INL/EXT-13-28223

Dynamic Analysis of Mobile Device Applications

Corey Thuen

January 2013

The INL is a U.S. Department of Energy National Laboratory
operated by Battelle Energy Alliance

INL/EXT-13-28223

Dynamic Analysis of Mobile Device Applications

Corey Thuen

January 2013

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Office of Electricity Delivery and Energy Reliability
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

ABSTRACT

The On-Device Dynamic Analysis of Mobile Applications (ODAMA) project was started in an effort to protect mobile devices used in Industrial Control Systems (ICS) from cyber attack. Because mobile devices hide as much of the "computer" as possible, the user's ability to assess the software running on their system is limited. The research team chose Google's Android platform for this initial research because it is open source and it would give us freedom in our approach, including the ability to modify the mobile device's operating system itself. The research team concluded that a Privileged Application was the right approach, and the result was ODAMA. This project is an important piece of the work to secure the expanding use of mobile devices within our nation's critical infrastructure.

EXECUTIVE SUMMARY

Securing the country's energy sector infrastructure from cyber-attack is critical to the well-being of the American people and is a central focus to the Department of Energy's (DOE) Office of Electricity Delivery and Energy Reliability (OE) Cybersecurity for Energy Delivery Systems (CEDS) program. The DOE program aims to enhance the reliability and resilience of the nation's energy infrastructure by reducing the risk of energy disruptions due to cyber-attacks. Technology in general, is quickly moving toward mobile devices and Industrial Control Systems are no exception to this fact. Unlike traditional computing systems, however, these devices are made to hide as much of the "computer" as possible. In doing so, manufacturers have taken away a user's ability to assess the software running on their system. This project was started was started to remedy that problem.

The first step in conducting this work was to evaluate and make a decision as to which mobile operating system to use. The research team chose Google's Android platform for this initial research mainly because it has the largest market share of any mobile operating system and, more important, it is open source, which means we would have a much easier time with any approach we deemed necessary (including modifying the operating system itself).

Several methods were explored, but the research team concluded that a Privileged Application was the right approach, and ODAMA (On-Device Dynamic Analysis of Mobile Applications) is the result. This tool fills a gap in the available toolbox of our nation's front-line defense for control systems mobile device applications. By taking away a user's ability to assess the software running on their system, carriers and manufacturers are inadvertently working to negatively impact our ability to secure mobile devices. ODAMA is one weapon in our defense arsenal. The more we know about what our devices are doing, the better we can identify problems and potential attacks.

The mobile space is only getting more important as our devices get more powerful and people are more familiar with them. We need tools and capabilities to identify security concerns in this space in order to be prepared for the future. This project is an important piece of that work.

CONTENTS

FIGURES

ACRONYMS

ARM	Application Response Measurement
CEDS	Cybersecurity for Energy Delivery Systems
DOE-OE	Department of Energy Office of Electricity Delivery and Energy Reliability
ICS	Industrial Control Systems
INL	Idaho National Laboratory
NSTB	National SCADA Test Bed program
ODAMA	On-Device Dynamic Analysis of Mobile Applications
R&D	Research and Development

Dynamic Analysis of Mobile Device Applications

1. Dynamic Analysis of Mobile Device Applications Project

The INL Dynamic Analysis of Mobile Device Applications Project is funded through the Department of Energy's (DOE) Office of Electricity Delivery and Energy Reliability (OE) Cybersecurity for Energy Delivery Systems (CEDS) Research and Development (R&D) Program. The DOE program aims to enhance the reliability and resilience of the nation's energy infrastructure by reducing the risk of energy disruptions due to cyber-attacks. The project is a one year effort, started in FY 2012, awarded as part of the INL NSTB Core Frontier R&D tasks.

Technology is quickly moving toward mobile devices. As ARM processors become more powerful and cheaper to make, the technology is, it seems, ubiquitous. ICSs are no exception. Mobile operating systems have come a long way in the past few years; these devices are popular, provide powerful portable computing tools and are being used by a large segment of the working population.

