Linux, the testing demonstrated the potential risks and impacts of these vulnerabilities, emphasizing their severity and the ease with which they could be exploited.

Key findings include the critical nature of XSS and SQL injection attacks and the high risk of privilege escalation, all of which pose substantial threats to the integrity and security of BadStore.net. These vulnerabilities, if not addressed, could allow unauthorized access and malicious users can control over the web application.

To tackle these risks, several strategies are recommended in the report such as regular updates and monitoring of the robots.txt file, disabling directory listings, implementing stringent access controls, restructuring the user registration process to prevent unauthorized privilege escalation, using whitelisting and input sanitization techniques to combat XSS, and strengthening defences against SQL injection through robust validation and encryption practices. Additionally, enhancing password security through strong policies and multi-factor authentication is crucial.

In essence, the penetration test underscores the critical need for comprehensive and proactive security measures in web applications to guard against the ever-evolving landscape of cyber threats.

# References

Anon, n.d. *Badstore.net User Manual - PDF.* [Online]   
Available at: https://docplayer.net/amp/12906628-Badstore-net-user-manual.html  
[Accessed 14 January 2024].

Anonymous, 2017. *Badstore Vulnerabilities.* [Online]   
Available at: https://cyb3r-cod3r.blogspot.com/2017/04/badstore-vulnerabilities.html?m=1  
[Accessed 15 January 2024].

Ken, L., 2023. *DVWA Penetration Testing Report.* [Online]   
Available at: https://systemweakness.com/dvwa-penetration-testing-report-2c7beb3395ff  
[Accessed 16 January 2024].

Vivona, A., 2018. *Pen-testing: Badstorev1.2.3 Walkthrough - Vulnhub.* [Online]   
Available at: https://medium.com/syscall59/badstore-1-2-3-walkthrough-vulnhub-7816f3001333  
[Accessed 14 January 2024].

Whittle, M., 2020. *OWASP Top 10 and DVWA.* [Online]   
Available at: https://levelup.gitconnected.com/ethical-hacking-part-1-owasp-top-10-and-dvwa-3f2d55580ba8  
[Accessed 16 January 2024].