Cyber Security Training

PREPARED BY

Khushi Bhatt (2018BTECHEEE004)

FACULTY GUIDE

Dr. Devika Kataria



Department of Electrical and Electronics Engineering
Institute of Engineering and Technology (IET)
JK Lakshmipat University Jaipur

May 2022

Cyber Security Training

Practice School-2

Submitted in partial fulfilment of the requirements

for the degree of

Bachelor of Technology in Electrical and Electronics Engineering

By:
Khushi Bhatt (2018BTECHEEE004)

Faculty Guide:
Dr. Devika Kataria



Department of Electrical and Electronics Engineering
Institute of Engineering and Technology (IET)
JK Lakshmipat University Jaipur

May 2022

CERTIFICATE

This is to certify that the Practice School-2 project work entitled "**Cyber Security Training**" submitted by **Khushi Bhatt (2018BTECHEEE004)**, towards the partial fulfilment of the requirements for the degree of **Bachelor of Technology in Electrical and Electronics Engineering** of JK Lakshmipat University Jaipur is the record of work carried out by them under my supervision and guidance. In my opinion, the submitted work has reached the level required for being accepted for the Practice School-II examination.

Dr. Devika Kataria

Associate Professor & Head,

Department of Electrical and Electronics Engineering

Institute of Engineering & Technology (IET)

JK Lakshmipat University Jaipur

Dr. Devika Kataria

Associate Professor & Head,

Department of Electrical and Electronics Engineering

Institute of Engineering & Technology (IET)

JK Lakshmipat University Jaipur

Date of Submission-

for (PS-2 Coordinator)

Prof. Dr. – Sanjay Goel

Director,

Institute of Engineering & Technology

JK Lakshmipat University Jaipur

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to Prof Sanjay Goel, Professor, Department of Computer Science & Engineering for his valuable time and guidance that made the project work a success.

My heartfelt thanks to Mr. Anil Kumar HM, Sr. Cyber Security Engineer at Intimetec Vision soft Pvt Ltd for his support in providing me the guidance during my training. Thanking all my friends and all those who have helped me carrying out this work directly or indirectly without whom completion of this training and project work was not possible.

I would also like to thank sincerely to Prof. Dr Devika Kataria, Head of Electrical and Electronics Department and Dr Sanjay Goel, Director, Institute of Engineering and Technology, JK Lakshmipat University for giving us a platform to carry out the learnings.

I also thank my faculty supervisor Dr. Devika Kataria for guiding me throughout the PS-2 process and keeping me on track with everything.

Sincerely yours,
Khushi Bhatt
(2018BTECHEEE004)

ABSTRACT

A penetration test is vital for a company that values cyber security. I learned new methods for performing VAPT in a more proactive manner. It was a proactive approach to maintaining a high level of security and protection from hackers because if a penetration tester can exploit the vulnerability and compromise the network, so can a real hacker. As we all know, the WannaCry ransomware attack affected over 2 lakh computers worldwide and demanded ransom payments in the form of bit coins to unlock the systems. Many large organizations have been impacted by this attack. With such big cyber-attacks occurring these days, it has become critical to perform penetration testing at regular intervals to maintain our information system secure from security breaches.

Once the test is completed, management must decide what the risk is and what they can do to minimize it, such as implementing a security measure. Occasionally, the loss caused by vulnerability is smaller than the cost of control. In certain instances, the company may choose to accept the risk. Being an excellent penetration tester requires time and effort; most penetration testers nowadays are just vulnerability analyzers. To be a good penetration tester, one must understand the art of exploitation. I learnt to hone my instincts for detecting what may be exploited and expanded.

LIST OF FIGURES

Figure 1 Intranetwork for two buildings 12

Figure 2 Level Progress of PortSwigger 13

Figure 3 Owasp Juice Shop Progress 13

Figure 4 Owasp Juice Shop View 14

Figure 5 DIVA Application 15

Figure 6 CPT Assignment 19

Figure 7 Vulnerable Code 1 22

Figure 8 Output1 22

Figure 9 Vulnerable Code 2 22

Figure 10 DIVA HardCoding Issues 23

Figure 11 Vulnerable Code 3 23

Figure 12 Insecure Data Storage 24

Figure 13 Output2 24

Figure 14 Vulnerable Code 4 26

Figure 15 DIVA Insecure Data Storage 2 26

Figure 16 Output3 27

Figure 17 Vulnerable Code 5 27

Figure 18 DIVA Insecure Data Storage3 28

Figure 19 Output4 28

Figure 20 Vulnerable Code 6 29

Figure 21 DIVA Insecure Data Storage4 30

Figure 22 Output5 30

Figure 23 DIVA Input Validation Issues1 32

Figure 24 DIVA Input Validation Issues2 33

Figure 25 Vulnerable Code 9 34

Figure 26 DIVA Acces Control 1 34

Figure 27 Output6 35

Figure 28 Vulnerable Code 10 35

Figure 29 DIVA Access Control 2 36

Figure 30 Output7 36

Figure 31 Vulnerable Code 11 37
Figure 32 Output7 38
Figure 33 Vulnerable Code12 38
Figure 34 DIVA Hardcoding Issues2 39
Figure 35 Output8 40
Figure 36 Vulnerable Code13 41
Figure 37 DIVA Input validation 3 41
Figure 38 MobSF report Snapshot 42
Figure 39 Burp Suite 46
Figure 40 Cisco Packet Tracer 47
Figure 41 MobSF Interface 48
Figure 42 Drozer Console 48
Figure 43 ADB 49
Figure 44 Nessus Interface of WebApp Scanning 50
Figure 45 AWS EC2 Instance 50

LIST OF TABLE

Table 1 IP Table 18

Contents

CERTIFICATE 3

ACKNOWLEDGEMENTS 4

ABSTRACT 5

LIST OF FIGURES 6

LIST OF TABLE 8

Chapter 1: About the Company 11

1.1 Company Name 11

1.2 What does the company do: 11

Chapter 2: Training workflow 12

2.1 Training Bundle 1: Fundamentals of Networking 12

2.1.1 Assignment: Develop an intranetwork for two buildings 12

2.2 Training Bundle 2: Web Application Security 12

2.2.1 Assignment: Portswigger Labs for understanding vulnerabilities 13

2.2.2 Assignment: Pen test a vulnerable application named Owasp Juice shop 13

2.3 Training Bundle 3: Learning Different Vulnerabilities 14

2.4 Training Bundle 4: Mobile Application Security 15

2.4.1 Assignment: Practicing on DIVA (Damn Insecure and Vulnerable App) 15

2.5 Training Bundle 5: Understanding the Tools 16

2.5.1 Assignment: ZAP Scan a test website and generate report. 16

Intercept and change an HTTP request. 16

Basic command-line usage in Kali Linux. 16

Scan a test mobile app using MobSF. 16

Chapter 3: Case Study on networking 17

Chapter 4: Some Major Web vulnerabilities in Owasp Juice Shop 20

Chapter 5: Damn Insecure and Vulnerable Application Report 22

Challenge – 1 [Insecure Logging] 22

Challenge – 2 [Hardcoding issue-1] 22

Challenge – 3 [Insecure Data Storage-1] 23

Challenge – 4 [Insecure Data Storage-2] 26
Challenge – 5 [Insecure Data Storage-3] 27
Challenge – 6 [Insecure Data Storage-4] 29
Challenge – 7 [Input Validation Issues-1] 32
Challenge – 8 [Input Validation Issues-2] 32
Challenge – 9 [Access Control Issues -1] 33
Challenge – 10 [Access Control Issues -2] 35
Challenge – 11 [Access Control Issues -3] 37
Challenge – 12 [Hardcoding Issues - 2] 38
Challenge – 13 [Input Validation - 3] 40
Chapter 6: Vulnerability Assessment and Penetration Testing 43
Vulnerability Assessment (VA) 43
Penetration Testing 43
Process of Vulnerability Assessment and Penetration Testing: 43
Goals and objectives 43
Information Gathering 44
Vulnerability Detection 45
Information Analysis and planning 45
Chapter 7: Tools and Technologies for WebApp and Mobile App Security 46
1. Burp Suite 46
2. Cisco Packet Tracer 46
3. MobSF 47
Static Analysis: 47
Dynamic Analysis 47
4. Drozer 48
5. ADB 48
6. Nessus 49
7. AWS Fundamentals 50
8. BitBucket 50
Chapter8 : Key Learnings and Conclusion 52
Referrences 53

Chapter 1: About the Company

1.1 Company Name

Intimetec Vision soft Pvt Ltd

1.2 What does the company do:

Intimetec Vision soft Private Limited was founded on February 13, 2009. It is a non-government corporation that is registered with the Registrar of Companies in Jaipur. It has a paid-up capital of Rs. 37,730,720 and an authorized share capital of Rs. 50,000,000. It is involved in software publication, consulting, and supply [Software publishing covers the manufacture, delivery, and documentation of ready-made (non-customized) software, operating system software, business & other application software, and computer gaming software for all platforms.

In Time Tec offers progressive software development services, allowing its clients to keep their best and brightest employees focused on innovation. In Time Tec's leadership team has an average of 15 years of experience in software/firmware R&D and 20 years of experience creating onshore/offshore R&D teams. Every organisation has core and context work. Core work, as described by author and speaker Geoffrey Moore, is your differentiator, your "secret sauce." Contextual work is non-core labour that must be completed. In Time Tec assists you in focusing on core tasks by handling context work. We provide the following software development services employing a hybrid model of onshore and offshore talent:

It takes a particular sort of individual to join us on our abundance platform. Everything we do is motivated by our goal and way of life. While serving our customers and communities, we want everyone to reach their maximum potential. It provides a broad variety of technical security services, ranging from examining existing procedures and documentation to assessing the security of present settings and designing new secure IT systems to suit modern security needs. It assists organisations in meeting their cyber compliance duties through the establishment of risk compliance evaluations, governance activities, policies, and action plans.