The problem, though, is that unlike a traditional computing environment, these devices were made to hide as much of the "computer" as possible. That is to say, they keep as much of the nitty-gritty undercarriage and inner-workings away from the user as possible. No one wants to see blue screens, error messages, or other nonsense. It should, users believe, "just work."

In creating such devices, manufacturers have taken away a user's ability to assess the software running on their system. The traditional tools for evaluating what software does are no longer useful because they don't run on the device and even if they did, they probably don't have permissions to actually gather any information. Well, our research team thinks that's bogus and intends to do something about it. That's why this project was started.

2. RESEARCH

2.1 Identify Environment

The first step in conducting this work was to evaluate and make a decision as to which mobile operating system to use. Apple iOS was considered because of its popularity but thrown out because of its highly controlled environment. Blackberry was discounted because we could find no one creating, or planning to create, ICS software for the platform.

The research team chose Google's Android platform for this initial research for multiple reasons. First, it has the largest market share of any mobile operating system. Second, and most importantly, it is open source, which means we would have a much easier time with any approach we deemed necessary (including modifying the operating system itself).

2.2 Choose Method of Approach

In trying to get access to the mobile device architecture undercarriage, a few methods were explored. Each has its pros and cons and we eventually concluded that a Privileged Application was the right approach. The others are documented here for comparison.

2.2.1 Unprivileged Application

The first approach considered is the use of an unprivileged application. This avenue was quickly discarded because there are some fundamental limitations. Applications on android (and other mobile platforms) are isolated from each other and prevented from interacting. This of course means that antivirus apps you can purchase are not that beneficial because they don't actually have the ability to do much on the system.

It may be possible to create an on-device **static** analysis program as an unprivileged application but that does not meet the dynamic requirement set forth by this project. The unprivileged application approach does not accomplish our goals of exposing the inner workings of the system to a user.

2.2.2 Kernel Module

The second approach is the use of a kernel module. This involves creating a module to insert into the kernel of the mobile operating system (that is the very core component to what the entire device **is)**. This method gives us the most control and visibility to what occurs on the system.

An app cannot sneeze without going through the kernel first. This means you would see every file touched, every network packet sent, every dirty thought an app has. Using this method, you could go beyond simply observing the inner-workings of the system and get into controlling program flow. Thus, should an app attempt to conduct activity you deem unworthy, you can prevent it from ever occurring.

The advantages of this method also mean that nothing would be missed. Further, android apps all run as separate users so tracking them in the kernel is awfully convenient and you can eliminate external noise. This method wasn't chosen, in the end, because of the limitations.

In order to successfully implement a kernel module solution a significant amount of work is necessary. The kernel source code must be acquired, the module must be compiled against it (that is, for each type of device and possibly version), and it must be installed correctly. Kernel modules also come with a danger; if a kernel module has a bug, the entire system could crash.

We implemented some of the work done in this area before abandoning it for door number 3. See Figure 1 for a picture of us watching the radio daemon to see what kind of communications the base operating system has with the radio driver. This kind of visibility is impossible with either of the other methods.

