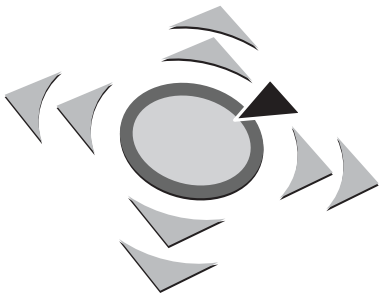
5. GI FG SIDAR Graduierten-Workshop über Reaktive Sicherheit

SPRING

Sebastian Schmerl, Simon Hunke (Hrsg.)

07. Juli 2010, Bonn



SIDAR-Report SR-2010-01

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 einen Tag 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 (SAP Research Center Karlsruhe) Ansprechpartner innerhalb der Fachgruppe SIDAR.

Nach der Premiere in Berlin fand SPRING in Dortmund, Mannheim und Stuttgart 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 Diplomarbeit 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: Intrusion Detection, Malware, Mobile Systeme und Bot-Netze.

Wir danken allen, die mitgeholfen haben.

Bonn, Juli 2010 — Sebastian Schmerl, Simon Hunke

Contents

Diesen Bericht zitieren als:
Sebastian Schmerl, Simon Hunke, editors. Proceedings of the Fifth GI SIG SIDAR Graduate Workshop on Reactive Security (SPRING). Technical Report SR-2010-01, GI FG SIDAR, Bonn, Juli 2010,

Beiträge zitieren als:
Autor. Titel. In Sebastian Schmerl, Simon Hunke, editors, Proceedings of the Fifth GI SIG SIDAR Graduate Workshop on Reactive Security (SPRING). Technical Report SR-2010-01, page xx. GI FG SIDAR, Bonn, Juli 2010.

PyBox - A Python approach to sandboxing

Felix Leder and Daniel Plohmann
University of Bonn, Germany
{leder, plohmann}{at}cs.uni-bonn.de

Over the last years, the diversity of malware has increased with new variants emerging on almost a daily basis. This means a serious challenge to analysts, raising the need for automation. One approach to gather information about suspicious files is running them inside of a controlled environment. These "sandboxes" provide functionality to monitor activities while executing their code, allowing insights into their behaviour. Known toolkits for this purpose are for example [CW07] or [SA07]. However, the existing solutions lack flexibility and allow necessary customization only to a certain extent because of their proprietary architecture. Other frameworks used for close monitoring are designed so heavy-weighing that they possibly influence the process of measurement itself. Examples like [LZ10] and [L08] are slowing down the executable of factors from thousand to tenthousand times. The overhead is caused by monitoring operations that are not relevant to the current analysis. Because of their static design, the possibility to control the tradeoff between execution speed for the analysis and level of detail with respect to the results, is very limited.

In addition to that, these analysis frameworks are difficult to extend. Depending on the programming language used, frequent compiling will be necessary, thus complicating rapid prototyping. Large amounts of development overhead in general interfere with achieving a short reaction time, which is essential when dealing with malware.

Our approach aims at providing this flexibility for semi-automated analysis while being as lightweight as possible. The foundation of our idea consists of injecting a software extension into the running, infected process carrying a Python interpreter. This offers the following advantages.

First of all, the injected code can be implemented with minimum complexity because the major functionality may be sourced out to external scripts. This eliminates the need for recompiling the framework after slight changes to the sourcecode and even allows to reconfigure the running script without the need to restart the process.

We extend the scripts with the ability to operate at system-level which is provided by a powerful library. Together with the injected code, full access to CPU registers, function parameters and results, as well as the process memory itself is gained. Also, individual API hooks can be managed dynamically. This in combination with the possibility to inject the monitoring on-the-fly allows to use the presented approach during systems forensics on live systems. A running process can easily be monitored with as much detail as wanted without the need to install additional drivers.

References

[CW07] Carsten Willems et al.: Toward Automated Dynamic Malware Analysis Using CWSandbox. *IEEE Security and Privacy 2007*, Pscataway, NJ, 2007.

