7. GI FG SIDAR Graduierten-Workshop über Reaktive Sicherheit

SPRING

Patrick Stewin und Collin Mulliner (Hrsg.)

5.-6. Juli 2012, Berlin

SIDAR-Report SR-2012-01
ISSN 2190-846X

Diesen Bericht zitieren als:
Patrick Stewin und Collin Mulliner, editor. Proceedings of the Seventh GI SIG SIDAR Graduate Workshop on Reactive Security (SPRING). Technical Report SR-2012-01, GI FG SIDAR, Berlin, Juli 2012,

Beiträge zitieren als:
Autor. Titel. In Patrick Stewin und Collin Mulliner, editor, Proceedings of the Seventh GI SIG SIDAR Graduate Workshop on Reactive Security (SPRING). Technical Report SR-2012-01, page xx. GI FG SIDAR, Berlin, Juli 2012.

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 (HFT Stuttgart) Ansprechpartner innerhalb der Fachgruppe SIDAR.

Nach der Premiere in Berlin fand SPRING in Dortmund, Mannheim, Stuttgart, Bonn und Bochum statt und ist dieses Jahr zurück in Berlin. Für die 7. SPRING konnten wir insgesamt 17 Einreichungen in diesen technischen Bericht aufnehmen, worüber wir sehr erfreut sind. 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 hohe Anzahl an Einreichungen für die diesjährige SPRING ermöglichte uns Sessions mit den Themenbereichen Web Security, Intrusion Detection, Security Analysis, Mobile Security, Obfuscation and Covert Channels sowie SCADA Security anzubieten.

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/`).

Besonderer Dank gebührt Ulrich Flegel und Michael Meier für ihre Unterstützung bei der Planung. Weiterhin möchten wir uns bei Claudia Petzsch und Jean-Pierre Seifert für die Unterstützung bei der lokalen Organisation und Durchführung bedanken.

Berlin, Juli 2012 Patrick Stewin und Collin Mulliner

Contents

On detecting fakeAV online scanners

Sascha Schimmler*

*Ruhr University Bochum
44780 Bochum, Germany
sascha.schimmler{at}rub.de

FakeAV rogueware is a form of computer malware that deceives or misleads users into paying for a simulated removal of bogus malware that has been installed on the user's computer. In the past few years (2008 – 2011), the prevalence and damage caused by fakeAV have increased [1] as a result of its success in infecting a large number of systems and the concluding profitability for the criminals behind those campaigns [2].

The most preferred way for the distribution of fakeAV is the use of websites that mimic the Microsoft Windows Explorer interface in which a simulated virus scan takes place. After the scan, a fakeAV binary is offered to the user to clean those bogus threats. The distribution websites have significant characteristics which distinguish them from benign websites. Each of those characteristics alone is not meaningful enough, but when combined together, in a smart way, it is possible to decide if a fakeAV website hides behind a given URL.

In my approach, I developed an analysis system that is able to detect if a website acts like a fakeAV online scanner. The first step in the analysis cycle is to decide if the website contains obfuscated javascript. If that is the case, the website is preprocessed through a low-interaction honeyclient [3] that produces an XML parse tree with HTML plaintext. This output is processed with static analysis methods. If no obfuscation is found, dynamic analysis is skipped. In the stage of processing static HTML, pictures are extracted and compared against a set of pictures commonly found in fakeAV websites with an image comparison algorithm[4]. Another step of the analysis is the extraction and comparison of embedded CSS elements and plaintext keywords.

This approach works fully automated and it is possible to analyse a large amount of URLs and to subsequently blacklist them, if they were flagged as fakeAV.

Literatur

[1] Rajab, M.; Ballard, L, *The Nocebo Effect on the Web: An Analysis of Fake Anti-Virus Distribution*, 2010.

[2] Brett Stone-Gross, B.; Ryan Abman, *The Underground Economy of Fake Antivirus Software*, 10th Workshop on Economics of Information Security (WEIS), June 2011.

[3] Dell'Aera, Angelo *Thug: a new low-interaction honeyclient*, The Honeynet Project, 2012.

[4] Olah, Attila *Fast image comparison with Python*, `http://aatiis.me/2010/08/12/fast-image-comparison-with-python.html`, 2011.

Lightweight Integrity Protection for Web Storage-driven Content Caching

Sebastian Lekies

SAP Research
sebastian.lekies@sap.com

Since the rise of Web 2.0 applications a shift from server-side to client-side functionality is perceivable on the Web. Especially new HTML5 features such as Web Messaging, Cross-Origin Resource Sharing or Offline Apps enrich the user-experience of modern Web applications. However, with the power of these new APIs comes the responsibility to utilize these features in a secure fashion. In the past some research work has already been conducted to reveal potential security issues with client-side technologies [1, 2, 3, 5, 6].

In this talk we investigate HTML5's Web Storage API that consists of the SessionStorage and LocalStorage attributes [4]. Web Storage is a mechanism that allows a Web application to store structured data within the user's Web browser via Javascript. While this API can be used for client-side state management, it is also often used for caching[7]. Especially, in mobile environments where bandwidth and latency matters Web-Storage-based caching can be a powerful technique to decrease loading times by saving and reusing frequently required scripts or style declarations on the mobile device[7].

However, caching such content in a storage that is accessible via scripting is a dangerous practice as it creates new attack vectors for adversaries. The cause of the problem is the fact that at one point in time, code written to the storage has to be executed again. Hence, if an attacker is able to exchange the cached code with his payload, the application automatically runs the malicious content. Well-known cross-site scripting defense techniques such as input validation or output encoding are not applicable in this scenario, as legitimate code fragments would also be rendered void by these XSS filters.

