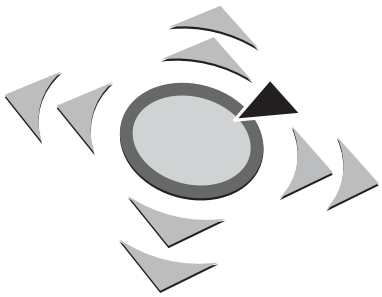
6. GI FG SIDAR Graduierten-Workshop über Reaktive Sicherheit

SPRING

Sebastian Uellenbeck (Hrsg.)

21.-22. März 2011, Bochum



SIDAR-Report SR-2011-01
ISSN 2190-846X

Vorwort

SPRING ist eine wissenschaftliche Veranstaltung im Bereich der Reaktiven Sicherheit, die Nachwuchswissenschaftlern die Möglichkeit bietet, Ergebnisse eigener Arbeiten zu präsentieren und dabei Kontakte über die eigene Universität hinaus zu knüpfen. SPRING ist eine zentrale Aktivität der GI-Fachgruppe SIDAR, die von der organisatorischen Fachgruppenarbeit getrennt stattfindet. Die Veranstaltung dauert inklusive An- und Abreise zwei Tage und es werden keine Gebühren für die Teilnahme erhoben. SPRING findet ein- bis zweimal im Jahr statt. Die Einladungen werden über die Mailingliste der Fachgruppe bekanntgegeben. Interessierte werden gebeten, sich dort einzutragen (`http://www.gi-fg-sidar.de/list.html`). Für Belange der Veranstaltung SPRING ist Ulrich Flegel (Hochschule Offenburg) Ansprechpartner innerhalb der Fachgruppe SIDAR.

Nach der Premiere in Berlin fand SPRING in Dortmund, Mannheim, Stuttgart und Bonn statt. Die Vorträge deckten ein breites Spektrum ab, von noch laufenden Projekten, die ggf. erstmals einem breiteren Publikum vorgestellt werden, bis zu abgeschlossenen Forschungsarbeiten, die zeitnah auch auf Konferenzen präsentiert wurden bzw. werden sollen oder einen Schwerpunkt der eigenen Abschlußarbeit oder Dissertation bilden. Die zugehörigen Abstracts sind in diesem technischen Bericht zusammengefaßt und wurden über die Universitätsbibliothek Dortmund elektronisch, zitierfähig und recherchierbar veröffentlicht. Der Bericht ist ebenfalls über das Internet-Portal der Fachgruppe SIDAR zugänglich (`http://www.gi-fg-sidar.de/`). In dieser Ausgabe finden sich Beiträge zur den folgenden Themen: Malwareanalyse, Betriebssystemsicherheit, Informationsfreiheit, Forensik und Smartphone-Security.

Besonderer Dank gebührt Christopher Wolf, Ulrich Flegel und Michael Meier für ihre Unterstützung bei der Planung, sowie Johannes Hoffmann und Ralf Hund bei der Durchführung der Veranstaltung.

Bochum, März 2011 Sebastian Uellenbeck

Contents

Diesen Bericht zitieren als:
Sebastian Uellenbeck, editor. Proceedings of the Sixth GI SIG SIDAR Graduate Workshop on Reactive Security (SPRING). Technical Report SR-2011-01, GI FG SIDAR, Bochum, März 2011,

Beiträge zitieren als:
Autor. Titel. In Sebastian Uellenbeck, editor, Proceedings of the Sixth GI SIG SIDAR Graduate Workshop on Reactive Security (SPRING). Technical Report SR-2011-01, page xx. GI FG SIDAR, Bochum, März 2011.

Verschleiernde Transformationen von Programmen

Michael Rex

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl VI, Informationssysteme und Sicherheit (ISSI)
michael.rex{at}tu-dortmund.de

Bösartige Programme, sogenannte Malware, stellen in der heutigen Zeit ein großes Problem dar. Die umfassende Vernetzung von Computern erlaubt es Malware, sich auf einfache Weise an eine Vielzahl potentieller Opfer zu verbreiten. Anti-Virus-Software ist deshalb für viele Computer eine Grundvoraussetzung zum sicheren Betrieb.

Der Einsatz von Anti-Virus-Software zwingt Malware-Autoren dazu, Strategien zu entwickeln, wie sich ihre Programme vor Anti-Virus-Software verbergen können, um sich weiterhin zu verbreiten. Auf der anderen Seite sind die Hersteller von Anti-Virus-Software gezwungen, Mittel gegen diese Strategien zu finden, um Malware weiterhin erkennen zu können. Dies führt zu einer Situation, die bildhaft als Wettrüsten bezeichnet wird, in der beide Seiten versuchen, durch immer ausgefeiltere Strategien die Bemühungen der jeweils anderen Seite zunichte zu machen.