[LZ10] Zhiqiang Lin, Xiangyu Zhang and Dongyan Xu: Automatic Reverse Engineering of Data Structures from Binary Execution. *Proceedings of the 17th Network and Distributed System Security Symposium (NDSS'10)*, San Diego, CA, February 2010.

[L08] Noé Lutz: Towards revealing attacker's intent by automatically decrypting network traffic. *Master's thesis*. ETH, Zürich, Switzerland, July 2008.

[SA07] David Zimmer: SysAnalyzer. *http://labs.idefense.com/software/malcode.php*, 2007.

Verwaltung von Signaturen fuer Malware-Gruppen

Sebastian Uellenbeck, Michael Meier

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl VI Informationssysteme und Sicherheit (ISSI)
sebastian.uellenbeck, michael.meier{at}udo.edu

Malware stellt eine große und zunehmende Bedrohung für die Sicherheit informationstechnischer Systeme dar. Aufgrund der von Malware-Autoren realisierten Techniken zur automatischen Erstellung einer Vielzahl verschiedener polymorpher Varianten von Malware stoßen klassische Malwareerkennungsansätze, die auf weitgehend manuell erstellten syntaktischen Signaturen basieren, an ihre Grenzen. Vor diesem Hintergrund wurde im Projekt AMSEL [1] ein System entwickelt, das automatisch kontinuierlich Malware sammelt, verhaltensbasiert analysiert und entsprechende Signaturen generiert und an Erkennungssysteme verteilt. Da polymorphe Varianten einer Malware grundlegend das gleiche Verhalten zeigen, wird nicht für jede Variante eine eigenständige Signatur erstellt, sondern zunächst werden Malware-Samples mit gleichartigem Verhalten automatisch mittels Clusterverfahren gruppiert und für diese Gruppen jeweils eine Signatur generiert. Die im System zu einer Malware vorliegende Informationsbasis ist dabei sehr dynamisch, da mit jedem gesammelten Malware-Sample die entsprechende Gruppe von Malware-Samples mit gleichartigem Verhalten wächst oder in mehrere Gruppen zerfällt. Entsprechend dynamisch sind die Signaturen für die Erkennung von Malware in diesen Gruppen. Daraus resultiert die Notwendigkeit, Strategien und Mechanismen zur Verwaltung von Signaturen zu entwickeln. Dazu gehören Verteilung, Aktualisierung, Rückruf, Versionierung und die Revisionierung von Signaturen.

Die Verteilung von Signaturen beschreibt, wie Erkennungssysteme mit Signaturen beliefert werden und gliedert sich in die Teilaufgaben Aktualisierung und Rückruf. Bei der Aktualisierung werden neue Signaturen zu einem vorhandenen Bestand hinzugefügt, sowie veraltete Signaturen gegen aktuellere ausgetauscht. Der Rückruf wird benötigt, um Signaturen, die in der Vergangenheit erzeugt wurden und zu einem späteren Zeitpunkt durch die Erweiterung des Goodpools als ungeeignet identifiziert wurden, aus den Erkennungssystemen zu entfernen. Da sich die Basis der Signaturen mit der Zeit ändert, ist es notwendig zu jedem Zeitpunkt zu wissen, welche Signaturen in einem konkreten Moment aktiv waren, was mit Hilfe einer Versionierung realisiert werden kann. Im Gegensatz dazu beschreiben Revisionen atomare Zwischenstände der Datenbasis, die als nicht verteilbar gelten. Eine Signatur wird beispielsweise erst erstellt und in späteren Schritten als korrekt und brauchbar verifiziert.

In dieser Arbeit werden Strategien und Mechanismen zur Verwaltung von Signaturen diskutiert und entwickelt.

Literatur

[1] M. Apel, J. Biskup, U. Flegel, and M. Meier. Early Warning System on a National Level - Project Amsel. In *Proc. of the European Workshop on Internet Early Warning and Network Intelligence (EWNI 2010)*, Hamburg, Germany, January 27th, 2010.