In this talk we first investigate the usage of Web Storage with regards to code caching by investigating the front pages of the Alexa top 500.000 Web sites. Thereby, we found out that 20,422 Web sites make use of client-side storage and that 386 Web sites store 2084 pieces of HTML, Javascript code or CSS style declarations within Local- or SessionStorage. Furthermore, we present a method that allows a Web application to securely store code fragments on the client-side. We achieve this by utilizing checksums that are calculated for cached code. Whenever the code is fetched and executed from Web Storage the application validates the checksum in order to ensure integrity of the stored content. Therefore, an attacker is not able to inject his payload into client-side storage capabilities and thus attacks are rendered void.

Literatur

[1] Adam Barth, Collin Jackson, and John C. Mitchel. Securing Frame Communication in Browsers. In *USENIX Security*, page 17, 2008.

[2] Philippe De Ryck, Lieven Desmet, Pieter Philippaerts, and Frank Piessens. A security analysis of next generation web standards. Technical report, European Network and Information Security Agency (ENISA), July 2011.

[3] Steve Hanna, Eui Chul, Richard Shin, Devdatta Akhawe, Arman Boehm, Prateek Saxena, and Dawn Song. The emperor's new apis: On the (in) secure usage of new client-side primitives. In *Web 2.0 Security and Privacy (W2SP 2010)*, 2010.

[4] Ian Hickson. Web storage. Available online: `http://www.w3.org/TR/webstorage/`, December 2011.

[5] Lin-shung Huang, Eric Y Chen, Adam Barth, Eric Rescorla, and Collin Jackson. Talking to yourself for fun and profit. In *Proceedings of W2SP*, 2011.

[6] Lavakumar Kuppan. Chrome and safari users open to stealth html5 appcache attack. Available online: `http://blog.andlabs.org/2010/06/chrome-and-safari-users-open-to-stealth.html`, June 2010.

[7] Steve Souders. App cache & localstorage survey. Available online: `http://www.stevesouders.com/blog/2011/09/26/app-cache-localstorage-survey/`, September 2011.

Time Is On My Side: Exploiting Timing Side Channels on the Web

Sebastian Schinzel

Lehrstuhl für Informatik 1 - IT-Sicherheitsinfrastrukturen
Universität Erlangen-Nürnberg, Germany

Timing side channels are vulnerabilities in software applications that leak sensitive information about secret values such as cryptographic keys. They differ from common intrusive vulnerabilities such as Buffer Overflows or SQL-Injection because the attacker sends normally looking requests to the server and infers secret information just from the time it took to process the request. Timing side channel attacks are well researched, especially against cryptographic hardware, but in day-to-day penetration testing, they are still widely ignored. One reason for this is that the timing differences are often small compared to the jitter introduced in networked environments. This makes practical timing side channel attacks challenging, because the actual timing differences blend with the jitter.

In this talk, I will present methods and tools to accurately measure response times despite the jitter in networked environments. I will introduce a programming library that enables penetration testers to measure accurate response times of requests send over networks. Furthermore, I will describe algorithms and statistical filters to reduce the jitter from measurements. For this, I will introduce a reporting tool that takes a dataset with network measurements as input, automatically applies the algorithms and filters, and produces a report with the results. This report enables even novice penetration testers to analyze a response time dataset for timing side channel vulnerabilities. In the end, I will show that timing side channels are practical by showing several attacks.

Verwundbarkeitsanalyse des Industrial-Ethernet Protokolls Profinet IO

Andreas Paul
Brandenburgische Technische Universität Cottbus
D-03046 Cottbus
andreas.paul{at}informatik.tu-cottbus.de

Zur Steuerung, Überwachung und Optimierung von Industrieanlagen werden seit geraumer Zeit SCADA-Systeme (supervisory control and data aquisition) eingesetzt. Während SCADA-Systeme früher ausschließlich aus proprietären Insellösungen bestanden, ist in den letzten Jahren ein starker Trend hin zu standardisierten und vollständig vernetzten Kommunikationsstrukturen erkennbar. Verbunden mit der zunehmenden Anbindung von SCADA-Systemen an öffentliche Netze steigt das Gefährdungspotential dieser Systeme durch Angriffe Dritter.

In SCADA-Systemen - speziell im Bereich der Automatisierungstechnik - ist mit der Entwicklung des Industrial Ehternet die standardisierte und offene Kommunikation sogar bis in die Feldebene vorgedrungen, wodurch theoretisch eine direkte Kompromittierung der unmittelbar am Automatisierungsprozess beteiligten Geräte, wie speicherprogrammierbare Steuerungen oder dezentraler Peripherie, ermöglicht wird. Da existierende Sicherheitskonzepte den speziellen Anforderungen im industriellen Umfeld, wie bspw. die Unterstützung von Echtzeitkommunikation, im Allgemeinen nicht gerecht werden, weisen aktuelle Industrial Ethernet-Technologien fehlende Mechanismen zur Gewährleistung einer authentifizierten und vertraulichen Kommunikation auf. Sicherheitsvorfälle aus der näheren Vergangenheit - wie beispielsweise die Sabotage vollständiger Industrieanlagen durch den Stuxnet-Virus [1] - zeigen jedoch, dass der Einsatz umfassender Schutzmechanismen in SCADA-Systemen als unabdingbar anzusehen ist.

Ein erster Schritt zur Entwicklung geeigneter Sicherheitskonzepte stellt eine ausführliche Verwundbarkeitsanalyse der eingesetzten Kommunikationstechnologien dar. In diesem Beitrag werden die Resultate der Analyse der Industrial Ethernet-Variante Profinet IO vorgestellt. Basierend auf der Erläuterung grundlegender Mechanismen und Kommunikationsabläufe des Profinet IO Protokolls werden unterschiedliche Angriffsszenarien diskutiert. Im Gegensatz zu einer rein theoretischen Betrachtung [3] werden dabei konkrete Nachrichtenfolgen zur Durchführung verschiedener Man-in-the-Middle-Angriffe beschrieben. Ferner werden Möglichkeiten zur Durchführung von Denial-of-Service-Attacken aufgezeigt.