In diesem Kontext wurden verschleiernde Transformationen von Programmen im Rahmen einer Diplomarbeit untersucht. Solche Transformationen werden von Malware-Autoren eingesetzt, um die Erkennung ihrer Programme zu verhindern oder zumindest zu erschweren [1, 2]. Es werden die *Komprimierung*, *Verschlüsselung*, *metamorphe Codeumformung* und *Virtualisierung* von Programmen betrachtet sowie hinsichtlich ihres Einflusses auf die Merkmale *Erscheinungsbild*, *Kontrollflussgraph*, *Verhalten* und *Funktionalität* eines Programms kategorisiert. Im Rahmen eines Experiments, in dem der Einfluss der Transformationen auf die Fähigkeit von Anti-Virus-Software, Malware zu erkennen, untersucht wurde, wurden mit Hilfe einer Experimentierumgebung mit fünf Packern weitgehend automatisiert Varianten von Malware-Samples erzeugt, welche die vorgestellten Transformationen abdecken. Mit Hilfe dieser transformierten Samples wurde dann die Erkennungsleistung von Anti-Virus-Software untersucht. Dieses Experiment zeigte, dass bereits einfache Transformationen wie die Komprimierung sich negativ auf die Erkennungsleistung von Anti-Virus-Software auswirken. Allerdings zeigte sich auch, dass das Laufzeitverhalten von Malware, auf der Ebene der Systemaufrufe betrachtet, gegen die Transformationen der im Experiment eingesetzten Programme resistent ist.

Literatur

[1] John Aycock. *Computer Viruses and Malware.* Springer, 2006.

[2] Peter Szor. *The Art of Computer Virus Research and Defense.* Addison Wesley, 2005.

Erkennung von bösartigen Netzwerkverbindungen mittels Verhaltensgraphenanalyse

Ralf Hund
Arbeitsgruppe Embedded Malware
Ruhr-Universität Bochum
ralf.hund{at}rub.de

Sogenannte *Bots* stellen seit Jahren eine erhebliche Gefahr für die Sicherheit des Internets dar. Sie unterscheiden von anderen Malwaretypen dadurch, dass sie in der Lage sind, eine dedizierte Netzwerkverbindung zu einem zentralen *Command & Control* (C&C) Server aufzubauen. Mittels dieser Verbindungen erhält der Bot zum Beispiel weitere Kommandos (Informationen über Updates) oder überträgt auf dem Rechner gesammelte Daten (Passwörter, Netzwerkdumps, etc.). Ein Bot alleine tätigt mittlerweile jedoch nicht nur solche C&C Verbindungen, sondern baut auch eine Vielzahl von „normalen“ Verbindungen auf, um unauffällig zu bleiben. Eine Unterscheidung von C&C Verbindungen von normalem Traffic ist hierbei einerseits für die weitere Analyse wichtig (zum Beispiel um Netzwerksignaturen zu erstellen), dient aber andererseits auch der generischen Erkennung von C&C Servern (zum Beispiel für Blacklisten).

Ein vielversprechender Ansatz um dieses Ziel zu verwirklichen ist die sogenannte *Verhaltensgraphenanalyse*. Hierbei wird das Bot-Binary in einer kontrollierten Umgebung (Sandbox) ausgeführt und beobachtet. In unserem Falle protokolliert die Sandbox die Systemaufrufe des Bots und markiert entsprechende Parameter und Rückgabewerte mit *Taintflags*. Somit ist es möglich, Beziehungen zwischen den Eingabeparametern verschiedener Systemaufrufe zu konstruieren. Diese Beziehungen werden anschließend im *Verhaltensgraphen* dargestellt. Konkret lässt sich damit zum Beispiel nachvollziehen, ob die Rückgabewerte einer Funktion (die zum Beispiel einen bestimmten Registry-Wert ausliest) über das Netzwerk nach außen versendet wurden.

Mit Hilfe der konstruierten Verhaltensgraphen man nun auf vielfältige Art und Weise gut- von bösartigen Verbindungen unterscheiden. Ein Ansatz besteht darin, eine Reihe von typischen Verhaltensmuster von C&C Verbindungen vorzugeben und diese dann anhand der Verhaltensgraphen zu finden. Ein typisches Verhaltensmuster ist zum Beispiel das Ausführen von vormals über das Netzwerk empfangenen Daten. Dies kann während einer Updateroutine des Bots geschehen.

Alternativ können auch Methoden aus dem Bereich des Data Mining angewandt werden. Eine Möglichkeit besteht darin, ein System automatisiert mit eine Menge von bekannten gutartigen und C&C Traffic zu füttern und mittels Graphanalyse bestimmte Merkmale letzterer Verbindungen zu identifizieren. Anschließend können die Ergebnisse weiter abstrahiert werden, so dass am Ende automatisiert Templates generiert werden, welche bestimmte Arten von C&C Verbindungen erkennen.