Clustering Malware for Generating Behavioral Signatures

Martin Apel, Michael Meier

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl VI, Informationssysteme und Sicherheit (ISSI)
martin.apel, michael.meier{at}udo.edu

Malicious software (malware) represents a major threat for computer systems of almost all types. In the past few years the number of prevalent malware binaries has increased dramatically due to the fact that malware authors started to deploy morphing techniques in order to hinder detection of such polymorphic malware by anti-malware products. Using these techniques numerous variants of a malware can be generated. All these variants have a different syntactic representation while providing almost the same functionality and showing similar behavior. In order to effectively detect polymorphic malware it is advantageous (if not required) to know which malware binaries are variants of a particular malware. Respective approaches for determining this relation between malware binaries automatically are currently investigated. In particular a number of approaches for clustering malware behavior have already been proposed which differ in focus as well as in the techniques applied. However, no systematic investigation of the question which of the available cluster algorithms is most appropriate for clustering malware behavior has been done yet. In order to answer this question we discuss desirable properties of cluster algorithms for the particular purpose of signature generation and comparatively evaluate cluster algorithms regarding these properties. We evaluated four different hierarchical cluster algorithms, namely Single-Linkage, Complete-Linkage, WPGMA, and UPGMA.

Using synthetic data as well as real world data sets we investigated, whether the algorithms respect some predefined equivalence relations, how they cope with a high amount of noise, and if the resulting cluster hierarchies have properties (deep and balanced) that are desirable for signature generation. We also examined how well the resulting clusterings are suited for signature generation. A cluster algorithm is assumed to be better, if based on the resulting clustering a better set of signatures can be generated. We measured the amount of behavior reports that are covered by each set of signatures as well as the number of signatures in the set. Signature sets covering more behavior reports are preferred. If two signature sets cover the same amount of behavior reports then the smaller set is favored. Results of our experimental evaluation indicate that the complete-linkage cluster algorithm is most suitable for clustering malware behavior and signature generation.

We also elaborate on additional requirements on clustering malware behavior for automatic signature generation. As malware often shows behavior which is common to multiple malware types support for overlapping clusters would be advantageous. Incremental clustering should be supported, e.g. incorporating new data without reclustering old data. For the specific purpose of signature generation the cluster algorithm should incorporate the resulting concepts (signatures in our case) into the clustering decisions, which is known as *conceptual clustering* [1]. We outline our plans on developing a cluster scheme fulfilling these requirements.

References

[1] R. E. Stepp and R. S. Michalski: Conceptual Clustering: Inventing Goal Oriented Classifications of Structured Objects, In Machine Learning: An Artificial Intelligence Approach, Vol II, 1986

Jitterbug 2.0

Benjamin Michéle*

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

An important property of malware such as keyloggers is to be invisible to the victim. This property is hard to achieve if the malware is running on the same CPU as the OS. Anti-malware software can identify the malicious code in memory or detect unsolicited network traffic. In [1] Shah et al. present the *Keyboard Jitterbug*, a physical keylogger device that uses a covert timing channel to send key logs to the adversary. Their approach has the advantage of not leaving any trace in the memory of the victim's machine as well as not adding extra network traffic. One of the downsides is the necessity to gain physical access to the victim's keyboard once, as well as adding extra hardware or replacing the keyboard, which might not go undetected.

Our Approach In order to remove these downsides, we combined the Jitterbug approach with a recent paper from K. Chen [2] which describes how to write custom firmware for an Apple Aluminium Keyboard. Combining these two approaches removes the above mentioned downsides: Neither does an attacker need physical access nor physical modification of the keyboard. Using our approach, different infection scenarios are possible. One scenario includes conventional malware which will infect the keyboard firmware and then erase all of its own traces on the host. Other scenarios include fake firmware updates as well as malicious manufacturers.

