

Angriffserkennung – Einführung und Überblick über aktuelle Ansätze

Mathias Wagner
mail mathias-wagner info
Universität Passau

Hauptseminar IT-Sicherheit
bei Dr. Bernhard Sick

Januar 2007

Inhaltsverzeichnis

1	Überblick	3
2	Was ist ein IDS?	3
2.1	Sensoren	3
2.2	Analyse	5
2.3	Alarmgenerierung	5
3	Analysemöglichkeiten	5
3.1	Erkennen von Missbrauch	6
3.2	Erkennen von anomalem Verhalten	7
4	Management	7
5	Performance	8
5.1	Geschwindigkeit	8
5.2	Verfügbarkeit	9
5.3	Anti-IDS-Techniken	9
5.4	Fehlalarmattacken	9
5.5	Datenschutz	10
6	Maschinelles Lernen	10
6.1	Lernen durch Example Classification	11
6.2	Neuronale Netze	11
6.3	Genetische Algorithmen	12
6.4	Datenfusion	13
6.5	Entscheidungsbäume	13
6.6	Zustandsübergänge und Petrinetz-Modellierung	15
6.7	Graphische Darstellung	16
7	Derzeit erhältliche IDS	17
7.1	Snort	17
7.2	Dragon	18
7.3	Cisco Intrusion Prevention System	18
8	Ausblick	18
9	Fazit	19

1 Überblick

Angriffe auf Netzwerke sind an der Tagesordnung. Unternehmen sowie Privatanutzer sind auf die Funktionsfähigkeit ihrer internen Netzstrukturen sowie deren Verbindung zum Internet angewiesen. Wie lässt sich das Risiko, Schäden an der eigenen Netzinfrastruktur oder dem Datenbestand zu erleiden, vermindern? Neben dem Schutz vor Viren, Trojanern und Ähnlichem ist eine Kontrolle des gesamten Netzwerkverkehrs, sowie anderen Quellen wie Logdateien und anwendungsspezifischen Meldungen nötig, um gezielte Angriffe oder interne Manipulationsversuche möglichst früh zu erkennen. Dann müssen, am besten automatisiert, Gegenmaßnahmen eingeleitet oder der bereits entstandene Schaden kompensiert werden. **Intrusion Detection Systeme (IDS)** sind für genau diese Aufgabe gedacht. Ihre Funktionsweisen, Möglichkeiten und Lernmethoden sowie deren Schwachstellen sollen im Folgenden erläutert werden. Diese Arbeit stützt sich hauptsächlich auf [7] und [8] Seiten 83-90.

2 Was ist ein IDS?

Ein IDS ist eine von mehreren Komponenten zum Schutz von Computernetzwerken. Dazu zählen Firewalls, Virens Scanner, sowie Honeypots. Dies sind dedizierte IT-Systeme (Server, Netze, Programme, Prozesse), die keine produktive Funktion erfüllen, sondern ausschließlich „Fallen“ für Angreifer darstellen, in dem sie produktive oder auch besonders sicherheitskritische Systeme vortäuschen (nach [5] 1.3.2). Diese Systeme alleine sind jedoch nicht geeignet, Netzwerkverkehr innerhalb des eigenen Netzes zu überwachen, Anomalien zu erkennen und möglichst schnell darauf zu reagieren, indem Alarm ausgelöst wird. Erst in Kombination mit einem IDS können alle Arten von Angriffen erkannt und bewertet werden. In Abbildung 1 ist ein weit verbreiteter Aufbau eines IDS abgebildet. In den folgenden Abschnitten werden die Sensor-, Analyse- und Managementeinheiten näher betrachtet.

Auf die Möglichkeiten und Probleme bei der visuellen Darstellung ausgewerteter Daten wird in 6.7 kurz eingegangen. Das Deception System, zu Deutsch Täuschungssystem soll hier vernachlässigt werden. Eine Möglichkeit für eine solche Komponente wären die erwähnten Honeypots. Die in diesem Beispiel verwendete Datenbank dient zur Speicherung von Mustern, die von der Analyseeinheit zur Bewertung herangezogen werden. Wie in 6 noch gezeigt wird, muss eine Datenbank nicht zwingend verwendet werden. Die Vergleichsdaten können auch in einer anderen Form vorliegen.

2.1 Sensoren

Sensoren sind im Netzwerk platziert und sammeln Daten. Diese können aus Netzwerktraffic, Verhalten der Nutzer oder einer Anwendung bestehen. Dabei werden zwei Arten von anfallenden Daten unterschieden. Zum einen *Netzwerkaktivität* und zum anderen *Hostaktivität*. Erstere besteht aus gesammelten Daten

der unteren ISO¹-Schichten, sowie Anwendungsprotokollen wie HTTP² oder FTP³ in der ISO-Anwendungsschicht. Außerdem kann auch, ähnlich zu einem Virenschanner, der Inhalt von Dokumenten wie HTML⁴-Seiten gesammelt und zur Auswertung weitergeleitet werden. Zweitere betrifft auffälliges Verhalten von Endsystemen, wie Clients, Servern aber auch Routern. Anwendungen können manipulierte Pakete verschicken und Netzwerkdienste wie Samba⁵ oder DNS⁶ von der Spezifikation abweichendes Verhalten zeigen.

