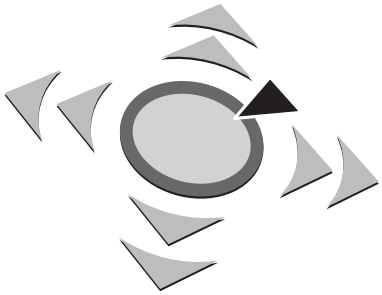
2. GI FG SIDAR Graduierten-Workshop über Reaktive Sicherheit

SPRING

Ulrich Flegel (Hrsg.)
Universität Dortmund, Fachbereich Informatik
D-44221 Dortmund

25. Juli 2007, Dortmund



SIDAR-Report SR-2007-01

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 (Universität Dortmund) Ansprechpartner innerhalb der Fachgruppe SIDAR.

Nach der Premiere in Berlin fand Spring in 2007 in Dortmund 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: Vorfallsbehandlung und Web-Services, Dynamische Umgebungen und Verwundbarkeiten, Anomalie- und Protokollerkennung sowie zu verschiedenen Aspekten der IT-Frühwarnung: Malware-Erkennung und -Analyse, Sensorik und Kooperation.

Besonderer Dank gebührt Michael Meier für die lokale Organisation der Veranstaltung und allen, die mitgeholfen haben: Alexander Burris, Johannes Hoffmann, György Kohut, Christoph Leuzinger und Özlem Sentürk.

Dortmund, Juli 2007 Ulrich Flegel

Contents

Diesen Bericht zitieren als:
Ulrich Flegel, editor. Proceedings of the Second GI SIG SIDAR Graduate Workshop on Reactive Security (SPRING). Technical Report SR-2007-01, GI FG SIDAR, Dortmund, July 2007, `http://hdl.handle.net/2003/24651`
Beiträge zitieren als:
Autor. Titel. In Ulrich Flegel, editor, Proceedings of the Second GI SIG SIDAR Graduate Workshop on Reactive Security (SPRING). Technical Report SR-2007-01, pages xx-yy. GI FG SIDAR, Dortmund, July 2007.

Begriffe und Modelle (nicht nur) für IT-Sicherheitsvorfälle

Tilmann Holst

Sicherheit in verteilten Systemen
Department Informatik
Universität Hamburg

DFN-CERT Services GmbH
Heidenkampsweg 41
20097 Hamburg

holst{at}dfn-cert.de

In der „Security"-Literatur werden grundlegende Begriffe wie z. B. „Incident" oder „Attack" unterschiedlich definiert, verwendet und verstanden. Das erschwert nicht nur die Zusammenarbeit, sondern auch Automatisierungen.

Ein reines Netzwerkmodell reicht für die Beschreibung von sicherheitsrelevanten Ereignissen oft nicht aus, es fehlt ein umfassenderes Modell. Hier soll ein ganzheitliches Modell vorgestellt werden, mit dem sich nahezu alle Vorfälle und Ereignisse beschreiben lassen. Drei Ebenen („tiers") werden unterschieden: „assessment tier", „analytical tier" und „observational tier". Innerhalb dieser Ebenen gibt es verschiedene Sichtweisen („views") um unterschiedliche Perspektiven abzubilden. Gesetze oder „Policies" werden im „assessment tier" gefasst, im „observational tier" können eine Netzwerk- und eine Host-Sicht verwendet werden. Auch nicht-technische Beobachtungen, z. B. von Menschen oder Objekten werden im „observational tier" behandelt.

Ausgehend von diesem Modell werden die Begriffe „event", „incident", „attack" sowie „accident" definiert und von einander abgegrenzt. Vorfälle, die einen Kausalzusammenhang haben, werden zu Fällen („Cases") zusammengefasst. Diese Begriffe lassen sich als Mengen abbilden und verdeutlichen. Durch entsprechende Definitionen lassen sich Konflikte oder Mehrdeutigkeiten ausschließen.

Für die automatisierte Verarbeitung von *Events* und *Incidents* sind folgende Kriterien notwendig:

Verlässlichkeit („Reliance") Wie gut ist die berichtende Quelle, wie sehr wird dem Betreiber getraut, ist der Transportweg angemessen geschützt?

Verarbeitbarkeit („Processabillity") Liegen die Informationen in einem geeigneten Datenformat vor? Sind sie die relevante Informationen vollständig? Stimmt die Semantik bei Schlüsselbegriffen überein?

Für Vorfälle („Incidents") ist ein drittes Kriterium notwendig:

Bewertung („Score") Wie gravierend ist der gemeldete Vorfall?

Die einzelnen Kriterien setzen sich aus Teilkriterien zusammen und werden aus diesen errechnet.

Abhängig von den eigenen Arbeitsmodellen und Policies können diese Kriterien herangezogen werden, um einzelne Ereignisse oder Vorfälle nach Relevanz zu sortieren, wichtiges von unwichtigem zu trennen. Abhängig von der Bewertung können unterschiedliche Anforderungen an die Verlässlichkeit eines Berichts gestellt werden.

Ziel der Arbeit ist es Methoden zu entwickeln, mit denen Vorfälle soweit möglich automatisch bearbeitet werden können. Begriffe und Modelle hierfür werden vorgestellt, Ansätze zur automatischen Verarbeitung werden skizziert.

Sicherheitsüberwachung entfernter Dienste in service-orientierten Architekturen

Lutz Lowis*

* Universität Freiburg, Institut für Informatik und Gesellschaft, Abteilung Telematik
Friedrichstr. 50, D-79098 Freiburg, lutz.lowis@iig.uni-freiburg.de

Die Sicherheit von IT-Systemen zu erhöhen ist nicht die Hauptmotivation hinter service-orientierten Architekturen (SOA). Vielmehr geht es darum, die Flexibilität zu erhöhen, mit der ein Unternehmen seine IT-Systeme auf neue geschäftliche Anforderungen ausrichten kann, indem Dienste verschiedener Hersteller und Standorte eingebunden werden. Wie ist es aber um die Vertraulichkeit, Integrität und Verfügbarkeit der Daten bestellt, die im Unternehmen verarbeitet werden, wenn kurzfristig potentiell gefährliche weil unbekannte Dienste eingebunden werden?

Unbekannt und damit potentiell gefährlich sind die Dienste deshalb, weil ihr Verhalten nicht genügend bekannt ist. Web Services als eine Umsetzungsmöglichkeit von SOA erlauben, weltweit nach passenden Diensten zu suchen und diese (semi-) automatisch einzubinden. Die Dienstbeschreibung, bswp. in WSDL, stellt dann die einzig vorhandene Informationen über die Dienste dar. Diesen Diensten stehen nun die Sicherheitsregeln für die Datenverarbeitung gegenüber, die z.B. im WS-Policy Framework festgehalten wurden. Es muß dann sichergestellt werden, daß etwa die Daten, denen eine hohe Vertraulichkeit zugeordnet wurde, auch wirklich vertraulich bleiben. Und das, obwohl sie von unbekannten Diensten, die oft sogar entfernt laufen, verarbeitet werden.