Challenges In contrast to the authors of the Keyboard Jitterbug which wisely chose PS/2 keyboards, we are forced to use USB keyboards. Instead of delivering key events immediately to the host, USB keyboards are polled regularly by the driver. This adds variable delay in the range of multiple milliseconds which we have to consider in our encoding. Furthermore, this is also a limiting factor for the available bandwidth of the covert channel. We are still investigating on how to solve this problem.

References

[1] Gaurav Shah, Andres Molina, Matt Blaze. *Keyboards and covert channels.* Proceedings of the 15th USENIX Security Symposium, pp. 59-75, 2006.

[2] K. Chen. *Reversing and exploiting an Apple firmware update.* Black Hat USA, 2009.

A Student Grade Man in the Middle Attack on the GSM Air-Link

Janis Danisevskis*, Kevin Redon†, Borgaonkar Ravishankar‡

Security in Telecommunications
Deutsche Telekom Laboratories
Technische Universität Berlin
D-10587 Berlin

*janis{at}sec.t-labs.tu-berlin.de †kredon{at}sec.t-labs.tu-berlin.de ‡ravii{at}sec.t-labs.tu-berlin.de

The attack in this talk is theoretically known since 2007 with the publication of [1]. Cryptanalysis of the GSM Ciphers A5/1 and A5/2 were published as early as 2000 [3] and 1999 [2] respectively. Our aim is to demonstrate that the man in the middle attack described in [1] can be practically mounted with a budget of around $500 and the programming skills of a college student.

Conventional IMSI catchers exploit the fact that GSM base stations do not need to authenticate with cellphones. However merely luring a victim into connecting to a fake base station is not transparent because the user is not registered to the regular network and thus can not be called. While professional spying equipment seems to exists with the claim to be transparent toward the user and the network, such equipment is not available on the free market and its complexity is unknown.

To maintain transparency toward the network the attacker must register with the network on the victim's behalf. This is achieved by retrieving the credentials needed to impersonate the victim from the victim's cellphone. In the case where the victim's phone supports A5/2 this can be done fairly easily and in real time.

In this talk we demonstrate our work in progress and the real world vulnerability implied by this man in the middle attack.

References

[1] Elad Barkan, Eli Biham, and Nathan Keller. *Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication.* Journal of Cryptology, 21(3):392–429, September 2007.

[2] Ian Goldberg, David Wagner, Lucky Green. *The (Real-Time) Cryptanalysis of A5/2*, presented at the Rump Session of Crypto'99, 1999.

[3] Jovan Golic. *Cryptanalysis of Alleged A5 Stream Cipher*, Advances in Cryptology, proceedings of Eurocrypt '97, Lecture Notes in Computer Science 1233, pp. 239–255, Springer-Verlag,1997.

Detecting malicious bot-induced network traffic using machine learning

Christian J. Dietrich

Univ. of Mannheim and Institute for Internet Security, Univ. of Applied Sciences Gelsenkirchen

Introduction and problem statement

Malicious remote controlled software, so called bots, cause lots of problems in todays Internet, such as spam, credential theft, denial of service attacks, click fraud and further infections. One possibility in order to avoid the damage bots cause is to detect them and possibly prevent them from doing harm. Nevertheless, only very few general methods for botnet detection exist. This work aims at designing a general botnet detection method.

On the one hand, some bots can be detected at the latest by the damage they induce. For example, spam bots can be identified by spam detection mechanisms. On the other hand, there are other types of bots that do not expose such an easily detectable attack or damage functionality. Some trojans or keyloggers for example steal credentials or confidential information without causing a stir. However, by definition, all bots have in common that they communicate with a command and control server (C&C). Thus, the network traffic of bots is commonly differentiated as either command and control traffic or attack traffic.