SEODisc: Ansatz zur Erkennung von SEO-Attacken

Matthias Meyer

Technische Universität Dortmund
matthias.meyer{at}tu-dortmund.de

Die Verbreitung von Malware im Internet geschieht über vielfältige Wege. Immer häufiger wird sie über Webseiten verbreitet, die den Benutzer mittels Fake-AV Kampagnen überzeugen, die als Virenscanner getarnte Malware auf dem eigenen Rechner zu installieren. Alternativ werden auch PHP-Kits eingesetzt, die die Schwachstellen des verwendeten Browsers erkennen und sie gezielt ausnutzen. Den oben genannten Infektionsvektoren gemein ist die Tatsache, dass möglichst viele potentielle Opfer auf die manipulierte Webseite gelockt werden müssen.

Zu diesem Zweck wird Black-Hat Search Engine Optimization (Black-Hat SEO) eingesetzt, um zu bestimmten Schlagworten manipulierte Webseiten in den Top Suchergebnissen bekannter Internetsuchmaschinen zu platzieren. Aufgrund der Popularität der Suchmaschine Google wird häufig auf diese optimiert. Um eine Platzierung in den Top 20 Suchergebnissen zu bestimmten Schlagwörtern zu erzielen, setzen Black-Hat SEOs unlautere Methoden wie Link-Spam [1] ein. Dabei werden Netzstrukturen aufgebaut, welche gezielt die Bewertungsalgorithmen der Suchmaschinen-Spider überlisten. Diese Netzstrukturen (SEO-Netze) bestehen aus der Verlinkung mehrerer Webseiten untereinander und lassen sich als gerichtete Graphen darstellen. Der Aufbau dieser SEO-Netze kann dabei teilweise automatisiert durch SEO-Kits erfolgen. Grundlage für den Erfolg dieser Methode besteht in der Unentdecktheit der Manipulation. Hierfür setzen Angreifer gezielt Cloaking Techniken ein. Eine der wichtigsten Methoden ist hierbei die Unterscheidung zwischen menschlichen Besuchern einer Webseite und Suchmaschinen-Spidern. Es wird unter anderem der in einer HTTP Anfrage mit übermittelte User Agent ausgewertet. In [2] wird das SEODisc System vorgestellt.

In diesem System erfolgt zunächst eine Kandidatengenerierung mittels Google Trends, um somit die aktuell trendigen Webseiten im Internet zu analysieren. Bei einer erfolgreichen SEO-Attacke ist die Wahrscheinlichkeit hoch, dass manipulierte Webseiten in den Trends enthalten sind.

Anschließend werden die trendigen Webseiten mittels verschiedener emulierter User Agents abgefragt und die Unterschiede in den Webseiten analysiert. Dabei werden gezielt differierende Links extrahiert (Δ-Links), um mithilfe dieser auffällige Netzstrukturen zu erkennen. Innerhalb dieser SEO-Netze wird weiterhin versucht Webseiten zu identifizieren, die durch das SEO-Netz gefördert werden, um diese einer gezielten Analyse durch spezialisierte Analysesoftware wie MonkeyWrench [3] genauer zu betrachten.

Literatur

[1] Zoltán Gyöngyi und Hector Garcia-Molina: *Link Spam Alliances*, In: *31st International Conference on Very Large Data Bases (VLDB 2005)*, 2005.

[2] Matthias Meyer: *Erkennung bösartiger Webseiten durch Analyse SEO vergifteter Suchmaschinenergebnisse*, Diplomarbeit, Technische Universität Dortmund, Januar 2011.

[3] Armin Büscher, Michael Meier und Ralf Benzmüller: *MonkeyWrench - Bösartige Webseiten in die Zange genommen.* In: *Sichere Wege in der vernetzten Welt - Tagungsband zum 11. Deutschen IT-Sicherheitskongress*, S. 459-472, 2009.

