Final Security Test Specification and Tools Report

Mobile Platform	Hybrid Application ; IoT System
Application domain type	Smart Air Quality
Authentication	Yes
Authentication schemes	Biometric-based authentication ; Channel-based authentication ; ID-based authentication ; Channel-based authentication
Has DB	Yes
Type of database	SQL (Relational Database)
Which DB	MySQL
Type of information handled	Personal Information ; Confidential Data
Storage Location	Remote Storage (Cloud Database)
User Registration	Yes
Type of Registration	The users will register themselves
Programming Languages	Dart
Input Forms	Yes
Upload Files	No
The system has logs	Yes
The system has regular updates	Yes
The system has third-party	Yes
System Cloud Environments	Public Cloud
Hardware Specification	Yes
HW Authentication	Basic Authentication (user/pass)
HW Wireless Tech	3G ; 4G/LTE ; 5G ; Bluetooth ; Wi-Fi ; Bluetooth ; GPS ; LoRa
Device or Data Center Physical Access	Yes

Cellular Jamming Attacks Testing

Cellular jamming disrupts wireless communication by interfering with radio signals. Hereâ€™s what you need to know:

Jamming Techniques

1. **Wideband Jamming**: Covers a broad frequency range.
2. **Narrowband Jamming**: Targets specific frequencies.
3. **Pulsed Jamming**: Intermittently disrupts signals.
4. **Continuous Jamming**: Sustained interference.

Testing Cellular Jamming

1. **Laboratory Testing**: Use controlled environments to assess jamming effects.
2. **Field Testing**: Evaluate real-world scenarios.
3. **Tools**: Custom-built jammers or software-defined radios (SDRs).

Mitigation Strategies

1. **Frequency Hopping**: Cellular systems that change frequencies dynamically.
2. **Spread Spectrum Techniques**: Distribute signal energy across a wide bandwidth.
3. **Authentication and Encryption**: Secure communication channels.
4. **Jammer Detection**: Monitor for jamming signals.

Cellular Jamming Testing Tools

Attack Type	Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
Cellular Jamming	Wireless communication systems	White-box, Grey-box, Black-box	Dynamic, Static	Laboratory Testing, Field Testing	Custom-built jammers, Software-defined radios (SDRs)	Mobile devices with wireless capabilities

Remember that testing DoS or jamming attacks should be conducted ethically and with proper authorization.

Reference

1. Kerrakchou, I., Chadli, S., Kharbach, A., Saber, M. (2021). Simulation and Analysis of Jamming Attack in IoT Networks. In: Motahhir, S., Bossoufi, B. (eds) Digital Technologies and Applications. ICDTA 2021. Lecture Notes in Networks and Systems, vol 211. Springer, Cham. https://doi.org/10.1007/978-3-030-73882-2_3;
2. MITRE ATT&CKÂ® Technique T1464: Network Denial of Service.

Testing the Wi-Fi Jamming Attack

1. Set up a Wi-Fi network (or multiple Wi-Fi networks) consisting of a variety of devices.
2. Create a packet capture device and capture the Wi-Fi network traffic.
3. Place the packet capture device in a central location.
4. Set up a jamming device near the Wi-Fi network(s) and activate it.
5. Monitor the packet capture device for any changes in the Wi-Fi network traffic.
6. Analyze the results and evaluate if the jamming device is successfully disrupting the Wi-Fi network(s).
7. Determine the effectiveness of the jamming device and take countermeasures to reduce or eliminate the jamming effect.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
Wi-Fi Jamming	White-box	Dynamic	Security Audit	Nessus	iOS, Android
	Grey-box	Static	Code Review	SonarQube	
	Black-box	Hybrid	Exploit	MetaSploit	
			Vulnerability	Acunetix	
			Stress Testing	LoadRunner, Jmeter	

Testing the Orbital Jamming Attack

Testing an orbital jamming attack involves multiple steps.

First, identify the target satellite or spacecraft. The types of systems that could be jammed vary depending on the mission, but generally include communication links, navigation systems, and sensor systems.

