Mobile Jailbreaking Cheat Sheet

What is "jailbreaking", "rooting" and "unlocking"?

Jailbreaking, rooting and unlocking are the processes of gaining unauthorized access or elevated privileges on a system. The terms are different between operating system, and the differences in terminology reflect the differences in security models used by the operating systems vendors.

For iOS, **Jailbreaking** is the process of modifying iOS system kernels to allow file system read and write access. Most jailbreaking tools (and exploits) remove the limitations and security features built by the manufacturer Apple (the "jail") through the use of custom kernels, which make unauthorized modifications to the operating system. Almost all jailbreaking tools allow users to run code not approved and signed by Apple. This allows users to install additional applications, extensions and patches outside the control of Apple’s App Store.

On Android, **Rooting** is the process of gaining administrative or privileged access for the Android OS. As the Android OS is based on the Linux Kernel, rooting a device is analogous to gaining access to administrative, root user-equivalent, permissions on Linux. Unlike iOS, rooting is (usually) not required to run applications outside from the Android Market. Some carriers control this through operating system settings or device firmware. Rooting also enables the user to completely remove and replace the device's operating system.

On Windows Phone OS, **Unlocking** is the process of editing specific keys of the Windows Phone OS registry or modifying the underlying platform to allow the execution of applications that are not certified by Microsoft or that use reserved capabilities. Different levels of unlocking exist depending on the OS and device version:

- **Developer-unlock**: Microsoft allows Independent Software Vendors (ISV) to unlock their systems to sideload and test homebrew apps onto physical devices, before their submission to the Store. Developer-unlock only allows to sideload applications that are not signed by the Windows Phone Store approval process and it is often a pre-condition to achieve a higher level of unlock (e.g., interop-unlock). A developer-unlocked device does not allow an app to escape its sandbox or tweak the system via registry editing. Windows Phone devices can be officially developer-unlocked for free using utilities provided by Microsoft;
- **Interop-unlock:** with the release of Windows Phone 7.5 Mango (7.10.7720.68), Microsoft introduced a new platform security feature, called Interop Lock, which restricted the access to drivers only to apps with the Interop Services capability (**ID_CAP_INTEROPSERVICES**). Moreover, Mango denies the sideloading of unsigned apps with that capability, thus limiting drivers’ access to Windows Phone Store *certified* apps only. Heathcliff74, the mind behind the WP7 Root Tools suite, researched the topic and found that by manipulating the value of the **MaxUnsignedApp** registry key (**HKLM\Software\Microsoft\DeviceReg\Install\MaxUnsignedApp**) it is possible to control the unlocking level of a Windows Phone device. A value between 1 and 299 means that the device is developer-unlocked, while a value equal or greater than 300 removes the restriction to sideload apps with the ID_CAP_INTEROPSERVICES capability, allowing apps to access restricted file system areas and registry editing, thanks to the use of high-privileged app capabilities. It has been hypothesized that the “magic number” involved in the MaxUnsignedApp register key is a feature introduced by Microsoft for OEMs and so at times referred to as **OEM developer-unlock**. It should be noted that typically the interop-

unlock by itself does not enable all of the system's available capabilities – condition that is also knows as **Capabilities-unlock**;

- **Full-unlock**: full-unlock aims at disabling a subset or all of the security mechanisms implemented by the OS to allow full access and the customization of the system (e.g., file system and registry unlimited access). Full-unlocking is usually achieved with custom ROMs flashing, where the OS binaries are patched to disable the OS security features, such as policy-checks. In a full-unlocked environment, apps are likely to be able to escape their sandbox because they can be run with elevated privileges.

Why do they occur?

iOS: many users are lured into jailbreaking to take advantage of apps made available through third party app sources, such as Cydia, which are otherwise banned or not approved by Apple. There is an inherent risk in installing such applications as they are not quality controlled nor have they gone through the Apple approval and application approval process. Hence, they may contain vulnerable or malicious code that could allow the device to be compromised. Alternately, jailbreaking can allow users to enhance some built in functions on their device. For example, a jailbroken phone can be used with a different carrier than the one it was configured with, FaceTime can be used over a 3G connection, or the phone can be unlocked to be used internationally. More technically savvy users also perform jailbreaking to enable user interface customizations, preferences and features not available through the normal software interface. Typically, these functionalities are achieved by patching specific binaries in the operating system. A debated purpose for jailbreaking in the iOS community is for installing pirated iOS applications. Jailbreaking proponents discourage this use, such as Cydia warning users of pirated software when they add a pirated software repository. However, repositories such as Hackulous promote pirated applications and the tools to pirate and distribute applications.