Evaluating "Ring -3" Rootkits

Patrick Stewin

Security in Telecommunications
Technische Universität Berlin, D-10587 Berlin, Germany
patrickx{at}sec.t-labs.tu-berlin.de

In 2009, security researchers discovered a new, very powerful rootkit environment on x86 platforms [1]. That environment is based on *Intel's Active Management Technology (iAMT)* [2], which is completely isolated from the host. One part of iAMT is implemented as an embedded μ-controller in the platform's memory controller hub. That μ-controller is called *Manageability Engine (ME)* and includes a processor (ARCtangent-A4), read-only memory (ROM), static random access memory (SRAM) and direct memory access (DMA) engines. Furthermore, iAMT provides an isolated network channel (out-of-band (OOB) communication). To illustrate the power of the stealth enviroment, [1] called the iAMT environment in conjunction with rootkits "ring -3", following the x86 ring protection model. For our evaluation we implemented a prototype in form of a USB keyboard keystroke logger [3].[1] Since we were unable to get an Intel developer board providing the "ring -3" environment, we had to use the exploit discovered by [1] to infiltrate our target platform. We monitor the keyboard buffer of the Linux based target platform via DMA. To find the physical address of the keyboard buffer we apply a search algorithm, that finds the USB product string and follows some pointers to the structure containing the buffer address. To exfiltrate captured keystrokes our prototype uses iAMT's OOB communication capabilities.

[1] discussed countermeasures against "ring -3" rootkits, but they also provide approaches to defeat such countermeasures. Furthermore, it is doubtful if all the proposed countermeasures can be applied in practice.[2] The goal of our evaluation is to find a reliable detection mechanism for "ring -3" rootkits. We assume that we can provoke delays when accessing the same resources as our prototype. For example, our prototype has to scan the host memory to find certain data structures and it also has use the network interface card to send keystroke codes. Another possibility is to initiate various DMA transfers using multiple devices. Only one device can be the bus master at a certain point in time. The next step is to design an experimental set-up that allows the measurement of delays and finally derive a reliable detection mechanism for "ring -3" rootkits.

References

[1] A. Tereshkin and R. Wojtczuk, "Introducing Ring -3 Rootkits," Black Hat USA, Jul. 2009. [Online]. Available: `http://www.blackhat.com/presentations/bh-usa-09/TERESHKIN/BHUSA09-Tereshkin-Ring3Rootkit-SLIDES.pdf`

[2] Kumar, A., Goel, P., Saint-Hilaire, Y.: Active Platform Management Demystified - Unleashing the power of Intel vPro Technology. Intel Press (2009)

[3] Stewin, P., Seifert, J.-P.: "In God We Trust All Others We Monitor" [Extended Abstract]. In: CCS '10: Proceedings of the 17th ACM Conference on Computer and Communications Security. ACM, p.639-641. (2010). Online verfÃ¼gbar: `http://portal.acm.org/ft_gateway.cfm?id=1866381&type=pdf&CFID=6743120&CFTOKEN=21999560`

[1] The rootkit provided by [1] is not suited for an evaluation: it reveals itself, does not work permanently, is unable to read from host memory and does not provide any network capabilities.

[2] Note: Intel fixed the bug responsible for the exploit discovered by [1], i.e., the known exploit cannot be used to infiltrate a patched x86 based target platform anymore.

OS Agnostic Sandboxing Using Virtual CPUs

Matthias Lange

Security in Telecommunications
Technische Universität Berlin, D-10587 Berlin
mlange {at} sec.t-labs.tu-berlin.de

Commodity operating systems have a poor track record when it comes to security. Malware and viruses aim at exploiting vulnerabilities and e.g. try to steal the users private data. Unfortunately most of todays operating systems do not allow to enforce the principle of least authority as their security mechanisms are to coarse grained.

In this work we show how we built an OS agnostic sandbox using virtual CPUs. It allows the execution of native code and thus does not wear the burden of an inherent performance penalty. To show the efficiency and usability of our solution we have built a secure execution container for web browser plugins.

Background Sandboxing techniques were developed to jail a program into a restricted execution environment [WLAG93]. Per default critical operations (e.g. syscalls) are disallowed and as such trap into the sandbox host. There for example arguments of the critical operation can be inspected and sanitized. Although e.g. the Java VM provides a restricted execution environment it is often disliked due to its performance penalty. Other sandboxing techniques like Googles Native Client (NaCl) rely on specific characteristics of the underlying platform.

Design The virtual CPU model (vCPU) is an execution abstraction that strongly resembles to physical CPUs [LWP2010]. It features upcalls, virtual interrupts, a state indicator and a state save area. The vCPU model allows for sequential execution while supporting control flow diversions upon events. We chose to base our architecture on the vCPU model as it allows the traditional thread model to be easily combined with an asynchronous event model.

Our architecture is designed to maximize data throughput with a zero-copy shared-memory interface between a sandboxed client and the host. The computational overhead is minimized by allowing native execution. To minimize event latency we designed an efficient event notification mechanism using event buffers.

First Results Our vCPU model is implemented using ptrace which allows the client to run natively. The measurements show that sandboxed clients perform comparable to native performance. The event latency is constant and does not depend on the number of concurrently running vCPU threads. Preliminary measurements indicate that the data throughput is sufficient for multimedia applications.