Once the target is identified, the next step is to simulate the attack using a radio frequency simulator. This will allow the tester to test the strength of the jamming signal to ensure that it is strong enough to interfere with the target's systems without causing permanent damage.

After the attack is simulated, the tester should conduct a real-time jamming test. This can be done by sending out a strong jamming signal at the target's frequency and monitoring its effects on the target systems.

Once the effects of the jamming signal on the target systems have been observed, the tester should analyse the results and document any system failures.

Finally, the tester should collect and analyse the data from the test to ensure that the jamming signal was effective and that no permanent damage was caused to the target systems.

Overall, these steps ensure that an orbital jamming attack can be properly tested before it is launched.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
Orbital Jamming Attack	White-box	Dynamic	Penetration Testing	Burp Suite	iOS, Android
Orbital Jamming Attack	Grey-box	Static	Code Review	SonarQube	iOS, Android
Orbital Jamming Attack	Black-box	Hybrid	Exploratory Testing	Maltego	iOS, Android

Testing the GPS Jamming Attack

1. Monitor the GPS devices for any abnormal behavior or erratic messages for an extended period.
2. Use a GPS signal jamming device to test the efficacy of the GPS antenna.
3. Use specialized software to check the GPS receiver for any errors.
4. Check if electromagnetic interference in the area is causing disruption in the GPS frequency.
5. Shut down the GPS and connect it with a different satellite receiver, in order to check if the device is still receiving data from other satellites.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
GPS Jamming Attack	White-box	Dynamic	Manual	N/A	iOS
GPS Jamming Attack	Grey-box	Static	Automated	Burp Suite	Android
GPS Jamming Attack	Black-box	Hybrid	Mixed	nmap	Windows Mobile

Testing the Bluesnarfing Attack

To test a bluesnarfing attack, the following steps should be taken:

Ensure that there are Bluetooth-enabled devices in the vicinity that can be targeted.

Use a Bluetooth sniffer to scan for and identify Bluetooth signals from the target device.

Use a Bluetooth attack tool, such as BlueSnarf, to connect to the target device.

Extract data from the target device, such as phone book, contacts, messages, calendars, and more.

Document the success or failure of the attack.

Analyze the results and advise the user on any security risks associated with using Bluetooth on their device.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
Bluetooth	White-box	Dynamic	Vulnerability Scanning	Nessus	Android
Bluetooth	Grey-box	Static	Source Code Analysis	Veracode	iOS
Bluetooth	Black-box	Hybrid	Penetration Testing	Metasploit	Android, iOS

Testing the Bluejacking Attack

Testing a Bluejacking attack consists of the following steps:

Identify potential targets in the area: Look for nearby Bluetooth devices that are turned on and discoverable.

Connect to the target device: Establish a Bluetooth connection with the targeted device.

Send the message: Send a short message or link to the target device using the deviceâ€™s Bluetooth sharing protocol.

Monitor the response: Observe if the target device responds to the message.

Analyze the response: Analyze the response from the target device to determine if the attack was successful.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
Bluejacking Attack	White-box	Dynamic	Unit Testing	Appium	iOS
Bluejacking Attack	Grey-box	Static	Risk Analysis	Jenkin	Android
Bluejacking Attack	Black-box	Hybrid	Security Testing	Wireshark	Windows
Bluejacking Attack			Performance Testing	Selenium	iOS

Testing the Wi-Fi Jamming Attack

Establish your test environment: - Create secure wireless network with a unique SSID. - Setup network tracking or logging capabilities to collect and analyze information. - Set different levels of access for different users and/or roles.

Deploy your wireless network and begin tracking traffic: - Provide access to all authorized users and install appropriate security protocols to protect the network from unauthorized access. - Monitor the network and log all wireless traffic, noting the SSIDs of all access points seen by the network.

Use an attacker tool to test your security and detect potential SSID Tracking attacks: - Utilize an attack tool like Aircrack-ng to simulate an attacker attempting to connect to the wireless network. - Use the attack tool to flood the network with SSID requests, and analyze the logs to see if any of them contain the unique SSID of the network.