Zur Ermittlung und ggf. Einschränkung des Verhaltens von Software bzw. Diensten stehen bereits verschiedene Ansätze zur Verfügung. Antivirensoftware sucht nach Signaturen bekannter Schadprogramme und verhindert ggf. die Ausführung des Dienstes; neue Schädlinge sind nur heuristisch zu erkennen, was mglw. eine Vielzahl false positives bedeutet. Digitale Signaturen liefern den Nachweis, ob ein Dienst wirklich vom angeblichen Hersteller stammt, sagen aber nichts über das Softwareverhalten aus. Proof Carrying Codes (PCC) beweisen zwar das Softwareverhalten, bisher aber nur für sehr maschinennahe Eigenschaften, und außerdem muß der Dienstanbieter für jede Policy einen speziellen Beweis erstellen. Firewalls werden eingesetzt, um bspw. zu verhindern, daß unerwünschte Kommunikationskanäle geöffnet werden, das lokale Verhalten eines Dienstes ist aber für sie weitgehend unsichtbar. Integrity Checker können die Unversehrtheit bekannter Dienste bestätigen, aber über unbekannte Dienste keine Aussage treffen. Type Safe Languages erfordern den Nachweis, daß der laufende Dienst auch tatsächlich dem typisierten Quellcode entspricht. Der Function Extraction (fx) Ansatz liefert eine abstrahierte Version des Quellcodes, die aber nicht für Endbenutzer lesbar ist, sondern eher eine Hilfestellung beim Quellcode-Review darstellt. Am geeignetsten erscheinen Sandbox- und Information Flow-Methoden, um die Einhaltung der eigenen Sicherheitspolicy bei Verwendung unbekannter Dienste sicherzustellen. Kombiniert könnten diese Methoden als Sicherheitsmonitor zur Überwachung entfernter Dienste eingesetzt werden.

Im Vortrag werden die Möglichkeiten zur Ermittlung und Einschränkung von Softwareverhalten im SOA-Szenario diskutiert, statische und dynamische Ansätze gegenübergestellt und eine günstige Kombination ausgesuchter Methoden vorgeschlagen. Weiterhin wird kurz auf die Benutzung von Trusted Computing-Ansätzen eingegangen, anhand derer entfernt ausgeführte Dienste verläßlich überwacht werden können. Abschließend wird eine Diskussion angeregt, welche Auswirkungen die Einführung von SOA für heute übliche Sicherheitsmechanismen und -ansätze hat und ob grundlegend neue Ansätze nötig erscheinen, um in SOA die Einhaltung von Sicherheitspolicies überwachen und durchsetzen zu können.

Eine zustandsbehaftete Web Service Firewall für BPEL

Meiko Jensen
Christian-Albrechts-Universität zu Kiel
Institut für Informatik, Arbeitsgruppe Kommunikationssysteme
mje@informatik.uni-kiel.de

Mit dem Aufkommen *service-orientierter Architekturen* (SOA) spielen *Web Services* und Web Service-Kompositionen eine immer bedeutendere Rolle. Eine wichtige Spezifikation für die Komposition von Web Services bildet die *Business Process Execution Language*, kurz BPEL. Mit dieser XML-basierten Sprache lassen sich Geschäftsprozesse unter Nutzung von Web Services als Kommunikationsschnittstelle abstrakt modellieren und in einer BPEL-Laufzeitumgebung automatisch ausführen. Solche *Prozessbeschreibungen* definieren das Verhalten einer Prozessausführung, in deren Verlauf verschiedene Web Service-Nachrichten zwischen den Prozessteilnehmern ausgetauscht werden. Dabei definiert der aktuelle Zustand des Geschäftsprozesses, welche Web Services aufgerufen werden dürfen und welche nicht.

Der Umstand, daß die von einer BPEL-Laufzeitumgebung angebotenen Dienste über ein (möglicherweise unsicheres) Netz bereitgestellt werden, macht die Laufzeitumgebung zu einem potentiellen Ziel von netzwerkbasierten Angriffen aller Art. Insbesondere die Verfügbarkeit einer solchen Laufzeitumgebung kann leicht zum Ziel eines Angriffs werden, da die Verarbeitung der XML-basierten Web Service-Nachrichten sehr aufwendig ist. Tests haben gezeigt, daß eine typische BPEL-Laufzeitumgebung mit Web Service-Nachrichten im Umfang von 500 KByte für über 2 Stunden vollständig ausgelastet wurde, wenn die Nachrichten nicht zum aktuellen Zustand des zugehörigen Geschäftsprozesses passten (*zustandsinvalide* Nachrichten) [GJL07].

Der von uns gewählte Ansatz zur Abwehr von Angriffen mit zustandsinvaliden Web Service-Nachrichten basiert auf einer vorgeschalteten Application Layer Firewall, die den jeweiligen Zustand innerhalb der BPEL-Laufzeitumgebung anhand der ausgetauschten Nachrichten mitverfolgt und gleichzeitig jede kommunizierte Web Service-Nachricht mit hoher Effizienz auf Validität im aktuellen Kontext überprüft. Dadurch wird es möglich, zustandsinvalide Nachrichten schnell und ressourcenschonend zu identifizieren und abzuweisen.

Für die Identifikation zustandsinvalider Nachrichten wird ein eigens entwickeltes Automatenmodell genutzt, das den Zustandsfolge-Graphen eines in BPEL beschriebenen Prozesses abbildet. Das Modell basiert auf klassischen Transitionssystemen, verfügt aber über einige Erweiterungen für spezielle BPEL-Sprachkonstrukte. Zur Erstellung eines solchen *Nachfolgermengen-Automaten (Successor Set Automaton, SSA)* dient ein zweistufiger Algorithmus, der eine BPEL-Prozessbeschreibung in einen entsprechenden Automaten transformiert.

Die Wirksamkeit des Schutzkonzeptes wurde mit weiteren Tests belegt. So konnte der obige Angriff auf die BPEL-Laufzeitumgebung unter Verwendung der beschriebenen Firewall effizient abgewehrt werden, ohne daß es für valide Kommunikationsvorgänge zu signifikanten Einbußen in der Verfügbarkeit der Laufzeitumgebung kam [Jen07].

Literatur

[GJL07] Nils Gruschka, Meiko Jensen, and Norbert Luttenberger. A Stateful web service firewall for BPEL. *Angenommen für: IEEE International Conference on Web Services (ICWS)*, 2007.

[Jen07] Meiko Jensen. Konzeption und Implementierung einer BPEL Firewall. *Diplomarbeit*, 2007.

SAX-basierte Validierung von WS-Security-angereicherten SOAP-Nachrichten gegen eine Security Policy

Ralph Herkenhöner

Arbeitsgruppe Kommunikationssysteme, Institut für Informatik
Christian-Albrechts-Universität zu Kiel
Email: rhk@informatik.uni-kiel.de

Der Standard *WS-Security* beschreibt Sicherheitsmechanismen für die SOAP-basierte Kommunikation von Web Services. Er bezieht sich auf die XML-Standards *XML Signature* und *XML Encryption* und führt sog. *Security Tokens* ein. Der Einsatz dieser Security Tokens wird mittels Sicherheitsregularien nach den beiden Standards *WS-Policy* und *WS-SecurityPolicy* reglementiert. Somit ist es für den Einsatz von WS-Security notwendig, die korrekte Verwendung der Security Tokens innerhalb einer Nachricht anhand des gegebenen Sicherheitsregelwerkes (einer sog. *Security Policy*) zu überprüfen (*Policy Decision*) und durchzusetzen (*Policy Enforcement*).