Cartos Suite is a cloud-based Managed Print Service (MPS) application suite that assists MPS sales teams and consultants in discovering current print fleets, designing future fleets, analysing Total Cost of Ownership (TCO), and simply finding and comparing groups of devices. Cartos Suite Fleet Design, TCO, Asset Advisor, and Device Data Service are used to accomplish this.

It may help your company examine and enhance its present cyber security resilience by utilising mature security architecture, design, development, deployment, vulnerability testing, and management methodologies. Through a continual sequence of articles and talks, as cyber security threats change and new vulnerabilities are uncovered.

Chapter 2: Training workflow

2.1 Training Bundle 1: Fundamentals of Networking

1. Understanding the concepts of IP addresses
2. OSI Model
3. TCP/IP Model
4. Switching/Routing
5. DNS
6. Basic Protocols
7. Security Testing
8. Cisco Packet Tracer Assignment

2.1.1 Assignment: Develop an intranetwork for two buildings

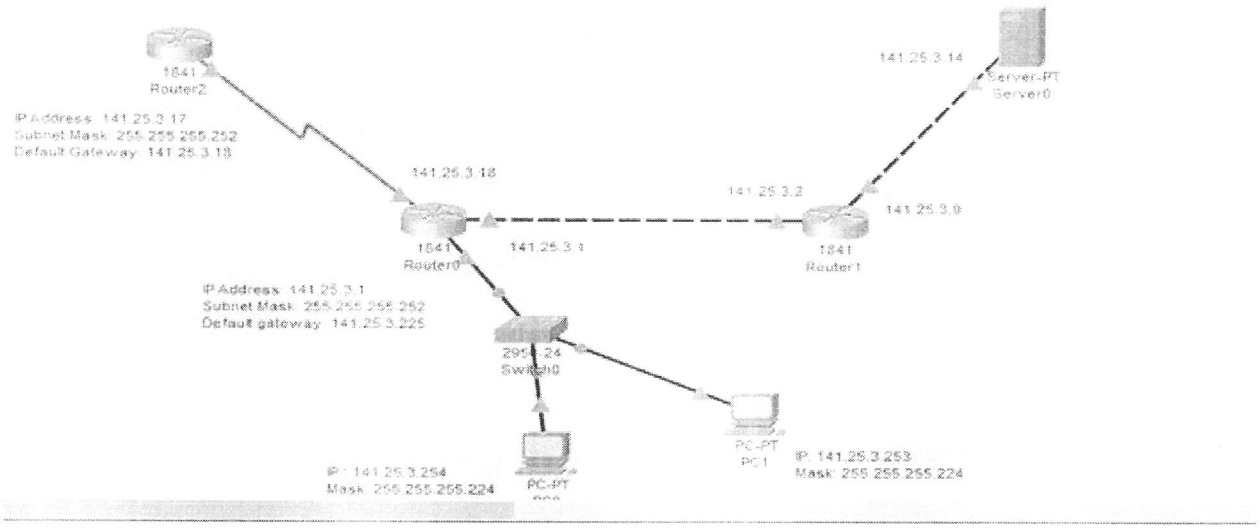


Figure 1 Intranetwork for two buildings

2.2 Training Bundle 2: Web Application Security

1. Web Application Basics
2. Understanding the HTTP protocol
   - HTTP Requests and Response Headers.
   - HTTP Request Methods
   - HTTP status Codes
3. Understanding the OWASP Top 10
4. Getting familiar with burp suite
5. Getting familiar with kali Linux
6. Getting familiar with juice shop
7. Portswigger labs of 22 vulnerabilities

2.2.1 Assignment: Portswigger Labs for understanding vulnerabilities

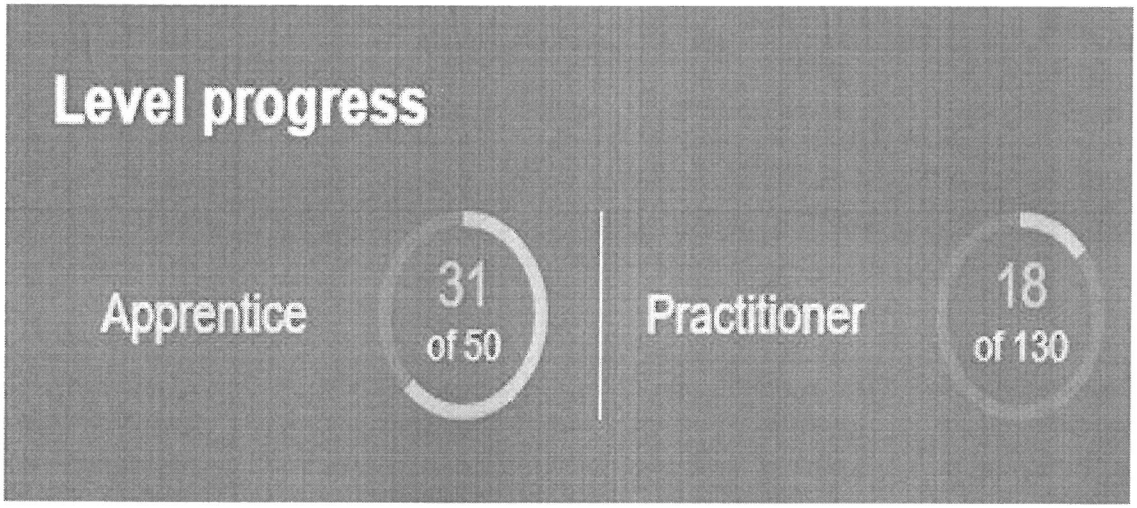


Figure 2 Level Progress of PortSwigger

2.2.2 Assignment: Pen test a vulnerable application named Owasp Juice shop

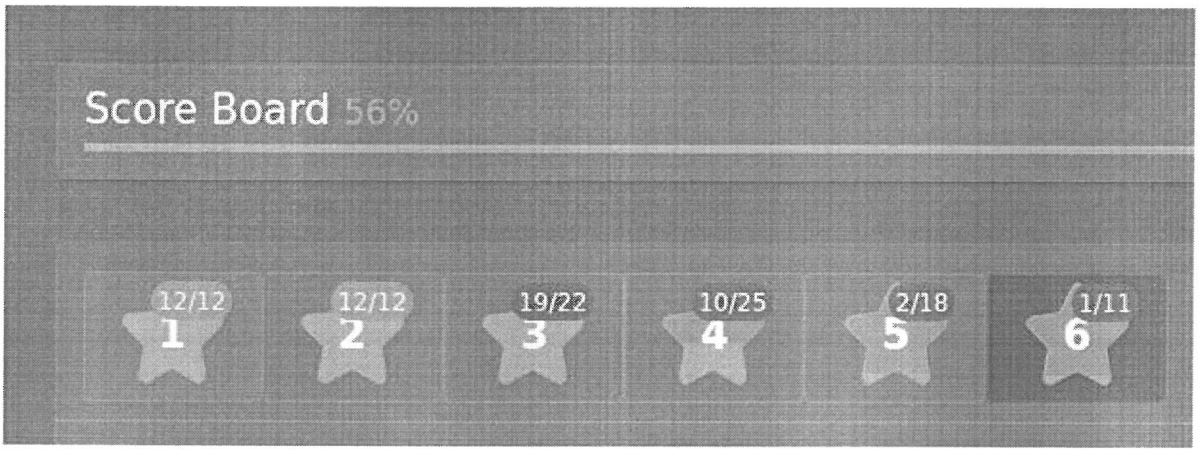


Figure 3 Owasp Juice Shop Progress

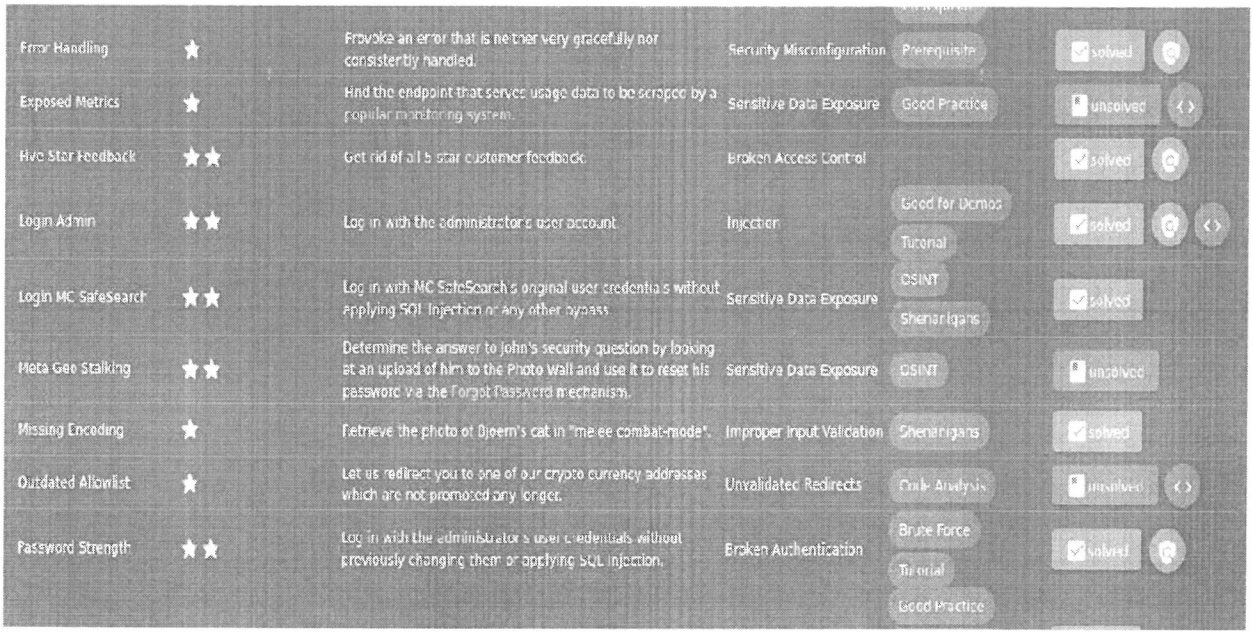


Figure 4 Owasp Juice Shop View

2.3 Training Bundle 3: Learning Different Vulnerabilities

1. SQL Injection
2. Broken Authentication
3. Directory traversal
4. Os command injection
5. XXS injection
6. Information disclosure
7. Broken Access control
8. File upload vulnerabilities
9. Server-side request forgery.
10. XXE Injection
11. Cross-site scripting
12. Cross-site request forgery
13. Cross-origin resource sharing
14. Clickjacking
15. DOM-Based vulnerabilities
16. Web sockets
17. Insecure Deserialization
18. Server-side template injection
19. Web cache poising
20. HTTP Host header attacks
21. HTTP request smuggling
22. OAuth authentication