Analyze results and adjust security accordingly: - If the SSID appears in the logs, the attack was successful and your security isn't sufficient to prevent tracking. - Adjust network security measures to ensure that unauthorized users cannot access the network and its resources.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
Wi-Fi SSID Tracking	White-box	Dynamic	Boundary Analysis	Qualys Network Inspector	iOS, Android, Windows
Wi-Fi SSID Tracking	Grey-box	Static	Source Code Analysis	Veracode	Android, Windows
Wi-Fi SSID Tracking	Black-box	Hybrid	Penetration Testing	Burp Suite	iOS, Android, Windows

Testing the Byzantin Attack

The purpose of testing for a Byzantine attack is to identify any malicious behavior within a system and to prevent the attack from taking place. There are a few different methods that can be used to test for Byzantine attacks. These include:

Network-Layer Analysis

One way to detect a Byzantine attack is through network-layer analysis. This involves examining the network traffic on a system to find any suspicious activity. This could include looking for abnormal traffic patterns or unexpected communication between nodes.

Cryptographic Analysis

Another way to detect a Byzantine attack is through cryptographic analysis. This involves examining the encryption methods used to protect data and ensuring that they are resistant to tampering and manipulation. It can also help identify any weaknesses or vulnerabilities in the system.

Security Audits

Security audits are another way to detect a Byzantine attack. This involves inspecting the system's security policies, processes, and tools to make sure that they are up to date and provide enough protection against malicious actors.

Logging and Monitoring

Logging and monitoring is another key tool for detecting a Byzantine attack. This involves collecting log data from the system and storing it in a secure repository. This allows for detailed analysis of activity on the system, which can help identify any potential security issues and malicious actors.

Simulation

Simulation is another method that can be used to test for Byzantine attacks. This involves running simulations of various scenarios and scenarios involving malicious actors to identify any weaknesses in the system. This is a useful tool for finding vulnerabilities and potential attacks before they take place.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
Functional	White-box	Dynamic	Component Testing	JUnit	Android
System	Black-box	Static	Integration Testing	UML	iOS
Security	Gray-box	Hybrid	Security Testing	Fuzzing Tool	Windows Phone
Performance	White-box	Dynamic	Regression Testing	Apache jMeter	Cross Platform

Testing the Malicious Insider Attack

Testing Malicious Insider Attacks

Monitor user behavior: Organizations should monitor user behavior for unusual activity and behavior, such as sudden spikes in data transfer or download activity or an increase in requests for data that would be outside of the userâ€™s normal job roles.

Physical security: Organizations should ensure that physical access to systems is limited to authorized personnel and that access controls are regularly reviewed and updated.

Conduct network access reviews: Regularly reviewing user access to resources and data can uncover potential malicious insiders.

Educate users on security: End users should be educated on security policies and procedures to ensure they understand the risks associated with malicious insider activity and understand how to protect themselves and the organization.

Network segmentation: Segmenting networks into different access tiers can limit the reach of malicious insiders.

Implement data encryption: Access to data should be encrypted to reduce the potential damage of a malicious insider attack.

Monitor access logs: Organizations should monitor user access logs to detect any unauthorized access to sensitive data or resources.

Use two-factor authentication: Organizations should implement two-factor authentication for accessing sensitive systems and data.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
Malicious Insider Attacks	White-Box	Dynamic	Fuzzing	Sulley	iOS NeuroMobi
	Grey-Box	Static	Penetration	Nessus	Android DroidRox
	Black-Box	Hybrid	Risk Assessment	Burp Suite	Windows Pranker

Testing the Sniffing Attack

To detect sniffing attacks, the following steps should be followed:

Monitor Network Activity: Monitor your network for unusually high levels of traffic, and compare it to what is normal. High amounts of traffic can indicate malicious activity.

Perform Packet Capture: Use packet capture techniques such as port mirroring or port spanning to monitor all the packets that travel between two locations or over a network. This will allow you to analyze the data in detail and detect any malicious activity.

Track Source IP Addresses: Track the source IP addresses of incoming packets to determine any suspicious activity. Malicious IPs can be blocked and monitored later.

