WIND™

Common Android Security Vulnerabilities in an Automotive Environment

WHEN IT MATTERS, IT RUNS ON WIND RIVER

EXECUTIVE SUMMARY

Since its first commercial release in 2008, the many security vulnerabilities found in Android have made it clear that basic Android Open Source Project (AOSP) Android is not secure enough for deployment in an automotive environment, where it is directly connected in a read/write manner to CAN bus networks that contain safety critical automotive infrastructure. Furthermore, developers for applications running on Android, even those that require high security such as for financial transactions, do not always employ accepted security practices. These two factors—Android vulnerabilities and inadequate implementation of security practices—will cause problems in an automotive environment without adequate protection.

Many of the known vulnerabilities from the cellular area are relevant to the use of Android in an integrated automotive environment. This paper explores a selected subset of the most relevant of these. The list is not exhaustive, but it provides broad coverage of the types of vulnerabilities and exploits that can be expected in an automotive environment—for both attacks and attackers could come in many different forms. Some examples include:

- Directed broad attacks targeted at applying brakes on only one side of the vehicle in order to put many vehicles into a spin while driving at highway speeds on congested freeways at a specific time
- Script kiddies driving down the road and causing all the windows of nearby cars to open and close while in range
- Attacks against the service center using the automobile as an attack vector

By first understanding these vulnerabilities, we can then create layers of strategies to prevent them from being exploited.

TABLE OF CONTENTS

VULNERABILITIES

This section examines a subset of the known Android vulnerabilities. The primary source of this information is the U.S. National Vulnerability Database (NVD). As of September 2013, the NVD lists more than 300 public entries that match the search "android."

However, the NVD does have some limitations as a source of information about Android vulnerabilities. First, other vulnerabilities exist that are general browser or Linux kernel vulnerabilities which affect Android systems, but are not included in that list.

Second, in some ways, the largest threat comes from social engineering attacks, where the user is duped into allowing activities that compromise the system. For example, typical cell phone users are not aware of the factors that make up a secure system, so they can be easily deceived. Social engineering attacks are not tracked in the NVD, nor listed prominently in this document. No technical solution is relevant, and the educational solutions attempted so far have had, at best, lackluster results. In an automotive environment, it may be acceptable to limit the choices that a user can make in order to prevent successful social engineering attacks from creating safety vulnerabilities.

Third, not all reported vulnerabilities are exploited, and not all exploited vulnerabilities are reported. Some vulnerabilities are discovered by white hat investigative hackers and then closed, and only after the vulnerability has been closed is it disclosed publicly. Other known vulnerabilities have not yet been, and may never be, reported to the NVD or accepted by their review board due to a lack of reporting requirements and lack of social pressure to report them. Some of those vulnerabilities are undoubtedly already exploited by malware. In this report, where no Common Vulnerabilities and Exposures (CVE) number is listed, the vulnerability is a candidate for this category. Also, in some cases the vulnerability is listed and a CVE number is assigned, but the report was made by the owning agency, Google in this case, and the CVE description is not made public. Vulnerabilities of this nature are not included in this report.

Fourth, many of the reported vulnerabilities are no longer applicable to the most recent versions of Android. They have been fixed. However, even those vulnerabilities are still open to the many deployed devices running older versions of Android. It is expected that updates to automotive Android devices will not occur so frequently or last for the full 15–20 year lifespan of the vehicle. Therefore, automotive systems should incorporate additional security capabilities that will continue to protect against attack well after the car is no longer able to update to newer versions of Android that have fixed known vulnerabilities.

Finally, the following classes of vulnerabilities have been selected due to their relevance to automotive uses of Android.

Rootkits and Other System-Level Threats

A very common and often well-advertised type of vulnerability allows a user to gain system-level privileges on the system. On standard phones, this allows operations such as overclocking the processor, abnormal management of SD cards and USB devices, and other system-level operations that are allowed only to the "root" user.