Botnet detection methods can furthermore be distinguished by their decision-making process. The majority of the existing botnet detection methods is based on characteristic criteria for botnets. Examples for these criteria are periodicity in the network behavior, destination access patterns or specific payload byte signatures. These criteria are often assembled manually in advance and do not change over time. The fact that these criteria are static is disadvantageous as bots can change their behavior quickly. An alternative approach to the manual extraction of criteria is to use machine learning techniques on bot network traffic samples. Thus, matching criteria can be extracted automatically and adapt as necessary.

Requirements and approach

As a consequence of the above mentioned, the following requirements should be met for a general behavior-based botnet detection mechanism:

- It is desirable to not solely depend on attack traffic. Instead, botnet detection should be based on any kind of traffic that is present even if no attack takes place, especially C&C traffic. However, the presence of attack traffic may support the detection.
- Behavior signatures should be dynamic and adapted as necessary.
- Detection should be accurate, i.e. it should have very low misclassification rates.

The following steps outline my approach:

1. Extract features from bot network traffic samples executed in a contained environment.
2. Build a machine learning model based on the extracted features.
3. Apply a classifier based on the model to live network traffic in order to detect bots at a network egress point.

Smartphone Botnets

Collin Mulliner*

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

Botnets - a collection of compromised computers - are one of the biggest problems in todays Internet. Botnets are used to send email spam, carry out DDoS attacks, and for hosting phishing and malware sites. Botnets are slowly moving towards smartphones since those devices are now powerful enough to run a bot and offer additional gains for a botmaster.

Therefore we investigate smartphone botnets. Our investigation leans towards designing a smartphone botnet in the best possible way in order to evaluate it and for predicting possible designs that could be encountered in the real world. The goal will be to come up with possible counter measures to stop smartphone botnet communication and to detect and/or prevent attacks carried out by smartphone botnets. Our focus of research on smartphone botnets are Command and Control and Payloads.

Command and Control (C&C) is the most challenging part in botnet design since it is the control channel for the botmaster. C&C has to be robust against attacks by defenders and possible by other botmasters who want to destroy and/or take over the botnet. In addition to robustness, stealthiness is another desirable feature since it slows down the detection of the botnet.

C&C for a smartphone botnet provides additional challenges that distinguishes it from desktop computer-based botnets. The main challenges are. Connectivity, communication costs, and computational power. Smartphones have many possibilities to communicate such as WiFi, Bluetooth, SMS, CSD (circuit switched data), and the packet-data service. All these possibilities have to be considered when designing a smartphone botnet. Some of the connectivity options such as SMS, CSD, and packet-data carry the burden of usage depended service charges. These services charges might prohibit communication of a bot and in addition will make the bot easily detectable by the owner of an infected device. Limited computational power is a limiting factor when it comes to encryption and authentication.

Payloads for mobile botnets are very interesting since a bot on a smartphone possesses many abilities not present on a desktop computer. Abilities such as access to the mobile phone service to place calls and to send SMS and MMS messages open the door for easy financial gain for a botmaster. Further since other ways exist to interact with the mobile phone operators network, attacks against the infrastructure are possible. Presents of hardware such as GPS and a microphone open the possibilities for tracking users and for eavesdropping. Access to specific private data such as the users phonebook, calendar, and SMS messages can be easily abused for extortion. If a mobile phone offers low level access to its communication (GSM/3G/LTE) hardware one can imagine very sophisticated attacks on the mobile phone network that will be very hard to prevent.

A Secure and Reliable OS for Automotive Applications

Matthias Lange*

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

In the upcomming generation of vehicles not only infotainment and navigation systems will become more complex but also a whole new class of applications will emerge. Because of the complexity of these new functionality and tight time-to-market requirements collaboratively developed software must be extensively reused. This introduces risks that were so far unknown in the automotive world.

In our research we work on an architecture that allows to consolidate multiple components that used to reside on dedicated hardware onto a single controller. Our architecture allows for deploying whole standard OS's which are securely contained in protection domains.