Compare Protocols: Compare the protocols used in the captured network traffic. If any unusual or unfamiliar protocols are used, the traffic should be investigated further.

Utilize Intrusion Detection Systems (IDS): Utilize Intrusion Detection Systems (IDS) to detect any anomalies in the network traffic. IDS systems analyze packets in real-time and look for any suspicious activity.

Use Network Scanning Tools: Utilize tools such as Nmap to identify open ports, services and vulnerabilities that need to be patched.

Use Antivirus Software: Use antivirus software to detect and prevent malicious activity. Antivirus software should be updated regularly for maximum protection.

Implement Encryption: Encrypt data before sending it over a network. This will prevent malicious actors from decrypting and accessing confidential data.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
Network	White-box	Dynamic	Network Sniffer	Wireshark/Ethereal	None
Network	Grey-box	Dynamic	Network & Host	Nmap	None
Network	Grey-box	Dynamic	Protocol Tests	Ncat	None
Host	White-box	Static	File Scanning	NESSUS	None
Host	Grey-box	Hybrid	Application	Burp Suite	iOS/Android
Application	Black-box	Dynamic	Code Analysis	FindBugs	iOS/Android

Testing the Man-in-the-Middle Attack

Testing Man-in-the-Middle Attack

Set up a virtual network using a virtual machine or other virtual environment.

Place a malicious node between two unsuspecting hosts within the same network.

Configure the malicious node to intercept and redirect all traffic it receives from the unsuspecting hosts.

Verify that the malicious node is effectively intercepting the data, by attempting to ping or connect to one of the unsuspecting hosts.

Attempt to gain access to data that is flowing through the malicious node.

Monitor the node for malicious activity.

Analyze the data logs to identify any suspicious activity.

Remove the malicious node from the environment.

Change any credentials, passwords, or other information that was intercepted.

Monitor the environment for any further malicious activity.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
MITM Attack	White-box	Dynamic	Dynamic Analysis	Mitmproxy, Wireshark	Android, iOS
	Grey-box	Static	Penetration Tests	Paros, Burp Suite	
	Black-box	Hybrid	Misconfiguration	Nmap, Scapy	

Testing the Eavesdropping Attack

Testing Eavesdropping attacks typically involve the following steps:

Set up the environment: - Choose a testing tool (ie Wireshark, Cain & Abel, etc) - Configure the network

Launch the attack: - Use the chosen tool to monitor the traffic on the network - Search for unencrypted data in transit

Analyze the results: - Investigate any suspicious packets to identify any confidential information - Review the logs to identify any unauthorized access attempts

Document the results: - Document any discovered confidential data and unauthorized access attempts - Present the analysis findings in a clear, organized format (Markdown is a great option)

Prevent further attacks: - Leverage the findings to identify any security vulnerabilities in the network - Implement appropriate security measures to protect against future eavesdropping attempts

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
Eavesdropping	White-box	Dynamic	Unit Testing	JUnit	iOS/Android
	Grey-box	Static	Penetration	Metasploit	
	Black-box	Hybrid	Security	Nmap	

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 isnâ€™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 didnâ€™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 Access Point Hijacking Attack

Reconnaissance: Utilize network reconnaissance techniques to identify wireless access points within range. These can include passive approaches such as wireless network scanning with a tool like Kismet, or active approaches such as using a tool like Aircrack-ng.

Enumeration: Connect to a legitimate access point on the network and run a tool like NetStumbler to enumerate the target.

Exploitation: Attempt to perform an access point hijacking attack by using a tool like AirJack. AirJack will capture valid authentication packets and can be used to take control of the target access point.

Verification: Verify the success of the attack by ensuring that the access point is controlled by the attacking machine. This can be done by pinging the IP address of the access point or using a tool like MDK3 to verify that the access point is now under the control of the attacker.

