Towards Dynamic Malware Analysis to Increase Mobile Device Security

Michael Becher, Felix C. Freiling

{becher,freiling}@informatik.uni-mannheim.de

Abstract: Mobile devices like smartphones have different properties compared to common personal computers. Therefore, the basic principles of mobile device security are also different and this situation is advantageous because it gives more possibilities of securing the mobile device, based on the more centrally controlled environment in which they operate, and a different role of the user. This paper describes the framework for a background monitoring system to collect software that a user is going to install on its device, and to automatically perform a dynamic analysis of the software. The implementation of a dynamic analysis module of the monitoring system is presented. It has the task of performing the dynamic analysis by transferring common sandboxing approaches to the Windows Mobile operating system on the ARM architecture.

1 Introduction

Malicious software, or *malware* for short, in the form of worms, viruses, bots, spyware etc. is omnipresent in the Internet [GHW07]. Computers attached to this network must implement special precautions like firewalls, anti-virus products etc. to prevent infection.

While the world of malware in standard desktop computers is very much alive, the situation is very different in the realm of mobile devices like smartphones or PDAs. Here, we hear of no large outbreaks or viruses or worms. This is surprising since these devices all are real (albeit small) multi-purpose computers. And so the fact that nothing really dangerous has happened yet does not mean that this will remain so in the future. Prophets like Pescatore/Girard (Gartner Group) in fact predicted more than two years ago, that the conditions for the first large-scale outbreak of a worm for mobile devices would be fulfilled at the end of the year 2007. Similar prophecies are documented elsewhere [Hyp06, TH06] too.

The state of the art of mobile device security is in fact difficult to assess. The studies of malware programs in mobile device contexts [FLJ$^+$07, BFL07] have been performed in rather artificial environments. It seems that only *real* malware for mobile devices can answer the question.

In this paper, we attempt to fundamentally analyse the similarities and differences between mobile devices like smartphones and common personal computers with respect to their security properties. This results in fundamental statements about the basic principles of mobile device security. We argue that the situation in mobile device security if for a couple

of reasons advantageous to the defender because it gives more possibilities of securing the mobile device. This is mainly due to the more centrally controlled environment, in which the mobile phones operate, and a different role of the user.

Resulting from these discussions, we develop in this paper an architecture for a background monitoring system to collect software that a user is going to install on its device, and to automatically perform a dynamic analysis of the software. The implementation of a dynamic analysis module called MobileSandbox is presented. It has the task of performing the dynamic analysis by transferring common sandboxing approaches to the Windows Mobile operating system on the ARM architecture.

Summarized, this paper makes three main contributions. It presents a compilation of the different conditions of mobile device security compared to common personal computer security in Section 2. Based on these conditions it describes a reasonable framework for dealing with the mobile malware threat in Sections 3 and 4. Finally, it gives details of the implementation of the MobileSandbox dynamic analysis module that creates a sequence of system calls of the analyzed program. Evaluating this sequence is possible but not within the scope of this paper.

2   Mobile Device Security

The topic of mobile device security develops somehow detached from common personal computer security. The differences and reasons are explained in this Section, three differences to PC security in Subsection 2.1 and some basic assumptions about the user Subsection 2.2, that led to the conclusions of this paper.

2.1   Differences to Personal Computer Security

There are three important difference of mobile device security compared to personal computer security. They are the basis to novel security measures especially designed for mobile devices and their infrastructure, that cannot be transferred from existing personal computer security solutions.

A first difference is the inherent possibility for malware to generate costs for the user and revenue to the malware author. This problem existed previously in PC security, when dial-up connections via modem or ISDN were common. Malware could dial premium rate numbers and with it directly benefit the malware author. With the appearance of DSL and flatrates, this problem mostly vanished, because the connection to the telephone system was not available anymore. In mobile devices however, it will most likely be a problem for a long time. Even if flatrates for data or voice services become common, separately charged premium services will always be available.

The second difference is the presence of the mobile network operator and its influence on the device. Different from PCs, where the network provider almost always has no

influence on the user's computer, the mobile network operator has a trusted device inside the mobile phone, the SIM card. In combination with the SIM Application Toolkit (SAT) it is possible to create trusted applications on the mobile phone with enhanced security. This difference remains true even despite of the facts, that trusted platform modules start to appear in PCs and that third-party trusted modules are available for mobile devices, e.g. embedded into a memory card. Both do not have the unique owner that the SIM card has.

