# Final Security Test Specification and Tools Report

| | |
|---|---|
| Mobile Platform | Hybrid Application |
| Application domain type | Entertainment |
| Authentication | Yes |
| Authentication schemes | Biometric-based authentication ; Factors-based authentication ; ID-based authentication |
| Has DB | Yes |
| Type of database | SQL (Relational Database) |
| Which DB | MySQL |
| Type of information handled | Personal Information ; Confidential Data ; Critical Data |
| Storage Location | Remote Storage (Cloud Database) |
| User Registration | Yes |
| Type of Registration | The users will register themselves |
| Programming Languages | HTML5 + CSS + JavaScript |
| Input Forms | Yes |
| Upload Files | Yes |
| The system has logs | Yes |
| The system has regular updates | Yes |
| The system has third-party | Yes |
| System Cloud Environments | Hybrid Cloud |
| Hardware Specification | No |
| HW Authentication | Basic Authentication (user/pass) |
| HW Wireless Tech | 3G ; 4G/LTE ; 5G ; Bluetooth ; Wi-Fi ; GPS ; RFID ; |

## Testing the DoS or Cellular Jamming Attack

1. **Assessment:** Set up a denial-of-service (DoS) or cellular jamming attack testing to check for potential vulnerabilities/risk.
2. **Preparation:** Before testing, make sure to have the necessary tools and materials, including a network-testing tool that can be used to simulate DoS/jamming attacks, a test environment for the attack, and a virtual machine environment.
3. **Testing Procedure:** * Step 1: Configure the test environment with the attack type chosen. * Step 2: Execute the attack on the test environment. * Step 3: Monitor for any signs of the attack's success such as decreased network performance. * Step 4: If the attack has been successful, record the data or screenshot the results. * Step 5: Repeat steps 1-4 as needed with different attack types and configurations.
4. **Analysis:** Analyze the results of the tests to determine any potential DoS/jamming vulnerabilities or risk in the system.
5. **Conclusion:** Analyze the results and make conclusions on the potential impacts of DoS/jamming attacks on the system.

### Testing Tools:

### Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform

DoS Attack | White-box | Dynamic | Penetration Testing | Nmap | iOS/Android Cellular Jamming | Gray-box | Static | Vulnerability Scan | Nessus | iOS/Android | Black-box | Hybrid | Activity Monitoring | Wireshark | iOS/Android

## Testing the Wi-Fi Jamming Attack

1. Test equipment

In order to test a Wi-Fi Jamming attack, the following test equipment is typically needed: - Wi-Fi Jamming device(s) - A laptop or desktop computer with a wireless adapter - External antennas

1. Set up test environment

The best way to test a Wi-Fi Jamming attack is to create a simulated Wi-Fi environment: - Create a dedicated wireless network or VLAN that has no other Wi-Fi traffic on it. - Connect the test laptop or desktop to the dedicated Wi-Fi network. - Connect the jamming device(s) to a power source.

1. Test the jamming attack

Once the test environment is set up and all the equipment is connected, the jamming attack can be tested:

- Verify that the jamming device(s) are broadcasting the jamming signal and that the test laptop or desktop can detect it.
- Try to make a connection on the test laptop or desktop.
- Verify that the jamming device(s) is blocking or degrading any attempts to connect to the Wi-Fi network.
- Try to connect to other Wi-Fi networks in range to verify that the jamming attack is blocking or degrading any attempts to connect to other Wi-Fi networks outside of the test environment.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Wi-fi Jamming Attack | White-box | Dynamic | Parameter Testing | InSSIDer | iOS, Android |
| Wi-fi Jamming Attack | Grey-box | Static | Code Review | Nmap | iOS, Android |
| Wi-fi Jamming Attack | Black-box | Hybrid | Network Sniffing | Wireshark | iOS, Android |

## Testing the Orbital Jamming Attack

### Testing of Orbital Jamming Attack

Orbital jamming attacks aim to hamper satellite communications and can be difficult to detect. Here are some test procedures that can be implemented to detect such attacks:

**Spectral Analysis using Software Defined Radios (SDRs)**: SDRs can be used to analyze the frequency spectrum and detect any deviations in the spectrum that may indicate the presence of the jamming signals.

**Direct Signal Detection:** If a direct signal is detected, then it can be used to detect the presence of a jamming attack.

**Power Analysis:** Power analysis can be used to identify any abnormalities in the received signal strength. A rapidly fluctuating signal could be an indication of an Orbital Jamming attack.

**Timing Analysis:** Timing analysis can be used to determine the timing of the jamming signals. If the timing is irregular or inconsistent, then it could be an indication of an attack.

**Signal-to-noise Ratio Analysis:** Signal-to-noise ratio analysis can be used to determine the accuracy of the received signal. If the signal-to-noise ratio is low or fluctuating, then it could indicate an orbital jamming attack.

These tests can help detect any presence of a jamming attack, and ultimately lead to further investigation.

### Testing Tools:

| Target | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Orbital Jamming Attack | White-box | Dynamic | Penetration Testing | Burp Suite | iOS |
| Orbital Jamming Attack | Grey-box | Static | Fuzz Testing | OWASP ZAP | Android |
| Orbital Jamming Attack | Black-box | Hybrid | Automated Security Testing | AppScan | iOS, Android |

## Testing the GPS Jamming Attack

#### Testing GPS Jamming Attack

Check if possible to jam satellite signals. This can be done by examining the radio frequencies around the target area.

Once a jamming source is spotted, use a GPS receiver to measure the impact of the interference.

Use a spectrum analyzer to further analyze the spectrum of the jammer. This will allow one to see any changes in the jammer power.

Test the GPS receiver to ensure that it is functioning properly, and that its connections are not being disrupted by the jammer.

Install different types of antennae to measure the range of the jamming attack and to determine how effective it is.

Take readings at different distances to measure the impact of the jammer. This will help determine the strength of the jammer and its likely range.