2.4 Training Bundle 4: Mobile Application Security

1. Introduction to Android
2. Android Architecture
3. Security Auditing
4. OWASP top 10 Mobile
5. Getting familiar with DIVA
6. Getting familiar with ADB
7. Getting familiar with MobSF
8. Getting familiar with drozer
9. Understanding iOS applications.
10. Understanding the Security Model.
11. Attacking iOS applications
    - Introduction to Transport Security.
    - Identifying Insecure Storage.
    - Attacking the iOS runtime.
    - Understanding Inter-Process Communication.
    - Identifying Data Leaks

2.4.1 Assignment: Practicing on DIVA (Damn Insecure and Vulnerable App)

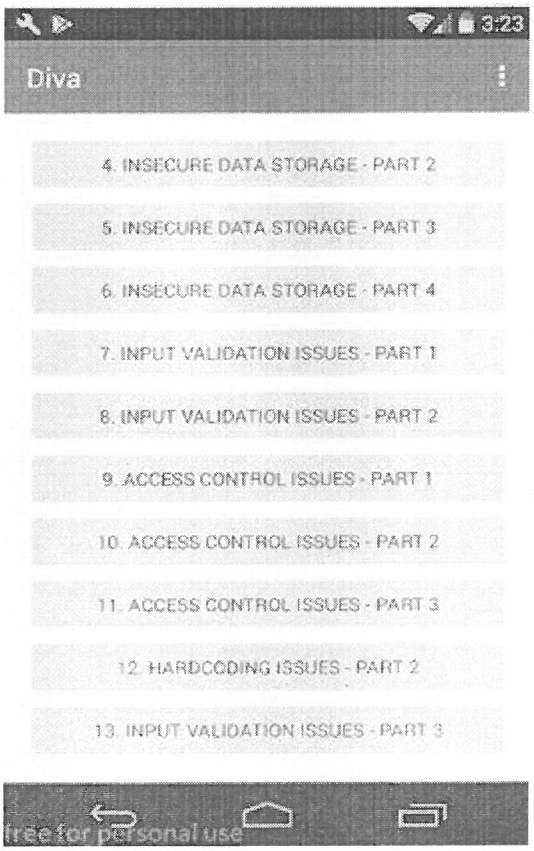


Figure 5 DIVA Application

2.5 Training Bundle 5: Understanding the Tools

1. OWASP ZAP.
2. Burp Suite.
3. Nessus Essentials.
4. Getting Familiar with Kali Linux.
5. Security Onion.
6. Mobile Security Framework (MobSF).
7. Drozer.
8. Elastic Stack fundamentals
9. AWS fundamentals

2.5.1 Assignment: ZAP Scan a test website and generate report.

Intercept and change an HTTP request.

Basic command-line usage in Kali Linux.

Scan a test mobile app using MobSF.

Chapter 3: Case Study on networking

Topology Story

Fast Ware is a software company that experienced a rapid growth and needs to implement an intranetwork for the two buildings that the main office is composed of and connect it to the link provided by the local ISP.

They have hired you to:
* Configure R1's link towards the ISP with the proper IP address
* Perform the cabling for the intranetwork
* Create an IP scheme using the 141.25.3.0 IP address space.
* Do host naming between all Cisco devices.
* Setup the connectivity to the ISP office using the IP that has been allocated (see the following table)
* Test all connections.

Note!
The ISP Router is already configured
The HTTP Server must be reachable only from inside the network.

IP Scheme: Fill in the correct values.

Device	Port	Network address	Mask	IP Number
ISP (first IP)	S0/0/0	141.25.3.16	/30	141.25.3.17
R1 (last IP)	S0/0/0	141.25.3.16	/30	141.25.3.18
R1 (first available subnet – first IP)	Fa0/1	141.25.3.0	/30	141.25.3.1
R1 (last available subnet - first IP)	Fa0/0	141.25.3.224	/27	141.25.3.225
R2 (the same subnet as R1 Fa0/1 - last IP)	Fa0/1	141.25.3.0	/30	141.25.3.2
R2 (first remaining subnet - first IP; server gets last	Fa0/0	141.25.3.0	/29	141.25.3.9 141.25.3.14

IP)				
PC0 (last available subnet - 30th IP)	Fa	141.25.3.224	/27	141.25.3.254
PC1 (last available subnet - 29th IP)	Fa	141.25.3.224	/27	141.25.3.253

Table 1 IP Table

Cabling and Interface configurations

* Perform the cabling of the intranetwork using R2 as the gateway for the HTTP server and R1 as a gateway for all the inning Fast Ware devices
* For the connections not indicated in the above table use the first available interface
* Configure the R1 and R2 routers using static routes for networks not known by the router
* Configure IP addresses with respect with the above table
* Configure the PC0 and PC1 with usable IP addresses.

* Ping from R1 to ISP

* Ping from R1 to R2

End-to-end connectivity configuration

* Ping from PC0 to The R1 interface S0/0/0
* Ping from PC0 to the ISP interface S0/0/0

Explain the difference in TTL from both replies
* Ping from PC1 to the HTTP Server

* Open the internet browser from PC1 with the ip address from the HTTP server

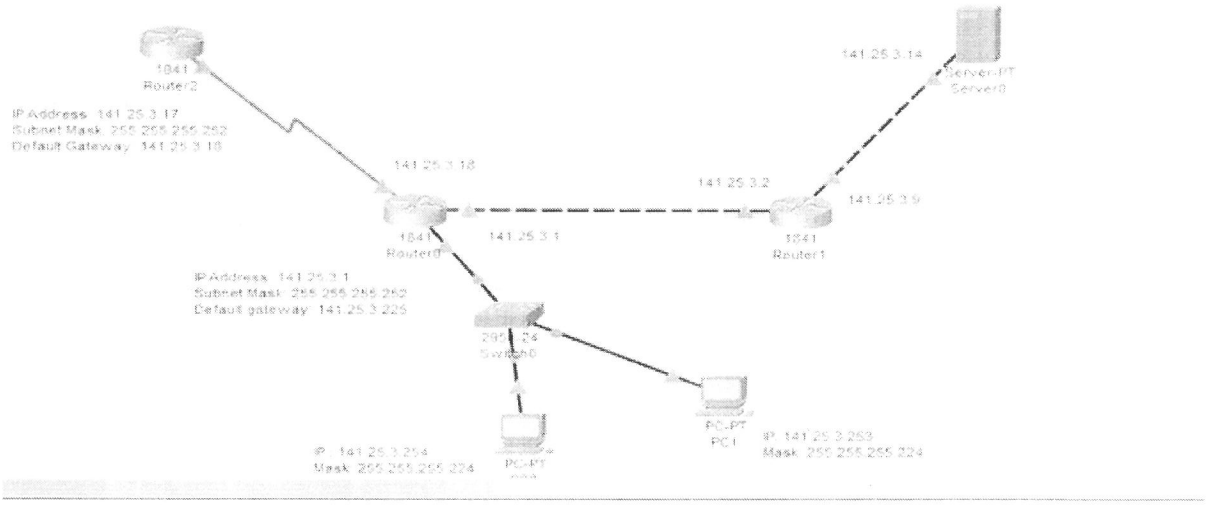


Figure 6 CPT Assignment

Chapter 4: Some Major Web vulnerabilities in Owasp Juice Shop

Injection Attack- Injection attacks are attacks in which data is injected into a web site in order to alter the programme and retrieve the desired information. SQL Injection, code Injection, log Injection, XML Injection, and so forth.

DNS Spoofing- It is a sort of computer security hacking. When data is introduced into a DNS resolver's cache, the name server reverts to an erroneous IP address, routing traffic to the attacker's computer or any other computer. DNS spoofing attacks can go undetected for lengthy periods of time and pose major security consequences.

Session Hijacking – It is a security attack on a user session that occurs across a secure network. Cookies are created by web applications to store the state and user sessions. An attacker can gain access to all of the user data by stealing the cookies.

Phishing- It is a sort of attack that tries to collect sensitive information such as user login passwords and credit card numbers. It happens when an attacker poses as a reliable entity in electronic communication.

Brute force is a sort of assault that relies on trial and error. This attack creates a huge number of guesses and validates them in order to access genuine data such as the user's password and personal identification number. This technique might be used by criminals to crack encrypted data or by security analysts to test the network security of an enterprise.

Denial of Service (DoS) - This is an attack designed to make a server or network resource inaccessible to users. It achieves this by either overwhelming the target with traffic or feeding it information that causes a crash. It attacks a server using a single machine and a single internet connection. It is divided into the following categories:

- **Volume-based assaults** are measured in bits per second and aim to overwhelm the bandwidth of the attacking site.
- **Protocol assaults** use actual server resources and are measured in packets.
- **Application layer attacks** have the purpose of crashing the web server and are measured in requests per second.

Dictionary assaults - This sort of attack saved a list of frequently used passwords and verified them in order to obtain the original password.