The third difference is a question of reputation, that is connected to the second difference. The mobile network operator will invoice every event that generated costs, even though it might have been generated by malware. Therefore, it can be expected that the mobile network operator will be held responsible from the user's point of view. In case of a widespread outbreak where several network operators are involved, mobile malware might even have an impact on the reputation of the entire mobile phone system in general.

2.2   The User of Mobile Devices

The implications of security measures like application frameworks (e.g. the Java framework J2ME) and signature schemes for different trust levels might not be understood by the average user. A recent example is the iPhone that was locked for third-party software. This was a good measure from the security point of view. But this measure was not accepted by its users (for usability reasons), forcing the Apple to unlock the devices, thus decreasing their security. So it can be expected, that devices will be open to extensions, and therefore malware, in the future.

The average user does not have extensive knowledge of security [WT99]. Even if his security-awareness should be increased by media coverage of security incidents (like worm or virus outbreaks), it seems questionable, if he is able to differentiate between different classes of security products to use them correctly. An additional proposition is that even if the user is security-aware, his will to get into the depths of security research and products might be close to zero [Gör06]. Therefore, it can reasonably be assumed, that the user of mobile devices will never become security-aware. Especially techniques of social engineering can be expected to be successfully applicable for an indefinite amount of time.

Additionally, it is assumed here that the appreciation of the mobile device is lower than for desktop PCs, and that it is more seen as a disposable item. Therefore, the cost of security solutions must be lower than for PCs, the solutions should especially work automatically and should not need much operator maintenance like adding new signatures to anti-virus databases.

Because of these facts, it is proposed that most of the users need a solution that is embedded into the normal handling of the used device rather than a separate solution.

But even though most of the users do not want to spend their time with security, they have different security needs. Some might especially want to prevent unwanted costs while others might be more concerned about their personal data.

3   Mobile Device Security System Requirements

It is the overall goal to increase the security of mobile devices all of the users. But as shown above, most users do not want to spend much time for installing and configuring a solution, and they do not want to spend much money for the solution. Therefore, it could be useful to give the user an easy possibility to express his security needs. That means to completely integrate security into a clear definition with an interface, that the user is accustomed to. Compared to ordinary computers, this interface could take advantage of the strong connection of mobile phone users to their mobile network operators. A user only should have to answer a few simple questions to define his *personal security profile*. Examples of possible questions are:

- allow applications access to private data

- allow cost-generating events (messaging, IP traffic)

- allow the network operator to monitor the software on his device (see next Section 4)

The security profile can serve two purposes. On the device level, security policies must be implemented to prevent the disallowed actions. On the level of the network analysis system, the profile settings are used to create the answer if an installable software is malicious or not.

The security policy might be overridden by certain applications. For example, the messaging program should usually be allowed to send messages, even if the user has disallowed it. And other applications might be approved by the network operator. These exceptions can technically be implemented by signature schemes, where a valid signature would override the personal security profile.

4   Mobile Dynamic Malware Analysis

A mobile network operator could legally be forced to refund any monetary damage that customers experience because of malware. If so, it could be justifiable to invade the user's privacy and analyze every piece of installable software, that reaches the user via the network. It could even be required that installable software from local sources like memory cards or a Bluetooth/infrared connection must be sent to the network operator before installing.

The analysis system here uses the mobile network as analysis place rather than the mobile device for two reasons. First, the mobile network has more computing power to perform a more thorough analysis. Only this makes the presented mobile dynamic malware analysis feasible. Second, it is assumed that most software will be delivered via the mobile networks, in part because of the easier handling compared to dealing with local connections.

Therefore, before a user installs software on his mobile device, the software will be analyzed in the mobile network for malicious behavior. It could be one of today's anti-virus

products. But they do the analysis most of the times based on their signature database with all their known drawbacks regarding completeness and susceptibility to small binary changes in the sample [MCDK06]. The use of formal verification methods could be thought of, but it can be expected that no formal specification for a software is available most of the times. Methods of static analysis could be used, but they are limited [MKK07b].

A promising approach is an automatic dynamic analysis, where system calls are logged and afterwards analyzed for malicious behavior. Because of mobile device limitations, this cannot be done efficiently on the mobile device. Recently, much work has been done in this direction for Windows malware. CWSandbox [WHF07] and TTAnalyze [BMKK06] monitor the execution of one sample and log the activities during one particular run. Even though this method misses more sophisticated malware that has some additional requirements for showing its malicious behavior, the absolute number of detected malware is still high because of the vast overall number of malware for Windows [GHW07]. Recent work enhances the dynamic analysis by being able to analyze multiple execution paths [MKK07a] which is a promising step towards a complete dynamic analysis, even though the approach has some limitations. The dynamic analysis of the sample should be as complete as possible, because this would relieve all the drawbacks of static analysis like scrambled or self-modifying binaries. Preliminary work towards a complete mobile dynamic malware analysis for the Windows Mobile operating system is MobileSandbox in Section 5, which at the moment implements analyzing a single run.