Eine bekannte Schwäche von Web Services ist ihre Anfälligkeit gegenüber *Denial-of-Service-Angriffen*. Um keine neuen Gelegenheiten für solche Angriffe zu bieten, gilt es, die erforderliche Überprüfung so effizient wie möglich zu gestalten. Eine Ursache für die Anfälligkeit gegenüber Denial-of-Service-Angriffen liegt in der gängigen Praxis XML-Verarbeitung, die Nachricht zwischenzuspeichern, als Baumstruktur im Speicher aufzubauen, und danach zu verarbeiten. Dadurch wird bei Verarbeitung großer Nachrichten zusätzlicher Speicher verbraucht. Diese Anfälligkeit kann erheblich abgemildert werden, wenn statt dessen eine SAX-basierte XML-Verarbeitung gewählt wird [1]. Dabei wird die Nachricht sukzessiv beim Einlesen verarbeitet.

Hierbei stellen sich jedoch neue Probleme. So stößt beispielsweise die SAX-basierte Verarbeitung an ihre Grenzen, wenn die für eine Signaturprüfung bzw. die für eine Entschlüsselung notwendigen Schlüssel in einer Nachricht erst hinter der Signatur bzw. dem verschlüsselten Inhalt auftreten. Dann müssen Nachrichtenteile zwischengespeichert werden, was Denial-of-Service-Angriffe erleichtert. Weiter ist es erforderlich, die Security Tokens bereits bei ihrer Verarbeitung bezüglich der Security Policy auszuwerten, um ein Zwischenspeichern zu vermeiden. Zudem stellt sich das Problem der Zuordnung (auch *Mapping*) der Security Tokens zu konkreten Einträgen in der Security Policy. Das Mapping wiederum erfordert eine geeignete Aufbereitung der Security Policy [2].

Vor diesem Hintergrund konnte mittels einer prototypischen Implementierung [3] gezeigt werden, dass WS-Security-angereicherte Nachrichten mit nahezu konstantem Speicherverbrauch gegen eine Security Policy validierbar sind, und fehlerhafte Nachrichten bereits vor ihrer vollständigen Verarbeitung als solche erkannt werden können.

Literatur

[1] N. Gruschka, N. Luttenberger, and R. Herkenhöner. Event-based SOAP message validation for WS-SecurityPolicy-Enriched Web Services. In *Proceedings of the 2006 International Conference on Semantic Web & Web Services*, 2006.

[2] N. Gruschka, R. Herkenhöner, and N. Luttenberger. WS-SecurityPolicy decision and enforcement for Web Service firewalls. In *Proceedings of the IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation*, 2006.

[3] R. Herkenhöner. Eventbased and policy-driven Web Service firewall. *Diplomarbeit*, 2005.

Detection of Attacks on Application and Routing layer in Tactical MANETs

Elmar Gerhards-Padilla and Nils Aschenbruck
University of Bonn
D-53117 Bonn, Germany
padilla, aschenbruck{at}cs.uni-bonn.de

Nowadays, in several areas of life there is the need for flexible and reliable communication. In most cases communication is provided using an existing, fixed infrastructure (e.g. access points), but there are scenarios (such as military or catastrophe scenarios) in which such an infrastructure is not available. In such scenarios mobile ad-hoc networks (MANETs) are used to provide reliable communication. In a MANET potentially every node in the network acts as router. The nodes are typically battery-driven and therefore have to regard battery- and processing-power limitations. To enable mobility of the nodes, wireless communication is used which leads to low bandwidth compared to wired networks.

In our work, we consider a special case of MANETs, so called tactical MANETs.Tactical MANETs are specialised for being used in military scenarios (e.g. infantry missions) or in disaster areas where existing communication infrastructure may have been destroyed. Similar to general MANETs, a tactical MANET has a dynamic topology, low bandwidth and limited security. Typical applications in tactical MANETs are voice communication as well as command & control systems which e.g. process and transmit geographic and topologic information of the operational area or tactical commands for the units.

There is typically some kind of hierarchical command structure in tactical MANETs which implies at least two types of nodes: supervising nodes and supervised nodes. These types differ especially in their geographic position and employed hardware. Supervising nodes typically stay in the background, have access to a power supply, and therefore can use more powerful hardware. In contrast to that, the supervised nodes move frequently, use battery powered handhelds and therefore less powerful hardware. The powerful hardware of the supervising nodes predestines them to serve as central instances in securing the network. At the supervising nodes we run an Intrusion Detection System (IDS). This IDS uses several sensors to detect different kind of attacks. One classifier for attacks is the layer the attack is performed on. We distinguish between attacks on the application and on the routing layer. For each class of attacks we have a specified sensor for our IDS. The sensor for detecting attacks on the application layer is called Cluster Based Anomaly Detector (CBAD) ([TJFM06]). It is well tested in wired networks and also shows promising simulation results for tactical MANETs in detecting Portscans, Worms and SYN-Flooding. The sensor for detecting attacks on the routing layer is called Topology Graph based Anomaly Detection (TOGBAD) ([PAMJT07]). It shows promising simulation results regarding detection of black and grey hole attacks.

Literatur

[TJFM06] J. Tölle, M. Jahnke, N. gentschen Felde and P. Martini; Impact of Sanitized Message Flows in a Cooperative Intrusion Warning System; *Proceedings of the 25th Military Communications Conference (MILCOM 2006).*

[PAMJT07] Elmar Gerhards-Padilla, Nils Aschenbruck, Peter Martini, Marko Jahnke and Jens Tölle Detecting Black Hole Attacks in Tactical MANETs using Topology Graphs; *to be published*; 2007.

Denial-of-Service Prevention with Peer-to-Peer Barter Trade

Jens Oberender*

* University of Passau
D-94032 Passau, Germany
jens.oberender{at}uni-passau.de

Barter trade, i.e. resource allocation in return to receiving service, provides prevention of Denial-of-Service attacks. *'... whatever you did for one of the least of these brothers of mine, you did for me.'* [Matthew, 25.40]

The consumption of resources is unbalanced in many network protocols. While the client has little CPU and bandwidth usage, its request can cause immense effort on the server side. Denial-of-Service attacks (DoS) are based on this weakness. By allocating large fractions of the victim resources attackers hinder authorized users and therefore threaten *availability*.

Peer-to-peer networks distribute load onto many peers. Each peer can request a service from other peers. Our approach investigates paths within this request tree. Communications within the network form a graph, where a directed edge (a, b) indicates that b has provided service to a. The graph reveals ongoing attacks where outgoing edges outnumber incoming edges. A *closed path* consists of edges where peer issued requests. If the path forms a cycle, every peer both provides service and earns benefit. This is called immediate remuneration. It is more universal than barter trade between two peers, because it copes with asymmetry of interests [FLSC04, AG04].