Test the GPS in different environments such as buildings, forests, and open areas to get an idea of its capabilities.

Finally, use a field-testing kit to measure the impact of the jammer in a more realistic environment. This will provide an accurate measure of the effectiveness of the jamming attack.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| GPS Functionality | White-box | Dynamic | Regression testing | Google Test | Android |
| GPS Functionality | Grey-box | Static | Security Testing | Nessus | iOS |
| GPS Jamming | Black-box | Hybrid | Penetration Testing | Metasploit | Cross-platform |
| GPS Jamming | Black-box | Hybrid | Exploratory Testing | Burp Suite | Cross-platform |

## Testing the Bluesnarfing Attack

1. First, set up a Bluetooth enabled device with a vulnerable version of Bluetooth and make sure the device is in Discoverable mode.

2. Scan the device with a tool such as BlueSnarf, a suite of Bluetooth security tools, to determine the services offered by the Bluetooth enabled device.
3. If the services offered contain authentication and/or authorization, identify and exploit vulnerabilities to complete a successful Bluesnarfing attack.
4. Once the attack is complete, a log of information should be generated and the activity should be monitored to verify the success of the attack.
5. Further testing of the security measures in place should be performed to reduce the likelihood of a successful Bluesnarfing attack.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| VulnerableBT | White-Box | Dynamic | Manual | No | Android, iOS |
| Vulnerable BT | White-Box | Static | Manual | No | Android, iOS |
| Vulnerable BT | Grey-Box | Dynamic | Manual | No | Android, iOS |
| Vulnerable BT | Grey-Box | Static | Manual | No | Android, iOS |
| Vulnerable BT | Black-Box | Dynamic | Manual | No | Android, iOS |
| Vulnerable BT | Black-Box | Static | Manual | No | Android, iOS |
| Vulnerable BT | White-Box | Hybrid | Manual | No | Android, iOS |
| Vulnerable BT | Grey-Box | Hybrid | Manual | No | Android, iOS |
| Vulnerable BT | Black-Box | Hybrid | Manual | No | Android, iOS |

## Testing the Bluejacking Attack

1. Set up the environment:
   - Install a Bluetooth-capable device, such as a laptop or a smartphone.
   - Install a Bluetooth scanner or proximity detection software such as Bluediving.
2. Perform the attack:
   - Enable Bluetooth on both the target device and the attacker's device.
   - On the attacker's device, scan for Bluetooth devices in the area.
   - Once the target device is detected, the attacker can send a message or picture to the target device.
3. Test the results:
   - Observe whether the message or picture was successfully sent.
   - Observe whether the target device has responded to the message or picture.
   - Verify that the data was successfully transferred between devices.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Bluejacking Attack | White-box | Dynamic | Fuzz Testing | Metasploit Framework | Android |
| Bluejacking Attack | Grey-box | Static | Security Code Review | Code Radar | iOS |
| Bluejacking Attack | Black-box | Hybrid | Exploratory Testing | Burp Suite | Windows Phone |

## Testing the Wi-Fi Jamming Attack

#### Testing a Wi-Fi SSID Tracking attack:

1. Scan the local wireless network to record all available SSID (Service Set Identifier) information, including the SSID name, MAC address, and signal strength.
2. Monitor the same wireless network to track changes in SSID information over time.
3. Compare the current and previous records of SSID information to identify any changes.
4. Collect data points using tracking software that can map a device’s route through an area based on the changes in SSID information.
5. Visualize the data points on a map and assess the accuracy of the location data.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Wi-Fi SSID Tracking | White Box | Dynamic | Penetration Testing | Nmap | Android/iOS |
| Wi-Fi SSID Tracking | Grey Box | Dynamic | Exploratory Testing | Selenium | Android/iOS |
| Wi-Fi SSID Tracking | Black Box | Static | Code Review | SonarQube | Android/iOS |
| Wi-Fi SSID Tracking | White Box | Hybrid | Security Testing | OWASP Zed Attack Proxy | Android/iOS |

## Testing the Byzantin Attack

One way to test for a Byzantine attack is to use the "Byzantine Fault Tolerance Test" (BFTT). This is a test designed to detect any malicious behavior in a distributed system. The test can be broken up into the following steps:

Setup the test environment: - Choose a testbed and deploy the system components according to the required configuration. - Install monitoring tools to measure the performance of the system.

Perform the test: - Launch the Byzantine Fault Tolerance Test to detect any instances of malicious behavior on the distributed system. - Generate test transactions that could be potentially malicious. - Measure the system's response to the malicious transactions.

Analyze the results: - Analyze the system's behavior in terms of speed, accuracy, and time to detect malicious transactions. - Compare the results of the test with the theoretical baseline or established standards.

Make changes to secure the system: - Make changes to the system configuration or software to ensure the system is secure. - Monitor the system's performance after the changes are implemented to ensure the system is secure.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tools | Mobile Platform |
|---|---|---|---|---|---|
| Byzantine Attack | White-box | Dynamic | Penetration Testing | Nmap, Wireshark | Android, iOS |
| Byzantine Attack | Grey-box | Static | Code Review | SonarQube | Android, iOS |
| Byzantine Attack | Black-box | Hybrid | Stress Testing | LoadRunner | Android, iOS |

## Testing the On-Off Attack

Testing an On-Off attack is done by conducting a series of tests against a target system to attempt to determine its susceptibility to the attack.

The testing process consists of the following steps:

1. Identify the target system and configure it for the attack.
2. Conduct active reconnaissance to discover the presence and types of services running on the target system.
3. Launch a series of On-Off attacks against the identified services.
4. Analyze the response from the target system to determine if the attack was successful.
5. If successful, note the attack vector(s) used and document the vulnerability that was exploited.
6. If unsuccessful, repeat the process until the target system has been fully tested.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Plataform |
|---|---|---|---|---|---|
| Endpoints | White-box | Dynamic | Exploitation | Burp | Android & iOS |
| Services | Grey-box | Static | Pen. Test | Zed Attack Proxy | Android & iOS |
| Network | Black-box | Hybrid | Scan | Nmap | Android & iOS |

## Testing the Malicious Insider Attack

Educate Employees: Provide regular training to employees on data protection, cybersecurity, and mitigation of malicious insider activity.