References

[LWP2010] Adam Lackorzynski, Alexander Warg, and Michael Peter. Generic virtualization with virtual processors 2010, Twelvth Realtime Linux Workshop

[WLAG93] Robert Wahbe, Steven Lucco, Thomas E. Anderson, and Susan L. Graham. Efficient software-based fault isolation 1993, ACM symposium on Operating systems principles

Practical P2P-Based Censorship Resistance

Benjamin Michéle

Technische Universität Berlin
ben{at}sec.t-labs.tu-berlin.de

People around the world are using the Internet to access news, to publish information, and to organize themselves. Recently, web services such as Facebook and Twitter have been used to organize peaceful demonstrations against totalitarian leaders, forcing them to resign and even leave the country. However, these regimes are aware of the power given to the people by the Internet and are therefore increasingly limiting access to these services [1]. Cutting off the Internet entirely is an option that is used only seldomly, as it severely impacts the country's economy.

The New York Times recently published an article [2] on a new US State Department policy that plans to support Internet freedom by financing various projects. Possible candidates for government support are projects like UltraSurf or TOR. UltraSurf along with many others, however, is proprietary and not well suited to serve a large amount of user requests due to a client/server based architecture. TOR on the other hand is open source and has a long history in providing anonymity to Internet users. However, its client/server approach has two drawbacks: Poor scalability and weak censorship resistance. Regarding these issues, there is active research and development improving TOR. Nevertheless, TOR was built with anonymity in mind and not censorship resistance.

In this work we propose a new P2P-based approach focusing on:

Censorship Resistance Our approach is entirely P2P-based, eliminating the need for central servers and therefore single points of failures. Participating nodes use a distributed hash table (DHT) to locate each other and necessary cryptographic certificates. Trusted peers can be used to detect attacks. Peer communication is normalized using SSL to impede traffic analysis.

Low Operator Risk One of TOR's strengths is at the same time a weak point: TOR servers can be used for a wide variety of TCP applications, with only a port-based filter built in. Running a TOR exit node can therefore have legal consequences for server operators. We propose a very light-weight approach allowing only HTTP traffic to a small selection of web sites that are legal in the operator's country.

Scalability Every user of the network offers the service to others, as well. This approach scales well and at the same time complicates IP-based censorship efforts.

Other success factors are ease-of-use and trustworthiness. All of these factors are addressed by our prototype implementation, which is being developed as an open source Firefox plugin.

References

[1] Reporters without Borders: *Internet censorship and attacks on journalists amid major street protests*, `http://en.rsf.org/egypt-internet-censorship-and-attacks-on-26-01-2011,39400.html`, January 26, 2011

[2] New York Times: *U.S. Policy to Address Internet Freedom*, `http://www.nytimes.com/2011/02/15/world/15clinton.html`, February 14, 2011

Tools and Processes for Forensic Analyses of Smartphones and Mobile Malware

Michael Spreitzenbarth

University of Erlangen-Nuremberg
spreitzenbarth{at}informatik.uni-mannheim.de

Malware is defined as computer programs that are used by an attacker to execute malicious code on the computer of a victim. In today's Internet malware constitutes a major problem and effective safety measures against this harassment are necessary. This problem looms as a new and future threat to smartphones, too. They contain many information which are of great interest for attackers. Several hundred different versions of malware for this type of device have already been noticed and it is expected that this number will increase even further within next years. Thus, effective and efficient protection measures against malware on mobile devices (mobile malware) become necessary, in order to have procedures for detecting and repelling these threats right from the beginning. Moreover, todays there is almost no criminal action in which information technology does not play a role. Increasingly, mobile devices become an object of investigation in the context of crime detection. Due to this reason two major research aspects have been defined within the scope of a BMBF project named 'MobWorm':

Automated Malware Analyses: In the scope of this question a prototype will be further developed. Therefore it is investigated which information from a mobile sandbox need to be collected. Afterwards, the corresponding implementation is executed. Moreover, methods are investigated, in how far the mobile sandbox may be used as a security measure, e.g. as a reference monitor for downloaded applications. Here, the mobile sandbox monitors activities of a program and terminates it directly if an unauthorized sequence of action occurs (e.g. the opening of a permitted network connection or the dialing of an expensive service number).

Mobile Phone Forensics: Within the frame of this research question we develop several methods to conduct forensic analysis on smart phones. In this context a major focus is put on Googles Android platform. In a first step various methods are researched how to create a memory dump of a mobile phone (e.g. with the help of Twister-Box, via JTAG or with specific software). These are documented in forensic processes, i.e. in detailed and exact activity rules. In a second step the methods for analyzing memory dumps are developed. As a result the usability and effectiveness of standard procedures like file carving and hash-value databases in the area of mobile phones should be investigated. The focus of the application examples is always put to the corresponding investigation of malware-infections. The methods and tools developed within the scope of this research question are intended to be an addition to already existing propriety systems and their functions which are often not well documented. With respect to the development we put great emphasis on the compliance with forensic principles and we gear to scientific standards in this area of research. The developed prototype as well as the fundamental research is important in order to understand the behavior of mobile devices and software in a detailed way in terms of malware analysis.