Mitigation: Implement security measures to prevent and detect access point hijacking attacks. These can include monitoring network traffic for suspicious activity, disabling SSID broadcast, enabling WPA2 encryption, implementing MAC address filtering and implementing a whitelisting protocol.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
Access Point Hijacking Attack	White-box	Dynamic	Packet Sniffing	Wireshark	Android / iOS
Access Point Hijacking Attack	Gray-box	Static	Code Review	static code analysis tool	Android / iOS
Access Point Hijacking Attack	Black-box	Hybrid	Penetration Testing	Burp Suite	Android / iOS

Testing the Cellular Rogue Base Station Attack

Install the necessary equipment:

- A cellular network access point (e.g., a mobile modem, a femtocell, or a base station simulator)
- An attack station (e.g., a laptop or a Raspberry Pi with a cellular modem)
- Software to generate and monitor rogue base station (e.g., KARMA)

2. **Test the equipment** by running standard tests to ensure that everything is working correctly.
3. **Enable KARMA** and configure the system settings to simulate a rogue base station.
4. **Run a scan** of the local environment to identify any other base stations that may be present and respond to rogue transmissions.
5. **Transmit Rogue Base Station Signals** over the local environment to detect any client devices that may be present.
6. **Monitor the response** of any detected devices to confirm that they are connecting to the rogue base station.
7. **Analyze the data** collected from the scan and the response of the devices to confirm whether or not the attack was successful.
8. **Document results** of the test and any other data collected to provide a comprehensive record of the attack.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
System	White-box	Dynamic	Penetration Testing	Metasploit	iOS / Android
Network	Grey-box	Static	Code Review	SonarQube	iOS / Android
Application	Black-box	Hybrid	Manual Testing	Selenium	iOS

Testing the GPS Spoofing Attack

Testing GPS spoofing involves running tests to ensure that the GPS receiver is correctly detecting the proliferation of fake or inaccurate GPS signals. Here are some steps to test GPS spoofing:

Create sample spoofed GPS signals: Use a simulator to generate GPS signals that contain incorrect location and timing data.

Feed sample GPS signals into the GPS Receiver: Connect the GPS receiver to the simulator and begin supplying it with the spoofed signals.

Analyze the data output: Monitor the output from the GPS receiver to ensure that it picks up the flaws in the spoofed signals.

Test the accuracy of the spoofed signals: Test the accuracy of the spoofed signals by comparing their location and timing data to known values.

Compare to a standard set of values: Compare the output of the GPS receiver with a standard set of values that have been obtained from a true GPS signal.

Look for discrepancies: Look for discrepancies in the output of the GPS receiver when compared to the standard set of values. These discrepancies will indicate whether or not the GPS receiver is correctly detecting the spoofed signals.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
GPS Spoofing Attack	White-box	Dynamic	Network	Nmap	Android
		Static	Code	SonarQube	iOS
	Grey-box	Hybrid	Device	OWASP ZAP	
				Burp Suite	
	Black-box			Appium	

Testing the Node Tampering Attack

Testing for node tampering can vary depending on the specific environment and the security measures that have been implemented. Unfortunately, there is no one-size-fits-all method for testing this type of attack. However, some of the steps that should be taken include:

Verify that integrity checking is enabled for files transferred or stored by the node. This includes the use of checksums, signature checking, or data verification.

Verify that the node has a robust access control policy in place and that it is continuously monitored and updated when necessary.

Monitor system logs for suspicious activities such as unexpected node communications, node access, and attempts to modify node-installed files.

Implement periodic scans for malicious software and malware.

Verify that the node is configured to run only approved, signed software.

Ensure that users are granted least access privileges necessary to perform their job duties.

Perform periodic vulnerability scans of the node in order to identify exploitable weaknesses.

Educate users on best security practices, such as setting strong passwords and avoiding untrusted sites or downloads.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
Node Tampering	White-box	Dynamic	Attacker-in-the-middle	OWASP ZAP	Android, iOS

Testing the Botnet Attack

Testing a Botnet attack can be done using a variety of different techniques and methods.

Honeypots: Honeypots are systems set up to passively monitor the network and can provide valuable information about the type of attack and its origin.

Network monitoring: A network monitoring tool such as a sniffer can be used to inspect traffic in order to assess whether a botnet attack is taking place.

Intrusion detection system (IDS): An IDS can be used to detect suspicious network traffic and alert the security team to a potential botnet attack.