Three parts are important for mobile dynamic malware analysis: collecting the software samples as complete as possible (see Subsection 4.1), analyzing the samples as complete as possible (see Subsection 4.2), and taking certain actions as a response to the analysis (see Subsection 4.3).

4.1   Collecting Samples

There are various ways, how installable software can reach the mobile device: messaging, data network, Wi-Fi, local connections via Bluetooth, infrared, memory cards and PC connections. Only few of them are under the control of the network operator (messaging, data network) and are instantly ready to be analyzed. At the moment, this is an argument for solutions like local anti-virus software. But it is possible to send the software to the network for analysis before actually installing it. This needs time and will create costs, but it could be necessary to ensure security for mobile device users. And it can be expected, that data traffic will be more common in the future, or it could be possible not to bill certain traffic in connection with the security system. In order to minimize costs it could be useful to send a hash value of the software first to see if the network already knows this particular software.

Of the local connections, PC connections and memory cards require active involvement of the user. Infrared connections need intervisibility. So all of them can be expected to play only a minor role in mobile malware propagation. Only Bluetooth with a wireless

communication range of several meters has the potential of being a major source for malware propagation, even though the mobility of the devices could successfully prevent some infections [YFC+07].

To be informed about current malware spreading via Bluetooth, implementing a Bluetooth honeynet could be a good idea. Especially crowded areas promise to be a good source of malware [CMZ07]. When proactively analyzing samples, the analysis of a particular submitted software might already be finished, when the user submits a sample.

4.2   Analyzing Samples

The collected samples must now be analyzed. This part of the system can incorporate different analysis modules as its sources, for example an anti-virus product. Another example of an analysis module is MobileSandbox in Section 5. The optimal case is a complete analysis that can be rated against several security profiles like the user's personal security profile. An additional network-wide security profile can exist, where e.g. known malicious behavior like malicious premium rate services can be stored.

In the example of refusal of messaging except for the messaging program, every access to messaging functions is a violation of the profile and therefore malicious behavior.

4.3   Responding to the Analysis

The actions because of this violation may be different. The mobile network operator might choose to disallow the installation of the software, send a message to the user that the program violates the user's or the network's security profile, maybe even as a voice confirmation of the user, depending on how severe the damage is expected to be. From an operator's point of view, the proceeding of asking the user for confirmation and recording his reply could relieve the operator of refund claims.

5   MobileSandbox Dynamic Analysis Module

The task of MobileSandbox in the mobile dynamic malware analysis system is the actual analysis of the collected samples. It executes the sample in an environment (the sandbox), where it can watch the steps of the investigated sample. This results in a sequence of API calls that the program used during its execution.
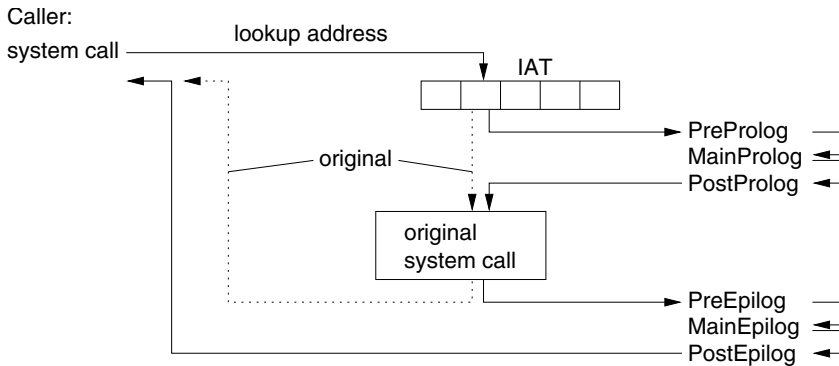
Figure 1: MobileSandbox

5.1 Import Address Table Patching

The concept of MobileSandbox is quite similar to CWSandbox, but it uses a different method to intercept the system calls. CWSandbox rewrites the first portion of the method in the DLLs. This is impossible in Windows Mobile because many DLLs are saved in read-only memory. It uses another standard method instead, patching of the import address table (IAT).

