

Technische Universität Berlin
Seminar Internet Sicherheit
Betreuer: Fabian Schneider und Bernhard Ager

DYNAMIC APPLICATION-LAYER PROTOCOL ANALYSIS FOR NETWORK INTRUSION DETECTION

Zusammenfassung

Dieser Text ist eine Zusammenfassung der Veröffentlichung: “Dynamic Application-Layer Protocol Analysis for Network Intrusion Detection” von H. Dreger, A. Feldmann, M. Mai, V. Paxson und R. Sommer [1], angefertigt für das Seminar “Internetsicherheit”, an dem Lehrstuhl für Intelligente Netze und Management verteilter Systeme (INET), Research Group Prof. Anja Feldmann, der Technischen Universität Berlin.

Elisa Jasinska
jasinska@informatik.hu-berlin.de
23. Januar 2007

1 Einleitung

Diese Zusammenfassung beschreibt eine dynamische Applikations-Layer Protokoll Analyse in Netzwerk-Basierten Intrusion-Detection-Systemen. Zunächst wird in Kapitel zwei eine Definition von Intrusion-Detection-Systemen gegeben, anschließend in Kapitel drei die Problemanalyse definiert. Kapitel vier geht genauer auf das Network-Intrusion-Detection-System (NDIS) “Bro” ein, auf dem das hier vorgestellte System basiert, und in Kapitel fünf werden die Funktionalität sowie in Kapitel sechs die Testergebnisse des beschriebenen Systems vorgestellt.

2 Intrusion-Detection-Systeme

Ein Intrusion-Detection-System (IDS) erkennt Angriffe auf Computersysteme. Im Gegensatz zu einem Intrusion Prevention System (IPS), welches Angriffen vorbeugt, hat ein IDS keinen Mechanismus um Angriffe zu bekämpfen. Es werden drei Arten von Intrusion Detection Systemen unterschieden:

- **Host-Basierte IDS (HIDS)**
Hostbasierte ID Systeme werden auf dem zu überwachenden Computer installiert und beobachten dieses System lokal.
- **Netzwerk-Basierte IDS (NIDS)**
Netzwerk-Basierte Intrusion-Detection-Systeme überwachen passiv den kompletten Paketverkehr eines Netzwerkes. Dabei versuchen sie Angriffe zu erkennen und zu melden.
- **Hybride IDS**
Ein Hybrides Intrusion-Detection-System verbindet die Eigenschaften der host- und netzwerkbasierten IDS.

Im folgenden geht es um eine Erweiterung des NIDS “Bro” [2] zur zuverlässigeren Erkennung von Anomalien in Netzwerken, wie zum Beispiel die Existenz von Bot-Netzen, Warez-FTP-Servern und Spam-Relays.

3 Problemanalyse

Existierende Network-Intrusion-Detection-Systeme benutzen meist Pattern Matching und Zustandsanalysen, um versuchte oder erfolgreiche Einbrüche in Rechnernetze zu erkennen.

Pattern-Matching untersucht das Paket auf das Vorkommen bestimmter für Angriffe typische Zeichenfolgen. Es lässt sich für eine beliebige Anzahl von zu erkennenden Mustern mit Hilfe von beispielsweise des Aho-Corasick-Algorithmus [3] in konstanter Zeit implementieren. Ein bekanntes und populäres NIDS, das mit Pattern-Matching arbeitet, ist Snort [4]. Bei Pattern-Matching wird jedoch der Zustand des Applikationsprotokolls nicht berücksichtigt, es wird also nicht gespeichert ob eine Verbindung¹ bereits besteht. Im Gegensatz hierzu wird dies bei einer Zustandsanalyse durchgeführt.

Eine Zustandsanalyse [5], *Stateful-Packet-Inspection* (SPI) genannt, ist die Zuordnung von Datenpaketen zu Protokollzuständen. Eine bestehende Verbindung und ihr Fortschritt wird gespeichert, so dass Pakete dieser Verbindung zugeordnet werden können und der Verbindungszustand in die Erkennung einfließen kann. Dies kann auf verschiedenen Protokollschichten durchgeführt werden, so wendet Snort [4] beispielsweise SPI auf der Ebene der TCP Verbindungen an. Mit einer tieferen Analyse, zum Beispiel auf der Applikations-Schicht, ist es auch möglich, Situationen wie das mehrfache Fehlschlagen eines Login-Versuches zu erkennen.

Zur Stateful-Inspection wird für jedes zu analysierende Protokoll ein sogenannter Protocol-Analyzer gebraucht, mit dem die SPI durchgeführt wird. Jedes ankommende Datenpaket muss dem passenden Protocol-Analyzer zugeordnet werden. Diese Zuordnung erfolgt herkömmlicherweise statisch anhand der Portnummern. Dies führt jedoch dazu, dass diese Analysetechnik versagt, wenn Protokolle auf anderen als den dafür reservierten Ports verwendet werden.