Denial-of-Service prevention can be realized by limiting resources for not closed paths. Peers that contribute to a closed path gain high service quality, while newbies and attackers are throttled. Compared to other incentive mechanisms this approach is more robust to attacks, because no historic data is required. The DoS prevention is fully distributed. Each peer can define its own thoughput and diversity limits.

Methodology: The simulation framework facilitates a number of fair peers and attackers (5:1). We apply Zipf-distributed requests and measure the response traffic and the request graph. First, we compute the ratio of closed graph connections between the two groups. Second, we measure the traffic throughput within closed paths and in other connections. Using several scenario topologies we show that attackers are unable to steal resources from fair peers.

References

[AG04] Kostas G. Anagnostakis and Michael B. Greenwald. Exchange-based incentive mechanisms for peer-to-peer file sharing. In *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pages 524–533, Washington, DC, USA, 2004. IEEE Computer Society.

[FLSC04] Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust incentive techniques for peer-to-peer networks. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 102–111, New York, NY, USA, 2004. ACM Press.

Predicting Vulnerable Software Components

Stephan Neuhaus, Thomas Zimmermann, Andreas Zeller*

*Universität des Saarlandes

Where do most vulnerabilities occur in software? Our Vulture tool automatically mines existing vulnerability databases and version archives to map past vulnerabilities to components. The resulting ranking of the most vulnerable components is a perfect base for further investigations on what makes components vulnerable.

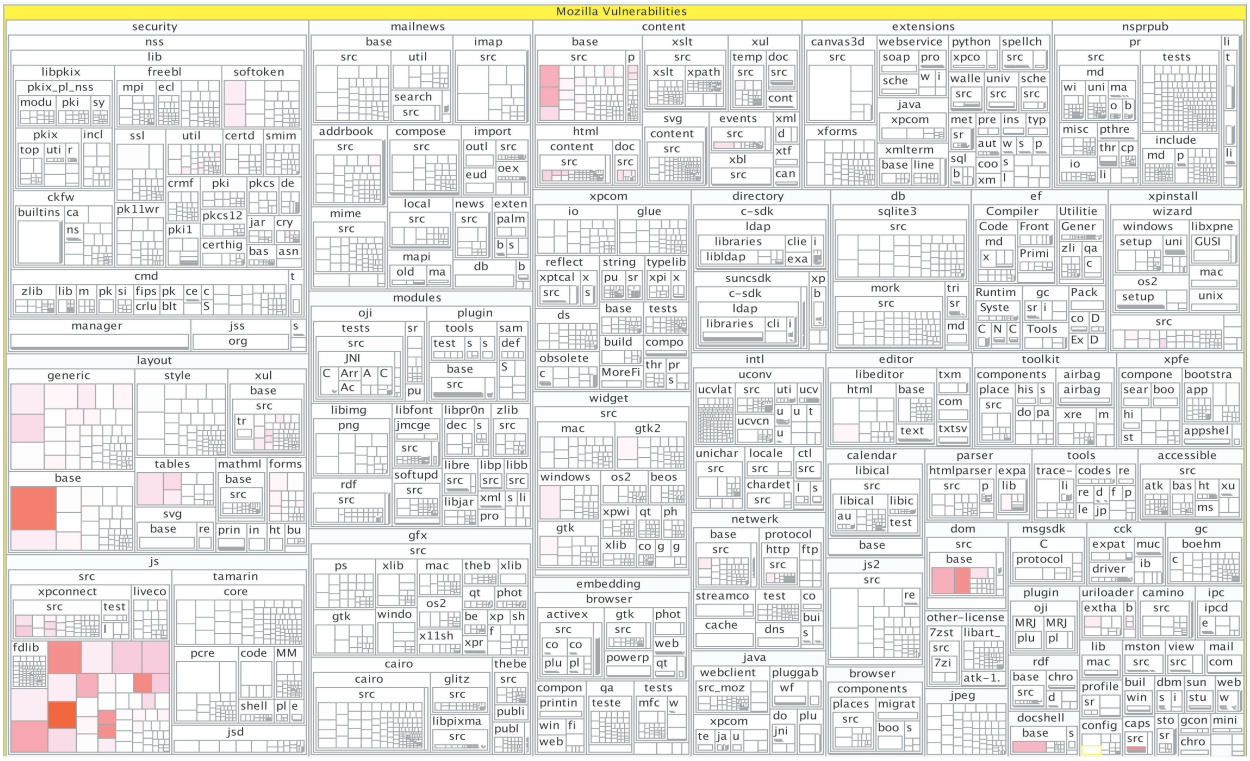


Figure 1: Distribution of vulnerabilities within Mozilla's codebase. A component's area is proportional to its size; its shade of gray is proportional to its number of vulnerabilities. A white box means no vulnerabilities, as is the case for 96% of the components.

In an investigation of the Mozilla vulnerability history, we surprisingly found that components that had vulnerabilities in the past were generally not likely to have further vulnerabilities. However, components that had similar imports were likely to be vulnerable.

Based on this observation, we were able to extend Vulture by a simple predictor that correctly predicts about two thirds of all vulnerable components. This allows developers and project managers to focus their their efforts where it is needed most: "We should look at *nsXPInstallManager* more closely, because it is likely to contain yet unknown vulnerabilities."

Also, most vulnerable components either occur in *clusters*, i.e., together in one directory, or as *singletons*, i.e., spread out across the directory tree. Components in clusters are likely to have more than one vulnerability, whereas singleton components generally have only one.

Targeting Physically Adressable Memory

Lexi Pimenidis*

*Lehrstuhl für Informatik IV
Ahornstr. 55, Aachen, Germany
pimenidis{at}i4.informatik.rwth-aachen.de

We introduce new advances in gaining unauthorised access to a computer by accessing its physical memory via various means. We will show a unified approach for using IEEE1394[Pro00, And99], also known as firewire, file descriptors and other methods to read from and write into a computer's memory in a more structured way than was possible before[BDK05].

Modern CPUs provide virtual address spaces for each process running on the system, where pages are scattered though the physical memory, or are even swapped to external media, like e.g. a hard drive. Our advances are based on searching and parsing data structures in a computer's physical memory in order to reassemble scattered pages and being able to read and write in each processes virtual address space.

To this end we propose a set of heuristics for finding page tables and page directories in physical memory, as well as simple to use and generic APIs to access the memory through various means. With this access we can e.g. list the processes running on a computer, and even dump their memory content.

The power of this ability can be shown, by finding and extracting secrets in memory, like private SSH keys, but can be extended to any other kind of credentials, passwords, or valuable information[Bur06]. Besides attacking hosts, this power can also be used by law enforcement agencies in order to collect evidence from live systems for forensic purposes – here we implemented a proof of concept that links incoming and outgoing connections on a Tor node.