URL Interpretation - This is a form of attack in which we may modify particular portions of a URL and cause a web server to send web pages that we are not permitted to view.

Chapter 5: Damn Insecure and Vulnerable Application Report

Challenge – 1 [Insecure Logging]

Objective: - Find out what is being logged where/how and the vulnerable code.

Description: - It is common that android apps logs sensitive information into logcat.

Vulnerable Code: -

```
public void checkout(View view) {
    EditText cctxt = (EditText) findViewById(R.id.ccText);
    try {
        processCC(cctxt.getText().toString());
    } catch (RuntimeException e) {
        Log.e("diva-log", "Error while processing transaction with credit card: " + cctxt.getText().toString());
        Toast.makeText(this, "An error occured. Please try again later", 0).show();
    }
}
```

Figure 7 Vulnerable Code 1

Output: -

```
05-10 16:46:26.260 18831 18845 W Binder  : reply too large data on java level: InterfaceDescriptor = null, code = 1
05-10 16:46:26.455   906   939 E ANDR-PERF-MPCTL: Invalid profile no. 0, total profiles 0 only
05-10 16:46:27.137   906   939 I chatty  : uid=0(root) perf@1.0-servic identical 4 lines
05-10 16:46:30.683   906   939 E ANDR-PERF-MPCTL: Invalid profile no. 0, total profiles 0 only
05-10 16:46:30.770  8546  8546 E diva-log: Error while processing transaction with credit card: 5253552766
05-10 16:46:32.816  1709  5725 W NotificationService: Toast already killed. pkg=jakhar.aseem.diva callback=android.app.ITransientNotification$Stub$Proxy@25c79ad
05-10 16:46:33.131   906   939 E ANDR-PERF-MPCTL: Invalid profile no. 0, total profiles 0 only
05-10 16:46:33.340   923  1965 W SurfaceFlinger: Attempting to destroy on removed layer: bbc21df Toast#0
05-10 16:46:39.861  1709  1735 E BatteryExternalStatsWorker: no controller energy info supplied for bluetooth
05-10 16:46:43.805  1709  1769 W WindowManager: removeWindowToken: Attempted to remove non-existing token: android.os.Binder@6ee5200
05-10 16:46:48.470 24430 26617 V TimeServer: Syncing TimeServer
05-10 16:46:48.476 24430 26617 D e       : request time failed: java.net.UnknownHostException: Unable to resolve host "0.de.pool.ntp.org": No address associated with h
ostname
```

Figure 8 Output1

Command: - adb logcat

Prevention: - This sensitive data is being stored in plain text, thus is visible over here. We should be cautious while saving such information and not store it in plain text.

Challenge – 2 [Hardcoding issue-1]

Objective: - The goal is to find out the vendor key and enter it into the application to gain access.

Description: - It is common that developers hardcode sensitive information in the application source code.

Vulnerable Code: -

```
public void access(View view) {
    if (((EditText) findViewById(R.id.hcKey)).getText().toString().equals("vendorsecretkey")) {
        Toast.makeText(this, "Access granted! See you on the other side :)", 0).show();
    } else {
        Toast.makeText(this, "Access denied! See you in hell :D", 0).show();
    }
}
```

Figure 9 Vulnerable Code 2

Application: -



Figure 10 DIVA HardCoding Issues

Prevention: - Sensitive Data shouldn't be hardcoded in the source code.

Challenge – 3 [Insecure Data Storage-1]

Objective: - Find out where the user-entered data is being stored and also vulnerable code.

Description: - Storing essential data like passwords and user-name needs a secure mechanism. Developers use files, databases, or saved setting to store these kinds of data in plain text.

Vulnerable Code: -

```
public void saveCredentials(View view) {
    SharedPreferences.Editor spedit = PreferenceManager.getDefaultSharedPreferences(this).edit();
    spedit.putString("user", ((EditText) findViewById(R.id.idslUsr)).getText().toString());
    spedit.putString("password", ((EditText) findViewById(R.id.idslPwd)).getText().toString());
    spedit.commit();
```

Figure 11 Vulnerable Code 3

Application: -



Figure 12 Insecure Data Storage

Command: - adb shell

cd data/data/jakhar.aseem.diva/shared_prefs

cat jakhar.aseem.diva_preferences.xml

Output: -

```
river:/data/data/jakhar.aseem.diva/shared_prefs # cd ..
river:/data/data/jakhar.aseem.diva # cd shared_prefs
river:/data/data/jakhar.aseem.diva/shared_prefs # ls
WebViewChromiumPrefs.xml jakhar.aseem.diva_preferences.xml
river:/data/data/jakhar.aseem.diva/shared_prefs # cat jakhar.aseem.diva_preferences.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="password">I am testing</string>
    <string name="notespin">5555</string>
    <string name="user">test</string>
</map>
```

Figure 13 Output2

Prevention: - Passwords should always be encrypted and then stored.

Challenge – 4 [Insecure Data Storage-2]

Objective: - Find out where the user-entered data is being stored and also vulnerable code.

Description: - Storing essential data like passwords and user-name needs a secure mechanism. Developers use files, databases, or saved setting to store these kinds of data in plain text.

Vulnerable Code: -

```
public void saveCredentials(View view) {
    try {
        this.mDB.execSQL("INSERT INTO myuser VALUES ('" + ((EditText) findViewById(R.id.ids2Usr)).getText().toStrin
        this.mDB.close();
    } catch (Exception e) {
        Log.d("Diva", "Error occurred while inserting into database: " + e.getMessage());
    }
    Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
}
```

Figure 14 Vulnerable Code 4

Application: -



Figure 15 DIVA Insecure Data Storage 2

Command: - adb shell

Cd /data/data/jakhar.aseen.diva/database

Cat ids2

Output: -

```
com.dolby.dax2appUI                              com.motorola.carriersettingsext
river:/data/data # cd jakhar.aseem.diva/
river:/data/data/jakhar.aseem.diva # ls
app_textures cache      databases    uinfo4429219546601230721tmp uinfo7381513305784900 6tmp
app_webview  code_cache shared_prefs uinfo5711914962279904930tmp uinfo9194591983163822459tmp
river:/data/data/jakhar.aseem.diva # cd databases
river:/data/data/jakhar.aseem.diva/databases # ls
divanotes.db divanotes.db-shm divanotes.db-wal ids2 sqli sqli-shm sqli-wal
river:/data/data/jakhar.aseem.diva/databases # cd ids2
/system/bin/sh: cd: /data/data/jakhar.aseem.diva/databases/ids2: Not a directory
2|river:/data/data/jakhar.aseem.diva/databases # cat ids2
test22testuserpasswordHiiiiAnshSoni)HelloHashedpasswordtrail2pass2trail2uuyyyyyyyttttttttrailllll6river:/data/data/jakhar.aseem.diva/databases # s (locale TEXT)
```

Figure 16 Output3

Prevention: - Passwords should be hashed and then stored.

Challenge – 5 [Insecure Data Storage-3]

Objective: - Find out where the user-entered data is being stored and also the code making vulnerable.

Description: - Storing essential data like passwords and user-name needs a secure mechanism. Developers use files, databases, or saved setting to store these kinds of data in plain text.

Vulnerable Code: -

```
public void saveCredentials(View view) {
    EditText usr = (EditText) findViewById(R.id.ids3Usr);
    EditText pwd = (EditText) findViewById(R.id.ids3Pwd);
    try {
        File uinfo = File.createTempFile("uinfo", "tmp", new File(getApplicationInfo().dataDir));
        uinfo.setReadable(true);
        uinfo.setWritable(true);
        FileWriter fw = new FileWriter(uinfo);
        fw.write(usr.getText().toString() + ":" + pwd.getText().toString() + "\n");
        fw.close();
        Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
    } catch (Exception e) {
        Toast.makeText(this, "File error occurred", 0).show();
        Log.d("Diva", "File error: " + e.getMessage());
    }
}
```

Figure 17 Vulnerable Code 5

Application: -



Figure 18 DIVA Insecure Data Storage3

Command: - adb shell

cd /data/data/jakhar.aseem.diva/

cat uinfo950407963180713039tmp

Output: -

```
river:/data/data # cd jakhar.aseem.diva/
river:/data/data/jakhar.aseem.diva # ls
app_textures uinfo13761721154321843 68tmp uinfo5608901209422639113tmp uinfo7381513057849006tmp  uinfo8191390853651632 84tmp  uinfo9194591983163822459tmp
app_webview  uinfo19810883385153162 13tmp uinfo5642455097674435892tmp uinfo7411951014198294 54tmp  uinfo8546854954505872144tmp uinfo950407963180713039tmp
cache        uinfo2434920497401369663tmp uinfo5711914962279904930tmp uinfo7484905391476637449tmp uinfo8690233272574767675tmp
code_cache   uinfo3986556230685624081tmp uinfo6411781490655318476tmp uinfo7642853534713434450tmp uinfo8695261645049595332tmp
databases    uinfo4429219546601230721tmp uinfo6467884831250579 83tmp  uinfo7710329331654878635tmp uinfo8885420131979390 99tmp
shared_prefs uinfo5053984821550209007tmp uinfo6499945707766854762tmp uinfo7744757584416166000tmp uinfo9033254228650278055tmp
river:/data/data/jakhar.aseem.diva # cat uinfo5711914962279904930tmp
trail3:trailpass
river:/data/data/jakhar.aseem.diva # cat uinfo950407963180713039tmp
khushi:khushi
river:/data/data/jakhar.aseem.diva #
```

Figure 19 Output4

Prevention: - Passwords should not be insecurely stored in a plain text format; encryption algorithms should be used.

Challenge – 6 [Insecure Data Storage-4]

Objective: - Find out where the user-entered data is being stored and also the code making vulnerable.

Description: - Storing sensitive data as a plain text is completely insecure.