Literatur

[1] Falliere, N., Murchu, L.O., Chien, E.: W32.Stuxnet Dossier, Version 1.4. Symantec Security Response, Cupertino, (February 2011), `http://tinyurl.com/36y7jzb`

[2] IEC 61158-6-10 Industrial communication networks - Fieldbus specifications - Part 6-10: Application layer protocol specification - Type 10 elements. (2007)

[3] Baud, M., Felser, M.: Profinet IO-Device Emulator based on the Man-in-the-Middle Attack. In: 11th IEEE International Conference on Emerging Technologies and Factory Automation. pp. 437–440. IEEE Computer Society, Washington, D.C. (2006)

Hardware Trojans

Christoph Bayer

Security in Telecommunications
Technische Universität Berlin
christoph{at}sec.t-labs.tu-berlin.de

Until lately, computer hardware was trusted without reservations. However, it is questionable whether this reliance is justified, because integrated circuits (ICs) can contain malicious logic, so-called hardware trojans. Since many in-house steps of an IC production process are being outsourced to third parties, malicious alterations of ICs are becoming more feasible and probable [1]. Especially, mission-critical areas like military and governmental systems, communication, transportation and financial infrastructure have to rely on its underlying hardware.

Hardware trojans vary e.g. with respect to their insertion phase during the production process, activation mechanism, effect or location in a system. A reasonable scenario and challenging question is the following [2]: Is it possible to design ICs in such a way that the functionality intended by the designers is guaranteed even if a malicious adversary is allowed to tamper with the ICs during certain subsequent production steps? So far, research has been focused on hardware trojan development, classification and detection after IC production. There are a few approaches to counteract and detect hardware trojans at runtime. However, these methods are not theoretically founded, and they do not cover all hardware backdoors.

This is why we motivate the development of a formal attack model. Such an approach can bring the issue of hardware trojans from the arena of heuristics into a well defined theoretical framework, and as such it has the potential to open up a new direction for theoretical security analysis.

References

[1] S. Adee. The Hunt for the Kill Switch. *IEEE Spectrum*, 45(5):34–39, 2008.

[2] Defense Science Board Task Force. On High Performance Microchip Supply, 2005.

Large Scale Analysis of Malware Downloaders

Christian Rossow*†

* VU University Amsterdam, c.rossow{at}few.vu.nl

† Institut fuer Internet-Sicherheit, rossow{at}internet-sicherheit.de

A downloader is a malicious program with the purpose to subversively download and install malware on a victim's machine. Once a downloader is executed, the running system will typically be compromised with multiple different malware families. Thus, downloaders represent a simple yet widely-used way to spread new malware, underlying a fraudulent underground service model, such as the pay-per-install model largely analyzed by Cabellero et al. [2].

We outline and analyze the landscape of what we think represents a snapshot of prevalent and current downloaders. We identified 23 downloaders, of which many – to the best of our knowledge – have not yet been documented. We characterize these downloaders concerning their communication model. For example, we discuss the communication architectures of downloaders (e.g., P2P), and outline the techniques used to encrypt or even camouflage the malicious activities. We then use dynamic analysis traces from Sandnet [1] to provide a long-term monitoring analysis on these 23 downloaders, identifying 18 downloaders to be still active as of February 2012.

Motivated by this observation, we investigate how attackers ensure the resilience of downloader infrastructures. Contrary to our expectation that IP address blacklists would force attackers to change their infrastructure frequently, we show that 219 C&C servers (20%) were actively operated for more than four weeks. For the remaining servers, we analyze how attackers use DNS and IP address fluxing to operate their downloaders, suggesting that isolating downloader infrastructures is much harder than it seems. Similarly, we observed that attackers settle C&C servers in multiple networks (Autonomous Systems) and deploy C&C domains from a diverse set of Top Level Domains, presumably to increase the complexity of takedown approaches.

All downloader infrastructures have one necessity in common: these services must be publicly accessible, as (with the exception of targeted attacks) fraudsters aim for large-scale deployment of their malware. Based on this assumption, we propose two automated methods to extract the downloaded malware (eggs) in a generic and scalable fashion. We evaluate these two malware acquisition techniques both on downloaders with plaintext and encrypted communication, acquiring a diverse set of malware in the wild.

References

[1] Christian Rossow and Christian J. Dietrich and Herbert Bos and Lorenzo Cavallaro and Maarten van Steen and Felix C. Freiling and Norbert Pohlmann: *Sandnet: Network Traffic Analysis of Malicious Software*, Proceedings of the 1st ACM EuroSys BADGERS Workshop, in Salzburg, Austria, April 2011

[2] Juan Caballero and Chris Grier and Christian Kreibich and Vern Paxson: *Measuring Pay-per-Install: The Commoditization of Malware Distribution, Proceedings of the 20th USENIX Security Symposium, in San Francisco, CA, August 2011*

The full downloader analysis was performed in collaboration with Christian J. Dietrich and Herbert Bos and will be presented at the 9th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA '12), in Heraklion, 26th–27th July 2012.

Encoding techniques for evading n-gram based Intrusion Detection Systems

Moritz Bechler

Wilhelm Schickard Institut
Universität Tübingen
72076 Tübingen, Germany
moritz.bechler@student.uni-tuebingen.de

While signature-based Intrusion Detection Systems are still commonly used today, they have major shortcomings when it comes to maintenance effort and 0-day or targeted attacks. Anomaly based detectors try to resolve these issues by modeling normal activity and alarming on deviation from this model. One simple kind of content anomaly detector uses byte frequencies to classify traffic, but this approach was shown to be easily evaded when the attack is adjusted to mimic a normal traffic profile by adding padding and performing some simple substitutions. One promising alternative, making these so called *mimicry* attacks more difficult, is n-gram based analysis. n-grams are sequences of n consecutive bytes, and because these are overlapping they are able to reflect some structural properties of the content. The ANAGRAM system [1] is a detector using a n-gram based model which counts the number of n-grams contained in traffic which are not part of the trained model.