On a cellular device, this may be acceptable or even desirable to users. But if the device is interconnected with automotive systems, this kind of control cannot be allowed.

Denial of Service

There are many levels of vulnerabilities on Android that can lead to denial-of-service conditions, ranging from a simple crash of a specific application to forcing the system into an infinite reboot loop that denies any service provided by the entire device.

The triggers for this kind of attack may be internal to the device, such as a buggy or malicious application that has been installed. Or they can be completely external to the device, such as specific network traffic patterns visible to the device. In either case, the end result is a service or device that is unavailable, which could be annoying and distracting to an automotive customer.

Middleware Vulnerabilities

The overall architecture of Android is as follows: The Linux kernel is used to manage the device hardware, but typical Linux user environments have been replaced with custom Android software. This custom Android layer can be referred to as Android middleware. This middleware includes init, the code that starts everything running; bionic, which is a library for applications to use to gain system services and functionality; dalvik, a system to run Java code; daemons such as vold, the system to manage SD cards and USB devices; and other modules.

Many of the known vulnerabilities in Android are related to this middleware layer. As a result, automakers who wish to use Android in integrated automotive environments would need either to perform rigorous inspection and certification of the middleware, which would be prohibitively costly, or to use mechanisms external to Android itself to limit the ability of Android to adversely affect automotive systems. Failure to take either of those actions would risk leaving vulnerabilities in place that might adversely affect safety critical automotive systems. However, a middle course is also possible, in which Android security is improved and hardware mechanisms are also used to limit adverse effects of attacks.

Android can be safely integrated with automotive systems—but care must be taken to ensure that the interfaces restrict the interaction appropriately and prevent adverse effects to safety critical systems. For example, it may be necessary to have an interlock between the Android device and the driver seat adjustment—a switch that would only allow the Android system to control seat position while the button is pressed, giving the driver full control over disconnecting the Android system from the control loop. Otherwise, a malicious Android app could wait for an especially distracting event such as a freeway entry to move the seat.

Image, Video, and Audio Vulnerabilities

Hundreds of vulnerabilities have been filed against Adobe Flash Player, including against the versions used on Android devices. Although some of these bugs have been around for a long time, they have not been fixed in recent releases and are still relevant to today's devices. There are also vulnerabilities in PDF viewers, JPEG support libraries, and other proprietary data format management tools.

In an automotive environment, a crafted audio file burned onto a CD and played by a specific car model's CD player can be created so that it inserts user-defined raw CAN bus packets into the in-vehicle network for that vehicle[1].

Browser Vulnerabilities

Browsers are a special form of middleware, and need to be treated separately. One primary aspect of a browser is that it downloads executable HTML code, and possibly other kinds of code, from a remote server and then executes that code. There are no guarantees that the external code is correct, nor that it is not malicious. And for the most part, users are not given the choice to examine code before it is executed.

Browser vulnerabilities are widespread, and the same vulnerability can be present on different browsers and on different operating systems with the same browser. So even though the NVD lists 17 known Android browser vulnerabilities, it is likely that some other known browser vulnerabilities are also relevant to Android.

One particularly nasty browser-based attack uses the telephony Unstructured Supplementary Service Data (USSD) functionality in a denial-of-service attack[2]. A USSD code can be issued to cause the system to perform a factory reset. This operation destroys all user data, and can keep the device out of service for several minutes while the factory reset is performed. The attack can happen from any website, and it is executed without warning to the user. The HTML code to implement this attack is a very simple one-line HTML statement. This vulnerability is not publicly listed or searchable in the NVD.

Application Vulnerabilities

Due to poor quality, general Android applications from various Android marketplaces should not be made available on any automotive system that might have interconnects with other automotive systems unless adequate protections are in place. Application developers, as a whole, do not have the experience, background, or willingness to follow best practice guidelines for high-security applications necessary to produce secure software. This does not indicate that the in-vehicle Android system should not have access to an app market, and this paper proposes that as part of a solution. However, general purpose applications, not designed for the automotive environment, should be avoided.