Security Aspects of Piecewise Hashing in Computer Forensics

Frank Breitinger* and Harald Baier**

* Hochschule Darmstadt
D-64295 Darmstadt, Germany
F.Breitinger{at}gmx.de

**Center for Advanced Security Research Darmstadt
D-64295 Darmstadt, Germany
harald.baier{at}cased.de

Hash functions are widely spread in computer science and used to map arbitrary large data to bit strings of a fixed length called a fingerprint. Cryptographic hash functions like SHA-1 or MD5 are established in various fields, but have one 'drawback' due to several security requirements: if one bit of the input is changed, this results in a completely different fingerprint.

In computer forensics cryptographic hashing represents the backbone in identifying files in sets of known-to-be-good (known as whitelisting) or known-to-be-bad files (known as blacklisting). Therefore, most investigators use hash databases, which are either created by themselves or maintained by institutions. Once they are in possession of such a database, the processing is straightforward: take a storage medium, compute all hash values, and perform look ups in the database.

An active adversary, however, can destroy this straightforward proceeding by changing one bit of each file. Conditioned by the pseudo-randomness of the cryptographic hash function, this leads to completely different hash values. A 'similarity hash function' for files, i.e. a fuzzy-hashing function, would be the perfect counterpart for this attack.

In 2006, Jesse Kornblum presented an implementation for this idea in [Kor06] named context triggered piecewise hashing (CTPH). Since then, he has been quoted numerous times (e.g. [Hur09]) and his approach has been improved with respect to performance (e.g. [SLC09]). However, there is no security analysis in terms of the reliability of the results.

In our research, we address some security aspects of CTPH:

1. Even if two files have absolutely different content, we show how to efficiently achieve high match scores of the respective CTPH values of both files. This approach may be used to circumvent CTPH whitelisting.

2. We show how to efficiently manipulate a file by a fixed number of modification. The CTPH of the manipulated version of the file differs significantly from the CTPH of the original, although both versions are perceptually identical. This approach may be used to circumvent CTPH blacklisting.

Besides these two anti-forensic issues, we also discovered weaknesses in the randomness of Kornblum's pseudo-random function. We showed how to improve Kornblum's pseudo-random function with respect to efficiency and randomness.

References

[Kor06] Kornblum, Jesse: Identifying almost identical files using context triggered piecewise hashing. In: *Digital Investigation* Bd. 3S, 2006, 91-97

[Hur09] Hurlbut, Dustin: Fuzzy Hashing for Digital Forensic Investigators / AccessData. Vers: Jan 09. `http://www.accessdata.com/downloads/media/Fuzzy\_Hashing\_for\_Investigators.pdf`.

[SLC09] Seo, K. ; Lim, K. ; Choi, J. ; Chang, K. ; Lee, S.: Detecting Similar Files Based on Hash and Statistical Analysis for Digital Forensic Investigation. In: *Computer Science and its Applications, 2009. CSA 09*, 2009, S. 1–6