Monitor Network Activity : Monitor and audit employees' access to data and system. Look for suspicious activity such as unusually large data transfers or out-of-the-ordinary access attempts.

Separate Duties : Thoroughly separate duties between different personnel to prevent one person from having too much access or control.

Implement Least Privilege Measures: Limit user access by granting only the privileges that are needed, rather than granting full access to all users.

Establish Access Protocols: Establish access protocols and rules regarding data access, such as when, what, and how users are allowed to access certain data.

Conduct Security Audits: Regularly perform security audits on employee accounts to detect malicious activity.

Utilize Network Logging : Utilize network logging to monitor user activity, such as web browsing and application usage.

Install SSL Monitoring: Install an SSL monitoring system to detect any suspicious activity, such as unauthorized access to sensitive data.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Detection | White-box | Dynamic | Functional Testing | OWASP ZAP Proxy | iOS |
| Prevention | Grey-box | Static | Fault Tree Analysis | HP Fortify | Android |
| Response | Black-box | Hybrid | Penetration Testing | Burp Suite | All |

## Testing the Sniffing Attack

Sniffing attacks can be tested using the tools included in most network security programs.

The following steps should be taken to test Sniffing Attacks:

Set up a network and create a designated test environment.

Monitor all of the traffic on the network and detect any signs of suspicious activity. This would include anything out of the ordinary, such as changes in IP addresses, traffic amounts, or unexpected errors.

Use the tools included in the security program to detect the sources of the suspicious traffic.

Actively scan the network in order to detect any malicious programs that might be present.

Use a packet sniffer to log all packets and analyze any anomalous behavior.

Detect and block any attempts to hijack communications or take control of any system on the network.

Perform a vulnerability assessment of the network to ensure that no weak spots are present that could be taken advantage of by a sniffing attack.

Test to see if your organization’s security measures are able to detect and block against any sniffing activity.

Once the tests have been completed, perform a validation process to ensure that all of the results are accurate and the attack can be blocked and remediated properly.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Sniffing Attacks | White-box | Static | Code Review | Wireshark | No |
| Sniffing Attacks | Grey-box | Hybrid | Fuzz Testing | Tcpdump | No |
| Sniffing Attacks | Black-box | Dynamic | Protocol Analysis | Scapy | No |

## Testing the Man-in-the-Middle Attack

### Testing Man-in-the-Middle Attack

Man-in-the-middle (MitM) attacks are a type of cyber attack where an attacker intercepts the communication between two parties, allowing them to listen in on, modify, or block the communication. In order to test for MitM attacks, you must be able to detect if a third-party is attempting to intercept or modify traffic between two parties.

1. Disable your firewall temporarily and ensure that any antivirus software is turned off.
2. Use command line tools such as **netstat** to monitor incoming and outgoing connections to your system.
3. Monitor inbound and outbound traffic for suspicious IP addresses or packets.
4. Use **traceroute** and **ping** tests to identify malicious intermediaries on the network.
5. Use **arp spoofing** to detect and stop man-in-the-middle attacks.
6. Monitor log files for suspicious or irregular behavior.
7. Implement solutions such as SSL/TLS or IPSec encryption to ensure the authenticity of data transmitted over the network.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Man-in-the-Middle | White-box | Dynamic | Penetration | Nmap | iOS |
| Man-in-the-Middle | Grey-box | Static | Source Code | Infer | Android |
| Man-in-the-Middle | Black-box | Hybrid | Vulnerability | Burp Suite | Windows |
| Man-in-the-Middle | Grey-box | Dynamic | Performance | WebLink | iOS |
| Man-in-the-Middle | White-box | Static | Network | Wireshark | Android |
| Man-in-the-Middle | Black-box | Hybrid | Fuzzing | Afl | Windows |

## Testing the Eavesdropping Attack

Eavesdropping attacks are one of the most common and widespread cyberattacks, in which an attacker attempts to gain access and collect sensitive or confidential information sent or stored by individuals or businesses.

Testing for an eavesdropping attack can be accomplished in several ways. Some of the methods that can be used are outlined below:

Network Sniffing: This involves monitoring network traffic for any suspicious activity. Network packet sniffing tools and protocols such as TCPdump and Wireshark can be used to help detect suspicious activity and potential eavesdropping attempts.

System Logs: Reviewing system logs can help to identify any suspicious activity that may be related to an eavesdropping attack. Administrators can also monitor application and network logs for any unauthorized access or changes that may have been made to the system.

Password Testing: Utilizing password-cracking tools such as John the Ripper, administrators can test the strength of passwords that are in use against a list of thousands of common passwords. This offers the best idea of how secure a network is against eavesdropping attacks.

Penetration Testing: This is a more advanced method of testing for eavesdropping attacks and involves launching sophisticated simulated attacks on an organization's system. Penetration testers use tools such as Metasploit and Nmap to assess the effectiveness of an organization's security measures against different types of attack vectors, including eavesdropping attacks.

Antivirus Scanning: This is an important step that should be undertaken regularly. Regularly scanning the system with a reliable antivirus will help to identify any malicious software that may have been installed as part of an eavesdropping attack.