In a second step, we extend our influence on the target system by injecting arbitrary code in order to obtain interactive access with administrator privileges on the victim's computer. Practical problems that were dealt with include a careful choice of injection location, because shared memory may be mapped into different processes.

The methods introduced are adaptable to all kinds of operating systems and hardware combinations. As a sample target, we have chosen Linux on an IA-32 system with the kernel-options `CONFIG_NOHIGHMEM` or `CONFIG_HIGHMEM4G`, `CONFIG_VMSPLIT_3G` and `CONFIG_PAGE_OFFSET=0xC0000000`.

Future work includes injecting kernel space root kits, and porting the libraries to different platforms, esp. Microsoft Windows, thus providing a unified interface for law enforcement bodies.

References

[And99] Don Anderson. *FireWire System Architecture - IEEE 1394*. Addison Wesley, February 1999. ISBN 0201485354.

[Pro00] Promoters of the 1394 Open HCI. *1394 Open Host Controller Interface Specification*, January 2000.

[BDK05] Michael Becher, Maximillian Dornseif, and Christian N. Klein. FireWire - all your memory are belong to us, 2005. http://md.hudora.de/presentations/firewire/2005-firewire-cansecwest.pdf.

[Bur06] Mariusz Burdach. Finding Digital Evidence In Physical Memory, 2006. http://forensic.seccure.net/pdf/mburdach_physical_memory_forensics_bh06.pdf.

Anomaly detection with protocol parsing – A case-study of the FTP protocol.

René Gerstenberger

Fraunhofer FIRST.IDA
Kekuléstr. 7, 12489 Berlin, Germany
rene.gerstenberger{at}first.fraunhofer.de

Network Intrusion Detection Systems (NIDS) are commonly used in today's setup of IT infrastructures and provide essential functionality to identify and parry attacks on IT systems. Whilst the detection of known attack patterns using signature based detection methods has made substantial progress, the future challenge is to identify unknown attacks. Therefore the search for anomalies in network traffic can possibly enable another alternative to improve the detection-rate of the so called zero-day attacks in order to allow faster commencing of counter-measures against them.

Parsing of network traffic on the application level to extract information from the data stream is part of the functionality of common NIDS like Bro or Snort. Since most of current exploits are designed for vulnerabilities of application level software, it seems not sufficient to take a look on IP header information only but to take further information into account. This additional knowledge can possibly be obtained from the connection payloads and further used to analyse protocol specific behavior. The discriminative properties to divide normal and abnormal traffic can be embedded in syntactical and semantical characteristics which are worth to be investigated.

The main goal of this work is to combine protocol parsing and methods of anomaly detection derived from machine learning techniques. Syntactic as well as semantic information can be gathered by the parsing process. It is of great importance to implement efficient methods to compare these representations of the extracted data. One possibility are similarity measures based on attributed parse-trees. Therefore we assume that the underlying language of a protocol specification can be described by a context-free grammar although this may not be entirely possible for certain problems. A parser analyzing network traffic according to the given grammar specification extracts parse-trees and as a result provides the required syntactical and semantical information. Methods to compare sequences and parse-trees are the techniques used to analyse the data.

The implementation of such a parser as described above, is presented using the FTP protocol as an example. Examples of computation of similarity measures for parse-trees [1] and sequences [2] are illustrated. The results of the experiments show that the machine learning techniques incorporating abstract representations of structured data provide new possibilities to improve signature-based NIDS and enable better performance on zero-day attacks.

References

[1] Shawe-Taylor, J., Cristianini, N.: Kernel methods for pattern analysis. Cambridge University Press (2004)

[2] Rieck, K., Laskov, P.: Language models for detection of unknown attacks in network traffic. Journal in Computer Virology **4** (2007)

Anomaly Erkennung unter dem Dach von Bro

Christian Gehl

Fraunhofer-FIRST.IDA
Kekulèstr. 7, 12489 Berlin, Germany
cgehl {at} first.fraunhofer.de

Attacken sind im Netzwerkverkehr häufig auffindbar ohne den genauen Typ der Attacke zu kennen, da sie sich vom normalen Verkehr unterscheiden, oder ähnlich zu bekannten Attacken sind. Die Fähigkeit ein Ähnlichkeitsmaß für ein Protokoll der Applikationsebene zu definieren, hängt von den Informationen ab, die man aus dem Netzwerkverkehr akquiriert. Die Aufgabe bei der Informationstransformation ist es, eine strukturierte Representation zu finden, welche die zugrunde liegende Semantik des Orginalverkehrs reflektiert. Die Möglichkeit binpac [2] im Rahmen des Intrusion Detection Systems Bro als ein allgemeines und effizientes Werkzeug zum analysieren von verschiedenen Applikationsprotokollen zu benutzten, erlaubt die semantische Auswertung des Netzwerkverkehrs und ist somit nicht auf eine Payload basierte Anomalie Erkennung beschränkt. Anomalien leiten sich von einem normalen Modell ab, welches eine Grundlinie für akzeptiertes Netzwerkverhalten wiedergibt und lassen sich somit auf jedes Verhalten zurückführen, das aus diesem Modell fällt.

Das normale Modell wird von einer inkrementellen one-class Support Vector Machine [1] konstruiert, um unüberwachtes online Lernen und Erkennung zu ermöglichen. Die SVM verarbeitet die strukturierten Netzwerkdaten indem sie Netzwerkereignisse auf Datenpunkte in einen Merkmalsraum abbildet und die Projektionen nach normal und anormal in diesen Raum separiert. Die Repräsentation von Netzwerkereignissen erhält man, während die Transformation der unverarbeitet Bytes in eine definierte grammatikalische Darstellung des Applikationsprotokolls von binpac ausgeführt wird. Ein Datenpunkt in solch einer Darstellungsweise ist vom Protokoll abhängig und Ähnlichkeit zwischen zwei Datenpunkten wird durch das innere Produkt der Projektionen in dem Merkmalsraum festgelegt. Die gewählte Abbildung und die extrahierten Merkmale sind die entscheidenden Faktoren für den Anomalieerkennungsalgorithmus, um eine gute Trennung zu erzielen.

Unsere Experimente, die diese Probleme behandeln, konzentrieren sich auf das HTTP Protokol und lassen sich sowohl für andere Protokolle, die binpac analysieren kann, als auch für jeden anderen high-level Netzwerkprotokollparser leicht adaptieren. Brauchbare Merkmale des HTTP Protokolls können die Länge, N-Gram Struktur oder andere Eigenschaften eines HTTP Requests oder jeder andere Bestandteil sein. Welche Merkmale am aufschlussreichsten für die Ähnlichkeit sind, ist noch Bestandteil der Arbeit. Der Algorithmus hat aber bereits jetzt seine Tauglichkeit für die Echtzeitverarbeitung in verschiedenen Konfigurationen mit überzeugenden Resultaten auf echten Datensätzen bewiesen Die gegenwärtige Arbeit konzentriert sich auf die Merkmalsextraktion, Merkmalsreduktion, Konfigurationseinstellungen und Erforschung der Robustheit gegen polymorphe vermischte Attacken, als auch den Einsatz im wirklichen Netzwerk.