Vulnerable Code: -

```
public void saveCredentials(View view) {
    EditText usr = (EditText) findViewById(R.id.ids4Usr);
    EditText pwd = (EditText) findViewById(R.id.ids4Pwd);
    try {
        File uinfo = new File(Environment.getExternalStorageDirectory().getAbsolutePath() + "/.uinfo.txt");
        uinfo.setReadable(true);
        uinfo.setWritable(true);
        FileWriter fw = new FileWriter(uinfo);
        fw.write(usr.getText().toString() + ":" + pwd.getText().toString() + "\n");
        fw.close();
        Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
    } catch (Exception e) {
        Toast.makeText(this, "File error occurred", 0).show();
        Log.d("Diva", "File error: " + e.getMessage());
    }
```

Figure 20 Vulnerable Code 6

Application: -

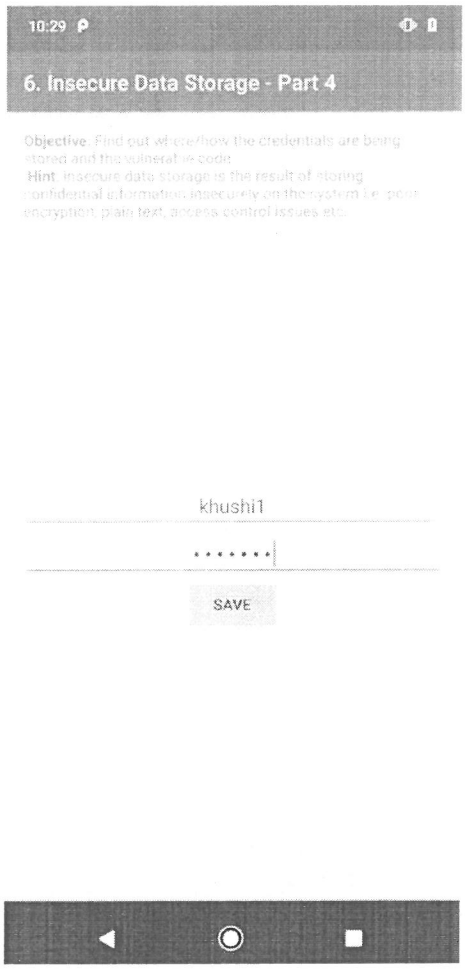


Figure 21 DIVA Insecure Data Storage4

Command: - adb shell

Cd mnt/sdcard

Cat /. uinfo.txt

Output: -

```
        C◆O   "#(◆P ◆zKb◆□B◆n
                          C◆O      "#(◆◆◆◆◆◆◆◆◆◆◆river:/mnt/sdc
river:/ $ cd mnt/sdcard
river:/mnt/sdcard $ cd /.uinfo.txt
/system/bin/sh: cd: /.uinfo.txt: No such file or directory
2|river:/mnt/sdcard $ cat .uinfo.txt
khushi1:khushi1
river:/mnt/sdcard $
```

Figure 22 Output5

Prevention: - The file name containing sensitive data should not be hardcoded in the source code.

Challenge – 7 [Input Validation Issues-1]

Objective: - Access all user data without knowing any username.

Description: - There are three users by default and your task is to output data of all the three users with a single malicious search.

Application: -



Figure 23 DIVA Input Validation Issues1

Prevention: - Proper validation is required and at the input area strings should not be concatenated.

Challenge – 8 [Input Validation Issues-2]

Objective: - Try accessing any sensitive information apart from web URL.

Description: - There are three users by default and your task is to output data of all the three users with a single malicious search.

Application: -

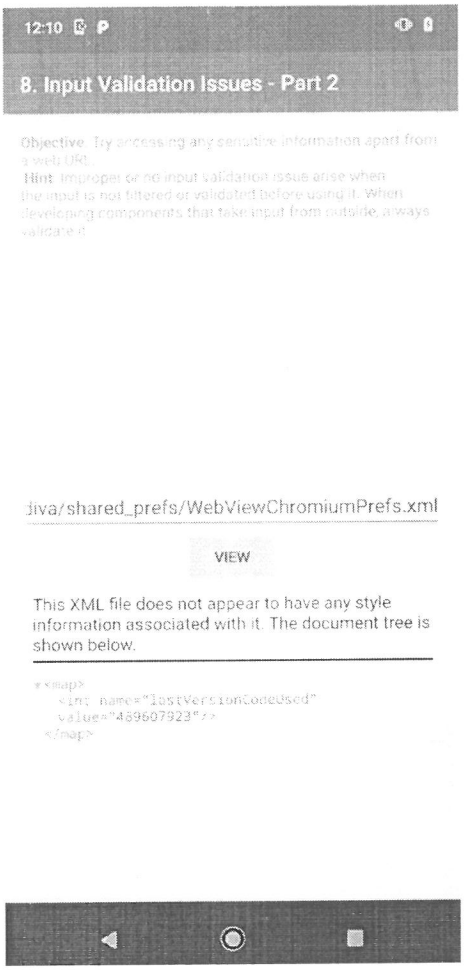


Figure 24 DIVA Input Validation Issues2

Prevention: - The file:/// should be blacklisted so no one can access a file through its location from here.

Challenge – 9 [Access Control Issues -1]

Objective: - Access the information, without clicking this button.

Description: - The intent-filter is generally used for a protection mechanism, but if you use it with activity then it can be even used to export publicly. Which will allow any application from outside to use this activity and view the API credentials.

Vulnerable Code: -

```
public void viewAPICredentials(View view) {
    Intent i = new Intent();
    i.setAction("jakhar.aseem.diva.action.VIEW_CREDS");
    if (i.resolveActivity(getPackageManager()) != null) {
        startActivity(i);
        return;
    }
    Toast.makeText(this, "Error while getting API details", 0).show();
    Log.e("Diva-acl", "Couldn't resolve the Intent VIEW_CREDS to our activity");
}
```

Figure 25 Vulnerable Code 9

Application: -



Figure 26 DIVA Acces Control 1

Command: - adb shell

am start jakhar.aseem.diva/.APICredsActivity

Output: -

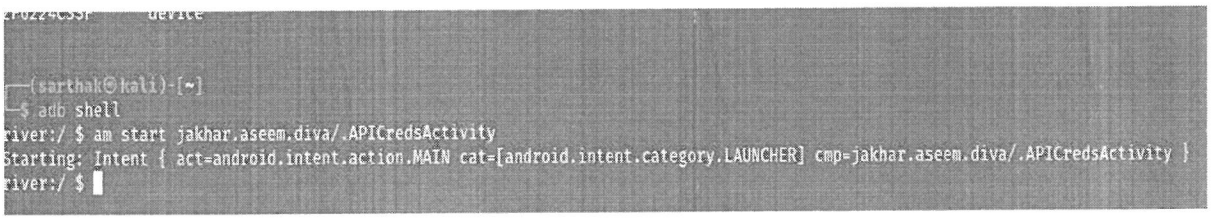


Figure 27 Output6

Prevention: - The access to such credentials should be accessed by some authorized user only.

Challenge – 10 [Access Control Issues -2]

Objective: - Access the information, without clicking this button.

Description: - The intent-filter is generally used for a protection mechanism, but if you use it with activity then it can be even used to export publicly. Which will allow any application from outside to use this activity and view the API credentials.

Vulnerable Code: -

```
public void viewAPICredentials(View view) {
    Intent i = new Intent();
    boolean chk_pin = ((RadioButton) findViewById(R.id.aci2rbregnow)).isChecked();
    i.setAction("jakhar.aseem.diva.action.VIEW_CREDS2");
    i.putExtra(getString(R.string.chk_pin), chk_pin);
    if (i.resolveActivity(getPackageManager()) != null) {
        startActivity(i);
        return;
    }
    Toast.makeText(this, "Error while getting Tweeter API details", 0).show();
    Log.e("Diva-aci1", "Couldn't resolve the Intent VIEW_CREDS2 to our activity");
}
```

Figure 28 Vulnerable Code 10

Application: -



Figure 29 DIVA Access Control 2

Command: - adb shell am start jakhar.aseem.diva/.APICreds2Activity "check_pin" "false"

Output: -

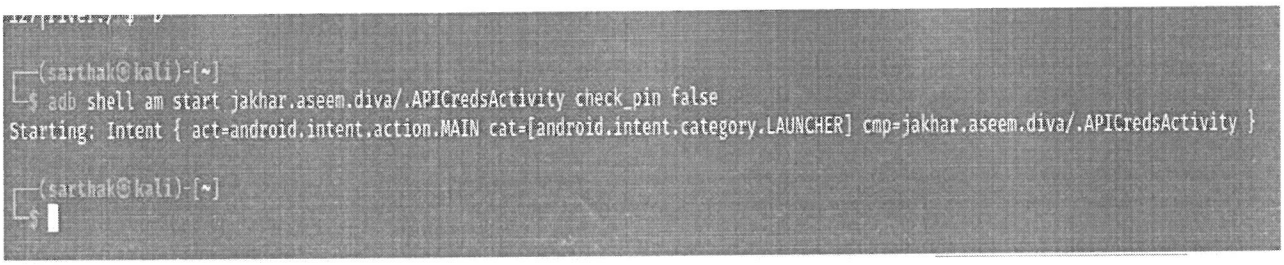


Figure 30 Output7

Prevention: - The access to such credentials should be accessed by some authorized user only.

Challenge – 11 [Access Control Issues -3]

Objective: - The Goal is to access these private notes, without entering the PIN

Description: - An androidManifest.XML file reveals that there is a content provider registered and it is explicitly exported using android_exported="true".