### Testing Tools:

| Target Testing | Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Eavesdropping Attack | White-box | Dynamic | Penetration Testing | Nexpose, Nmap, Metasploit | Android, iOS |

## Testing the CSRF Attack

Below are the steps to test for a CSRF attack:

1. Ensure that the application is properly allowing and validating CSRF tokens.
2. Identify all form entry points in the application.
3. Use a proxy tool such as Burp Suite to intercept the request.
4. Modify the request on the proxy tool with an incorrect CSRF token.
5. Submit the request and observe the response.
6. If an error is returned, the application is properly validating the token and the attack is not successful.
7. If the request is accepted, the application is not properly validating the CSRF token and the attack is successful.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| CSRF Attack | White-Box | Dynamic | Manual | Burp Suite | iOS & Android |
| | Grey-Box | Static | Automated | Nikto | IOS & Android |
| | Black-Box | Hybrid | Automated | Owasp Zap | iOS & Android |

## Testing the SQLi Attack

### Testing SQLi Attack

#### 1. Cross-Site Scripting (XSS)

Cross-site scripting (XSS) is a type of attack that involves a malicious script being injected into a website. When a user visits the website, the script is executed in their browser. To test for XSS, we can insert a malicious script into the website and see if the script is executed.

#### 2. SQL Injection

SQL injection is a type of attack where malicious code is inserted into the SQL query in order to gain access to data stored in a database. To test for SQLi attacks, we can insert a malicious query into the website and check if it is executed on the database.

#### 3. Parameter Tampering

Parameter tampering is a type of attack where parameters of a URL are modified to gain access to restricted information or data. To test for parameter tampering, we can change the values of URL parameters and see if the website is responding with different outputs.

#### 4. Brute-Force Attack

Brute-force attacks are attempts to guess the username and password of a system using automated tools. To test for brute-force attacks, we can use tools that try different combinations of usernames and passwords and check if the system is able to detect the attempts.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| SQLi Attack | White-box | Dynamic | Manual | SQLMap | iOS / Android |
| SQLi Attack | Grey-box | Static | Automated | OWASP ZAP | iOS / Android |
| SQLi Attack | Black-box | Hybrid | Exploration | Burp Suite | iOS / Android |

## Testing the XSS Attack

#### Testing XSS Attack

1. Identify the potential injection points: - Take a look at webpages for available input fields, parameters in a URL, cookies, and source code.
2. Input test payloads: - One of the most common XSS payloads is the `<script>alert(1)</script>` tag.
3. Monitor the response: - If your payload is successfully executed, you'll likely get an alert box or some other type of notification.
4. Assess for vulnerabilities: - Once you've confirmed where the vulnerabilities are, you can start to control and mitigate the attack.

#### Prevention of XSS Attack

1. Employees: - Train employees to be aware of XSS attacks so they can spot and report suspicious activities.
2. Validate input: - Validate and filter any user input to ensure that only properly formatted input is accepted.
3. Sanitize input: - Sanitizing data is the process of removing Javascript, HTML, and other malicious code from user input.
4. Use Trusted data: - Be sure to use trusted input whenever possible to minimize the chances of XSS attacks.
5. Use Security Headers: - Use security headers, such as HTTP Strict Transport Security (HSTS), to help prevent XSS attacks.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| XSS Attack | White-box | Dynamic | Manual | Burp Suite | iOS, Android |
| | Grey-box | Static | Automated | HP WebInspect | |
| | Black-box | Hybrid | | Acunetix | |

## Testing the SSRF Attack

#### Testing SSRF Attack

Set up an attacker controlled server: This server needs to be able to accept connections from the target server and should log the requests from the target server like the source IP address, used port, timestamp, and any data sent over with the request.

Prepare a crafted URL: This URL should lead to the attacker controlled server and is used for the SSRF attack.

Send the crafted URL to the target server: This can be done manually by providing the URL as an input for a user or programmatically by sending the URL with a POST request to the target server.

Check the logs of the attacker controlled server: After the URL is sent to the target server, the attacker controlled server should log the requests sent by the target server. This is used to analyze the type of requests being sent, the data being sent, and the source IP address.

Monitor the environment of the target server: When the target server makes requests to the attacker controlled server, the attacker can set up environment variables, files, resources, etc. in order to exploit the target server.

Try to gain access to the target server: If the SSRF attack was successful, the attacker can try to access the target server from the attacker controlled server by using the access gained from the exploited environment of the target server.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Server | White-Box | Dynamic | Input Validation | DSSS - Detection of SSRF | No |
| Server | Gray-Box | Dynamic | Fuzzing | Burpsuite | No |
| Server | Black-Box | Dynamic | Vulnerability Scanner | NTOSpider | No |
| Server | White-Box | Static | Source Code Analysis | Code Editors | No |
| Server | Gray-Box | Static | Source Code Auditing | Vega | No |
| Server | Black-Box | Static | Manual Review | Manual Review | No |
| Server | White-Box | Hybrid | Interactive Testing | Manual Review | No |
| Server | Gray-Box | Hybrid | Penetration Testing | Metasploit | No |
| Server | Black-Box | Hybrid | Penetration Testing | Zed Attack Proxy | No |
| Mobile | White-Box | Dynamic | Input Validation | DSSS - Detection of SSRF | Yes |
| Mobile | Gray-Box | Dynamic | Fuzzing | Burpsuite | Yes |
| Mobile | Black-Box | Dynamic | Vulnerability Scanner | NTOSpider | Yes |
| Mobile | White-Box | Static | Source Code Analysis | Code Editors | Yes |
| Mobile | Gray-Box | Static | Source Code Auditing | Vega | Yes |
| Mobile | Black-Box | Static | Manual Review | Manual Review | Yes |
| Mobile | White-Box | Hybrid | Interactive Testing | Manual Review | Yes |
| Mobile | Gray-Box | Hybrid | Penetration Testing | Metasploit | Yes |
| Mobile | Black-Box | Hybrid | Penetration Testing | Burp Suite | Yes |

## Testing the Command Injection Attack

1. **Testing for Command Injection**

Command Injection is an attack in which malicious commands are injected into a vulnerable application in order to gain access or obtain privileged information. This type of attack involves taking user input and executing it as code on the system, making it highly dangerous and difficult to detect.