Challenges Various challenges arise if classical control applications are to be supplemented with new complex applications on a single control unit. Modern infotainment units are inconceivable without access to the internet, which, on the downside, opens an attack vector for external adversaries. As such, the system cannot rely on the well-behavior of all components but has to enforce isolation. In this context, isolation is not limited to the spatial aspect but also includes the assurance of timely execution of the real-time components.

A system that used to be closed now becomes vulnerable to deliberate attacks. Experience from the desktop world shows that the mere existence of malware is a manifest to the shortcomings of contemporary standard operating systems. Therefore, a new system architecture is needed that can leverage the rich features of existing operating systems while, at the same time, compensate for their shortcomings regarding security.

Approach As of yet, dedicated hardware is the predominant method to ensure non-interference among software components. While this addresses safety and security concerns, it also leads to a fair number of control units, which stresses the power budget, adds to the bill-of-material, increases weight, and introduces complexity. Given these shortcomings, it is unlikely that an approach primarily relying on physical separation has a good prospect.

Microkernel based systems allow the construction of systems with an exceptional small trusted computing base (TCB). That makes them a good foundation for reliable systems. Dependability derives from the system being decomposed into small units that are placed into dedicated protection domains. On top of the microkernel and a layer of infrastructure servers we use OS rehosting to employ encapsulated instances of Linux.

Proof-of-Concept Implementation The microkernel approach does not only allow for spatial but also for temporal isolation. To illustrate the point, an example AUTOSAR application runs alongside. Since the microkernel is in charge of scheduling, this component is assured to be dispatched in a manner that allows it to meet all its deadlines. Another component that turns rogue, say a subverted instance of Linux, can neither corrupt the memory state nor impair the timely execution.

Learning from Rootkits

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

A rootkit is malicious code with certain stealth capabilities. It is placed on a target platform by an attacker. Stealth is an important rootkit property, since the attacker's goal is to hide the malicious code from the user. Therefore, rootkit developers try to find more *advanced* environments to hide their rootkits as documented by the rootkit evaluation: Rootkits moved from user space to kernel space and beyond.

One approach to obtain stealthiness is to somehow isolate the rootkit from the host platform. Our focus is on modern x86 platforms. On such platforms security researchers demonstrated rootkits (according to the ring protection model) not only running in user space (ring 3) or kernel space (ring 0) but also in ring -1 (Virtual Machine Monitor [2]), ring -2 (System Management Mode [1]) and ring -3 (Intel Active Management Technology [3]) in recent years. Obviously, the lower the ring the more isolated is the rootkit from the user.

Goals Our goal is to understand these isolated execution environments to: (i) develop countermeasures against such powerful and stealthy rootkits and to (ii) use them to enhance the platform's security properties. These isolated execution environments could be perfectly used to run code related to the Trusted Computing Base (TCB) such as a runtime monitor.

Challenges As yet, security research in this area has been done mainly on rootkits. To run code related to the TCB in an isolated environment, such as ring -3, certain challenges arise. For example, we need evidence that the environment is *bullet proof*. The herein before mentioned rootkits show, that the isolated environments are not bullet proof. Furthermore, we must understand the properties of these environments to realize an appropriate measurement strategy, i.e., how, when and what to measure when monitoring the platform. An important challenge arising in this context is the time-of-check-time-of-use (TOCTOU) problem.

References

[1] S. Embleton, S. Sparks, and C. Zou, "Smm rootkits: a new breed of os independent malware," in *SecureComm '08: Proceedings of the 4th international conference on Security and privacy in communication networks*. New York, NY, USA: ACM, 2008, pp. 1–12.