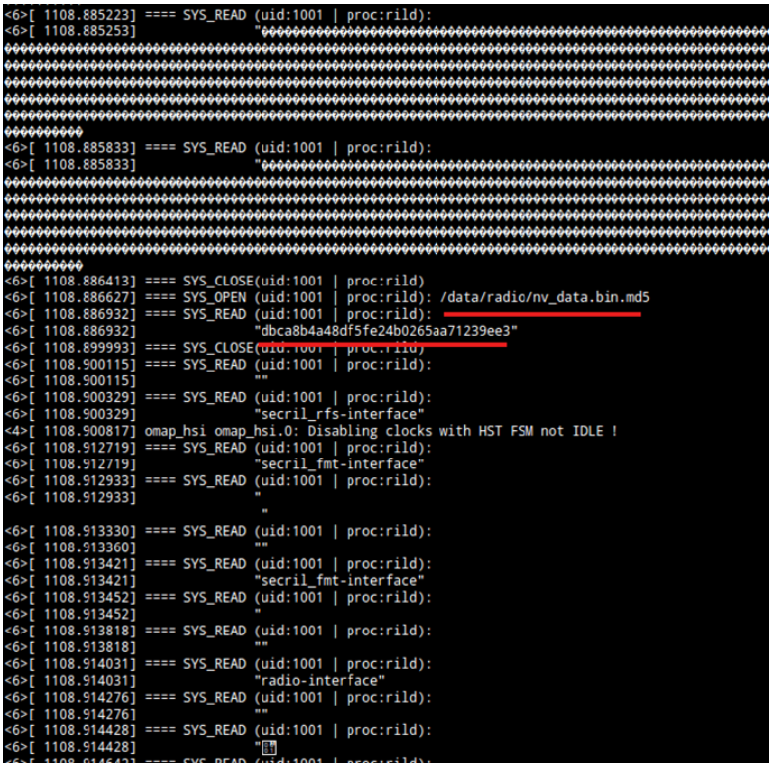

Figure 1. Kernel module watching radio daemon.

2.2.3 Privileged Application

The privileged app approach is the sweet spot (and has the most pretty pictures). Here we get most of the benefits of a kernel module with few of the drawbacks. This approach was selected and ODAMA was produced. (ODAMA = On-Device Dynamic Analysis of Mobile Applications.)

ODAMA is a privileged app that runs right in the mobile operating system. It differs from an unprivileged app in that it requires superuser permissions. The problem with that, if you recall, is that a majority of users do not have these permissions on their device (or even know what they are). This means the user of ODAMA will need to either acquire a device that already has those permissions built in or "root" their device. Neither of those solutions are particularly difficult, just uncommon.

Once installed, ODAMA is an app just like any other. You fire it up and select the app you would like to monitor. ODAMA conducts its monitoring in the background while you use and abuse your suspect app. In the end, the results are stored in a database that you can browse (or save, copy, etc) from the results screen of the ODAMA app.

ODAMA works by using the linux utility "strace" to watch for system calls made by a given process. Compared to the kernel module this method has a slightly higher chance of missing some activity (though only at the beginning) but is applicable to a much larger set of devices and doesn't require the user to know anything about the device or have any of its source code (aside from having superuser access).

We used ODAMA to analyze android applications (both ICS oriented and not). In order to avoid calling out specific applications, we created an example app to analyze, called "Badroid," that shares some characteristics with apps found in the real world.

In Figure 2, the user chooses the app they would like to trace from the list of installed apps.

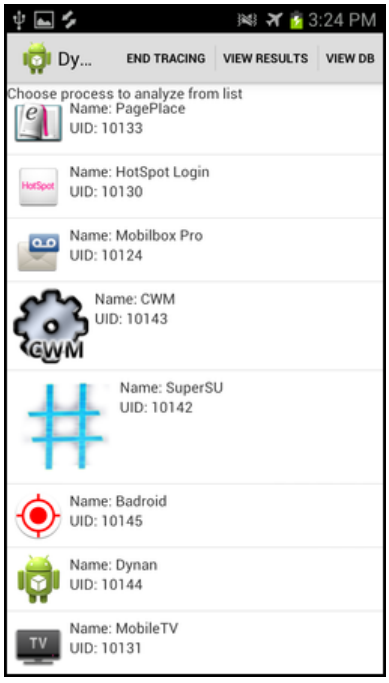

Figure 2. Main ODAMA screen.

In Figure 3, the user selects our example app, Badroid, and chooses "Do bad," which acts as hidden malicious activity.

Figure 3. User selects Badroid app.

In Figure 4, the user chooses the “Do File,” which opens up and reads a file on the SDcard.

Figure 4. User chooses "Do File."

In Figure 5, the user returns to Dynan and taps “END TRACING” to discontinue tracing the example app.