Die hier zusammengefasste Veröffentlichung [1] beschreibt ein Verfahren, Stateful Packet Inspection auch bei Verwendung von beliebigen Portnummern durchzuführen. Dazu wurde das Intrusion Detection System “Bro” [2] um eine im Folgendem näher beschriebene Funktionalität erweitert.

4 Bro Intrusion Detection System

Das Bro Network-Intrusion-Detection-System [6] ist in drei Schichten aufgebaut: die libpcap Schicht, die Event-Engine und der Policy-Script-Interpreter. Die Pakete werden durch diese drei Schichten durchgereicht, dabei wird durch libpcap [7] der Verkehr mitgeschnitten und vorgefiltert. In der Event-Engine werden weitere Filter angewandt und Events generiert und im Policy-Script-Interpreter Ereignisse ausgelöst (siehe [Abbildung 1](#)).

¹ Eine Verbindung wird definiert durch zwei IP-Adressen und zwei Port-Nummern.

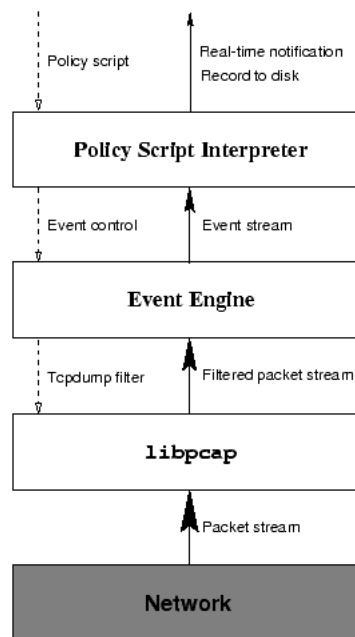


Abbildung 1: Struktur des Bro Systems

4.1 libpcap

libpcap ist eine für viele Betriebssysteme verfügbare Schnittstelle, die ein Framework für die Überwachung von Netzwerkdaten bereitstellt. Bro kann mit Hilfe von libpcap den Datenverkehr mitschneiden. Die Anwendung von libpcap ist von Vorteil, da Bro dadurch plattform- und netzwerkunabhängig ist. Weiterhin stellt libpcap einen Filter bereit, der die für die Event-Engine uninteressanten Pakete basierend auf statischer Port-Nummern-Zuordnung verwirft. Pakete für die die Event-Engine keine Analyzer bereithält können nicht weiterverarbeitet werden, daher wird uninteressanter Verkehr schon im ersten Schritt herausgefiltert und der Einsatz der Event-Engine reduziert.

4.2 Event-Engine

Die Event-Engine, an die die ausgewählten Pakete weitergeleitet werden, überprüft zunächst, ob das Paket wohlgeformt ist. Beispielsweise wird im Falle einer falschen Prüfsumme das Paket verworfen und ein Event generiert. Ist das Paket syntaktisch korrekt, wird es entweder einer bereits bekannten Verbindung zugeordnet oder ein neuer Verbindungseintrag generiert. Daraufhin wird das Paket an die der Verbindung zugeordnete Instanz des Packet Analyzers übergeben, welcher aus den Zustandswechseln des Application Layer Protocols weitere Events generiert.

4.3 Policy-Script-Interpreter

Der Policy-Script-Interpreter führt Skripte aus, die in der speziellen Bro-Sprache geschrieben sind. Diese Skripte reagieren auf Events und lösen Ereignisse aus, wie zum Beispiel die Benachrichtigung des Administrators im Fall von mehreren fehlgeschlagenen Login-Versuchen.

5 Dynamische Application-Layer Protokollanalyse

In dem im Folgendem beschriebenen System wird in einem ersten Schritt anhand von Signaturen² heuristisch eine Aussage über das verwendete Protokoll getroffen. Im zweiten Schritt wird dann durch einen passenden Protocol-Analyzer die getroffene Vermutung verifiziert oder widerlegt. Um das gleichzeitige Testen mehrerer Hypothesen über das verwendete Protokoll zu ermöglichen, wurden Methoden hinzugefügt, um eine baumförmige Struktur von Analyzer-Modulen erzeugen zu können.

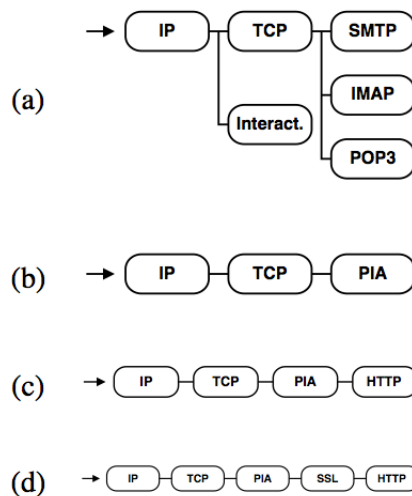


Abbildung 2: Analyseebäume

² Signaturen sind reguläre Ausdrücke, die Charakteristiken von bestimmten Protokollen beschreiben.