Vulnerable Code: -

```
<provider android:name="jakhar.aseem.diva.NotesProvider" android:enabled="true" android:exported="true"
android:authorities="jakhar.aseem.diva.provider.notesprovider"/>
```

Figure 31 Vulnerable Code 11

Command: - adb forward tcp:31415 tcp:31415

Drozer console connect

run scanner.provider.finduris -a com.jakhar.aseem.diva

run app.provider.query

content://jakhar.aseem.diva.provider.notesprovider/notes/

Output: -

```
Accessible content URIs:
  content://jakhar.aseem.diva.provider.notesprovider/notes/
  content://jakhar.aseem.diva.provider.notesprovider/notes
dz> run scanner.provider.finduris -a jakhar.aseem.diva
Scanning jakhar.aseem.diva...
Able to Query    content://jakhar.aseem.diva.provider.notesprovider/notes/
Unable to Query  content://jakhar.aseem.diva.provider.notesprovider
Unable to Query  content://jakhar.aseem.diva.provider.notesprovider/
Able to Query    content://jakhar.aseem.diva.provider.notesprovider/notes

Accessible content URIs:
  content://jakhar.aseem.diva.provider.notesprovider/notes/
  content://jakhar.aseem.diva.provider.notesprovider/notes
dz> run app.provider.query content://jakhar.aseem.diva.provider.notesprovider/note
s
| _id | title     | note                                    |
| 5   | Exercise  | Alternate days running                  |
| 4   | Expense   | Spent too much on home theater          |
| 6   | Weekend   | b333333333333r                          |
| 3   | holiday   | Either Goa or Amsterdam                 |
| 2   | home      | Buy toys for baby, Order dinner         |
| 1   | office    | 10 Meetings. 5 Calls. Lunch with CEO    |
```

Figure 32 Output7

Prevention: - Specific credentials should be required to check the who all are authorized to access the notes.

Challenge – 12 [Hardcoding Issues - 2]

Objective: - The goal is to find out the vendor key and enter it into the application to gain access.

Description: - It is common that developers hardcode sensitive information in the application source code.

Vulnerable Code: -

```
package jakhar.aseem.diva;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class Hardcode2Activity extends AppCompatActivity {
    private DivaJni djni;

    /* access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.supp
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hardcode2);
        this.djni = new DivaJni();
    }
```

Figure 33 Vulnerable Code12

Application: -



Figure 34 DIVA Hardcoding Issues2

Command: - cd diva/lib/arm64-v8a

ls

strings libdivajni.so

Output: -

```
┌──(root💀kali)-[/home/…/diva-apk-file/diva/lib/arm64-v8a]
└─# strings libdivajni.so
__cxa_finalize
__cxa_atexit
Java_jakhar_aseem_diva_DivaJni_access
strncmp
Java_jakhar_aseem_diva_DivaJni_initiateLaunchSequence
strcpy
JNI_OnLoad
libstdc++.so
libm.so
libc.so
libdl.so
_edata
__bss_start
__bss_start__
__bss_end__
__end__
_end
libdivajni.so
olsdfgad;lh
.dotdot
GCC: (GNU) 4.9 20140827 (prerelease)
.shstrtab
.hash
.dynsym
.dynstr
.rela.dyn
.rela.plt
.text
.rodata
.eh_frame_hdr
.eh_frame
.init_array
```

Figure 35 Output8

Prevention: - Sensitive data should not be hardcoded in source code and secret keys should be encrypted and then stored in libraries.

Challenge – 13 [Input Validation - 3]

Objective: - To crash the system.

Description: - Sometimes developers don't implement the length limit. Hackers can use this to perform a DOS attack and crash the system.

Vulnerable Code: -

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_input_validation3);
    this.djni = new DivaJni();
}