In this work, practical mimicry attack against ANAGRAM are explored. An approximate n-gram model of a detector can be described using a deterministic finite automaton which is subsequently used to find a substitution producing a perfect match on this n-gram profile. A small portion of code is later used to reproduce a shellcode on a target system. Unfortunately finding fixed-length substitutions achieving this is closely related to the NP-complete subgraph isomorphism problem. In this work, it was tried to find single-byte substitutions using a brute-force approach on restricted alphabets and also a variation of this scheme where some padding is added. These approaches are both computationally hard and were found to fail on most of the inputs used. Therefore this work proposes a variable-length encoding scheme which is a improved version of one introduced in [2]. This scheme uses a set of 2^k distinct cycles sharing a common node in the model graph and, by choosing from one of these cycles, encodes k bit at a time. To optimize for size, shorter cycles are used for input symbols occurring more often. This turned out to work well with an optimum of $k = 4$ resulting in an upper bound on space expansion of 21.

While this is worse than the expansion to be expected from single-byte substitutions it might very well be in the practical range. Future work may be able to turn this into a full mimicry attack against ANAGRAM which will successfully evade this detector.

References

[1] Wang, K., Parekh, J.J., Stolfo, S.J.: ANAGRAM: A content anomaly detector resistant to mimicry attack. In: In Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID) pp. 226-248 (2006)

[2] Fogla, P., Sharif, M., Perdisci, R., Kolesnikov, O., Lee, W.: Polymorphic blending attacks. In: In Proceedings of the 15 th USENIX Security Symposium. pp. 241-256 (2006)

Intrusion Detection for Automation

Franka Schuster
Brandenburg University of Technology Cottbus
D-03046 Cottbus, Germany
Franka.Schuster{at}informatik.tu-cottbus.de

Automation systems are usually part of a supervisory control and data aquisition (SCADA) infrastructure for control of industrial, infrastructural, or facility-based processes. Also critical infrastructures, such as power and water supply, are SCADA systems that depend on the faultless operation of automation systems in plants and distribution networks.

Since automation systems of a SCADA infrastructure are usually connected to a corporate network, operators apply firewall techniques to seal main parts of the automation system off from unauthorized access. From the limited security perspective of the operators, this measure is often assessed as sufficient. However, this measure cannot help to identify attacks that already have overcome firewall protection or are initiated from inside the automation system. Recent incidents [1, 2] show that without security measures within such an automation system an attacker can successfully obfuscate malicious behaviour by manipulating status information from field devices provided for higher control and monitoring. Thus, secure automation requires the implementation of multistage security for which standard firewalls must only be the first step. This should be complemented by the application of reactive security means, such as intrusion detection techniques.

For this purpose, we investigate in the development of a tailored intrusion detection system for automation traffic, which is to the best of our knowledge not available, yet. It consists of a detection component, intended to be placed on a PC or switch, that monitors communication between several automation devices. It analyses the traffic regarding anomalies in packet structure, content and sequences. By placing multiple instances of this component on control and field level analysis events from different points in the automation system can be propagated between the instances. On the one hand, a collection of these events provides a differenciated view on the current state of individual parts of the automation system regarding anomal communication. On the other hand, an incremental aggregation and correlation of analysis events during propagation allow an assessment of the overall system state.

Our approach differs from related solutions [3, 4, 5] in the distributed manner ([3, 4]) and the direct analysis of automation traffic instead of applying indirect data collection, such as log file parsing ([5]).

References

[1] Nicolas Falliere, Liam O Murchu, and Eric Chien. W32.Stuxnet Dossier, Version 1.4. `http://tinyurl.com/36y7jzb`, February 2011.

[2] W32.Duqu – The precursor to the next Stuxnet, Version 1.4. `http://tinyurl.com/6jxael9`, November 2011.

[3] Quickdraw SCADA IDS. `http://www.digitalbond.com/tools/quickdraw/`.

[4] Snort 2.9.2 with SCADA protocol support. `http://www.snort.org/assets/166/snort_manual.pdf`.

[5] AlienVault ICS SIEM. `http://alienvault.com/docs/AlienVault-Datasheet-ICS-SIEM.pdf`.

GAIAS-IDS: Architekturkonzept zur Integration szenarienspezifischer Datenquellen in dynamische Intrusion Detection Systeme

Felix von Eye, Stefan Metzger, Wolfgang Hommel und Helmut Reiser
Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften
D-85748 Garching
[voneye, metzger, hommel, reiser]@lrz.de

In sehr großen, heterogenen IT-Infrastrukturen geht der Einsatz von Intrusion Detection Systeme (IDS) mit einer Komplexität einher, die signifikant über die in bisheriger Literatur betrachteten Laborumgebungen und kleinen bis mittelgroßen Rechnernetzen hinaus geht. Hinzu kommt, dass viele IDS eigene, d. h. dedizierte und zum Teil proprietäre Sensoren voraussetzen, bereits vorhandene Datenquellen für Sicherheitsereignisse wie System- und Anwendungsprotokolldateien aber nicht erschließen können.

Als Beispiel und Motivation dient uns das Münchner Wissenschaftsnetz (MWN), das vom Leibniz-Rechenzentrum betrieben wird und aktuell schon mehr als 100.000 angeschlossene Systeme umfasst. Bedingt durch die große Anzahl an zu schützenden Hosts und deren Nutzungscharakteristik ist die Menge sicherheitsrelevanter Ereignisse so groß, dass ihre Auswertung selbst bei Verwendung von heutzutage üblichen hochperformanten Security Information und Event Management (SIEM) Systemen nur eingeschränkt, also z. B. durch Reduktion der Anzahl verwendeter Korrelationsregeln, möglich ist und vor allem ständiger und personalintensiver manueller Pflege bedarf.