Innerhalb der Baumstruktur wird das Ergebnis des jeweils vorherigen Analyzers in den Nachfolgenden übergeben. Somit wird die Anwendung von parallelen Analyzern (siehe [Abbildung 2](#) (a) für die Erkennung eines beliebigen Emailprotokolls), und auch die Erkennung von verkapselten Paketen (siehe [Abbildung 2](#) (d)) ermöglicht.

Nach der Erkennung des Transportprotokolls³ wird PIA (Protocol-Identification-Analyzer) initialisiert, um das Applikationsprotokoll zu erkennen (siehe [Abbildung 2](#) (b)). PIA verwendet eine Reihe von Heuristiken um das entsprechende Applikationsprotokoll zu identifizieren. In der derzeitigen Implementation werden hierfür Signaturen eingesetzt, aber auch die Möglichkeit vorbehalten weitere Methoden zur Erkennung mit einzubinden (wie zum Beispiel auch statische Portnummern oder spezielle Erkennungsfunktionen). Weiterhin wurde eine “Prediction Table” (Voraussagungstabelle) implementiert, in der die Voraussagungen über erwartete Verbindungen samt der dazugehörigen Analyzer gespeichert werden. Beispielsweise wird bei der Erkennung einer FTP-Control-Session davon ausgegangen, dass darauf eine FTP-Daten Verbindung folgen wird; diese wird dann in der “Prediction Table” gespeichert, und die heuristische Erkennung dadurch beschleunigt.

Wurde ein Protokoll von PIA erkannt, so wird der jeweils dazugehörige Analyzer instanziiert (siehe [Abbildung 2](#) (c)). Zusätzlich setzt PIA die heuristische Analyse fort, bei einem weiteren Treffer wird ein weiterer Analyzer instanziiert. Schlägt ein Analyzer fehl, so wird dieser beendet. Daher verändert sich die Baumstruktur dynamisch.

Da zur korrekten Analyse einer Verbindung die initialen Pakete benötigt werden, muss auch später instanziierten Analyzern die Möglichkeit gegeben werden, auf den Anfang einer Verbindung zugreifen zu können. Um dies zu realisieren bietet PIA die Möglichkeit den Anfang einer Verbindung, bis zu einer konfigurierbaren Grösse⁴, zu puffern. Dieser Puffer wird dann als erstes an einen neu instanziierten Analyzer übergeben, bevor die neuen Daten weitergeleitet werden.

6 Ergebnisse

Die Implementierung des beschriebenen Systems umfasste zum Zeitpunkt der Veröffentlichung FTP, HTTP, IRC und SMTP-Analyzer. Mit diesen Analyzern wurde sie in den folgenden Netzwerken eingesetzt: University of California,

³ Im Moment existieren nur TCP Analyzer, somit wird immer zuerst IP und TCP initialisiert.

⁴ Voreingestellt sind 4KB in dieser Implementation

Berkeley (UCB), Münchener Wissenschaftsnetz (MWN) und Lawrence Berkeley National Laboratory (LBNL).

Alle drei Netzwerke zeichnen sich durch hohen Datendurchsatz aus. An der UCB nutzen ca. 45 000 Hosts drei mal 2 Gbps Uplinks, woraus ein Durchsatz von ca. 5TB/Tag entsteht. Das Münchener Wissenschaftsnetz bietet 1Gbps Uplink für ca. 50 000 Nutzer mit daraus resultierenden 1-3 TB Durchsatz pro Tag. Das LBNL hat 1Gbps Uplink für 15 000 Nutzer, resultierend in einem Durchsatz von 1.5 TB pro Tag.

Die folgenden Szenarien wurden mit der erläuterten Implementierung untersucht: Die Erkennung der Protokolle unabhängig von der Portnummer, die Nutzlastbetrachtung (Payload) von FTP-Verkehr und Erkennung von IRC-Basierten Botnetclients und -servern, wie im folgenden erläutert.

6.1 Nicht standardisierte Portnummern

Bei der Untersuchung von Paketen mit nicht standardisierten Portnummern wurden Datenpakete mit bekannten Portnummern⁵ verworfen, um die Last zu reduzieren. Sowohl an der UCB als auch am MWN entdeckte das System Server, die zuvor nicht von NIDS gefunden worden wären.