public void push(View view) {
    if (this.djni.initiateLaunchSequence(((EditText) findViewById(R.id.ivi3CodeText)).getText().toString()) != 0) {
        Toast.makeText(this, "Launching in T - 10 ...", 0).show();
    } else {
        Toast.makeText(this, "Access denied!", 0).show();
    }
}
```

Figure 36 Vulnerable Code13

Application: -



Figure 37 DIVA Input validation 3

Output: -

The screen after clicking on the red button started appearing white and the diva application got hung.

Prevention: - Put a limit to the input being given

MobSF Report of DIVA Application

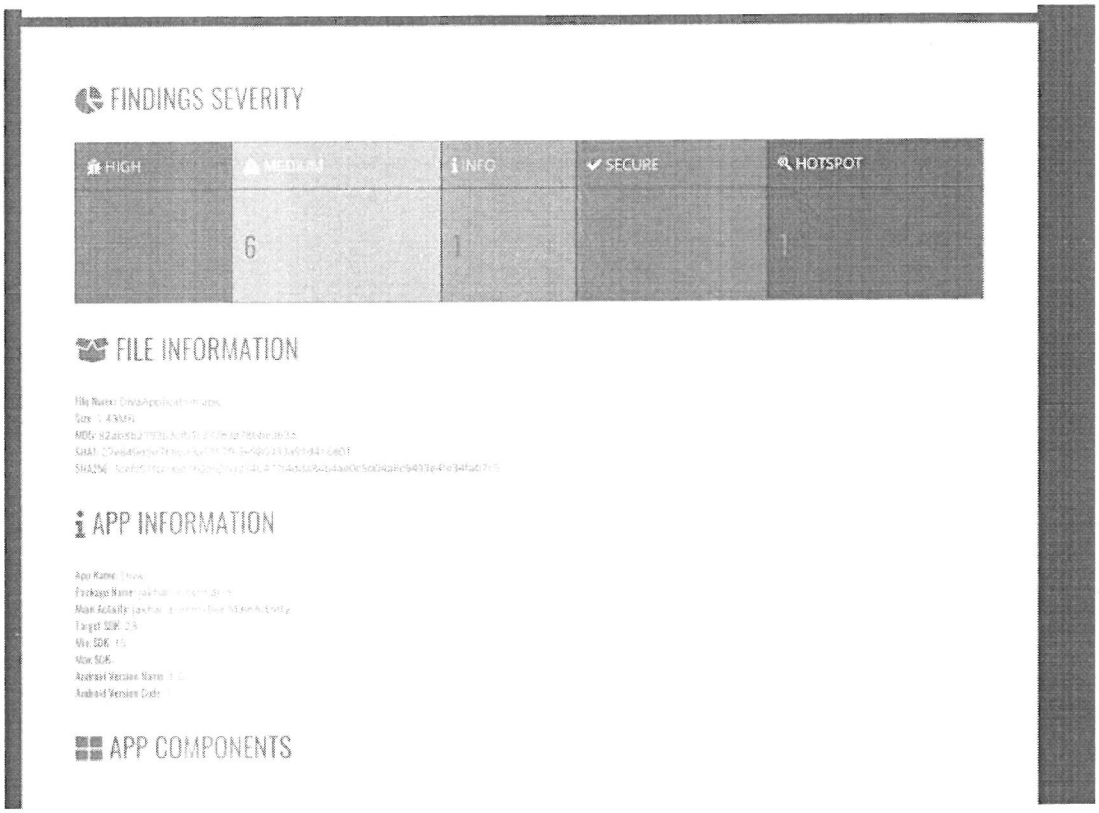


Figure 38 MobSF report Snapshot

Chapter 6: Vulnerability Assessment and Penetration Testing

Vulnerability Assessment (VA)

A Vulnerability Assessment is an automatic examination of network devices, servers, online applications, and systems to discover critical vulnerabilities and configuration flaws that an attacker may exploit. VA is often performed within the network on internal devices and can occur as frequently as once every day. The solution to the question "What is the possible weakness in my network?" is vulnerability assessment.

Penetration Testing

A Penetration Test is a comprehensive expert-led exercise that identifies numerous probable avenues an attacker may use to get into the network. In addition to identifying vulnerabilities, it also highlights the possible damage and subsequent compromise that an attacker may carry out once they have breached the security perimeter. Penetration testing provides a response to the question, "What is a plausible path an attacker may take to exploit the vulnerability?"

Process of Vulnerability Assessment and Penetration Testing:

Goals and objectives

The Goals and Objectives of Vulnerability Assessment and Penetration Testing are defined here. Scope: When administering exams and evaluations, we must explicitly describe the scope of the task. There are three distinct scopes available:

White Box Testing: During this testing, the software's underlying structure, architecture, and code are examined in order to validate the flow of input-output and to enhance security, design, and so on. It's also referred to as transparent box testing, glass box testing, and open box testing.

White box testing often evaluates the following aspects of software code:

- Security flaws
- The flow of data via the code
- The anticipated result
- The operation of conditional loops

- Each statement, object, and function are tested.

The primary purpose of this testing is to ensure that the functioning flow is correct. A series of specified inputs are compared to the expected output; if they do not match, we have found a problem. There are two basic steps: -

- Recognize the source code
- Create and run the test cases

Black Box Testing: Black Box Testing is a sort of software testing in which the functionality of software programs is tested without knowledge of their underlying structure, routes, or implementations. It is primarily concerned with the input and output of the program.

The testing procedure is as follows: -

1. The specifications and requirements are being reviewed.
2. Checks legitimate inputs to see whether the system checks them correctly, then selects incorrect inputs to see if the system under test can identify them.
2. Checks legitimate inputs to see whether the system checks them correctly, then selects incorrect inputs to see if the system under test can identify them.
3. Determine the predicted outcomes based on the inputs.
4. The tester creates test cases
5. Test cases are being carried out.
6. A comparison between actual and predicted production.
7. All flaws have been corrected.

Grey Box Testing: This is a hybrid of black box and white box testing. With knowledge of the internal network and system, we test from either the exterior or internal network.

Information Gathering

Collecting as much information on the IT environment as possible, such as networks, IP addresses, operating system versions, and so on, is relevant to all of the above-mentioned scope kinds.

Vulnerability Detection

Scanners are used at this phase. Vulnerability scanners such as Nessus, OpenVAS, and Nmap, among others, are used to search for vulnerabilities in the network and IT environment.

Information Analysis and planning

Analyze the discovered vulnerabilities in the preceding step and devise a plan to breach the network and system.

Chapter 7: Tools and Technologies for WebApp and Mobile App Security

1. Burp Suite

This tool is designed primarily for testing web applications. Assume you discovered a test web application that is no longer in use after the production release. This tool may be used to go further into the programme and seek for vulnerabilities. The high severity vulnerabilities can be exploited further to advance the assault. It is the most widely used tool among professional web application security researchers and bug bounty hunters. Its ease of use makes it preferable to free alternatives such as OWASP ZAP. Burp Suite is a collection of tools for web application penetration testing. BurpSuite includes an intercepting proxy that allows the user to view and edit the contents of requests and answers as they travel.

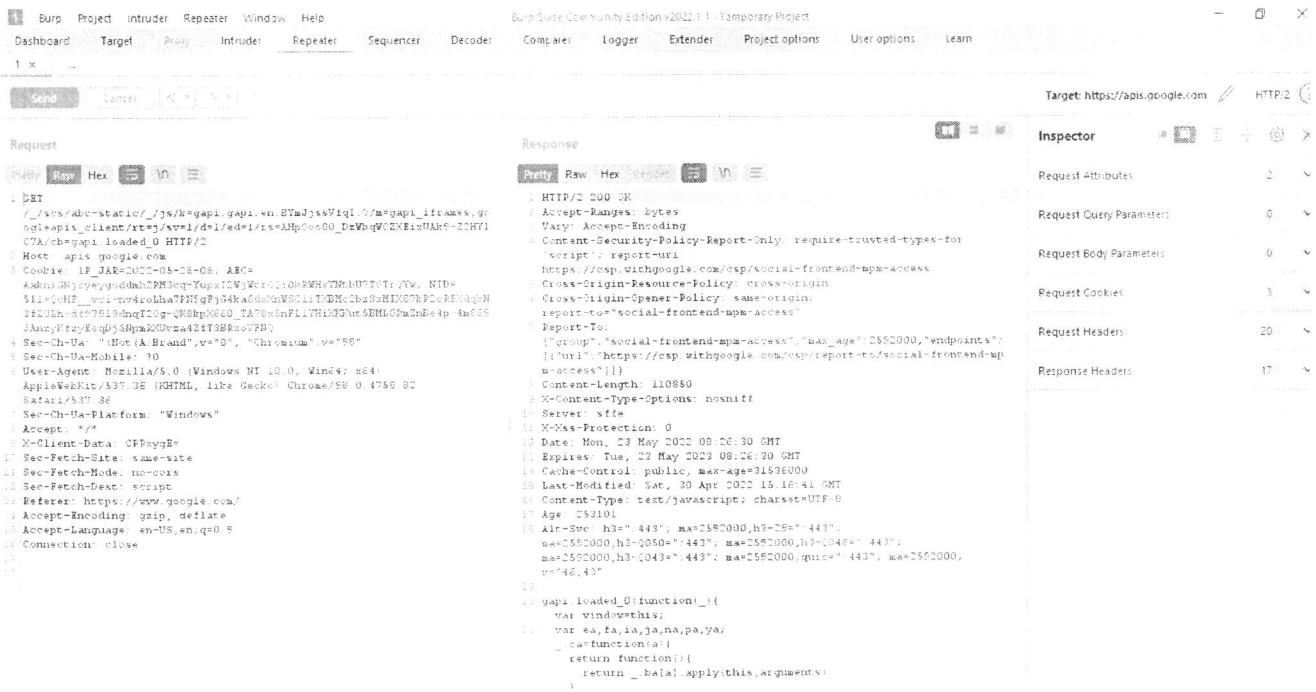


Figure 39 Burp Suite

2. Cisco Packet Tracer

Cisco Systems' Packet Tracer is a cross-platform visual simulation application that allows users to develop network topologies and emulate current computer networks.

Using a simulated command line interface, users may replicate the setup of Cisco routers and switches.

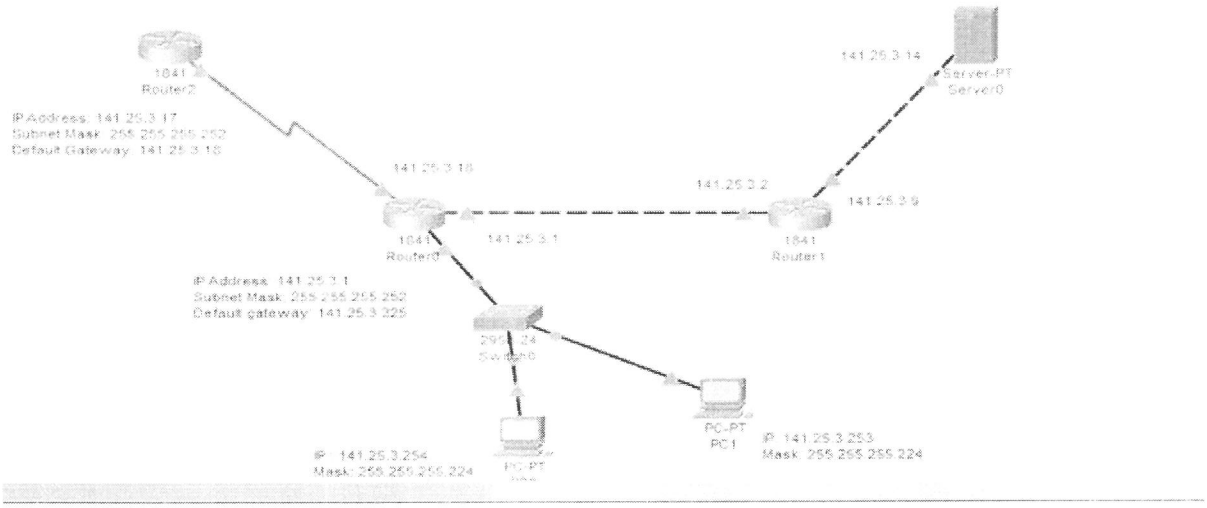


Figure 40 Cisco Packet Tracer

3. MobSF

MobSF, or Mobile Security Framework, is a free and open-source security assessment tool that can conduct both dynamic and static analysis.

Once MobSF is operational, you can now upload any mobile application file and MobSF will examine it.

Static Analysis: MobSF will do source code-based analysis without running the programme through static analysis or static testing to avoid relying on the runtime environment. As a result, static analysis may be utilized in conjunction with application development. Static testing will be performed more efficiently on a regular basis within a set time frame, so that every time an update or release of code is performed, the test is completed without having to run the program.

Dynamic Analysis: Dynamic Analysis testing is a method of testing an application in real time, or while it is running. Nevertheless, the main aim of Dynamic Analysis is to evaluate and search for security flaws in Android applications that are functioning. Dynamic analysis can be used after application development has entered the production phase or before it.



Figure 41 MobSF Interface

4. Drozer

Drozer is the most popular Android security evaluation framework. Drozer allows you to look for security flaws in applications and devices by taking the role of a programme and communicating with the Dalvik Virtual Machine and other apps.

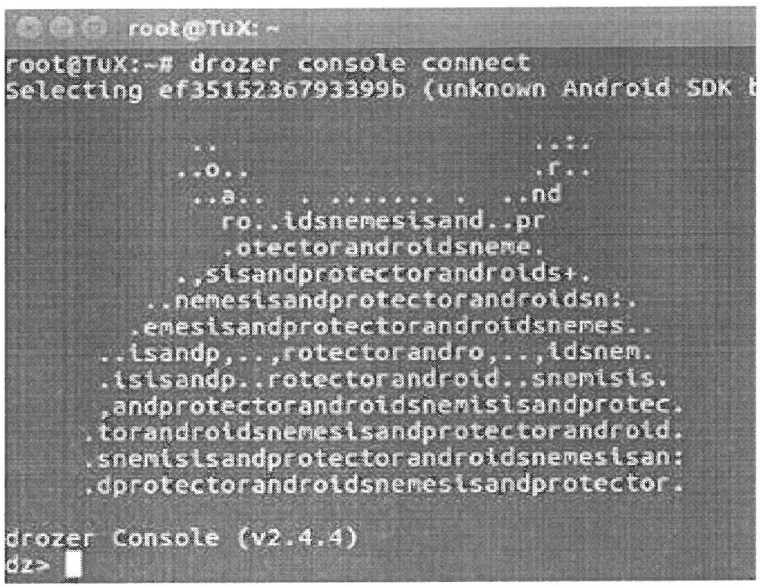


Figure 42 Drozer Console

5. ADB

Android Debug Bridge is a powerful command-line tool for communicating with devices. The adb command simplifies device tasks like as app installation and debugging.

It gives you access to a Unix shell, which you may use to perform various commands on a device.