Antiforensik auf mobilen Endgeräten

Stefan Lambertz
FH Aachen
D-52066 Aachen
Stefan.Lambertz{at}alumni.fh-aachen.de

Forensische Methoden werden immer öfter herangezogen, um Straftaten mit oder auf Computern aufzuklären. Computer durchdringen immer mehr unser alltägliches Leben und so gewinnt die Computerforensik eine immer höhere Bedeutung. Die angewendeten forensischen Methoden sind zum Teil gut dokumentiert und es gibt auf dem Markt nur eine sehr begrenzte Anzahl von forensischen Analyse Tools. Der technisch versierte Straftäter kann sich also über die Methoden der Strafverfolgung informieren und diese nachvollziehen. Mit diesem Wissen kann der Angreifer Methoden ersinnen, die es dem Ermittler erschweren oder unmöglich machen, den Täter zu finden oder überhaupt eine Tat nachzuweisen. Die Methoden, mit denen man versucht eine forensische Untersuchung zu erschweren oder die Spuren einer Tat zu verwischen, fasst man unter dem Begriff Antiforensik zusammen.
Durch diesen Sachverhalt haben die Täter einen Wissensvorsprung. Ein Reaktion auf neue Verschleierungsmethoden kann erst erfolgen wenn sie das erste Mal entdeckt wurden. Ein Anpassen der Methoden und der Tools kann dann für viele Straftaten zu viel Zeit in Anspruch nehmen. Daher ist es sinnvoll, die Entwicklungen der Antiforensik vorwegzunehmen, also selber in diesem Bereich zu forschen, und die forensischen Methoden und Tools auf die möglichen Fortschritte der Antiforensik anzupassen bevor diese das erste Mal außerhalb eines Labors auftreten. Dies kann zwar nicht allen antiforensischen Methoden entgegenwirken, jedoch gehen Entwicklungen oft in die selbe Richtung. Weiterhin ist eine Betrachtung der Forensik aus anderer Sicht sicherlich auch für die Forensik hilfreich. Nicht zuletzt gibt es auch Personen, die ein berechtigtes legales Interesse daran haben, dass ihre Geräte, und damit ihre Daten, einer forensischen Analyse gegnerischer Kräfte widerstehen.
Beim Einsatz antiforensischer Methoden müssen nicht alle Spuren hundertprozentig gelöscht werden. Da Ermittler nur über begrenzte Ressourcen, wie Zeit, Mitarbeiter und Tools verfügen, reicht es aus, wenn das Auffinden der Spuren mehr Ressourcen in Anspruch nehmen würde, als den Ermittlern zur Verfügung steht. Die Antiforensik findet zumeist in zwei Feldern Anwendung. Zum Einen wird sie genutzt, um belastendes Material, das auf dem Rechner vorliegt, zu verstecken, zum Anderen wird sie genutzt, um Angriffe auf ein System oder kompromittierte Dateien zu verschleiern.
Moderne Smartphones nähren sich in ihrem Funktionsumfang immer mehr stationären Computern an, sie sind jedoch stark proprietäre Systeme, die einer forensischen Untersuchung entgegenstehen. Durch ihre Mobilität und den mit ihnen gepflegten Umgang speichern sie, zum Einen Daten die sich auf Computern nicht finden lassen, zum Anderen speichern sie sehr sensible Daten.
Ich habe am Beispiel des iPhone 4 untersucht welche Daten sich auf einem Smartphone finden lassen. Diese Daten sollen von antiforensichen Methoden so versteckt werden, dass sie durch bekannte forensische Methoden nicht mehr gefunden werden können. Daraufhin habe ich verschiedene antiforensiche Methoden zusammengetragen und ihre Einsetzbarkeit auf dem iPhone betrachtet, dabei wurden auch auf die vom Hersteller implementierten Funktionen zum Schutz der Nutzerdaten zurückgegriffen. Durch eine selbst geschriebene Applikation habe ich ausgewählte Nutzerdaten, wie Kontakte, Kalender und Notizen vor gebräuchlichen forensischen Tools versteckt.

Smartphone Honeypots

Collin Mulliner

Security in Telecommunications
Technische Universität Berlin, D-10587 Berlin
collin{at}sec.t-labs.tu-berlin.de

Mobile and smartphone security is a fast moving field. New vulnerabilities and resulting attacks need to be detected and analyzed as fast as possible. Unfortunately the attacker side is always a step ahead. To catch both, vulnerabilities and attacks, we aim to apply the technique of *honeypots* to the area of smartphones. For regular computer systems this has been done on large scale by [HP].

Honeypots: A honeypot is computer system that is meant to be attacked in order to study the attacker's behavior during and after the attack. Honeypots have been created in many different flavors. From single computer to whole networks of fake machines – called honeynets.

We determined multiple challenges while setting up a smartphone-honeypot:

`System Setup`: How to build an actually smartphone honeypot system. ¿From real devices to development-emulators and maybe complete simulation [P04]. This largely depends on the OS we want to run as a honeypot and on the communication types we want to support. Compared to regular computers we have additional hardware and software capabilities that need to be present or simulated.

`Monitoring`: Monitoring the honeypot is one of the essential parts. The honeypot is only useful if we can exactly determine what the attacker is doing. Depending on the system setup monitoring can be highly complicated.

`Containment`: After compromise of the honeypot we need to make sure that the attacker can not use the honeypot for carrying out attacks. Furthermore, the honeypot should not be abused for fraud such as premium SMS/calls.

`Visibility`: To make the honeypot useful it needs to be visible for attackers. This can happen in many ways such as publishing the phone number, email address, instant messaging account name and a like in as many ways a possible. The honeypot then needs to inspect message content and and such to e.g. open links contained in them in order to get infected.

References

[P04] Niels Provos: *A Virtual Honeypot Framework* In *13th USENIX Security Symposium* (San Diego, CA, August 2004.)