Innerhalb eines Tages wurden an der University of California, Berkeley, die folgenden Server auf nicht standardisierten Ports gefunden: 6 interne und 17 externe FTP Server, 568 interne und 54 830 externe HTTP Server, 2 interne und 33 externe IRC Server, 8 interne und 8 externe SMTP Server (siehe [Abbildung 3](#)).

Protokoll	gefundene lokale Server	gefundene externe Server
FTP	6	17
HTTP	568	54830
IRC	2	33
SMTP	8	8

Abbildung 3: Nicht standardisierte Portnummern an der UCB

Am MWN wurden während des gleichen Zeitraumes die folgenden Auswertungen gesammelt: 3 interne und 40 externe FTP Server, 108 interne und 18 844 externe HTTP Server, 3 interne und 58 externe IRC Server, 8 interne und 5 externe

⁵ 21, FTP; 6667/6668, IRC; 80/81/631/1080/3128/8000/8080/8888, HTTP; 25, SMTP; ausserdem zusätzlich 6665/6666/6669/7000, IRC und 587, SMTP

SMTP Server (siehe [Abbildung 4](#)).

Protokoll	gefundene lokale Server	gefundene externe Server
FTP	3	40
HTTP	108	18844
IRC	3	58
SMTP	3	5

Abbildung 4: Nicht standardisierte Portnummern am MWN

6.2 Nutzlastbetrachtung von FTP

Nachdem der Verkehr auf Nicht-Standard-Ports gefunden wurde, ist eine weitere Analyse der Daten möglich. Dazu wird ein spezieller Datei-Analyzer eingebunden, der die Nutzlast von FTP-Paketen übergeben bekommt. So könnte etwa libmagic⁶ genutzt werden, das eine Reihe an dateispezifischen Charakteristiken zur Verfügung stellt, um zu prüfen, welche Dateitypen übertragen werden. Leider finden sich zu dieser Analyse keine weiteren Angaben über deren Anwendungswerte.

6.3 IRC-basierte Botneterkennung

Botnetze sind Netze aus Computern (Bots) die ferngesteuert Anweisungen ausführen. Eine gängige Methode Befehle an Bots zu verschicken, ist das IRC Protokoll. Via IRC können zum Beispiel Befehle in den Topics an Bots gerichtet sein, daher wurde die Analyse der Botnetze und die der IRC Datenerkennung gekoppelt.

Die derzeitige Implementierung umfasst drei Heuristiken, um Bots zu erkennen: Der Nickname des Clients wird untersucht. Wenn dieser zum Beispiel einem bestimmten regulären Ausdruck entspricht, könnte es sich um einen Bot handeln. Weiter wird der Topic im Channel auf bekannte Kommandos untersucht. Wenn sich ein Client an einem bereits als Botserver identifizierten Server einloggt, wird dies vermerkt.

Am Münchener Wissenschaftsnetz wurden damit schnell 100 Bots entdeckt. An der UCB wurden innerhalb von 2 Wochen nur 15 interne Hostsysteme an den Administrator gemeldet.

⁶ Magic Number Recognition Library, <http://magicdb.org/>

7 Fazit

Eine dynamische Analyse von Protokollen bietet verbesserte Erkennungsleistungen für Bedrohungen von Rechnernetzen. Die Baumstruktur - die parallele Anwendung von verschiedenen Analyzern ermöglicht - und der Einsatz verschiedener Heuristiken sind dabei von grossem Vorteil.

In den getesteten Szenarien hat sich im Besonderen die dynamische Analyse von Protokollen, die nicht auf den dafür vorgesehenen Ports laufen, als sehr effektiv herausgestellt.

Es erscheint offensichtlich, dass dynamische Protokollanalysen in der Praxis die Erkennung von Sicherheitsrisiken in Zukunft erheblich verbessern können.

Literatur

- [1] H. Dreger, A. Feldmann, M. Mai, V. Paxson & R. Sommer. Dynamic Application-Layer Protocol Analysis for Network Intrusion Detection, http://www.net.in.tum.de/~hdreger/papers/USENIX_SEC06-DynAppAnalysis.pdf , In Proceedings of the 15th Usenix Security Symposium.
- [2] Bro Intrusion Detection System
<http://www.bro-ids.org/> .
- [3] Aho-Corasick-Algorithmus,
<http://de.wikipedia.org/wiki/Aho-Corasick-Algorithmus> .
- [4] Snort,
<http://www.snort.org/> .
- [5] Stateful Packet Inspection,
http://de.wikipedia.org/wiki/Stateful_inspection .
- [6] Vern Paxson, Bro: A System for Detecting Network Intruders in Real-Time,
<http://www.cs.rutgers.edu/dataman/552dir/papers/pax99.pdf> .
- [7] tcpdump & libpcap,
<http://www.tcpdump.org/> .