When an executable starts, the Windows loader looks up the addresses of each used system call and inserts them into the IAT, because these addresses are not known at compile time. MobileSandbox afterwards does some steps that lead to the setup of Figure 1. The address of every entry in the IAT is changed. For every changed address four functions are set up (`PreProlog`, `PostProlog`, `PreEpilog`, `PostEpilog`). They handle saving and restoring the current processor state and calling the two main functions. The IAT entry for each system call now points to its corresponding `PreProlog` function, which is the unique entry point for every system call.

When the sample uses a system call, its execution is wrapped by the `MainProlog` and `MainEpilog` functions. `MainProlog` is responsible for logging the system call to the host and to handle special system calls for creating processes and retrieving system call addresses. The parameters of `CreateProcess` need to be changed, so that the new process starts in suspended mode (see Subsection 5.2). Calls to `ShellExecuteEx` need to be simulated by `CreateProcess`, because `ShellExecuteEx` has no possibility to select suspended mode. A call to `GetProcAddress` is noted here for subsequent handling in `MainEpilog`.

`MainEpilog` first logs the system call's return value. Sometimes the return value needs to be changed. For example, if `GetProcAddress` was called to retrieve the address of a system call, its return value needs to point to the system call's custom `PreProlog` function which will be set up at the same time. In the case of `CreateProcess`, the DLL `MSandboxDLL` is injected into the new process.

The usefulness of logging parameters is limited when they are only pointers to data structures. This is especially true, if the results of the system call are transferred in a data structure referenced by a pointer in the parameters. MobileSandbox is able to dereference the pointers and additionally log the data structure when it is required. This is especially true for the mobile messaging methods.

Unfortunately, a malware sample is able circumvent the MobileSandbox method. A program does not need to use the IAT, but may calculate the system call address itself in advance. Whenever it wants to use a system call, it sets the address and sets the system into kernel mode. MobileSandbox is not able to log this event, because like other current sandbox solutions it has no access to the kernel structures at the moment. This extension is currently being developed. It seems to be possible to inject custom DLLs even in kernel processes using an undocumented debugging API. With the same technique as above, the kernel level calls can then be logged.

5.2   DLL Injection

The actual injection is based on the well documented DLL injection procedure on Win32 systems. But it needs a small adaption, because WinCE does not offer all the debug API methods that Win32 does (esp. `CreateRemoteThread` and `VirtualAllocEx`):

1. The sample is started in "suspended mode", i.e. the EXE is loaded into device memory but the main thread is not started.

2. MobileSandbox saves a part of the sample's program code and overwrites it with own instructions.

3. The context is changed with `SetThreadContext`, so that the PC register points to the custom code of MobileSandbox.

4. Now the sample's main thread is started. The custom code uses `LoadLibrary` to load `MSandboxDLL` into the sample's address space.

5. Finally, the sample's original state is restored and its main thread is started.

5.3   Communication to Host

An important requirement to ensure the integrity of the analysis is logging to a remote place rather than saving the log on the device only. MobileSandbox implements this communication of the device to the host system with a TCP connection over ActiveSync. The ActiveSync connection is established automatically, when a device is connected to the host via USB. The connection with the emulator is set up in the Device Emulator Manager. Both endpoints get an IP address, so TCP communication is possible.

The host system can use the Remote API (RAPI) to access the device. It is possible to perform file system operations or to start processes. After the successful injection of `MSandboxDLL` a TCP connection to the host system is established and every log entry is sent immediately.

5.4   Analysis

MobileSandbox can use the device emulator or an actual device. The emulator has two main advantages. First, restoring the original state is simple after a sample has been executed. It just needs restarting and restoring its directories on the host file system. Second, it is easily possible to execute the sample on a variety of different operating system versions, because the emulator uses original ROM images. It is just limited by the number of obtainable images and there is no need to handle with a large number of devices. And our experience shows, that within the scope of our work the emulator reacted like an actual device. Even though, one drawback of the emulator is the possibility that malware recognizes it and because of that does not show its malicious behavior. Another drawback is the limited network functionality for the messaging and phone APIs. However, this problem can be solved by adapting the parameters and return values of these functions in the `MainProlog` and `MainEpilog` methods.

A restriction of the current MobileSandbox implementation is its ability to only handle EXE files. In the context of an automatic analysis system, it should be able to handle installation archives too.