Behavioral analysis: Analyzing the behavior of the botnet can help identify its purpose and intent, and mitigate the risks associated with it.

Network forensics: Network forensics can help identify the sources of a botnet attack, as well as the malicious activities occurring on the network.

Web application vulnerability tests: Application vulnerability tests can identify weaknesses and potential entry points into the network that can be targeted by botnets.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
Botnet Attack	White-box	Dynamic	Penetration	Nessus	Android
	Grey-box	Static	Fuzzing	SqlMap	iOS
	Black-box	Hybrid	Exploitation	DroidBox	
			Diagnostics	nmap	

Testing the Malware-as-a-Service Attack

Testing a Malware-as-a-Service attack is a multi-step process:

Prepare test environment: Firstly, create an isolated test environment, separate and independent from a live environment. This will help ensure that the malicious files and services do not affect users in the live environment.

Configure a honeypot: Next, set up a honeypot to capture and analyze the incoming malicious traffic. A honeypot is a decoy system designed to imitate a production environment and identify malicious activity.

Execute Malware-as-a-Service attack: After setting up the honeypot, execute the Malware-as-a-Service attack to assess its effectiveness. You can use a virtual machine or run the attack in a sandbox environment.

Monitor and analyze results: Lastly, monitor the honeypot for incoming malicious traffic and analyze the results. This should help you understand the attack profile and assess its effectiveness. Additionally, you can use security tools such as anti-virus and intrusion prevention systems to detect malicious activity.

By following these steps, you can efficiently test a Malware-as-a-Service attack.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
Network	White-box	Dynamic	Traffic Simulation	Wireshark/Snort	Not Applicable
Application	Grey-box	Static	Code Analysis	Burp Suite/Nmap	App Scanner
System	Black-box	Hybrid	Exploitation	Metasploit/OWASP ZAP	XCode & Android Studio

Testing the Bypassing Physical Security Attack

Testing Physical Security Bypass Techniques

Physical security bypass is a type of attack where a malicious user attempts to access assets, data, or resources by circumventing physical access controls. Bypassing physical security measures can be done in several ways, and it is important to test for these attacks in order to protect your organization. Here are a few key techniques for testing physical security bypasses:

Perform a security walkthrough of the physical premises: This includes inspecting the external and internal perimeter for any potential weaknesses or exposures. Look for any open windows, inadequate locks, unlocked or malfunctioning doors, and other security lapses.

Test for duplicate keys or key overrides: This includes testing if keys are kept in secure locations, if duplicate keys are being issued, and if any employees have illegally duplicated their keys.

Check for any unauthorized devices in the area: This includes testing for cameras, microphones, recording devices, and other electronic surveillance equipment that may have been planted inside of the building.

Ensure that all exterior doors and windows are locked: Check to make sure all exterior doors and windows cannot be easily picked or bypassed.

Test for wireless network vulnerabilities: Wireless networks can be easily used to bypass physical security measures, so test for any wireless security weaknesses that may be present.

By testing for these vulnerabilities, you can ensure that your organization is not vulnerable to physical security bypass attacks.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
Processes	White-box	Dynamic	Fuzz Testing	Spike	Android
Hardware	Grey-box	Static	Penetration Testing	Metasploit	iOS
Locks	Black-box	Hybrid	Statical Analysis	AppDiffer	Windows
Perimeters			Code Review	Codacy	