Android: rooting Android devices allows users to gain access to additional hardware rights, backup utilities and direct hardware access. Additionally, rooting allows users to remove the pre-installed "bloatware", additional features that many carriers or manufacturers put onto devices, which can use considerable amounts of disk space and memory. Most users root their device to leverage a custom Read Only Memory (ROM) developed by the Android Community, which brings distinctive capabilities that are not available through the official ROMs installed by the carriers. Custom ROMs also provide users an option to 'upgrade' the operating system and optimize the phone experience by giving users access to features, such as tethering, that are normally blocked or limited by carriers.

Windows Phone OS: Windows Phone users generally unlock their devices to tweak their systems and to be able to sideload homebrew apps. Depending on the level of unlocking, the OS can be customized in term of store OEM settings, native code execution, themes, ringtones or the ability to sideload apps that are not signed or that use capabilities normally reserved to Microsoft or OEMs. Developers unlock their devices to test their products on real systems, before the submission to the Store. An interop-unlocked device allows users to access file system areas where Store apps are installed, thus allowing DLL extraction, reverse engineering and app cracking.

What are the common tools used?

iOS: jailbreaking software can be categorized into two main groups:

1. **Tethered**: requires the device to be connected to a system to bypass the iBoot signature check for iOS devices. The iOS device needs to be connected or tethered to a computer system every time it has to reboot in order to access the jailbreak application, such as redsn0w, and boot correctly;
2. **Un-tethered**: requires connection for the initial jailbreak process and then all the software, such as sn0wbreeze, is on the device for future un-tethered reboots, without losing the jailbreak or the functionality of the phone.

Some common, but not all of the iOS jailbreaking tools are listed below:

- Absinthe
- blackra1n
- Corona
- greenpois0n
- JailbreakMe
- limera1n
- PwnageTool
- redsn0w
- evasi0n
- sn0wbreeze
- Spirit
- Pangu

A more comprehensive list of jailbreaking tools for iOS, exploits and kernel patches can be found on the iPhoneWiki website.

Android: there are various rooting software available for Android. Tools and processes vary depending on the user's device. The process is usually to:

1. Unlock the boot loader;
2. Install a rooting application and / or flash a custom ROM through the recovery mode.

Not all of the above tasks are necessary and different toolkits are available for device specific rooting process. Custom ROMs are based on the hardware being used; examples of some are as follows:

- **CyanogenMod** ROMs are one of the most popular aftermarket replacement firmware in the Android world. More comprehensive device specific firmwares, flashing guides, rooting tools and patch details can be referenced from the homepage;
- **ClockWorkMod** is a custom recovery option for Android phones and tablets that allows you to perform several advanced recovery, restoration, installation and maintenance operations etc. Please refer to XDA-developers for more details.

Windows Phone OS: several tools and techniques exist to unlock Windows Phone devices, depending on the OS's version, the specific device vendor and the desired unlocking level:

- **Microsoft Official Developer Unlock:** the Windows Phone SDK includes the "Windows Phone Developer Registration" utility that is used to freely developer-unlock any Windows Phone OS device. In the past, free developer unlocking was limited to recognized students from the DreamSpark program;