In order to test for command injection, it is important to first identify any areas of the application that may be vulnerable to attack. Applications which take user input and execute it as code are particularly at risk.

Once potential vulnerable areas have been identified, test cases can be created to simulate attempts at injecting malicious code. This can be done by attempting to insert special characters or meta-characters into input fields, such as semi-colons, backticks, or quotation marks. If the application responds in an unexpected manner or creates an externally visible result, such as creating an extra file in the directory, it is likely that command injection is possible.

It is important to note that command injection may not always be easy to spot, as the application may attempt to sanitize user input or where malicious code can be 'hidden' within benign input. As such, testing using all potential input combinations is recommended in order to ensure that vulnerabilities have been adequately identified and addressed.

Once all potential vulnerabilities have been assessed and remediated, it is recommended to repeat the testing process in order to validate the changes.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Command Injection Attack | White-box | Dynamic | Input Fuzzing | Metasploit | iOS/Android |
| Command Injection Attack | Grey-box | Static | Variant Analysis | CodeSonar | iOS/Android |
| Command Injection Attack | Black-box | Hybrid | Interactive Fuzzing | Burp | iOS/Android |

## Testing the Code Injection Attack

Testing code injection attacks involve attempting to inject malicious code into a web application. Testing code injection is typically done using a few different methods.

Manual Tests: Manual tests involve manual input of potentially malicious code into user inputs and forms within the application. If the application does not appropriately filter or reject suspicious input, this can lead to the vulnerability.

Automation Tools: Automation tools, such as Security Scanners, can be used to test for code injection attacks. Tools such as these can detect code injection vulnerabilities with very high accuracy, allowing security teams to quickly identify and mitigate such risks.

Penetration Testing: Penetration testing is a manual assessment of security. During this process testers attempt to gain access to restricted areas within an application, by entering or manipulating code in user input fields. This will allow testers to identify potential security weaknesses in an application, and recommend remediation strategies.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Code Injection | White-box | Dynamic | Unit Testing | HP QTP | Android |
| Code Injection | Grey-box | Dynamic | Penetration Testing | HP Fortify | Android |
| Code Injection | Black-box | Static | Security Scanning | Burp Suite | iOS |
| Code Injection | White-box | Hybrid | Code Review | Splint | iOS |

## Testing the Phishing Attack

1. Create a mock Phishing website: Create a fake website that looks and behaves like the real one in order to fool users into providing their login credentials or other sensitive information.
2. Set up traffic monitoring: Monitor incoming traffic to your website to ensure that the fake site isn’t receiving any legitimate requests.
3. Analyze the behavior of malicious users: Monitor user behavior on your site and look for any suspicious activity that may indicate a potential attack.
4. Send out test emails: Send out phishing emails to members of your staff and monitor their responses. If the email is opened and a link is clicked, you can be sure that your staff needs more education on Phishing attacks.
5. Analyze the results: Once all of the test emails have been sent out and the responses analyzed, prepare a report outlining what worked and what didn’t.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Phishing Attack | White-box | Dynamic | Penetration | Metasploit | iOS, Android |
| Phishing Attack | Grey-box | Static | Source Code | OWASP ZAP | iOS, Android |
| Phishing Attack | Black-box | Hybrid | Fuzzing | Wfuzz | iOS, Android |

## Testing the Pharming Attack

Testing a pharming attack requires several steps:

1. Use a vulnerability scanner to identify and assess the security of the computer systems, networks, and applications.
2. Access the system and perform a detailed analysis of the system logs and audit trails.
3. Perform a packet sniffing test to detect any malicious activity.
4. Check for any Domain Name System (DNS) anomalies and any suspicious domain redirection.
5. Check if the DNS cache poisoning technique has been employed.
6. Use a URL analyzer tool to detect any malicious redirects or domain name changes.
7. Run a malware scan on the system to detect any malicious code or program.
8. Use a honeypot system to detect any suspicious activity on the network.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Network | White-box | Dynamic | Automation | Nmap | N/A |
| Network | White-box | Static | Manual | Wireshark | N/A |
| System | Grey-box | Dynamic | Automation | Nessus | N/A |
| System | Grey-box | Static | Manual | Regshot | N/A |
| Host | Black-box | Dynamic | Automation | Core Impact | Android, iOS |

## Testing the Spoofing Attack

Testing for Spoofing Attacks

Scanning For Open Ports: By using network scanning tools, such as NMAP, one can identify open ports that may be vulnerable to spoofing attacks.

Creating Honeypots: Honeypots can be created and deployed on the network to detect and identify malicious behavior.

Disabling Unused Services: Unused services can be disabled on the system to reduce the attack surface available to an attacker.

Restrict Network Traffic: Network traffic can be restricted on the system to avoid spoofing attacks.

Whitelisting: Only applications, IP addresses and services that are known and trusted should be allowed to access the system. All other traffic should be blocked.

Implement Firewalls and Intrusion Detection Systems: Firewalls and Intrusion Detection Systems (IDS) should be implemented to detect and prevent unauthorized access and activity.

Monitor Network Activity: The network activity should be monitored on a regular basis to identify any suspicious activity that may indicate a spoofing attack.

Regularly Update System Software: Systems should be updated regularly with the latest security patches to address known vulnerabilities that can be exploited by attackers.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Spoofing Attack | White-box | Dynamic | Exploratory Testing | SoapUI | iOS |
| Spoofing Attack | Grey-box | Static | Source Code Analysis | Veracode | Android |
| Spoofing Attack | Black-box | Hybrid | Penetration Testing | Nessus | iOS & Android |

## Testing the Session Fixation Attack

### Testing for Session Fixation Attack

Start by testing for session fixation with known tokens. Generate a known token, log into the application and submit the known token in the login form. If the application successfully logs into the system, the application may be vulnerable to session fixation.

