Internal Web Penetration Testing

When performing internal web assessments, we focus on identifying misconfigurations, weak credentials, and exposed services within an internal network. Here's a comprehensive guide for conducting this type of assessment.

1. Recon with EyeWitness and GoWitness

EyeWitness and **GoWitness** are tools used to capture screenshots, collect information, and even identify active services based on Nmap scan results. This helps get a visual overview of web applications running on the network.

- **EyeWitness Setup**:
  - Capture screenshots of live web servers after an Nmap scan:

```
nmap -p80,443,8080 -sV -oA webscan <target_ip_range>
eyewitness --web --threads 10 --input nmap_scan.xml --o
utput eyewitness_results
```

- **GoWitness Setup**:
  - Similar usage for GoWitness:

```
gowitness nmap -f nmap_scan.xml
```

Explanation:

- **EyeWitness/GoWitness** can quickly help you identify vulnerable web services by providing screenshots and accessible information from the target web servers identified by Nmap.

2. Login - Default Password Check

After identifying services, check if they have **default login credentials**. Many services come with **default usernames and passwords** that might not have been

changed, such as routers, printers, or web applications.

- **Common default credentials**:
  - **admin/admin**
  - **admin/password**
  - **root/toor**
- Check lists such as default-password.info for specific devices or services.

3. Directory Busting

Directory busting helps you discover hidden pages or directories in a web application. Tools like **Gobuster**, **DirBuster**, and **FFUF** automate this process.

- **Gobuster Example**:

```
gobuster dir -u http://<target-ip> -w /usr/share/wordlist
s/dirbuster/directory-list-2.3-medium.txt -t 50
```

  - `u`: Target URL.
  - `w`: Wordlist.
  - `t`: Number of threads.

Targets for Directory Busting:

- **Splash Page**: Pages that are exposed but not linked from the main site (e.g., `/admin`, `/login`).
- **404/403 Responses**:
  - **404 Not Found**: The requested resource doesn't exist.
  - **403 Forbidden**: The resource exists but is restricted.

4. Directory Listing

Sometimes, misconfigured web servers enable **directory listing**, allowing attackers to browse directories that are not intended to be exposed. This can reveal sensitive files such as:

- Configuration files (`config.php`, `.env`)

- Backup files (`backup.zip`, `db_backup.sql`)
- Source code

Example:
Simply navigating to
`http://<target-ip>/uploads/` might reveal a list of files if directory listing is enabled.

5. Unencrypted HTTP Login Page

If the login page is served over **HTTP instead of HTTPS**, the credentials submitted by users are **transmitted in cleartext**. This exposes the credentials to any attacker capable of intercepting traffic via **man-in-the-middle** (MITM) attacks.

- **Vulnerability**:
  - Attackers can intercept HTTP traffic using tools like **Wireshark** or **Ettercap**, capturing the credentials in cleartext.

Example:

```
ettercap -T -M arp:remote /victim_ip/ /gateway_ip/
```

Internal Vulnerabilities

1. **Default Passwords**:
   - Leaving default passwords unchanged allows easy access to services and devices. Always test default credentials on identified login pages.
2. **Unencrypted HTTP Login**:
   - Any HTTP login page should be flagged as a high risk since credentials are sent in cleartext. The fix is to enforce **HTTPS** for all login pages.
3. **Outdated Devices/Services**:
   - Services running on old, unpatched versions may be vulnerable to **known CVEs**. After identifying the version, cross-check for any publicly available exploits (e.g., using `searchsploit`).

External Web Penetration Testing

In external web app testing, the focus shifts towards vulnerabilities that can be exploited from the outside, such as **access control issues**, **IDORs**, or **path traversal** vulnerabilities.

1. IDOR (Insecure Direct Object References)

IDOR occurs when an application allows users to access resources by modifying a parameter value, such as a user ID, without proper authorization checks.

- **Example**:If changing the `12345` to another user's ID (`12346`) allows access to that user's data without proper permission checks, then the web application is vulnerable to IDOR.

```
GET /api/v1/users/12345
```

How to Test:

- Try changing URL parameters to access other users' data.
- Use **Burp Suite** to automate this by capturing requests and modifying ID values.

2. Access Control Bypass (Account Takeover)

Access control vulnerabilities allow unauthorized users to access or modify resources they shouldn't have access to, potentially resulting in **account takeover**.

- **Test Scenario**:
  - Check if you can perform actions like changing another user's password, accessing another user's dashboard, or elevating privileges by modifying roles.

Example:

```
http

POST /api/v1/account/change_password
{
    "user_id": "12345",
```

```
    "new_password": "newpassword"
}
```

By changing the `user_id` to another account (`user_id: 12346`), you might take over someone else's account.

3. Path Traversal

Path traversal allows an attacker to navigate directories on the web server by manipulating file path inputs. This can lead to the exposure of sensitive files such as `/etc/passwd` or web configuration files.

Example:

- **Vulnerable URL**:This URL might allow the attacker to retrieve the `/etc/passwd` file from the server.

```
http

https://example.com/download?file=../../../../etc/passwd
```

How to Test:

- Use `../` sequences in file download or image loading URLs to see if the application is vulnerable to directory traversal.
- Automate testing with tools like **FFUF** or **Burp Suite's Intruder**.
- **Real-World Example**:

```
http

https://api.redacted.tld/getImage?path=../../../etc/passwd
```

External Web Vulnerabilities

1. **IDOR (Insecure Direct Object Reference)**:
    - Allows attackers to access data belonging to other users by modifying identifiers (e.g., user IDs). Proper access controls and object-level authorization are essential.
2. **Access Control Bypass**:
    - A lack of proper access control could allow unauthorized users to perform actions or gain access to restricted data. Ensure access controls are strictly enforced at every level.
3. **Path Traversal**:
    - Path traversal attacks exploit file access vulnerabilities to navigate outside the intended directories. This can expose sensitive files or allow code execution if not mitigated