As a result of the lack of background and security-aware practices, many applications contain bugs and vulnerabilities. The NVD shows more than 127 matches to a search for "android application" (as of September, 2013). But this number is a drop in the bucket compared to the number of critical reviews available directly from Google Play that appear to report bugs.

When standards are enforced, it may be completely acceptable to allow general developers to create applications for use on an infotainment system[3]. But in order to accomplish this, strict enforcement of the application marketplace is required, as well as strict

enforcement of devices so that they can only install applications from the secure market, and not from the Web or other unreliable markets.

Viral Aspects of Malware

One kind of threat in an automotive environment is the use of malware to attack not just the automobile or driver, but the auto manufacturer. This might happen in a number of ways. For the purposes of this paper, we use the example of a service center's computer as the target of the attack.

In the security community, directed attacks are considered rare, but they are also known to be much more difficult to defend against. Whereas the opportunistic attacker can find a vulnerability and use it wherever it is most convenient, the directed attacker finds information specific to a given target and tries to find vulnerabilities in it. Thus, the job of the attacker is more difficult, but defending against this kind of attack requires absolute, comprehensive security.

Use of an automobile's computers to attack service centers is a credible scenario, just as is the opposite: attacking an automotive computer via the service center computer. In both variants, the results in an automotive environment would include denial of service while the service center's computer is being investigated, unhappy customers when their vehicle service cannot be performed at the scheduled time, direct costs associated with finding and eradicating the malware, other direct costs that depend on the characteristics of the malware being used, and potential reduction in the safety of vehicle operation for any vehicle serviced between the time the service center's computer is infected and the time the infection is detected.

Botnets

Android devices can be compromised by those who want to use them for sending spam, phishing attacks, and other money-making schemes. They typically accomplish this by adding a layer to a well-known game or application. The game or app performs its normal function, and the malicious layer works in the background to send SMS messages, email, or other types of messages to destinations determined by a remote server.

The effects on the user can include sluggish system performance, denial of service when the provider discovers that the device is being used in a way that violates the usage agreement, and charges to the account for data or SMS overages.

In addition, botnets are often used to collect and disseminate personal information about the device owner, which is fed back into their database to provide higher returns on spam sent to that individual.

An automotive application that is infested with botnet code would result in the same effects listed above as for a cellular or tablet device. In a connected automotive infotainment system, these symptoms would result in a negative impact on the auto manufacturer's reputation. In addition, sluggish performance may result in safety issues in cases where Android is integrated with automotive systems; for example, when the Android system is used to display the backup camera.

Other Vulnerabilities

Near field communication (NFC) has been shown to be insecure[4]. Although the direct attack surface of NFC itself is small, with a few thousand lines of code and very short packets, the integration of NFC with other applications means that NFC can be used to silently force a system to visit a compromised website or take other dangerous action. So the indirect attack surface of NFC is huge.

With care and security-conscious review of designs and implementations, NFC can be used effectively and securely.

SECURITY ENHANCEMENT STRATEGIES

Android was not developed with high levels of security as a design requirement. Therefore, care must be taken when using Android in an automotive environment to avoid creating high-risk situations inadvertently, and avoid opening vulnerabilities that can be exploited by malicious code. Figure 1 shows common guidance for improving software security. In a separate paper, we will propose ways to alleviate the known Android problems in an automotive environment by following a multi-layer defense-in-depth strategy.