Literatur

[1] Pavel Laskov, Christian Gehl, Stefan Krüger, and Klaus-Robert Müller. Incremental support vector learning: Analysis, implementation and applications. *J. Mach. Learn. Res.*, 7:1909–1936, 2006.

[2] Ruoming Pang, Vern Paxson, Robin Sommer, and Larry Peterson. binpac: a yacc for writing application protocol parsers. In *IMC '06: Proceedings of the 6th ACM SIGCOMM on Internet measurement*, pages 289–300, New York, NY, USA, 2006. ACM Press.

Applying Image Analysis Methods to Network Traffic Classification

Thorsten Kisner*, Alex Essoh and Prof. Firoz Kaderali

* FernUniversität in Hagen, D-58084 Hagen, Germany
thorsten.kisner, alex.essoh, firoz.kaderali{at}fernuni-hagen.de

The issue of IT-Awareness has been of great interest in the last few years. Not surprisingly when the dramatic increase of incidents and vulnerabilities over the past ten years is taken into consideration. Therefore, many organisations try to protect themselves by enforcing security guidelines or even a security policy for their network. In most cases a security officer (network administrator) is responsible for the implementation of this security policy. For example the firewall policy of an organisation could be to block diverse ports: the telnet port (23), the standard port where the MySQL server runs (3306) and common P2P ports or only allow incoming traffic on Port 80. This solution only provides limited protection with regard to enforcing security policies for several reasons. Firstly, malicious users are able to bypass the firewall by configuring their server to listen to port 80 although the service offered is not HTTP and forbidden. Secondly using HTTP Tunneling, encapsulated in HTTP packets, data packets of applications which are normally not permitted can reach their destination by bypassing the firewall and violating the security policy of the organisation. To address this problem, diverse methods have been proposed with the aim of identifying the service related to traffic flow (see [1] and related literature).

In our work we examine the in- and outgoing network traffic of two different servers (SMTP and HTTP). The network traffic is measured at the gateway to the external network with the built-in packet and byte counter of `iptables`. We use these traces to build time series and are able to map them to input data for digital image processing. Haralick et al. [2] proposed 14 criteria extracted from the GLCM (Grey Level Co-occurrence Matrix) to describe a texture and used them as an input vector for a classificator. Conners et al. pointed out six significant parameters from the original 14 in [3] and we use the *Correlation (CORR)* as the seventh GLCM parameter of our dataset. Four of our parameters (*Angular Second Moment (ASM)*, *Entropy (ENT)*, *Correlation* and *Inverse Difference Moment (IDM)*) expose a significant difference between SMTP and HTTP traffic which we use to categorise the type of network traffic. We then use the k-Nearest-Neighbors (kNN) algorithm to classify our data with an accuracy of 90%.

References

[1] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin Soule and Kave Salamatian (Hrsg.): *Traffic Classification On the Fly*, *Computer Communication Review*, volume 36, number 2, April 2006, 23-26. ACM Press NY.

[2] Robert Haralick, K. Shanmugam and Its'hak Dinstein (Hrsg.): *Textural features for image classification*, *IEEE Transactions on Systems, Man, and Cybernetics*, volume 3, number 6, November 1973, 610-621.

[3] R.W. Conners, M. M. Trivedi and C.A. Harlow (Hrsg.): *Segmentation of a High-Resolution Urban Scene using Texture Operators*, *Computer Vision, Graphics and Image Processing*, volume 25, 1984, 273-310.

Exploring New Ways in Malware Detection

Thorsten Holz

Laboratory for Dependable Distributed Systems
University of Mannheim
thorsten.holz@informatik.uni-mannheim.de

Malicious software (*malware* for short) is an increasingly severe plague of the Internet. While the first computer virus, *Elk Cloner*, which started to spread in 1982, was a relatively simple program that infected boot sectors of floppy disks, malware today can spread autonomously over the network, exploiting weaknesses of network services, generating new variants of itself on the fly, and preventing analysis with the help of executable packers or crypters. In the form of botnets – networks of compromised machines that can be remotely controlled by an attacker – malware is one of the biggest threat to today's Internet.

The common approach to detect malware which prevails today is to use *blacklisting*. Blacklisting is the technique commonly employed by common antivirus software: the antivirus scanner maintains a database of malicious files and checks a given system against this database. For efficiency reasons, not the whole file is compared by only characteristic strings or *signatures* of malicious files. Signature-based malware detection has inherent drawbacks. Firstly, it is necessarily *reactive*: the attacker is always one step ahead because a new signature can only be entered into the database *after* a new malware sample has been caught and analyzed. Signature creation and signature distribution are therefore also a question of time. Secondly, signature-based detection does not scale. Modern antivirus engines already store a database of more than 200,000 entries (e.g., the antivirus engine by McAfee), more are added every day. It is clear that eventually the scanning process will become infeasible for normal computers. Finally, every antivirus scanner can be turned against itself: An attacker can use available scanners as "test oracles" for his own malware and tune it until it is not detected.

In this project, we claim that the attackers are eventually winning the battle against blacklisting approaches. Using honeypot technologies, we collected large amounts of current malware and used this malware to test current antivirus products. We show that the detection rates of malware for three current threats in the Internet are unacceptably low: only roughly 50% of malware distributed via botnets is detected, malware available through drive-by downloads is detected in 60% of the cases, and polymorphic worms are detected in roughly 98% of the cases. Based on these detection rates we argue that signature-based detection is not able to respond sufficiently to current threats and that new alternative ways should be explored.

We speculate on three possible ways to improve the detection rates of antivirus engines and therefore take back the initiative from the attackers. We explore how *whitelisting*, *machine learning*, or *behavior-based* approaches can be used to enhance the detection capabilities. Briefly spoken, whitelisting is the opposite of blacklisting and identifies "good" binaries instead of bad ones. Machine learning tries to analyse typical patterns in malware and detect them in a much more flexible and automatic way than signatures. Behavior-based methods finally detect malware based on automatic behavior analysis, not based on static memory layouts. For each method, we look at some experimental results and discuss their advantages and limitations.

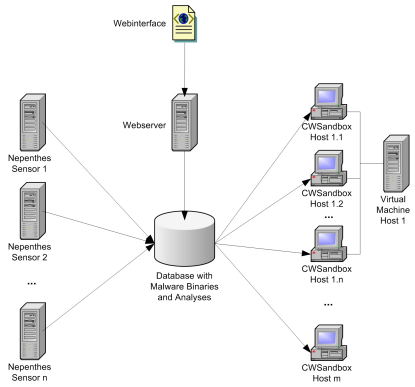
Automatic Behavior Analysis with CWSandbox

Carsten Willems

Laboratory for Dependable Distributed Systems
University of Mannheim

Malicious software like viruses, worms, and bots are currently one of the largest threats to the security of the Internet. Such malware must be analyzed to determine its effects on the running and remote systems and the infection methods it uses for further spreading. Nowadays, most malicious applications use multiple techniques (compression, encryption, code obfuscation, invalid opcodes, ...) for making a traditional static code analysis very difficult and time consuming. Accordingly, other methods - like behavior analysis - are very helpful. Furthermore, the speed in which malware spreads and the large number of new malware samples which appear every day, calls for automation. CWSandbox is an approach to automatically analyze dangerous malware which is based on behavior analysis: malicious samples are executed for a finite time in a simulated environment where all system calls are closely monitored. From these observations, the CWSandbox is able to automatically generate a detailed report which greatly simplifies the task of a malware analyst. Amongst other things, such a report contains information about all accesses to the local filesystem, the registry, virtual memory, the network, or other processes.