If the form does not accept known tokens and does not reissue the token, the application may be vulnerable to session fixation.

Insert a tracking or logging code into the application and try to use different known tokens and see if they are accepted by the application.

After logged in, renew the session and check if the session ID is changed. If it is same as before, it might be vulnerable to session fixation.

Log out of the application and try to use the session ID that was captured during login. If the application continues to use the same session ID, it might be vulnerable to session fixation.

Create a honeypot page that only registered users can access. Log in with a known token and try to access the page. If the application allows access to the page using the known token, the application might be vulnerable to session fixation.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Session Fixation Attack | White-box | Dynamic | Automated | Web security scanner (e.g. Acunetix, Netsparker, Burp Suite) | N/A |
| Session Fixation Attack | Grey-box | Dynamic | Automated | Web security scanner (e.g. Acunetix, Netsparker, Burp Suite) | N/A |
| Session Fixation Attack | Black-box | Dynamic | Automated | Web security scanner (e.g. Acunetix, Netsparker, Burp Suite) | N/A |
| Session Fixation Attack | White-box | Static | Automated | Code security scanner (e.g. Checkmarx, Veracode, SonarQube) | N/A |
| Session Fixation Attack | Grey-box | Static | Automated | Code security scanner (e.g. Checkmarx, Veracode, SonarQube) | N/A |
| Session Fixation Attack | Black-box | Static | Automated | Code security scanner (e.g. Checkmarx, Veracode, SonarQube) | N/A |
| Session Fixation Attack | White-box | Hybrid | Automated | Code and web vulnerability scanner (e.g. Appscan, Appspider, Arachni) | N/A |
| Session Fixation Attack | Grey-box | Hybrid | Automated | Code and web vulnerability scanner (e.g. Appscan, Appspider, Arachni) | N/A |
| Session Fixation Attack | Black-box | Hybrid | Automated | Code and web vulnerability scanner (e.g. Appscan, Appspider, Arachni) | N/A |
| Session Fixation Attack | White-box | Dynamic | Manual | N/A | N/A |
| Session Fixation Attack | Grey-box | Dynamic | Manual | N/A | N/A |
| Session Fixation Attack | Black-box | Dynamic | Manual | N/A | N/A |
| Session Fixation Attack | White-box | Static | Manual | N/A | N/A |
| Session Fixation Attack | Grey-box | Static | | | |

## Testing the Session Hijacking Attack

Testing Session Hijacking attack requires two machines connected to the same network. The attacker would need to intercept the network traffic from the victim's machine where the session data is being exchanged. It is also important to note that it is possible to hijack cookies from other machines on the same network.

Here are the steps to test a Session Hijacking attack:

Install a network sniffing tool (such as Wireshark) on both machines to monitor network traffic.

Use the sniffing tool to identify any cookies used in the session interaction between the two machines. Note that certain browsers may encrypt the cookie data so the cookie might appear to be encrypted.

Intercept the cookie data on the attacker’s machine - the attacker should now have temporary access to the session data.

Use the sniffing tool to analyze the data transferred between the two machines and confirm if the session data has been compromised.

Try to login to the victim’s session with the stolen credentials or cookies.

If successful, the attacker has hijacked the session and should be able to access the victim’s session data.

Log out of the session on the attacker's machine and remove all evidence of the session hijacking attack.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Session Hijacking Attack | White-box | Dynamic | Code review | Burp Suite | N/A |

| | | | | | |
|---|---|---|---|---|---|
| Session Hijacking Attack | Grey-box | Static | Vulnerability assessment | Nessus | N/A |
| Session Hijacking Attack | Black-box | Hybrid | Penetration testing | Nmap | N/A |

## Testing the Access Point Hijacking Attack

### Testing Access Point Hijacking Attack

To test an Access Point (AP) Hijacking attack, follow these steps:

1. Set up a rogue AP that mimics the legitimate AP. - Ensure that both SSIDs (Service Set Identifiers) and MAC addresses are identical.
2. Create a beacon frame that advertises the rogue AP.
3. Capture the beacon frame using a network monitoring tool. - This will give you an idea of when the legitimate AP is being impersonated.
4. Monitor for clients that attempt to authenticate with the rogue AP. - This will provide further evidence of the attack.
5. Confirm that the rogue AP is functioning by trying to access it using a client device.
6. Review logs for a record of the attack.
7. Attempt to decrypt any traffic captured from the attack. - This will give you an understanding of the data that was stolen.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Access Point Hijacking Attack | White-box | Dynamic | Manual | N/A | N/A |
| Access Point Hijacking Attack | Grey-box | Dynamic | Automation | Nessus | N/A |
| Access Point Hijacking Attack | Black-box | Static | Manual | Nmap | IOS/Android |

## Testing the Cellular Rogue Base Station Attack

**Prerequisite**:

- Wireless Access Point
- Wireless testing tool/software

**Setup**:

- Place the wireless access point with in the testing environment.

**Testing Procedure**:

- Using wireless testing tool/software, create a wireless environment and record the signal strength and wireless connection status of the legitimate wireless base station.
- Place the rogue base station in the same environment and power it up.
- Again measure the signal strength and wireless connection status of the rogue base station and compare it with the legitimate base station.
- If the rogue base station is showing higher signal strength than the legitimate one, then it indicates that the rogue base station is stronger than the legitimate one.

**Analysis**:

- Analyze the result of the comparison and determine whether a cellular rogue base station attack is taking place in the environment or not.
- Use the information gathered from the testing to focus the countermeasures for mitigating the attack.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Cellular Rogue Base Station Attack | White-box | Dynamic | Manual testing | N/A | Android/iOS |
| Cellular Rogue Base Station Attack | Grey-box | Static | Automated testing | Veracode | Android/iOS |
| Cellular Rogue Base Station Attack | Black-box | Hybrid | Penetration testing | Metasploit | Android/iOS |

## Testing the GPS Spoofing Attack

Testing GPS spoofing can be done in several ways:

Open Path Verification: This involves verifying the reported path by comparing it to the landscape or terrain around it. This method can be used to identify any inconsistencies or anomalies in the path reported by the GPS device.