- **The ChevronWP7 Unlocker and Tokens**: in the early days of Windows Phone hacking, ChevronWP7 Labs released an unlocker utility (ChevronWP7.exe) that was used to unofficially developer-unlock Windows Phone 7 devices. The unlocker changed the local PC hosts file in order to reroute all the "developerservices.windowsphone.com" traffic to a local web server served with the HTTPS protocol. A crafted digital certificate (ChevronWP7.cer) was also required to be imported on the target Windows Phone device: the so configured environment allowed the unlocker to perform a Man-in-The-Middle (MiTM) attack against the USB attached device, simulating of a legitimate uncloking process. Basically, the utility exploited a certificate validation issue that affected the early version of Windows Phone platform. Lately, ChevronWP7 Labs established a collaboration with Microsoft, allowing users to officially developer-unlock their devices by acquiring special low-price unlocking tokens;
- **Heathcliff74's Interop-unlock Exploit**: Heathcliff74 from XDA-developers developed a method to load and run custom provisioning XML files (provxml) to interop-unlocked Windows Phone 7 devices. The idea behind the method was to craft a XAP file (which is a simple compressed archive) containing a directory named "**../../../../provxml**", and then extract the content of the folder (a custom provxml file) within the \provxml\\ system folder: abusing vulnerable OEM apps (e.g., Samsung Diagnosis app) the provxml file could then have been run, thus allowing changing registry settings (e.g., the MaxUnsingedApp key) and achieving the desired unlock. The method requires the target device to be developer-unlocked in order to sideload the unsigned XAP-exploit;
- **The WindowsBreak Project**: Jonathan Warner (Jaxbot) from windowsphonehacker.com developed a method to achieve both the developer and the interop unlock, while using the technique ideated by Heathcliff74, but without the need to sideload any unsigned apps. The exploit consisted of a ZIP file containing a custom provxml file within a folder named "**../../../../provxml**": the extraction of the custom provxml file in the \provxml\\ system folder was possible thanks to the use of the ZipView application. The original online exploit is no longer available because the vulnerability exploited by WindowsBreak has been patched by Samsung;
- **WP7 Root Tools:** the WP7 Root Tools is a collection of utilities developed by Heathcliff74 to obtain root access within a interop-unlocked or full-unlocked platform. The suite provides a series of tools including the Policy Editor, which is used to select trusted apps that are allowed to get root access and escape their sandbox. The suite targets Windows Phone 7 devices only;
- **Custom ROMs**: custom ROMs are usually flashed to achieve interop or full unlock conditions. A numbers of custom ROMs are available for the Windows Phone 7 platforms (e.g., RainbowMod ROM, DeepShining, Nextgen+, DFT's MAGLDR, etc.). The first custom ROM targeting Samsung Ativ S devices was developed by -W_O_L_F- from XDA-developers, providing interop-unlock and relock-prevention features among other system tweaks;
- **OEMs App and Driver Exploits**: unlocked access is often achieved exploiting security flaws in the implementation or abusing hidden functionalities of OEM drivers and apps, which are shipped with the OS. Notable examples are the Samsung Diagnosis app – abused in the Samsung Ativ S hack - that included a hidden registry editor, and the LG MFG app: both have been used to achieve the interop-unlock by modifying the value of the MaxUnsignedApp registry value.

Why can it be dangerous?

The tools above can be broadly categorized in the following categories:

- **Userland Exploits:** jailbroken access is only obtained within the user layer. For instance, a user may have root access, but is not able to change the boot process. These exploits can be patched with a firmware update;
- **iBoot Exploit:** jailbroken access to user level and boot process. iBoot exploits can be patched with a firmware update;
- **Bootrom Exploits:** jailbroken access to user level and boot process. Bootrom exploits cannot be patched with a firmware update. Hardware update of bootrom required to patch in such cases;

Some high level risks for jailbreaking, rooting or unlocking devices are as follows.

Technical Risks

1. **General Mobile**
    1. Some jailbreaking methods leave SSH enabled with a well-known default password (e.g., alpine) that attackers can use for Command & Control;
    2. The entire file system of a jailbroken device is vulnerable to a malicious user inserting or extracting files. This vulnerability is exploited by many malware programs, including Droid Kung Fu, Droid Dream and Ikee. These attacks may also affect unlocked Windows Phone devices, depending on the achieved unlocking level;
    3. Credentials to sensitive applications, such as banking or corporate applications, can be stolen using key logging, sniffing or other malicious software and then transmitted via the internet connection.
2. **iOS**
    1. Applications on a jailbroken device run as root outside of the iOS sandbox. This can allow applications to access sensitive data contained in other apps or install malicious software negating sandboxing functionality;
    2. Jailbroken devices can allow a user to install and run self-signed applications. Since the apps do not go through the App Store, Apple does not review them. These apps may contain vulnerable or malicious code that can be used to exploit a device.
3. **Android**
    1. Android users that change the permissions on their device to grant root access to applications increase security exposure to malicious applications and potential application flaws;
    2. 3rd party Android application markets have been identified as hosting malicious applications with remote administrative (RAT) capabilities.
4. **Windows Phone OS**
    1. Similarly to what is happening with other mobile platforms, an unlocked Windows Phone system allows the installation of apps that are not certified by Microsoft and that are more likely to contain vulnerabilities or malicious codes;
    2. Unlocked devices generally expose a wider attack surface, because users can sideload apps that not only could be unsigned, but that could also abuse capabilities usually not allowed to certified Windows Phone Store applications;
    3. Application sandbox escaping is normally not allowed, even in case of a higher level of unlocking (e.g., interop-unlock), but it is possible in full-unlocked systems.