Im Rahmen eines Forschungsvorhabens wird als Lösungsansatz eine Gateway-basierte Architektur konzipiert, die, wie es für dynamische Intrusion Detection Systemen üblich ist, nicht immer alle verfügbaren Sensormeldungen über Sicherheitsereignisse an ein zentrales SIEM-System weiterleitet. Vielmehr reduziert das Gateway durch lokale Vorfilterung anhand dynamisch festgelegter Kriterien die Masse an Informationen auf ein handhabbares Maß und bietet zusätzlich die Möglichkeit, der SIEM-Lösung nur auf explizite Anfrage, d. h. in der Rolle eines dynamisch konfigurierbaren Sensors, Detailinformationen zur Verfügung zu stellen. Neben einer Verbesserung der Skalierbarkeit wird dadurch auch eine Integration nahezu beliebiger bereits vorhandener Datenquellen möglich, so dass die Vorteile von IDS und SIEM-Systemen miteinander kombiniert werden können, ohne dass zusätzliche Sensoren geeignet platziert und betrieben werden müssen.

Einen wesentlichen praktischen Mehrwert bietet die GAIAS-IDS-Architektur im Hinblick auf organisationsübergreifende Kooperationen, was wiederum am Beispiel des MWN deutlich wird: Das LRZ ist als Betreiber des MWN für Außenstehende primärer Ansprechpartner für alle möglichen Arten von Netzmissbrauchsfällen. Innerhalb des MWN sind jedoch die einzelnen Universitäten, Lehrstühle und Forschungseinrichtungen eigenverantwortlich für die ihnen zugewiesenen Subnetze. Oftmals ist es dem LRZ daher nicht möglich, beispielsweise bei einem vireninfizierten Rechner dessen Nutzer direkt zu kontaktieren, sondern es ist auf die Unterstützung der Administratoren vor Ort angewiesen. Durch die Nutzung von GAIAS-IDS kann in diesen Fällen ohne wiederholten manuellen Konfigurationsaufwand ein zeitlich befristeter Transfer von Angriffsmeldungen und institutslokalen Sensordaten erfolgen, wohingegen bei regulärem Nutzungsverhalten das Gatewaysystem keine institutsspezifischen Informationen an das LRZ-interne SIEM-System weitergibt.

Im Vortrag werden das Szenario, der aktuelle Stand der GAIAS-IDS-Architektur, Vorteile gegenüber bisherigen IDS-/SIEM-Systemen und das praktische Anwendungspotential dargelegt.

StackIDS - Catching Binary Exploits before they execute a System Call

Jakob Lell*
* TU Berlin
D-10623 Berlin, Germany
jakob@cs.tu-berlin.de

If an attacker gains control of a program by exploiting some flaws in the software, the next natural step is to gain control of the whole system with the privileges of the compromised program. At least one system call but in general a series of system calls is needed to achieve this next step. By detecting exploits before executing system calls issued by a program it is therefore possible to prevent successful exploitation even if the attacker has already compromised the control flow of a vulnerable application. While existing approaches focus on detecting anomalies in the sequence of system calls or on limiting the resources accessible to a process using profiles, we propose a novel approach to inspect the integrity of the stack data. The calling stack of a program contains many hints about the actual state of the program and the calling hierarchy. If the stack at the time of the system call does not match with the program executed, e.g., is not well-formed for all stack frames or the return addresses do not match with call instructions, this is an indication for an exploit. We implemented this concept in our new monitoring tool StackIDS. Experiments showed that our prototype was able to detect and block a variety of exploit techniques. StackIDS has also been successfully tested against two real-world exploits.

Tests with a random selection of programs from a typical Linux installation showed that the prototype implementation can be used with a majority of programs without any modifications of the protected programs. In some cases it is however necessary to recompile the program with reduced compiler optimization in order to prevent false positives.

The performance of the prototype implementation of StackIDS is not sufficient for most real-world applications. This is mainly due to the fact that the prototype was implemented in a scripting language as a userspace program using the ptrace() system call. A reimplementation of StackIDS in the kernel could drastically reduce the performance impact of protecting applications with StackIDS.

Automatically Detecting Backdoors In Software

Felix Schuster

Ruhr-University Bochum

Backdoors in software have probably been around since the very first access control mechanisms were implemented in software. Recent discoveries of purposely placed backdoors in software [1] underline the importance of the topics of prevention, identification and elimination of such backdoors. Our group started researching these topics several months ago. The focus of our work is on finding and disabling backdoors in binary software without the aid of the corresponding source code with the focus on embedded firmwares. Embedded firmwares appear to be especially predestined for the installment of backdoors since they in many cases process, contain or restrict access to valuable data (consider for example network switches, corporate printers or VoIP-telephones) but are on average rarely or never updated. Besides in most cases analysis by reverse-engineering of embedded firmwares for backdoors is hindered by proprietary data-formats and obscure system architectures.

In this context one of our currently most promising approaches for the identification of malicious backdoor code is the employment of platform independent *differential debugging*. The term differential debugging was first coined by the company *Zynamics* [2]. It refers to the technique of recording runtime traces of a software for different inputs. Given for example the runtime traces for a valid and an invalid password it is often possible to identify the function that determines the validness of an entered password in an automatic way. Analogous it is in turn often possible to determine the concrete basic-blocks that decide on the validness of a password or set an authentication flag to either *true* or *false* accordingly. As a foundation for such analysises we have created a library for Python that offers differential debugging features abstracted from the actual target platform. This library communicates with a remote GDB server in order to collect traces. It is thus theoretically applicable for all platforms that GDB server can be compiled for and executed on, which should include the majority of interesting embedded devices. Small adapter code is needed for each processor platform though. The novelty of our differential debugging approach is that it operates with respect to what we call *virtual basic-blocks*. Virtual basic-blocks are not actually present in a given binary program but are implicitly induced by conditional instructions like e.g. the CMOVZ instruction on x86. Besides our library features an interface for the generic description of network protocols. Our library is able to use such descriptions to automatically identify points of interest in an to be audited server application.

At the current point of research, we are already able to automatically identify authentication functions and command-dispatcher rountines in complex and well-known software like *ProFTPd* but the actual discovery of even toy backdoors still lies ahead of us.