Dual Path Verification: This involves using two different methods to verify the accuracy of the path reported by the GPS device. The two methods are usually triangulation and celestial navigation.

Data Acquisition and Evaluation: This method involves collecting and analyzing data from the GPS device in order to identify any discrepancies or errors in its reported path. This is particularly useful when multiple GPS devices are being used in order to verify their reported paths.

Monitoring: This technique involves continuously monitoring the reported path of the GPS device in order to detect any suspicious behavior or inconsistencies. This method is often used when attempting to detect a spoofed signal.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| GPS Spoofing Attack | White-box | Dynamic | Automation | Appium | Android & iOS |
| GPS Spoofing Attack | Grey-box | Static | Manual | GPSLogger | Android & iOS |
| GPS Spoofing Attack | Black-box | Hybrid | Automation | MonkeyTalk | Android & iOS |

## Testing the Botnet Attack

## Testing a Botnet Attack

Botnets are malicious networks of computers infected with malware. They are typically used to generate unwanted traffic on websites, servers, networks, and services. In this article, we will discuss how to test a botnet attack.

### Prerequisites

- Understanding of Botnet attacks
- Knowledge of common botnet testing tools

### Preparation

1. Install and configure any necessary software, such as emulators and virtual machines.
2. Prepare a secure environment for testing the botnet attack. This includes establishing security measures for the testing environment and isolating it from the production environment.
3. Configure the necessary hardware and software.

### Running the Test

1. Establish a connection between the test environment and an external source system.
2. Introduce the botnet code into the test environment.
3. Activate and monitor the botnet code, tracking how it propagates and gathers information.
4. Test the botnet's resilience against security measures.
5. Collect data on botnet activity and performance and analyze it.

### Conclusion

Testing a botnet attack is essential in order to assess the effectiveness of security measures and ensure that all systems are adequately protected. With the right preparation and the right tools, it is possible to conduct effective tests and prevent botnet attacks.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Botnet Attack | White-box | Dynamic | Protocol fuzzing | nmap | Android |
| Botnet Attack | Black-box | Dynamic | Activity monitoring | Wireshark | iOS |
| Botnet Attack | Grey-box | Static | Source code review | CheckMarx | Android/iOS |

## Testing the Malware as a Service Attack

Testing Malware as a Service (MaaS) involves assessing the efficacy of a MaaS solution in detecting, preventing, and responding to potential malicious actors. It is essential to ensure that the MaaS provider has the resources and capabilities to provide effective security and protection while adhering to industry best practices.

The following steps should be taken in order to test the efficacy of a MaaS solution:

Define test objectives: The first step before testing is to clearly define what needs to be tested and what is to be achieved by the end of the testing process. For example, the objective could be to test and verify the MaaS solution's ability to detect, prevent, and respond to malicious activity.

Select test environment: Selecting the appropriate environment for testing is essential to ensure a successful test. The environment should be as similar to the actual production environment as possible.

Execute test scenarios: Once the environment is set up, the next step is to execute the various test scenarios which are designed to evaluate the effectiveness of the MaaS solution.

Record observations and results: During the testing process, the tester should make a record of the observations and results from the various test scenarios. This data can then be used to analyze the efficacy of the MaaS solution.

Analyze test results and draw conclusions: After the testing is complete, the testers should analyze the results in order to determine the efficacy of the MaaS solution. They should also draw conclusions on the ability of the solution to detect, prevent, and respond to various types of malicious activity.

Testing the efficacy of a MaaS solution is essential to ensure that the solution can provide the appropriate level of security and protection. Following the steps outlined above will help ensure that tests are successful and meaningful.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| API | White-Box | Dynamic | Automated | Webproxy | Android & iOS |
| Network Environment | White-Box | Static | Manual | Nessus Security Center | Android & iOS |
| Application Environment | Grey-Box | Hybrid | Automated | Burp Suite | Android & iOS |
| End-User Environment | Black-Box | Dynamic | Automated | Nmap | Android & iOS |

## Testing the Bypassing Physical Security Attack

### Testing Bypassing Physical Security

Physical security is an important part of any security system and must be tested regularly to ensure it is effective. Here are some tips for testing the physical security of your system:

Perform regular patrols around your premises to assess the effectiveness of your physical security measures.

Monitor your security cameras to ensure they are in proper working order and can adequately capture any suspicious activity.

Use physical penetration tests to evaluate the security of doors, windows, and other access points.

Regularly inspect locks, alarms and other security equipments.

Test access control systems to verify that identities can be authenticated and access is denied properly.

Cart tests can also be used to determine if it is possible to bypass interaction with security personnel.

Assess your system for weaknesses, such as unlocked doors, weak locks, unmonitored cameras and access points not adequately secured.

Regularly review your system’s response to alarm systems and other security breaches.

Use physical security audit to track changes in physical security measures and document any breaches.

By following these tips and regularly testing your physical security systems, you can ensure the safety of your premises and the data contained within.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Physical Security | White-box | Dynamic | Application Penetration Test | Metasploit, Core Impact | Android, iOS |
| Physical Security | Black-box | Static | Code Review | Retire.js, Checkmarx | Android, iOS |
| Physical Security | Grey-box | Hybrid | Hybrid Analysis | Burp Suite Proxy, Acunetix | Android, iOS |

## Testing the Physical Theft Attack

Testing physical theft is not something that can be easily tested, as it requires physical guards and surveillance of the premises. However, here are some steps that can be taken to reduce the likelihood of theft:

1. Install motion detectors at all entry and exit points of the building.
2. Ensure adequate lighting in and around the premises.