Testing the Physical Theft Attack

Testing Physical Theft

1. Ensure that all physical assets of the organization are properly protected. - Invest in alarms, CCTV, or other security devices to protect assets in the office. - Create an inventory of all physical assets and store it in a secure location. - Identify areas of risk and take steps to minimize them.
2. Train staff on proper security procedures. - Regularly remind staff about security policies and procedures. - Ensure that all personnel are aware of the signs of physical theft and have the resources to respond if necessary. - Provide training on how to protect physical assets from theft.
3. Investigate any reports of physical theft. - Take any reports of physical theft seriously and investigate them thoroughly. - Follow up on any leads or suspicious activity. - Interview staff and collect any relevant evidence.
4. Monitor access to physical assets. - Keep track of who has access to physical assets and who is entering and exiting the premises. - Limit access to physical assets to only those personnel who need it.
5. Monitor security tools and measures. - Test alarms and CCTV systems regularly to ensure they are working properly. - Invest in additional measures where possible to further protect physical assets.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Plataform
Physical Theft	White-box	Static	Source Code Review	PMD/Checkstyle	iOS/Android
Physical Theft	Grey-box	Dynamic	Regression Testing/Exploratory Testing	Selenium/Appium	iOS/Android
Physical Theft	Black-box	Hybrid	Performance Testing	Apache JMeter	iOS/Android

Testing the VM Migration Attack

Testing VM Migration

VM Migration is a process of migrating virtual machines from one physical host to another. The process is usually done either manually or through automated tools. It is important to test the migration procedure before putting it into production to be sure that it is working correctly.

In order to properly test VM Migration, the following steps should be followed:

Prepare a test environment with two physical hosts that are connected to a local network.

Create a virtual machine on one of the physical hosts.

Configure the virtual machine to be migrated with the necessary information, such as network address, data storage, user access, etc.

Perform a test migration of the virtual machine from one physical host to the other.

Monitor the migration process to make sure that all operations are successfully completed.

Once the migration process has completed, verify that the virtual machine is working in the new environment, including checking all the configurations and data.

Finally, test the functionality of the virtual machine in the new environment to ensure that all applications and services work as expected.

By following these steps, organizations can ensure that the migration process works correctly and that any issues are addressed promptly.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
Server	White-box	Dynamic	Exploratory	Nessus	N/A
Server	White-box	Static	Code Review	Fortify	N/A
Server	Grey-box	Static	Comparing Security Policies	nmap	N/A
Client	Black-box	Dynamic	Vulnerability Scanning	Burpsuite	iOS/Android

Testing the Side-Channel Attack

First, you should define the types of side-channels you would like to test. Examples of side-channels might include power, electromagnetic, timing, acoustic, and leakage.

Then, you should decide which data gathering tools you will use to record the information associated with each side-channel. Depending on your environment, these tools can vary from devices such as oscilloscopes to software programs such as spectral analyzers or logic analyzers.

Once you have determined the tools needed, you should set up the environment in which your tests will occur. Make sure to carefully plan the physical location of each component, such as the device being tested and the monitoring equipment, to ensure accurate measurements.

Once the environment is set, you should begin recording data. Output from the side-channel should be captured in an organized manner, such as separating the data into multiple files or creating a log.

Lastly, the data should be analyzed to identify any potential issues. This can be done by using various analysis techniques, such as manually examining the data or using statistical algorithms. This analysis should then be reported in a format that is easy to interpret, such as tables or graphs.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Plataform
White-Box	Dynamic	System Call	Penetration Testing	Kali Nmap	Android iOS
Grey-Box	Static	Dynamic Trace	Regression Testing	HPFortify Metasploit	
Black-Box	Hybrid	Security Scan	Fuzz-testing	Coreaudit	

Testing the Spectre Attack

Determine if your system is vulnerable - The first step in testing Spectre is to determine whether your system is vulnerable. You can use the Spectre Variant 1 Detector utility to check for potential vulnerabilities.

Test for Vulnerability - Once you have established that your system could be vulnerable, you can test for specific vulnerabilities using vulnerability scanners like the National Vulnerability Database (NVD).

Check for Updates - In addition to testing for vulnerabilities, it is important to make sure that your system has the latest patches and security updates to protect against Spectre. You can use the Windows Update or Mac OS Update to check for any relevant patches.

Check for Processors or Firmware that Need an Update - Spectre can also affect your system's processor and firmware. It is important to ensure that these are up-to-date to avoid potential problems. Check with your system's manufacturer for any relevant updates or patches.

Install Firewalls or Update Security Settings - Firewalls and other security software can also help protect against Spectre. Make sure to install or update any relevant programs.

Use the Instruments to Monitor for Suspicious Behavior - Lastly, you can use various instruments to monitor your system for suspicious activity or processes. This could include monitoring the system for unrecognized processes, suspicious network traffic, memory usage and more.