References

[1] JC CREW. RuggedCom - Backdoor Accounts in my SCADA network? You don't say... `http://seclists.org/fulldisclosure/2012/Apr/277`, April 2012.

[2] Sebastian Porst, Zynamics GmbH. BinNavi 3.0 Preview: Improved Differential Debugging. `http://blog.zynamics.com/2010/01/19/binnavi-3-0-preview-improved-differential-debugging/`.

Ad-hoc-Vertrauen zwischen Smartphones auf Basis von Kommunikations-Logs

Sebastian Trapp und Matthias Wählisch

Freie Universität Berlin, Institut für Informatik
AG Technische Informatik & Telematik
Takustr. 9, 14195 Berlin
{sebastian.trapp, m.waehlisch}@fu-berlin.de

Die spontane Kommunikation zwischen Geräten setzt Vertrauen unter den Teilnehmern voraus. Die Etablierung von Vertrauen zwischen zwei Endgeräten wird gegenwärtig mittels Verfahren der Authentifizierung oder der Reputationsüberprüfung gelöst. *Authentifizierungsmechanismen* stellen indirekt die Identität des Kommunikationspartners oder seine Zugehörigkeit zu einer Gruppe sicher. Existierende Mechanismen basieren entweder auf einem vorher vereinbarten Schlüssel (*Pre-shared Secret*) oder einer zentralen Instanz. Beide Aspekte widersprechen dem Ad-hoc-Paradigma. Der Gewinn von Vertrauen durch *Reputation* ist häufig langwierig und birgt das inhärente Problem der „Vorspiegelung falscher Tatsachen“. Zudem erhöhen solche Verfahren die Kosten und benötigen Zeit bis die Vertrauenszuordnung vollständig konvergiert ist. Dies widerspricht dem Anwendungsfall von Ad-hoc-Netzen, in denen spontane, typischerweise kurzlebige Verbindungen präsent sind. Insbesondere können sich bösartige Knoten über lange Zeit gutwillig verhalten, um Vertrauen zu gewinnen und einen Angriff vorzubereiten.

Inspiriert durch soziologische Erkenntnisse [1] versuchen neuere Verfahren [2] soziales Vertrauen zwischen den Nutzern von Smartphones abzuleiten und auf ein Ad-hoc-Vertrauen zwischen den Geräten abzubilden. Die Vertrauensermittlung basiert ausschließlich auf Basis lokal verfügbarer Daten (Kommunikations-Logs). Der vorherige Austausch von geheimen Schlüsseln, explizite Nutzerinteraktion oder zentrale Komponenten kommen nicht zum Einsatz, wodurch sich der Ansatz insbesondere für Ad-hoc-Szenarien eignet. Der Erfolg solcher Protokolle hängt aber maßgeblich von einem genauen Verständnis ab, welche Kommunikationsdaten Vertrauen beschreiben.

In diesem Beitrag berichten wir über die erste, umfangreiche Messstudie [3], welche technische Daten von Smartphones auf soziales Vertrauen abbildet. Hierfür ließen wir mittels Amazon Mechnical Turk 217 Probanden jeweils 20 ihrer persönlichen Kontakte hinsichtlich Vertrauen bewerten. Auf Basis dieser Bewertungen und der vollständigen (anonymisierten) Kommunikations-Logs der involvierten Smartphones haben wir Kommunikationsmuster identifiziert, die vertrauenswürdige Personen beschreiben. Des Weiteren skizzieren wir die Anwendung dieser Erkenntnisse in einem Sicherheitsprotokoll zum Aufbau von vertrauenswürdigen Ad-hoc-Verbindungen zwischen Smartphones.

Literatur

[1] M. S. Granovetter, “The Strength of Weak Ties,” *The American Journal of Sociology*, vol. 78, no. 6, pp. 1360–1380, 1973.

[2] S. Trapp, M. Wählisch, and J. Schiller, “Short Paper: Can Your Phone Trust Your Friend Selection?” in *Proc. of the 1st ACM CCS Workshop on Security and Privacy in Mobile Devices (SPSM).* New York: ACM, 2011, pp. 69–74.

[3] ——, “Bridge the Gap: Measuring and Analyzing Technical Data for Social Trust between Smartphones,” Open Archive: arXiv.org, Technical Report arXiv:1205.3068, May 2012. [Online]. Available: `http://arxiv.org/abs/1205.3068v1`

Secure Smartphone Online Authentication using Softkeyboards

Sebastian Uellenbeck*, Hugo Gascon†, Christopher Wolf*, Konrad Rieck⋆

*Ruhr-University Bochum, †TU Berlin, ⋆University of Göttingen

Nowadays Smartphones are not only used for communication but also to store personal data like contacts, photos, and videos. Apart from actively storing data there is also some automatically generated private data on the Smartphone's storage such as browser history, personal text messages like SMS or email, and online banking data, to name but a few. Latest studies [1] show that 89% of lost Smartphones were accessed by an anonymous finder for personal related apps and informations. Therefore protection of private data on a Smartphone is a serious problem if the device gets lost or stolen. At present there are some attempts to hinder a walk-in-thief from accessing private data like PIN, pattern or password. All of these attempts have in common that they only protect the data if the user has activated them or a predefined amount of time has elapsed. However they are useless if a Smartphone is still unlocked and also cumbersome to enter because of the Smartphone's small keyboard. In addition, this login methods are easy to circumvent because they only consist of knowledge that can be obtained through shoulder surfing (PIN, password) or more sophisticated smudge attacks (pattern) [2]. To improve the Smartphone's authentication capabilities, Smartphones contain a lot of sensors to measure environmental changes. More precisely they often have an accelerometer that measures acceleration, a gyroscope that is able to detect torque, and an orientation sensor providing informations about the direction the Smartphone is held.

Given this, we developed an Android Softkeyboard that collects sensor data while the user is entering freely chosen text. First, we collect sensor data for each special key event like pressing or releasing a key and keeping a key pressed. In addition, we also save the exact pixel a user hits and of course the key for each keystroke. Finally, we collect all sensor data before, during, and after each keystroke.