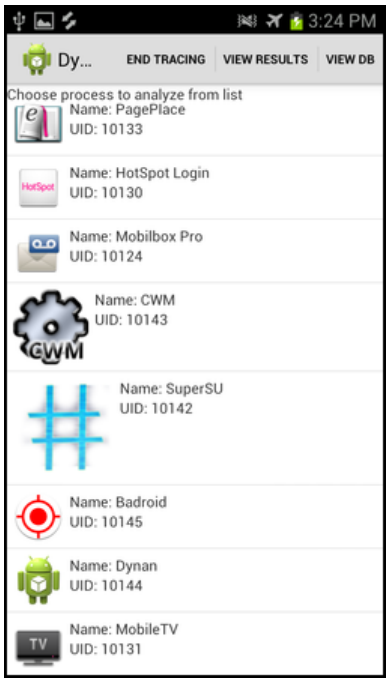

Figure 5. User returns to Dynan.

In figure 6, the user selects DB view to see a list of all trace results (only one in our case).

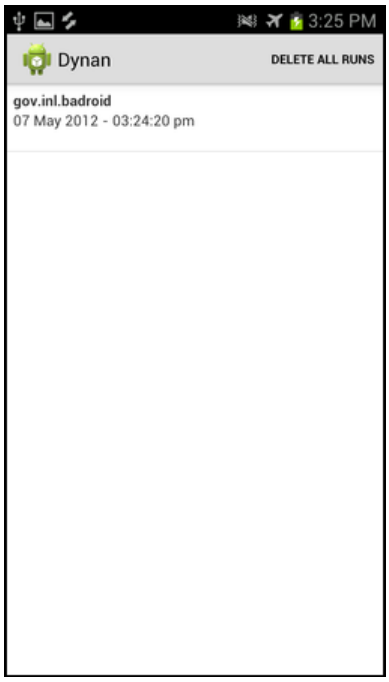

Figure 6. User selects DB.

In Figure 7, the user sees something suspicious in the results list on a socket send. The user selects that syscall to see more information. In Figure 8, the user investigates web.thuen.org because it looks nefarious.

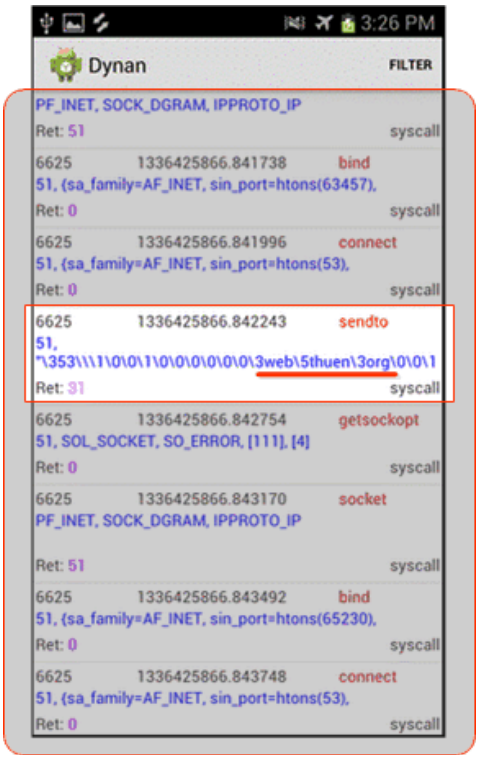

Figure 7. Results list.

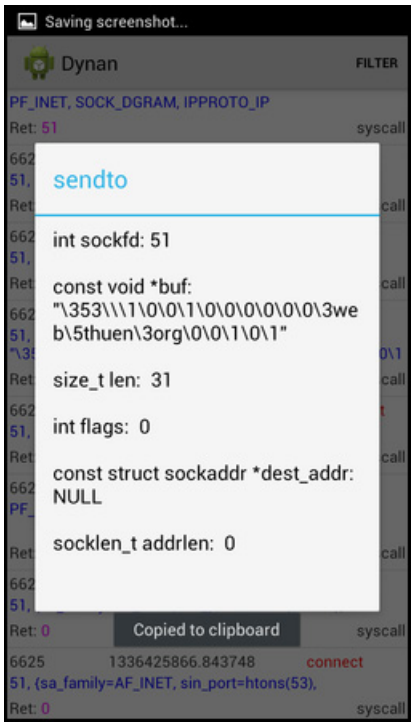

Figure 8. web.thuen.org.