The output is an XML file with all information for further processing. Figure 2 shows how the log looks like for dereferenced pointers, with the `smsAddress` structure as an example.

```
SMS_ADDRESS smsAddress;
SmsGetSMSC(&smsAddress);

<api name="SmsGetSMSC" id="24" dll="SMS.dll" ordinal="0"
      argsavailable="true" returns="0">
<arg name="psmsaSMSCAddress" type="int" value="6712820"/>
<info name="psmsaSMSCAddress.smsatAddressType"
      value="SMSAT_INTERNATIONAL"/>
<info name="psmsaSMSCAddress.ptsAddress"
      value="14250010001"/>
<info name="ReturnValue" value="S_OK"/>
</api>
```

Figure 2: Example Analysis

6   Conclusion and Future Work

This paper described some differences between common personal computer security and mobile device security. Based on these differences, the framework of a mobile dynamic malware analysis system was proposed as a reasonable solution for the different environment that mobile devices like smartphones operate in. MobileSandbox was introduced as a module of the analysis system, that is able to analyze a run of executables for the Windows Mobile operating system.

Future work will comprise realizing the mobile dynamic malware analysis system, in which MobileSandbox can be embedded. A major task is collecting samples for analysis, for example, with a Bluetooth honeynet. MobileSandbox will be extended, on the one hand to log kernel level events. On the other hand, its capabilities will be extended to analyze more than one particular run of the sample. This will increase the size of the logs, so reducing the log complexity is another task for future work.

Acknowledgements

References

[BFL07]   Michael Becher, Felix C. Freiling, and Boris Leidner. On the Effort to Create Smartphone Worms in Windows Mobile. In *Information Assurance and Security Workshop, 2007*, pages 199–206, 20-22 June 2007.

[BMKK06] Ulrich Bayer, Andreas Moser, Christopher Kruegel, and Engin Kirda. Dynamic Analysis of Malicious Code - TTAnalyze. *Journal of Computer Virology*, 2006.

[CMZ07]   Luca Carettoni, Claudio Merloni, and Stefano Zanero. Studying Bluetooth Malware Propagation: The BlueBag Project. *IEEE Security and Privacy*, 5(2):17–25, 2007.

[FLJ+07]   Chris Fleizach, Michael Liljenstam, Per Johansson, Geoffrey Voelker, and András Mehés. Can you infect me now? Malware Propagation in Mobile Phone Networks. In *WORM*, 2007.

[GHW07]   Jan Goebel, Thorsten Holz, and Carsten Willems. Measurement and Analysis of Autonomous Spreading Malware in a University Environment. In Bernhard M. Hämmerli and Robin Sommer, editors, *DIMVA*, volume 4579 of *Lecture Notes in Computer Science*, pages 109–128. Springer, 2007.

[Gör06]   Stefan Görling. The Myth of User Education. In *Virus Bulletin Conference*, October 2006.

[Hyp06]   Mikko Hypponen. Malware goes Mobile. *Scientific American*, pages 70–77, 2006.

[MCDK06] Jose Andre Morales, Peter J. Clarke, Yi Deng, and B. M. Golam Kibria. Testing and evaluating virus detectors for handheld devices. *Journal in Computer Virology*, 2(2):135–147, 2006.

[MKK07a] Andreas Moser, Christopher Kruegel, and Engin Kirda. Exploring Multiple Execution Paths for Malware Analysis. In *SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 231–245, Washington, DC, USA, 2007. IEEE Computer Society.

[MKK07b] Andreas Moser, Christopher Kruegel, and Engin Kirda. Limits of Static Analysis for Malware Detection. In *Proc. Twenty-Third Annual Computer Security Applications Conference ACSAC 2007*, pages 421–430, 10–14 Dec. 2007.

[TH06] Sampo Töyssy and Marko Helenius. About malicious software in smartphones. *Journal in Computer Virology*, 2(2):109–119, 2006.

[WHF07] Carsten Willems, Thorsten Holz, and Felix Freiling. Toward Automated Dynamic Malware Analysis Using CWSandbox. *IEEE Security and Privacy*, 5(2):32–39, 2007.

[WT99] Alma Whitten and J. D. Tygar. Why Johnny can't encrypt: a usability evaluation of PGP 5.0. In *SSYM'99: Proceedings of the 8th conference on USENIX Security Symposium*, pages 14–14, Berkeley, CA, USA, 1999. USENIX Association.

[YFC+07] Guanhua Yan, Hector D. Flores, Leticia Cuellar, Nicolas Hengartner, Stephan Eidenbenz, and Vincent Vu. Bluetooth worm propagation: mobility pattern matters! In *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 32–44, New York, NY, USA, 2007. ACM Press.