Combining this data, we have a huge feature set where we have to find a collection of features that can be given to a learning algorithm to automatically discriminate between distinct users.

This talk gives an overview of the created framework, the feature sets we have tested so far, the pitfalls we were not able to skip, the results until now, and an outlook on future work.

References

[1] Scott Wright. The Symantec Smartphone Honeystick Project. Symantec, 2012.

[2] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith. Smudge Attacks on Smartphone Touch Screens. In *4th USENIX Workshop on Offensive Technologies, Washington, DC.* USENIX, 2010.

Bytecode Obfuscation-Techniken unter Android

Patrick Schulz
Rheinische Friedrich-Wilhelms-Universität Bonn
D-53113 Bonn
schulzp{at}informatik.uni-bonn.de

Android ist heutzutage das am häuftigsten eingesetzte mobile Betriebssystem. Der daraus resultierende Markt für Android Anwendungen ist sowohl für Entwickler als auch Malware Autoren von Bedeutung geworden. Ein Schutz der Anwendungen wird in vielen Fällen durch die Anwendung von Obfuscation-Techniken erreicht. Diese werden insbesondere verwendet, um Analysen z.B. zum Zweck des Reverse Engineerings zu erschweren. Die Android Architektur ist noch recht jung und die eingesetzten Obfuscation-Techniken noch einfach, im Vergleich zu den verwendeten Techniken unter der x86-Architektur.

Android Anwendungen sind größtenteils in Java geschrieben und werden in Form von Dalvik Bytecode ausgeliefert. Dieser ist dem Java Bytecode ähnlich und wurde auf die Anforderungen auf mobilen Geräten optimiert. Der schlanke Instruktionssatz und die Verwendung eines Bytecode-Validators auf den Geräten macht die Anwendung von vielen bekannten Obfuscation-Techniken auf Bytecodeebene schwierig. Daher setzen existierende Obfuscation-Tools Techniken ein, die nur wenige Änderungen am Bytecode selbst benötigen. Hierbei kommen meist Identifiermangeling, Stringobfuscation und Entfernung von ungenutztem Code zum Einsatz.

Die Anwendung von Techniken, die gezielt die Veränderung des Bytecodes beabsichtigen, lassen sich dennoch mit Einschränkungen anwenden. Die Auswirkungen auf Analyseprogramme sind hier sehr deutlich an den Ausgaben zu erkennen und haben starke Auswirkung auf deren Verständlichkeit und Konsistenz. Die untersuchten Obfuscation-Techniken beinhalten unter anderem das Einfügen von Junkbytes, die Anwendung von Kryptern/Packern und die Machbarkeit von Selfmodifying Code auf der Android Architektur.

Der Einsatz eines Bytecode-Validators und die Verwendung eines einfachen Instruktionssatzes schränken die Anwendbarkeit von Obfusction-Techniken unter Android nur in Teilen ein. Die Ergebnisse zeigen welche Schwächen aktuelle Analyseprogramme haben und welche Obfuscation-Techniken anwendbar sind, sowie deren Einschränkungen und Effektivität.

In-Memory-Scanning mit Virensignaturen und ein generischer Unpacking-Ansatz

Robert Michel
Ruhr-Universität Bochum
robert.michel{at}rub.de

In der AntiVirus-Industrie wird per Virensignaturen versucht, bekannte Malware anhand von Binärketten zu identifizieren. Jedoch gibt es inzwischen ausgeklügelte Methoden um die Erkennung zu umgehen. Der gängiste Trick ist das Komprimieren von Malware, was dazu führt, dass die dem AV-Produkt bekannte Signatur nicht mehr greift. Um die Signaturen zur Erkennung dennoch anwenden zu können, ist es nötig die Malware wieder zu entpacken. Der in dieser Arbeit verfolgte Ansatz versucht im ersten Schritt den Zeitpunkt abzupassen, an dem Malware entpackt im Speicher vorliegt, um Virensignaturen auf die entpackte Datei anzuwenden. Der Fokus liegt hierbei im Gegensatz zu bekannten Arbeiten wie z.B. Omniunpack [1] auf der Anwendung bei Kunden in einer Produktivumgebung - im Kontrast zu einer komplett emulierten Umgebung.

Der zweite und auch deutlich größere Teil dieser Arbeit befasst sich mit der Rückgewinnung der Malware im entpackten Zustand. Hierfür wird ein genereller Ansatz verwendet, der den Einsatz von Emulations- und Virtualisierungstechniken zwar zulässt, jedoch nicht voraussetzt - im Gegensatz zu anderen Arbeiten. Das Ziel besteht darin, mit bekannten und unbekannten Packern umgehen zu können um eine möglichst weite Abdeckung zu erreichen. Ein großes Augenmerk liegt auf der korrekten Wiederherstellung der Imports einer PE-Datei, bei dem alle bekannten generischen Unpacker bisher große Probleme aufweisen. Auch auf ein Verwendung der Debugging-API wird verzichtet, um nicht mit der weiten Vielzahl an anti-debugging-Techniken in Berührung zu kommen.

Für das Unpacking wird die Analyseumgebung entsprechend manipuliert, um die Ausführung der zu entpackenden Datei an ihrem ursprünglichen Entrypoint anzuhalten. Anschließend wird die Import Address Table (IAT) gesucht und wiederhergestellt. Hierfür werden vor der Ausführung alle Exports der im Prozess verwendeten Dlls mit Hooks versehen, deren Zielfunktion entscheidet, ob der ursprüngliche Funktionsaufruf ausgeführt wird, oder ob eine Identifikation der gehookten Funktion zurückgeliefert wird.

Viele Packer kapseln den eigentlichen Funktionsaufruf in eigenen Funktionen, was verhindert, dass die Adressen in der IAT direkt zu ihren Funktionsnamen aufgelöst werden können. Da diese eigenen Funktionen jedoch erst zur Laufzeit erstellt werden, ist die letztendliche Zieladresse dieser Funktionen immer der zuvor gesetzte Hook.