In Figure 9, we see that ODAMA also supports syscall filtering via the filtering menu. Let's look only at file options. In Figure 10, we can easily see the file our target app opened and can evaluate whether this is a desired behavior.

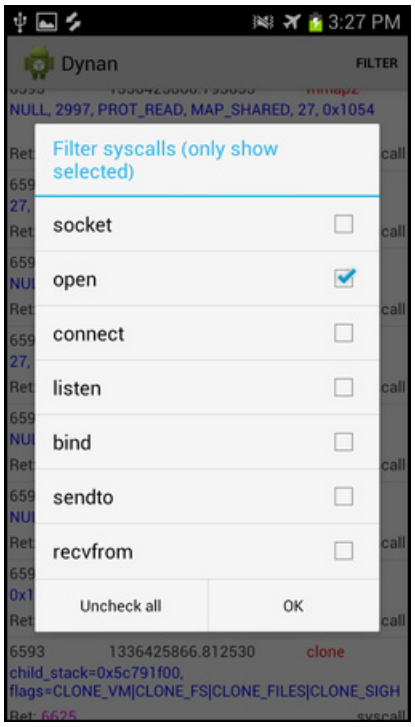

Figure 9. ODAMA supports syscall filtering.

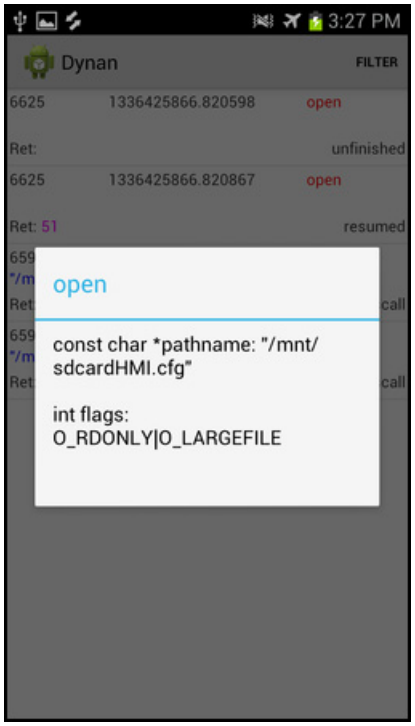

Figure 10. File our target app opened.

ODAMA also supports viewing the raw syslog output. Figure 11 shows a list of traces.

Figure 11. Raw syslog output.

Figure 12 shows the raw output from our example trace.

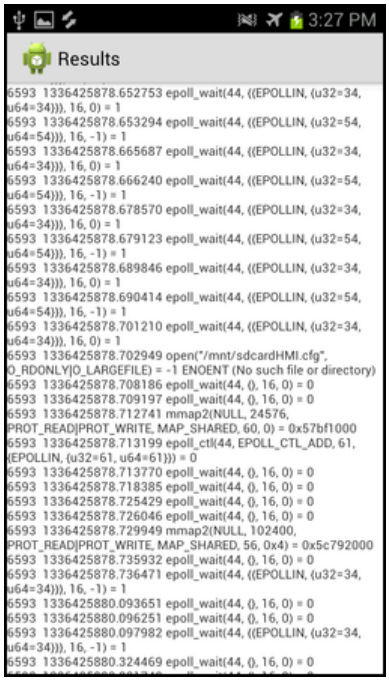

Figure 12. Raw output from example trace.

3. Impact to Industry

This tool fills a gap in the available toolbox of our nation's front-line defense. Carriers and manufacturers are actively, though inadvertently, working to negatively impact our ability to secure mobile devices. ODAMA is one tool in our mobile device defense toolbox. The more we know about what our devices are doing, the better we can identify problems and potential attacks.

4. Conclusion

The mobile space is only getting more important as devices get more powerful and people are more familiar with them. We need tools and capabilities to identify security concerns in this space in order to be prepared for the future. This project is an important piece of that work, but only a piece.

It has been a pleasure to work on this project and we are thankful for the opportunity.