To perform an automated analysis on a larger scale, the sandbox is embedded into the Automatic Analysis Suite[AAS], which is based on a central malware binary and analysis repository. New malicious samples can be submitted to it automatically by Nepenthes [Nep] sensors or manually via a web interface. Several physical and virtual hosts are used to run CWSandbox automatically on new submitted samples. After their analysis has been completed, the results are written back to the repository and the sandbox hosting machine is brought back into a clean initial system state to undo the side effects of the malware execution.



References

[Nep] Markus Koetter, Paul Baecher, Thorsten Holz, Felix Freiling, and Maximillian Dornseif. The Nepenthes Platform: An Efficient Approach to Collect Malware. In 9th International Symposium On RAID 06, Hamburg, Germany, Sep 20-22, 2006, Proceedings, LNCS. Springer, 2006.

[AAS] Carsten Willems, Thorsten Holz, Felix Freiling. Toward Automated Dynamic Malware Analysis Using CWSandbox. IEEE Security and Privacy, vol. 05, no. 2, pp. 32-39, Mar/Apr, 2007.

Truman Box: Safe Malware Analyses by Simulating the Internet

Christian Gorecki*

*University of Aachen (RWTH)
D-52056 Aachen, Germany
christian.gorecki{at}post.rwth-aachen.de

Nowadays, there are several software tools allowing us to collect malicious software, so called *malware*, in a secure environment. The *honeypot* implementation called *nepenthes* (`http://nepenthes.mwcollect.org/`) is only one example how to gather malicious programs spreading over the Internet by simulating known vulnerabilities. Once a binary has been isolated, there are other tools doing further analyses in an automated manner to a certain degree. Considering for example the *CWSandbox* (`http://www.cwsandbox.org/`), we can see that good results can be obtained by this approach.

Aiming for the best possible dynamic analyses of malware, analyses can be improved considerable allowing certain traffic to the Internet so the malware is able to process updates and receive further instructions. By doing so, we have to take the risk of infecting other systems during the analyses phase. In order to avoid this, we need a *simulation of the Internet*. This work is about implementing a transparent bridge providing different generic network services and redirecting traffic to those. Inserted between Internet and client (network), the system is meant to work in four different modes:

- *Simulation*: the system emulates typical network services and is able to bind those dynamically to different ports. In particular, we achieve that standard protocols can be served on an arbitrary port.
- *Half-Proxy*: extension of the simulation mode, where usefull information are gathered from the Internet in order to provide a behavior which is closer to what the clients would see, without interception of the Truman Box.
- *Full-Proxy*: the system is operating as a transparent proxy and has filtering capabilities as well as the possibility to alter payloads.
- *Transparent*: only passive logging of in– and outgoing data traffic without any blocking or redirecting.

Depending on the needs, the system can be configured to process the bypassing traffic in one of these modes. The network services should be reachable on any arbitrary port to grant port independence, *e.g.*, for standard services offered on non-standard ports. Therefore we use a hybrid protocol identification driven not only by header information, but also by connection payloads. To gather as much information as possible, the services are generic and give reasonable response even to uncommon requests. So far the following services are supported: DNS, FTP, HTTP, IRC and SMTP.

Interception of data packets happens on layer 2 of the ISO Reference Model for OSI, in order to guarantee the transparency of a bridge. The entire system should operate as a standalone device, *i.e.*, it should not depend on another system.

Distributed monitoring and analysis for reactive security

Tobias Limmer and Falko Dressler

University of Erlangen-Nuremberg, Department of Computer Science 7
(tobias.limmer| dressler){ at } informatik.uni-erlangen.de

An ever growing number of attacks to sites on the Internet increases the need for effective systems to detect those incidents and initiate countermeasures. Botnets are one of the most successful methods used by hackers. Those networks are difficult to detect and neutralize, but an especially difficult task is to find the individuals controlling the attacking hosts. A crucial part of successful dissection is the analysis of monitored data in several separate networks. Recently, we introduced our monitoring toolkit Vermont to enable distributed aggregation of such information.

Analysis of network monitoring data can be divided into two different types: payload-based and flow-based analysis. On the one hand, network monitors like Snort and Bro analyze complete network packets including payload. In this area usually signature-based detection methods are used. On the other hand, flow-based analysis summarizes monitored packet data to only include header information and to aggregate packets with common attributes to one record. This method achieves great information reduction but still provides enough data for following analysis to generate meaningful results [1]. It is mostly used in high-speed environments, where payload-based monitoring is not suitable due to performance problems. The most popular data format for transferring flow-based data is Cisco's Netflow v9, which is widely supported in hardware-based routers, and its successor IP Flow Information Export (IPFIX).

We are using Vermont, which runs under Unix-based operating systems, for flow-based analysis of network streams. It supports direct observation of packet data using the PCAP system library with subsequent aggregation and export as IPFIX flows. Special attention was paid to high flexibility and support of dynamic reconfiguration of its aggregation parameters. The application is also able to receive IPFIX flows, so it is hierarchically stackable. This enables it to be used in distributed high bandwidth environments, as load can be shared among several hosts. For example, the tasks for portscan detection, statistical analysis and anomaly detection can be performed on separate hosts, which work on identical data. If the task is still computationally too expensive, it could be split up in two parts and executed on two machines. Further possibilities of attack detection lie in the areas of detecting DoS-attacks, vertical and horizontal portscans, hosts sending spam mails or hosts, just in the process of being infected when downloading malware code. Those methods benefit from distributed monitoring stations which gather data in different networks and forward it to a centrally managed correlation system. Possible reactions to detected incidents may include the reconfiguration of firewalls which could help to prevent malware from spreading or to prevent the overload of analysis systems from a targeted DoS attack to cover more subtle attacks.

References

[1] Falko Dressler and Gerhard Münz: *Flexible Flow Aggregation for adaptive Network Monitoring*, pages 702-709 in *31st IEEE Conference on Local Computer Networks (LCN): 1st IEEE LCN Workshop on Network Measurements (WNM 2006)*, Tampa, Florida, November 2006.

Internet-Frühwarnung auf Basis des Internet-Analyse-Systems

Sascha Bastke

Institut für Internet-Sicherheit
D-45877 Gelsenkirchen
Sascha.Bastke@internet-sicherheit.de

Mit steigender Bedeutung des Internet wächst auch das Schadenspotenzial, das durch Ausfälle der Infrastruktur Internet verursacht werden kann. Einher mit dieser Feststellung geht die Feststellung, dass Systeme zum Schutz des Internet fehlen bzw. unzureichend sind. Unternehmen schützen ihre Netze zwar durch Intrusion Detection Systeme aber ein System, das eine globale Sicht auf das Internet bzw. dessen Sicherheitslage zulässt und im Bedarfsfall zuverlässig Alarme generiert existiert nicht.