[2] J. Rutkowska, "Subverting Vista kernel for fun and profit," Black Hat USA, Aug. 2006. [Online]. Available: `http://blackhat.com/presentations/bh-usa-06/BH-US-06-Rutkowska.pdf`

[3] 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`

Topologie-angepasste Overlays für Peer-to-Peer Intrusion Detection

Michael Vogel*

* Brandenburgische Technische Universität
mv{at}informatik.tu-cottbus.de

Heutige IT-Infrastrukturen weisen eine schnell wachsende Leistungsfähigkeit der Endsysteme und Kommunikationsnetze auf. Die Anzahl und Komplexität der genutzten Anwendungen wächst kontinuierlich. Regelmäßig werden Verwundbarkeiten in diesen Anwendungen und Betriebssystemen entdeckt, die kontinuierlich durch Patches geschlossen werden. Signatur-basierte Intrusion Detection Systeme (IDS), wie Snort und Bro werden eingesetzt um Angriffe auf bekannte Verwundbarkeiten zu erkennen und agieren meist als zentrale Komponenten an Uplinks in Weitverkehrsnetze. Die Beobachtungsmöglichkeiten sind jedoch beschränkt. Die interne Kommunikation in einem Netz wird nicht erfasst. Ein umfassenderer Schutz durch ein IDS kann durch die Platzierung vieler verteilter IDS-Sensoren in einzelnen Subnetzen und LANs erreicht werden. Diese erzeugen jedoch umfangreiche, zu analysierende Beobachtungsdatenströme. Zur Analyse werden die Möglichkeiten der Kooperation einzelner Endsysteme genutzt, um eine leistungsfähige, verteilte und redundante IDS-Infrastruktur aufzubauen. Hierzu werden die Beobachtungsdaten der Sensoren verteilt durch die Vielzahl in einer Domäne vorhandener Endsysteme mit freien Ressourcen (Server, Desktop-Hosts) analysiert. Die Analyse erfolgt möglichst quell-nah, also auf Endsystemen, die sich bezüglich der Netzwerktopologie im gleichen LAN, Subnetz usw. wie der Sensor befinden.

Zur Suche nach geeignete Analyseressourcen bauen die kooperierenden Analysesysteme ein Peer-to-Peer-Informationsoverlay auf. Dieses verwaltet Informationen zu verfügbaren Analysekomponenten und freien CPU- und Speicherressourcen. Die Peers identifizieren die eigene Position in der physischen Netztopologie möglichst detailliert, um diese im Informationsoverlay zu vermerken. So können z. B. aus IP- Adressen und Subnetzmasken Zuordnungen zu verschiedenen LANs identifiziert werden. Durch whois-Anfragen werden Peers in Netzen verschiedener Organisationen unterschieden und schließlich verschiedene Autonome Systeme (AS) und ihre Uplinks bzw. Peerings mit anderen AS identifiziert. Dies erfolgt mit dem Ziel durch eine Suche im Informationsoverlay Kooperationspartner aufzufinden, die sich bezüglich der physischen Netztopologie möglichst in der Nähe befinden und dadurch die Kommunikationsverbindung kurze Latenzen und hohen Datendurchsatz erwarten lässt. Ergänzt werden diese Informationen durch gezielte Messungen zwischen Repräsentanten (Peers) einzelner Subnetze, bzw. Organisationen.

Die Topologie des Informationsoverlays spiegelt die physische Netztopologie des Internets, mit LANs, Netzen einzelner Organisationen, WANs, Peering- und Uplink-Beziehungen zwischen Autonomen Systemen wieder. So bilden die Peers eines Subnetzes (z. B. LAN) ein eigenes lokales Informationsoverlay. Ein oder mehrere Peers dieses Subnetzes sind als Repräsentanten ebenso Teil eines übergeordneten Overlays auf Organisationsebene (z. B. Campusnetz, Firmennetz). Wiederum sind ausgewählte Peers einer Organisation Repräsentanten in einem Overlay für das Autonome System dem die Organisation zuzuordnen ist. Die Suche nach Kooperationspartnern beginnt zunächst im lokalen Overlay und wird nur bei Bedarf auf physisch entferntere Peers ausgedehnt. Der Ausfall von Kommunikationsverbindungen, z.B. Uplinks beeinträchtigen das Overlay nur insofern, dass Informationen zu Peers, die durch den Ausfall ohnehin nicht mehr erreichbar sind, nicht gefunden werden können.