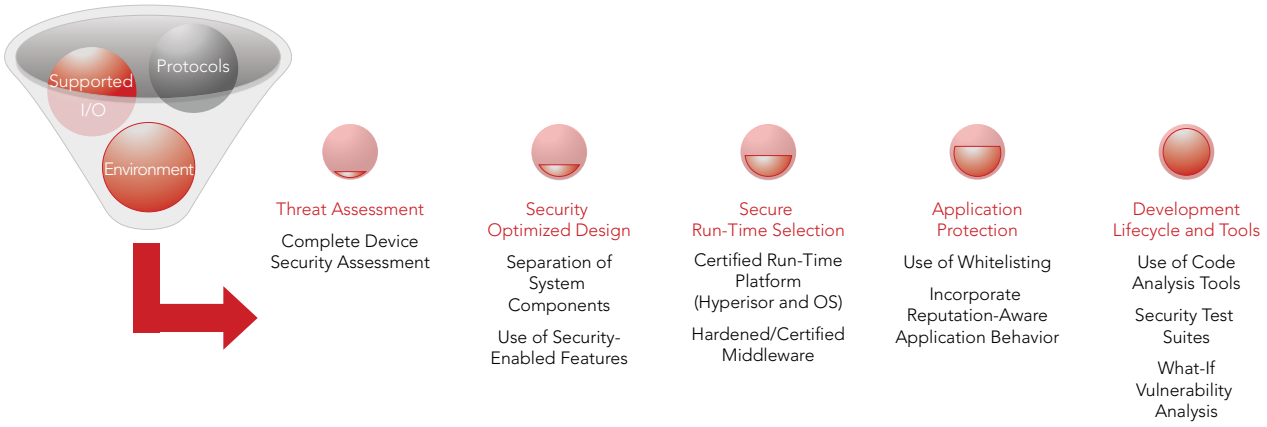

Figure 1: Five steps to enhance embedded security

CONCLUSION

The Android operating system is quickly becoming a platform of interest for building next-generation in-vehicle infotainment (IVI) systems. But as Android devices and technologies continue to evolve, the embedded Android market is facing a greater need for stronger security capabilities.

Standard Android, as released by Google, may be suitable as a starting point for deployment as a stand-alone infotainment system. But when the infotainment system needs to be connected to other automotive components such as driver controls, vehicle status indicators, backup cameras, or environmental controls, basic AOSP Android does not provide the necessary level of protection. Security enhancements must be made to minimize risk and ensure driver safety.

By adding appropriate enhancements, it is possible to create an environment that is sufficiently secure. For best results, a defense-in-depth (DiD) strategy provides separation of convenience and infotainment features from safety critical systems using non-programmable, hardware-based barriers, plus software enhancements. Software enhancement measures are discussed in more detail in a separate white paper from Wind River®.

[1] Koscher et al., 2010. "Experimental Security Analysis of a Modern Automobile." www.autosec.org/pubs/cars-oakland2010.pdf Checkoway et al., 2012. "Comprehensive Experimental Analyses of Automotive Attack Surfaces." www.autosec.org/pubs/cars-usenix-sec2011.pdf

[2] Vince, September 25, 2012. "TouchWiz-specific hack can hard reset Galaxy S III and other Galaxy phones through their web browsers." http://blog.gsmarena.com/touchwiz-specific-hack-can-hard-reset-galaxy-s-iii-and-other-galaxy-phones-through-their-web-browsers/ http://news.ycombinator.com/item?id=4569686

[3] J. Hill, January 18, 2013. "Ford AppLink opens floodgates to in-car iOS, Android, and BlackBerry apps." http://arstechnica.com/gadgets/2013/01/ford-applink-opens-floodgates-to-in-car-ios-android-and-blackberry-apps/

[4] Dan Goodin, July 25, 2012. "Android, Nokia smartphone security toppled by Near Field Communication hack." http://arstechnica.com/security/2012/07/android-nokia-smartphone-hack/

WIND

Wind River is a global leader in delivering software for the Internet of Things. The company's technology is found in more than 2 billion devices, backed by world-class professional services and customer support. Wind River delivers the software and expertise that enable the innovation and deployment of safe, secure, and reliable intelligent systems.