3. Implement security cameras both inside and outside the building.
4. Install tamper-proof locks on all entrance and exit points.
5. Issue swipe cards or security badges to all personnel.
6. Vet all personnel before hiring them.
7. Ensure that staff do not leave valuables unattended.
8. Carry out regular security audits.
9. Make sure that all personnel report any suspicious activities.
10. Ensure that valuables are not stored in unsecured areas.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Hardware Security | White-box | Dynamic | fuzz testing | [Oss Fuzz] (https://oss-fuzz.com/) | Android, iOS |
| Hardware Security | White-box | Static | code reviews | [Black Duck] (https://www.blackducksoftware.com/) | Android, iOS |
| Hardware Security | White-box | Hybrid | vulnerability assessment | [Acunetix] (https://www.acunetix.com/) | Android, iOS |
| Hardware Security | Grey-box | Dynamic | payload-based attack | [Burp Suite Pro] (https://portswigger.net/burp/pro) | Android, iOS |
| Hardware Security | Grey-box | Static | architecture review | [Microsoft Security Risk Detection] (https://www.microsoft.com/en-us/security-risk-detection/) | Android, iOS |
| Hardware Security | Grey-box | Hybrid | penetration testing | [OWASP Zed Attack Proxy] (https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project) | Android, iOS |
| Hardware Security | Black-box | Dynamic | fuzz testing | [Sulley Fuzzer] (https://github.com/OpenRCE/sulley) | Android, iOS |
| Hardware Security | Black-box | Static | code reviews | [CheckMarx] (https://www.checkmarx.com/) | Android, iOS |
| Hardware Security | Black-box | Hybrid | vulnerability assessment | [Netsparker] (https://www.netsparker.com/) | Android, iOS |

## Testing the VM Migration Attack

**Testing VM Migration**

To test Virtual Machine (VM) Migration, the following steps should be taken:

Prepare the source and target environment: Set up the source/target environments and make sure all necessary applications and services that the VM needs is installed and running properly.

Configure Migration: Configure the VM's settings such as network, CPU, memory, storage, etc. to be compatible with the source/target environment and verify they are the same between both environments.

Test Network Connectivity: Make sure that the source and target environments are properly connected over the network and that the necessary ports are open and accessible.

Test Migration Tool: Test the migration tool being used, if any, to make sure it is working properly.

Perform Initial Backup: Take an initial backup of the source environment to ensure the VM is properly backed up and can be restored if the migration fails.

Begin Migration: Begin migrating the VM from the source to the target environment.

Monitor Progress: Monitor the migration process and make sure no errors or warnings occur during the migration.

Validate Migration: Once the migration is completed, validate that all services, applications, and settings are intact and working properly.

Restore Backup: Restore the initial backup as a contingency plan against any unforeseen errors or failures.

Clean Up Source: Clean up the source environment, including removing any unnecessary files, services, or applications that are no longer needed or can be moved over to the target environment.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| VM Migration | White-box | Dynamic | Manual Testing | N/A | N/A |
| VM Migration | Grey-box | Dynamic | Exploratory Testing | N/A | N/A |

| | | | | | |
|---|---|---|---|---|---|
| VM Migration | Black-box | Dynamic | Automated Testing | Burp Suite | N/A |
| VM Migration | White-box | Static | Code Review | Veracode | N/A |
| VM Migration | Grey-box | Static | Risk Assessment | N/A | N/A |
| VM Migration | Black-box | Static | Penetration Testing | Metasploit | N/A |
| VM Migration | White-box | Hybrid | Traceability Testing | N/A | N/A |
| VM Migration | Grey-box | Hybrid | Security Testing | HP WebInspect | N/A |
| VM Migration | Black-box | Hybrid | Application Security Testing | Acunetix | iOS/Android |

## Testing Rowhammer Attack

Rowhammer is a security vulnerability due to which a malicious user can gain access to memory by repeatedly accessing certain memory locations (rows of memory cells).

The best way to test for Rowhammer is to use a specialized tool known as RAMBleed. RAMBleed is an open source tool developed by Google researchers which can detect Rowhammer attack traces in the RAM. The following steps can be followed to use RAMBleed to test for Rowhammer:

1. Install RAMBleed on the system.
2. Start the RAMBleed server.
3. Configure the RAMBleed client on the system to be tested.
4. Execute the RAMBleed client and wait to see if any attack traces are detected.
5. Follow the instructions to interpret the output of RAMBleed.
6. If any attack traces were detected, take the necessary steps to fix the vulnerability.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Memory | White box | Dynamic | Assertion | DRAMstress | Not applicable |
| Memory | Grey box | Static | Verification | Memsmash | Not applicable |
| Memory | Black box | Hybrid | Execution | Hammertime | Not applicable |

## Testing the VM Escape Attack

There are a few approaches to testing for VM Escape (also known as Virtual Machine Escape).

**Code Review**: A comprehensive code review can help identify potential vulnerabilities present in the code which, if exploited, could lead to a VM Escape. This involves a thorough, line-by-line examination of the source code, using techniques such as manual inspection, automated static code analysis and fuzzing.

**Exploit Testing**: A series of exploitation techniques can be used to try to break out of the virtualized environment. These could include things such as exploiting buffer and account overflow vulnerabilities, command injection and malicious software attempts.

**Penetration Testing**: Penetration testing involves the use of specialized tools and techniques to break into the virtual environment and gain access to critical resources. This could involve standard methods such as brute force attacks, port scanning, and social engineering.

**External Auditing**: External auditing involves examining the operating environment from the outside, examining access controls, security protocols and other measures. This can identify any weak points that would allow for a successful VM Escape.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| VM Escape Attack | White-box | Dynamic | Fuzzing | Peach Fuzzer | N/A |
| VM Escape Attack | Grey-box | Static | Signature Detection | Codenomicon Defensics | N/A |
| VM Escape Attack | Black-box | Hybrid | Exploitation | Metasploit | N/A |