```
┌──(sarthak㉿kali)-[~]
└─$ adb
Android Debug Bridge version 1.0.41
Version 28.0.2-debian
Installed as /usr/lib/android-sdk/platform-tools/adb

global options:
 -a         listen on all network interfaces, not just localhost
 -d         use USB device (error if multiple devices connected)
 -e         use TCP/IP device (error if multiple TCP/IP devices available)
 -s SERIAL  use device with given serial (overrides $ANDROID_SERIAL)
 -t ID      use device with given transport id
 -H         name of adb server host [default=localhost]
 -P         port of adb server [default=5037]
 -L SOCKET  listen on given socket for adb server [default=tcp:localhost:5037]

general commands:
 devices [-l]             list connected devices (-l for long output)
 help                     show this help message
 version                  show version num
```

Figure 43 ADB

6. Nessus

Nessus is a network and web application vulnerability scanner that can do many sorts of scans and assist a penetration tester in identifying flaws. The attacker can next spend time determining what else can be abused. Some noteworthy aspects of the tool are disabled in the free version. The complete version is robust and packed with capabilities that will aid in the scanning portion of the penetration test.

Nessus can scan for the following vulnerabilities and exposures:

Vulnerabilities in a system that might allow unauthorised control or access to sensitive data.

Configuration Error (e.g., open mail relay, missing patches, etc.).

Some system accounts include default passwords, a few popular passwords, and blank/absent passwords.

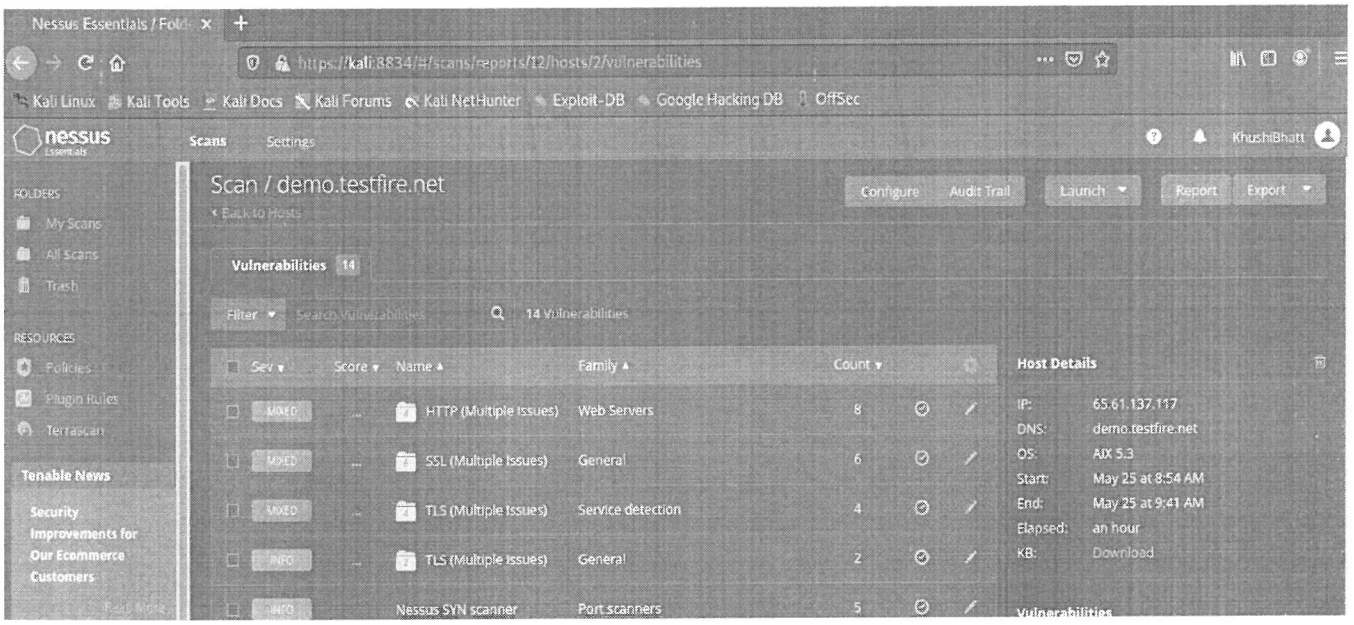


Figure 44 Nessus Interface of WebApp Scanning

7. AWS Fundamentals

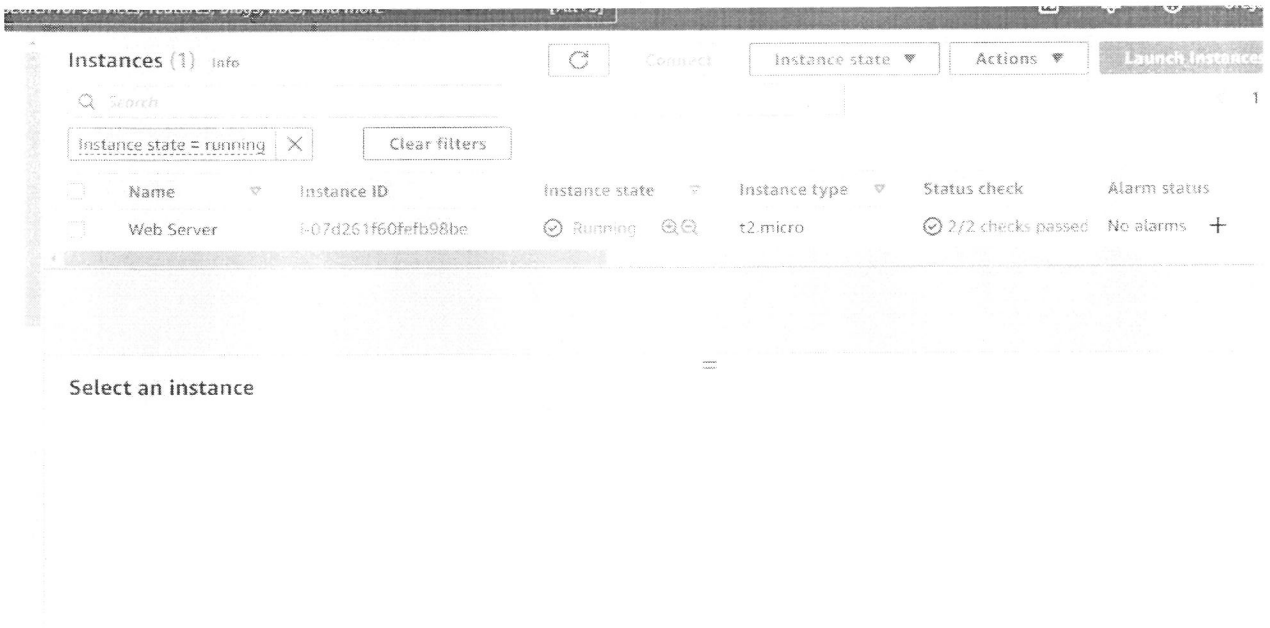


Figure 45 AWS EC2 Instance

8. BitBucket

Bitbucket is our professional-grade Git repository management service. It provides us with a central location to manage git repositories, collaborate on source code, and assist you through the development process. It has fantastic characteristics such as:

Use access control to limit who has access to your source code.

Process control is used to impose a project or team workflow.

Pull requests with in-line commentary for code review cooperation.

Integration with Jira for complete development traceability.

If the features you need aren't already accessible in our Marketplace, you may use our full Rest API to create them.

Chapter8 : Key Learnings and Conclusion

In my PS-II, I covered networking fundamentals as well as securi
progressed through the Basics of Web Application, I learned about the
that exist in a web application. Owasp (Open Online Application Security P
profit organisation that provides a list of the top ten vulnerabilities detected in we
applications. After putting my hands on many vulnerable applications such as OWAS
Shop and DIVA, I learned about a plethora of tools that can be used to pen test an online sit
or a mobile application. Daily SCRUM meetings and the discussion has given me great
learnings on my domain.

InTimeTec has taught me important lessons regarding various behavioural traits that will help me succeed in life. On the road to success, InTimeTec has made me aware of all the abilities that are necessary and will be useful in everyday life. The difficulties that we create in our own lives, InTimeTec has made me aware of them and given me the opportunity to develop myself in both my personal and professional lives.

Referrences

1. What is Security Testing? Types with Example. (2020). Retrieved 26 May 2022, from https://www.guru99.com/what-is-security-testing.html#1

2. OWASP Top Ten Web Application Security Risks | OWASP. (2022). Retrieved 26 May 2022, from https://owasp.org/www-project-top-ten/

3. (2022). Retrieved 26 May 2022, from https://www.cronj.com/blog/what-are-http-requests-and-response/

4. Download Nessus Vulnerability Assessment. (2022). Retrieved 26 May 2022, from https://www.tenable.com/products/nessus

5. Kali Linux Tutorial for Beginners: What is, How to Install & Use. (2020). Retrieved 26 May 2022, from https://www.guru99.com/kali-linux-tutorial.html

6. Getting Started with AWS Cloud Essentials. (2022). Retrieved 26 May 2022, from https://aws.amazon.com/getting-started/cloud-essentials/

7. GitHub - MobSF/Mobile-Security-Framework-MobSF: Mobile Security Framework (MobSF) is an automated, all-in-one mobile application (Android/iOS/Windows) pen-testing, malware analysis and security assessment framework capable of performing static and dynamic analysis. (2022). Retrieved 26 May 2022, from https://github.com/MobSF/Mobile-Security-Framework-MobSF

8. Download Burp Suite Community Edition - PortSwigger. (2022). Retrieved 26 May 2022, from https://portswigger.net/burp/communitydownload

9. Security Onion Documentation — Security Onion 2.3 documentation. (2022). Retrieved 26 May 2022, from https://docs.securityonion.net/en/2.3/

10. (2022). Retrieved 26 May 2022, from https://labs.f-secure.com/tools/drozer/

Plot No. 1/2, Kanakpura Industrial Area, Kanakpura, Sirsi Road, Jaipur-302034
Site No.271, Sri Ganesha Complex, Hosur Main Road, BTM 1st Stage, Bangalore- 560068.

Date: 26th May'22

TO WHOMSOEVER IT MAY CONCERN

Employee Name: Khushi Bhatt

Employee ID: ITTV/EMP/2574
Address: Plot no. 1 / 2, Kanakpura Industrial Area, Kanakpura. Sirsi Road, Jaipur-302034.

This is to certify that **Ms. Khushi Bhatt** is pursuing internship followed by employment with **In Time Tec Visionsoft Pvt. Ltd** from **1st Feb'22** at our **Jaipur** office located at **Plot no. 1/2, Kanakpura Industrial Area, Kanakpura. Sirsi Road, Jaipur-302034.**
During his Internship period, She is entitled to a stipend of Rs. 15,000/- per month, other details are valid as per the offer letter.

For InTimeTec Visionsoft Pvt. Ltd.
Neha Dubey
Authorised Signatory

Regards,

Neha Dubey
Sr. HR Associate
In Time Tec Visionsoft (p) Ltd.

Confidential

This is a Computer Generated Document. No Signature Required.