Basierend auf dieser Erkenntnis hat das Institut für Internet-Sicherheit das Internet-Analyse-System ([1]) entwickelt. Es handelt sich um ein System, das auf Basis von Daten, die durch im Internet verteilte Sonden gesammelt werden, den Zustand des Internet erfassen und bewerten soll. Das Prinzip der Datenerfassung ist relativ simpel. Für jede beobachtete Eigenschaft eines Netzwerkpaketes wird ein Zähler verwaltet. Diese geben an, wie viele Pakete mit der Eigenschaft bereits an der Sonde vorbei gekommen sind. Dabei werden Eigenschaften, die datenschutzrechtliche Relevanz haben könnten, von der Zählung ausgeschlossen. Die Zählung selbst erfolgt periodisch, d.h. nach Ablauf eines Intervalls werden die Daten an eine Zentrale übertragen und die Zählung beginnt von neuem. Diese Art der Datenerfassung hat zwei Vorteile. Erstens ist die anfallende Datenmenge relativ gering. So wird es möglich, Daten über einen langen Zeitraum zu speichern und somit eine Wissensbasis aufzubauen. Zweitens wird der Datenschutz sichergestellt, was einen realen Einsatz des Systems erleichtert.

Im Mittelpunkt dieses Projekts steht aktuell die automatisierte Erkennung von Anomalien und Angriffen. Dazu werden derzeit zwei Fragestellungen intensiv behandelt. Einmal wird die Frage nach der Ausdrucksmächtigkeit der gesammelten Daten untersucht. Es geht darum festzustellen, welche Angriffe mit den gesammelten Daten erkannt werden können und welche nicht. Teil dessen ist auch die Frage nach Erweiterungen des Prinzips der Datensammlung, um auch bisher nicht erkennbare Angriffe erkennen zu können. Erste Ergebnisse zeigen das insbsondere Angriffe, die sich durch starke Verhaltensänderung im Netzwerkverkehr wiederspiegeln, erkannt werden können. Mögliche Erweiterungen gehen in die Richtung nach Angriffssignaturen in den Netzwerkpaketen zu suchen und deren Auftreten zu zählen.

Bei der zweiten Fragestellung geht es um die Entwicklung von Methoden zur Anomalieerkennung. Hier liegt ein Augenmerk auf dem Einsatz von Methoden der KI, insbesondere von Neuro-Fuzzy-Systemen. Diese bieten eine Möglichkeit Lernverfahren mit Expertenwissen zu koppeln und so die Erkennungsleistung zu verbessern. Weiterhin steigt die Nachvollziehbarkeit der Entscheidungen, da das System auch als Menge interpretierbarer Fuzzy-Regeln darstellbar ist.

Ein weiterer Schwerpunkt des Projekts sind Prognosen, die es erlauben Technolgietrends zu bestimmen und die Verkehrsentwicklung zu beobachten. Hier werden Zeitreihenansätze und Neuronale Netze eingesetzt.

Literatur

[1] Prof. Dr. Norbert Pohlmann and Dipl.-Inform. (FH) Marcus Proest: *Die globale Sicht auf das Internet*, 2006

Ein nationales IT-Frühwarnsystem - Herausforderungen, Technologien und Architekturen

Alexander Burris, Ulrich Flegel, Johannes Hoffmann, György Kohut, Christoph Leuzinger, Michael Meier

Universität Dortmund
{burris, flegel, hoffmann, kohut, leuzinge, meier}{at}ls6.cs.uni-dortmund.de

Neben präventiven Sicherheitsmaßnahmen gewinnen Maßnahmen und Mechanismen der reaktiven Sicherheit an Bedeutung. Voraussetzung für eine Reaktion auf Sicherheitsvorfälle ist jedoch deren rechtzeitige und zuverlässige Erkennung. Ein kooperativer Ansatz verschiedener Institutionen ermöglicht den Austausch von Beobachtungen sicherheitsrelevanter Aktivitäten, Erkennungsergebnissen und Informationen zu möglichen Reaktionen auf erkannte Vorfälle.

Die Entwicklung eines solchen IT-Frühwarnsystems stellt wesentliche Herausforderungen. Einerseits ist eine Zusammenführung von Bedrohungen und Alarmen aus den einzelnen Institutionen in einem zentralen Repository notwendig. Andererseits haben die beteiligten Institutionen ein Interesse daran, dass Dritte die übermittelten Daten nicht der entsprechenden Institution zuordnen können. Ansätze zur Lösung dieses Interessenkonflikts bieten Informationsreduktionsverfahren (z.B. Pseudonymisierungsverfahren). Desweiteren müssen Kriterien zur Erkennung bekannter und unbekannter automatisierter Angriffe, etwa durch Malware, zeitnah automatisch generiert werden.

Die Architektur des Frühwarnsystems koppelt verschiedene Basistechnologien um den genannten Anforderungen gerecht zu werden (s. Abb. 2). Zur Aufzeichnung bereits bekannter und noch unbekannter Malware werden Honeypots eingesetzt (Malware-Kollektor), z.B. Nepenthes. Die Malware wird zur Beobachtung in einer kontrollierten Umgebung ausgeführt (Sandbox), z.B. CW-Sandbox. Aus diesen Beobachtungen werden unter mittels maschineller Lernverfahren Malware-Erkennungskriterien generiert (Signatur-Generator). Diese werden zentral bereitgestellt und automatisch an signaturbasierte Erkennungssysteme verteilt, z.B. Intrusion Detection Systeme (IDS). Die von Erkennungs- und Alarmboxen auf der Grundlage der bereitgestellten Signaturen generierten Alarmmeldungen werden zur Erstellung eines Lagebildes an ein Lagezentrum übermittelt. Zur Berücksichtigung von Datenschutz- und Vertraulichkeitsinteressen werden geeignete Informationsreduktionsverfahren entwickelt.

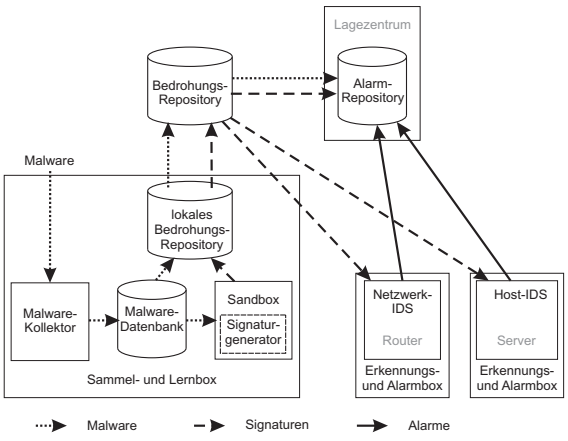


Abbildung 2: Architektur IT-Frühwarnsystem