Parallelisierung von Network Intrusion Detection Systemen

René Rietz
Brandenburgische Technische Universität Cottbus
D-03013 Cottbus
rrietz{at}informatik.tu-cottbus.de

Die derzeit eingesetzten Intrusion Detection Systeme (IDS) basieren auf Technologien, die in den 90iger Jahren des vorangegangenen Jahrhunderts entwickelt wurden. In den meisten dieser Systeme wird eine Signaturanalyse realisiert, bei der protokollierte Ereignisse mit vordefinierten Mustern (Signaturen) verglichen werden. Bei der signaturbasierten Network Intrusion Detection (NIDS) werden dabei bekannte Protokoll-Header überprüft oder einfache Zeichenkettenvergleiche bzw. reguläre Ausdrücke auf dem Protokoll-Payload der Vermittlungs- und Transportschicht angewendet.

In den letzten zehn Jahren sind durch die zunehmende Übertragungskapazität lokaler Netze und des Internets sowie durch die Einführung neuer Kommunikationsparadigmen (wie z.B. P2P und Web 2.0) für NIDS schwierig zu bewältigende Analyselasten entstanden. Da in den 90iger Jahren vorwiegend IDS mit nur einem Prozessorkern eingesetzt wurden, sind die zugrunde liegenden Technologien auf sequentielle Analysen optimiert. Um mit der steigenden Netzlast und der zunehmenden Komplexität der eingesetzten Kommunikationsprotokolle Schritt halten zu können, sind entweder neue parallele Analysemethoden notwendig oder die Überführung bestehender IDS in parallele Analysesysteme. In den bereits bestehenden NIDS, die sich im breiten praktischen Einsatz befinden, wurden bislang 44 (Bro) bzw. 50 (Snort) Personenjahre Entwicklungszeit investiert (COCOMO-Schätzung). Eine Anpassung bestehender NIDS hin zu parallelen Analysemethoden ist somit einer Neuentwicklung vorzuziehen.

Es wurden bereits verschiedene Methoden untersucht, die bestehende IDS-Systeme durch Kapselung von Funktionalitäten parallelisieren. Die meisten dieser Verfahren betrachten die eigentliche IDS-Funktionalität als Black Box, indem z.B. zu analysierende Netzwerkflüsse auf zwei identisch konfigurierte NIDS aufgeteilt (Load-Balancing) oder innerhalb eines minimal modifizierten NIDS parallel verarbeitet werden. Eine Parallelisierung der eigentlichen IDS-Funktionalität wurde bisher nicht betrachtet oder auf die ersten Stufen der Pipeline-ähnlichen Verarbeitung (z.B. parallele Erkennung von Portscans) beschränkt. Neuentwicklungen wie z.B. Suricata, die Signaturen bestehender NIDS (Snort) verarbeiten können, sind trotz im Software-Design verankerter Parallelisierung bisher langsamer als das schnellste bekannte sequentielle NIDS (Snort).

Mittels eigener Messungen wurde für das NIDS Snort untersucht, in welchen Verarbeitungsstufen die höchste Analyselast anfällt. Dabei wurde überraschend festgestellt, dass das einfache Pattern-Matching mehrerer Zeichenketten/Binärdaten (mittels Suchbäumen bzw. Suchautomaten) den größten Analyse-Anteil umfasst (70% – 90%). Die Verarbeitung von regulären Ausdrücken aus den Angriffssignaturen und die Auswertung der Abhängigkeiten zwischen den Signaturteilen ist mit 0.1% – 2,5% Analyselast vernachlässigbar gering. Aufbauend auf diesen Ergebnissen wird zur Zeit ein Konzept zur Parallelisierung der Suchautomaten von Snort entwickelt, das die Eigenschaften der sequentiellen Suche (insbesondere die effiziente Nutzung des Prozessorcaches) beibehalten soll.