Testing Tools:

Target Testing	Testing Technique	Test Analysis	Test Method	Test Tool	Mobile Platform
Spectre Attack	White-box	Dynamic	Fuzzing	AFL fuzzer	iOS/Android/Windows
Spectre Attack	Grey-box	Dynamic	Bounding Box	Pitbull	iOS/Android/Windows
Spectre Attack	Black-box	Hybrid	Penetration Testing	Metasploit	iOS/Android/Windows

Testing the Meltdown Attack

Preparation * Check whether you have a processor that is vulnerable to Meltdown:

- Intel
- AMD
- ARM
- Download and install the Verifiable Builds of Meltdown Checker

Test * Run the Meltdown Checker:

```
shell $ ./meltdown_checker.py
```

- If your processor is vulnerable to Meltdown attack, you'll get an output that looks like this:

```shell System check (hardware & OS version) ...................... [OK]

Checking for vulnerability to Meltdown attack .............. VULNERABLE ```

- If your processor is not vulnerable to Meltdown attack, you'll get an output that looks like this:

```shell System check (hardware & OS version) ...................... [OK]

Checking for vulnerability to Meltdown attack .............. NOT VULNERABLE ```

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Meltdown Attack | White-box | Dynamic | Penetration | Metasploit | iOS/Android |
| | Grey-box | Static | Code review | Veracode | iOS/Android |
| | Black-box | Hybrid | Fuzz Testing | InsightVM | iOS/Android |
| | | | | Burp Suite | iOS/Android |

## Testing Hardware Integrity Attack

Testing hardware integrity can be done by running a series of tests to check the validity of a hardware system.

Visual Inspection: Visually inspect the hardware system for any signs of physical damage such as corrosion, breaks and loose connectors.

Memory Test: Check the amount of RAM installed by running a memory test utility. Ensure the amount of RAM installed is sufficient to meet your system requirements.

Hard Drive Test: Run a hard drive test utility to check for bad sectors and ensure the drive is not overly fragmented.

BIOS Test: If your hardware needs a specialized driver, you should test the BIOS to make sure it is properly configured.

Disk Drive Test: Run a disk drive test utility to ensure the drive is functioning properly and is not corrupted.

Power Supply Test: Test the power supply to make sure it is correctly routed and can provide your hardware system with sufficient power.

Temperature and Noise Test: Monitor the temperature and noise levels of the system to ensure the components are not overheating or producing too much noise.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Plataform |
|---|---|---|---|---|---|
| Hardware Integrity | White-box | Dynamic | Penetration Test | Kali Linux | iOS/Android Devices |
| Hardware Integrity | Grey-box | Static | Vulnerability Scanning | Nessus | iOS/Android Devices |
| Hardware Integrity | Black-box | Hybrid | Source Code Analysis | CodeInspect | iOS/Android Devices |

## Testing Rowhammer Attack

1. Choose a system with vulnerable DRAM modules:

- It is important to have a system with vulnerable DRAM modules to test for Rowhammer.

2. Set up stressor application (e.g. memtest86+):
   - To test for Rowhammer, a stressor application is needed. A popular one, often used for this type of testing, is memtest86+.
3. Run the stressor application repeatedly for a longer period of time:
   - The stressor application should be run repeatedly for a longer period of time, usually several hours.
4. Monitor system response:
   - During the test, the system should be monitored to check for any errors or abnormalities.
5. Analyze results:
   - Once the testing period is over, the results should be analyzed for any evidence of Rowhammer attacks.

### Testing Tools:

| Target Testing | Testing Technique | Test Analysis | Test Method | Test Tool | Mobile Platform |
|---|---|---|---|---|---|
| Hardware | White-box | Dynamic | Hardware-in-the-Loop | Babblar | Android |
| Software | Grey-box | Static | Fuzz Testing | Windmill | iOS |
| Firmware | Black-box | Hybrid | Dynamic Web Testing | Syhunt | |
| Application | | | Penetration Testing | Metasploit | |

## 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 |