Non-technical risks

1. Under the current Digital Millennium Copyright Act (DMCA), jailbreaking is termed as 'legal' in the US, which can provide some users with a false sense safety and jailbreaking as being harmless. Please refer to 'Rulemaking on Anticircumvention' for more details;
2. Software updates cannot be immediately applied because doing so would remove the jailbreak. This leaves the device vulnerable to known, unpatched software vulnerabilities;
3. Users can be tricked into downloading malicious software. For example, malware commonly uses the following tactics to trick users into downloading software:
    1. Apps will often advertise that they provide additional functionality or remove ads from popular apps but also contain malicious code;
    2. Some apps will not have any malicious code as part of the initial version of the app but subsequent "Updates" will insert malicious code.
4. Manufacturers have determined that jailbreaking, rooting or unlocking are breach of the terms of use for the device and therefore voids the warranty. This can be an issue for the user if the device needs hardware repair or technical support (Note: a device can be restored and therefore it is not a major issue, unless hardware damage otherwise covered by the warranty prevents restoration).

What controls can be used to protect against it? Before an organization chooses to implement a mobile solution in their environment, they should conduct a thorough risk assessment. This risk assessment should include an evaluation of the dangers posed by jailbroken devices, which are inherently more vulnerable to malicious applications or vulnerabilities such as those listed in the OWASP Mobile Security Top Ten Risks. Once this assessment has been completed, management can determine which risks to accept and which risks will require additional controls to mitigate.

Below are a few examples of both technical and non-technical controls that an organization may use.

Technical Controls

Some of the detective controls to monitor for jailbroken devices include:

1. Identify 3rd party app stores (e.g., Cydia);
2. Attempt to identify modified kernels by comparing certain system files that the application would have access to on a non-jailbroken device to known good file hashes. This technique can serve as a good starting point for detection;
3. Attempt to write a file outside of the application's root directory. The attempt should fail for non-jailbroken devices;
4. Generalizing, attempt to identify anomalies in the underlying system or verify the ability to execute privileged functions or methods.

Despite being popular solutions, technical controls that aims to identify the existence of a jailbroken system must relay and draw conclusions based on information that are provided by the underlying platform and that could be faked by a compromised environment, thus nullifying the effectiveness of the mechanisms themselves. Moreover, most of these technical controls can be easily bypassed introducing simple modifications to the application binaries; even in the best circumstances, they can just delay, but not block, apps installation onto a jailbroken device.

Most Mobile Device Management (MDM) solutions can perform these checks but require a specific application to be installed on the device.

In the Windows Phone universe, anti-jailbreaking mechanisms would require the use of privileged APIs that normally are not granted to Independent Software Vendors (ISV). OEM apps could instead be allowed to use higher privileged capabilities, and so they can theoretically implement these kind of security checks.

Non-Technical Controls

Organizations must understand the following key points when thinking about mobile security:

1. Perform a risk assessment to determine risks associated with mobile device use are appropriately identified, prioritized and mitigated to reduce or manage risk at levels acceptable to management;
2. Review application inventory listing on frequent basis to identify applications posing significant risk to the mobility environment;
3. Technology solutions such as Mobile Device Management (MDM) or Mobile Application Management (MAM) should be only one part of the overall security strategy. High level considerations include:
    1. Policies and procedures;
    2. User awareness and user buy-in;
    3. Technical controls and platforms;
    4. Auditing, logging, and monitoring.
4. While many organizations choose a Bring Your Own Device (BYOD) strategy, the risks and benefits need to be considered and addressed before such a strategy is put in place. For example, the organization may consider developing a support plan for the various devices and operating systems that could be introduced to the environment. Many organizations struggle with this since there are such a wide variety of devices, particularly Android devices;
5. There is not a ‘one size fits all’ solution to mobile security. Different levels of security controls should be employed based on the sensitivity of data that is collected, stored, or processed on a mobile device or through a mobile application;
6. User awareness and user buy-in are key. For consumers or customers, this could be a focus on privacy and how Personally Identifiable Information (PII) is handled. For employees, this could be a focus on Acceptable Use Agreements (AUA) as well as privacy for personal devices.

Conclusion

Jailbreaking and rooting and unlocking tools, resources and processes are constantly updated and have made the process easier than ever for end-users. Many users are lured to jailbreak their device in order to gain more control over the device, upgrade their operating systems or install packages normally unavailable through standard channels. While having these options may allow the user to utilize the device more effectively, many users do not understand that jailbreaking can potentially allow malware to bypass many of the device's built in security features. The balance of user experience versus corporate security needs to be carefully considered, since all mobile platforms have seen an increase in malware attacks over the past year. Mobile devices now hold more personal and corporate data than ever before, and have become a very appealing target for attackers. Overall, the best defense for an enterprise is to build an overarching mobile strategy that accounts for technical controls, non-technical controls and the people in the environment. Considerations need to not only focus on solutions such as MDM, but also policies and procedures around common issues of BYOD and user security awareness.