In Verbindung mit der Eigenschaft von Packern, dass die vom gepackten Programm verwendete IAT erst zur Laufzeit vom Entpacker gefüllt wird ergibt sich, dass das entpackte Programm eine IAT besitzt die vollständig mit Hooks gefüllt ist - oder mit packer-eigenen Funktionen, die letztendlich auch wieder auf Hooks zeigen. Da in den Hooks die Funktionsnamen zurückgegeben werden können wird hiermit ermöglicht die IAT vollständig wiederherzustellen, bevor die PE-Datei im Normalfall vollständig und lauffähig gespeichert wird.

Literatur

[1] Lorenzo Martignoni, Mihai Christodorescu und Somesh Jha (Hrsg.): *OmniUnpack: Fast, Generic,*

and Safe Unpacking of Malware in *In Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2007

AntiRE - An Executable Collection of Anti-Reversing Techniques

Daniel Plohmann and Christopher Kannen

Fraunhofer FKIE
D-53113 Bonn, Germany
⟨*plohmann*, *kannen*⟩{at}cs.uni-bonn.de

Reverse engineering is a methodology used to recover high-level information about structure and semantics of low-level analysis subjects. In the context of software, it is applied e.g. to establish understanding of embedded algorithms and protocols in order to achieve interoperability with software that is only available in compiled, binary form. This procedure is not always desired by creators of software as their products may contain intellectual property that they want to guard from competitors. Thus, protection schemes for binary code have been developed since decades.

Nowadays, such protection schemes are massively applied to malicious software (malware). The main reason is that the modifications performed on the binary code allow evasion from detection by security products. As a side effect, if the employed techniques or their effects are not known to malware analysts performing reverse engineering on malware samples, the analysis time can increase dramatically.

AntiRE is a collection of such anti analysis approaches, gathered from various sources like [1] and [2]. While these techniques by themselves are nothing new, we believe that their integration into a single, executable file provides a comprehensive overview, suitable for directly studying their behaviour in a harmless context without additional efforts. AntiRE includes different methods to detect or circumvent debuggers, fool execution tracing, and disable memory dumping. Furthermore, it can detect the presence of different virtualization environments and gives examples on how to twarth static analysis.

The techniques are implemented as isolated, minimal examples and accompanied by descriptions. Depending on their mode of operation, they are either treated as test or demonstration. A technique is defined as test if it yields information about presence of an analysis environment, in this case the result is returned. All test results are summarized at the end of execution. Otherwise, if the technique only serves as a demonstration, it does not influence the test statistics. To provide a better overview, the techniques are organized in groups ordered by similarity. When executed in a debugger, the tests and demonstrations can be easily accessed through a binary "table of contents", realized as a long sequence of commented function calls. When executed, our tool additionally generates output to a console window, showing the descriptions and optionally the test result. This can be used to quickly check an analysis environment for robustness against the included mechanisms.

AntiRE has been released as an open source project [3] and willingly accepts feedback and participation. The project will be extended continually.

References

[1] Peter Ferrie: *The "Ultimate" Anti-Debugging Reference*, 2011.
`http://pferrie.host22.com/papers/antidebug.pdf`

[2] Ange Albertini: *corkami.com - reverse engineering experiments and documentations*, 2011.
`http://www.corkami.com`

[3] Daniel Plohmann and Christopher Kannen: *AntiRE*, 2012.
`https://bitbucket.org/fkie_cd_dare/simplifire.antire`

Control Protocols for Network Covert Channels

Steffen Wendzel
FernUniversität in Hagen
D-58097 Hagen
SteffenWendzel{at}web.de

Covert channels are security policy breaking communication techniques based on channels which were not designed for a communication. Applied to today's computer networks, covert channels provide a means to i) exfiltrate confidential information of organizations, ii) to control botnets, as well as they iii) enable the unrestricted communication of political parties and journalists in monitored networks, just to mention the most important use-cases.

Within the last years, first ideas for the improvement of network covert channels arose, such as autonomous covert channels, self-adapting covert channels and covert channels with internal control protocols (so called *micro protocols*).

This work presents protocol engineering techniques which help to optimize the micro protocol design in a way that the covert channel raises as little attention as possible.

We demonstrate that covert channel overlays can utilize multiple network protocols to enable mobile covert channel usage, as well as we optimize the forwarding of data within the overlay network [1]. In our work, mobile covert channel overlays enable the usage of multiple access devices (e.g. smart phones), protocols and access points, as well as backward-compatible micro protocols.

We present a new approach by applying context-free and regular grammar to verify the conformance of micro protocols to the utilized network protocol (e.g. a covert channel's micro protocol should not break the standard-conform behavior of ICMPv6 if it is embedded within the protocol) [2]. We ensure such standard-conform micro protocols by verifying the inclusion of the micro protocol's language within the language of the utilized protocol.

Furthermore, we demonstrate calculations as well as an implementation for a first *active warden* able to decrease the bandwidth of protocol switching covert channels [3]. Our active warden introduces delays using a Linux netfilter extension to counter the covert channel performance. An important goal of the project is to prevent mentionable effects on the performance of normal (i.e. non-covert) network traffic.

References

[1] Steffen Wendzel and Jörg Keller: *Low-attention forwarding for mobile network covert channels*, in Proc. 12th Conference on Communications and Multimedia Security (CMS 2011), Ghent, Belgium, LNCS vol. 7025, pp. 122-133, Springer, 2011.

[2] Steffen Wendzel and Jörg Keller: *Systematic Engineering of Control Protocols for Covert Channels*, in Proc. 13th Conference on Communications and Multimedia Security (CMS 2012), Kent, 2012 (accepted).

[3] Steffen Wendzel and Jörg Keller: *Design and Implementation of an Active Warden Addressing Protocol Switching Covert Channels*, in Proc. 7th International Conference on Internet Monitoring and Protection (ICIMP 2012), Stuttgart, 2012 (accepted).