[HP] The Honeynet Project: *http://www.honeynet.org*

Taming the Robot: Efficient Sand-boxing of the Android OS

Steffen Liebergeld

Security in Telecommunications
Technische Universität Berlin, D-10587 Berlin
steffen{at}sec.t-labs.tu-berlin.de

Smart phones contain a lot of private information, such as contact lists, email, SMS and the browsing history. Their operating systems are based on traditional desktop operating systems and have all the attack vectors known from desktop computers, but typically employ less means of defence. As smart phones become more and more wide spread they are increasingly targeted by attackers, setting private user data at risk [1]. A compromised smart phone may cost money if the attacker is able to send premium SMS without the user's consent. The attacker may also target the carrier with denial of service attacks through bot-nets built out of smart phones.

In our research we focus on a secure smart phone architecture. Our architecture allows the reuse of existing software, and enables us to integrate components that require high security.

Challenges Smart phone operating systems such as iOS and Android are based on traditional desktop operating system kernels. As such, any flaw in the kernel can allow an adversary to take control of the phone. Due to the kernels complexity, such flaws are common. A recent study found 88 exploitable bugs in the Android kernel [2]. Typical countermeasures known from desktop computers such as virus scanners and firewalls are not applicable to smart phones due to their limited performance and power envelope. Security updates are installed less frequently which increases the time until a vulnerability is fixed. Securing the phone by disabling functionality, e.g. disabling the camera or disallowing installation of custom applications, is not possible because it would severely cut down on features, which appeal to the normal user.

Approach Our secure system is based on a microkernel, which provides fine grained resource access control and strict isolation between components. However, due to timing and financial constraints, creating a secure operating system and a healthy ecosystem (app store, developer community) from scratch is not feasible. Thus, we employ virtualization technology to run Android in a sand-box on top of the micro kernel. This enables us to run high-security applications such as a cryptographic key store side-by-side with Android in a secure way. A compromised Android does not allow an adversary to compromise high security applications to, for example, steal the user's private keys. An interesting scenario features two instances of Android in parallel, in different isolated partitions. One partition runs a hardened business Android that, for example, does not allow the user to install applications. The other partition runs an unrestricted Android to be used as the user's private environment.

References

[1] Vincenzo Iozzo and Philipp Weinmann: *iPhone Safari vulnerability allowed to steal the SMS database http://www.iphonehacks.com/2010/03/iphone-hacked-via-mobile-safari-exploit-at-pwn2own-hijacks-sms-database.html*

[2] Coverity Report 2010 *http://www.coverity.com/scan_android*

Android Sicherheit
Überprüfung und Gewährung von Rechten

Daniel Bußmeyer

Ruhr-Universität Bochum
D-44801 Bochum
daniel.bussmeyer{at}rub.de

Durch die steigende Leistung der Mobilfunkgeräte finden darauf immer komplexere Applikationen Anwendung. Diese beinhalten immer sensitivere Daten, wie z.B. E-Mail-Konten, Exchange-Zugänge, VPN-Zugänge, etc. Dadurch wird es zunehmend wichtiger die Mobilfunkgeräte besonders zu schützen.
Android verfolgt dabei ein Konzept aus einer Mischung des Linux-Rechtesystems und eines darauf aufsetzenden eigenen Verfahrens. Dabei sind prinzipiell Zugriffe auf Hardware und auf Inhalte zu unterscheiden. Zum einen werden I/O- und Hardware-Zugriffe aufgrund von Zugehörigkeit zu Benutzergruppen aufgelöst, wohingegen Zugriffe auf Kontakte oder SMS durch einen System-Service realisiert werden. Dieser Service überprüft dann, ob eine Applikation die angeforderten Rechte zugeteilt bekommt.
Festgelegt werden die Rechte, die eine Applikationen für sich beansprucht, zur Installationszeit. Diese Rechte werden separat vor der Installation angezeigt, wobei einige hervorgehoben werden, um den Benutzer auf evtl. Gefahren hinzuweisen (Zugriff auf Kontaktdaten, SMS, etc.). Danach hat ein Benutzer die Möglichkeit zu entscheiden, ob er der Software alle geforderten Rechte erlaubt oder das Programm nicht installiert.
Aus den diversen Überprüfungsroutinen des Rechtesystems, ergeben sich nun verschiedene Möglichkeiten, Zugriff auf Daten oder Systemkomponenten zu bekommen, ohne diesen vorher bei der Installation explizit zugeteilt bekommen zu haben.
Durch das Ausnutzen der unterschiedlichen Funktionen der Rechteüberprüfung innerhalb Androids, ist es so möglich, unberechtigt Informationen eines Geräts auszulesen oder diese an einen entfernten Server zu schicken. Es können auch Daten auf dem Gerät ungeachtet vom Benutzer manipuliert und verändert werden, bis zur Kontrolle des Geräts durch potentielle Angreifer.

Literatur

[BHUS10] Anthony Lineberry, David Luke Richardson und Tim Wyatt (Hrsg.): *These aren't the permissions you're looking for (BlackHat USA 2010)*