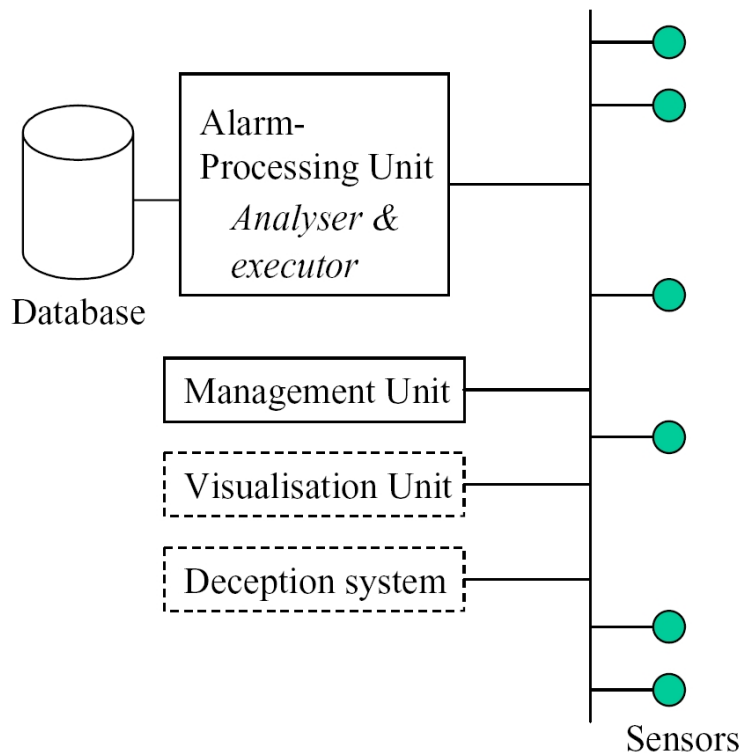


Abbildung 1: Möglicher Aufbau eines IDS (aus [7] Seite 3.)

Ein entscheidender Punkt bei der Installation von Sensoren ist die Art der Anbindung an das Netzwerk, um gesammelte Daten an die Analysestellen weiterzuleiten. Entweder die Sensoren verwenden die bereits vorhandene Infrastruktur oder für das IDS wird ein eigenes autonomes System errichtet. Ein zweites Netz ist äußerst kostenintensiv und benötigt zudem zusätzlichen Platz für Verkabelung und Router. Jedoch kann ein Angreifer, der das Hauptnetzwerk mittels Denial of Service (DoS) quasi deaktiviert, dennoch erkannt werden, da das IDS

¹Open Systems Interconnection Reference Model: Einteilung des Netzwerkstacks in sieben Schichten

²Hypertext Transfer Protocol

³File Transfer Protocol

⁴Hypertext Markup Language

⁵Open source Software, die SMB (Server-Message-Block-Protokoll) auf Unixsystemen verfügbar macht.

⁶Domain Name System

auf eigene Ressourcen zurückgreifen kann. In realen Systemen wird des Netzwerk, wegen der hohen Kosten geteilt. Im eigenen Netz können leicht Regeln für die Quality of Service (QoS) implementiert werden. Diese priorisieren die Kommunikation des IDS am höchsten. Sobald die Sensoren jedoch das eigene Netz verlassen und ausserhalb installiert werden, ist die Durchsetzung der QoS Regeln schwieriger und auf jeden Fall mit zusätzlichen Kosten verbunden. Dort kann eine eigene Leitung sinnvoll sein.

2.2 Analyse

Analyseeinheiten werden in zwei Klassen eingeteilt. Zum einen „Erkennen von Missbrauch“ und zum anderen „Erkennen von anomalem Verhalten“. Diese beiden Verfahren sind in Kapitel 3 näher beschrieben. Die Analyse kann zu unterschiedlichen Zeitpunkten stattfinden:

1. *Fortwährend*: Ereignisse werden sofort analysiert. Bei ausreichender Prozessorleistung ist dies in Echtzeit möglich.
2. *Periodisch*: Ereignisse werden gesammelt und in Intervallen analysiert. Beispielsweise können Logfiles täglich oder stündlich eingelesen werden.
3. *Manuell*: Vermutet der Administrator einen Angriff, so kann er die Analyse manuell starten.

2.3 Alarmgenerierung

Wird während des Analysevorgangs ein Angriff erkannt, so erzeugt das System einen Alarm. Dieser kann passiv oder aktiv sein. Dabei muss ein vom IDS erkannter Angriff nicht tatsächlich eine echte Attacke sein. Dies führt zu den in Abschnitt 3 beschriebenen Fehlalarmen.

Passiv: Der Alarm wird in Logfiles vermerkt, erscheint bei verantwortlichen Administratoren auf dem Bildschirm oder wird als Pager/SMS-Nachricht versandt.

Aktiv: Dazu müssen im System aktive IDS-Komponenten installiert sein. Diese können weitere Analysedaten einholen, einzelne Geräte abschalten oder Netzwerkverkehr zu Täuschungssystemen umleiten, um den weiteren Angriff schadlos für spätere Analyse Zwecke weiter zu verfolgen. Der einfachste, jedoch sehr wirkungsvolle aktive Eingriff wäre, die als schädlich eingestuft Pakete einfach zu löschen und somit den Angriff auf das Zielsystem zu verhindern.

3 Analysemöglichkeiten

Im Folgenden werden die zwei meistgenutzten Verfahren zur Angriffserkennung erläutert. Doch zuerst muss auf das Problem der „Falschen Alarme“ eingegangen werden.

Bei der Analyse können Vorgänge nicht nur korrekt, als erlaubt oder nicht erlaubt, erkannt, sondern auch fälschlicherweise erlaubt, oder zu Unrecht blockiert werden. Diesen Sachverhalt stellt Tabelle 1 dar.

	<i>Angriff</i>	<i>kein Angriff</i>
<i>IDS erzeugt Alarm</i>	Angriff erfolgt und Alarm ausgelöst (korrekt)	Kein Angriff erfolgt und Alarm erzeugt (falscher Alarm)
<i>Keine IDS Reaktion</i>	Angriff erfolgt und kein Alarm ausgelöst (fehlender Alarm)	Kein Angriff erfolgt und kein Alarm ausgelöst (korrekt)

Tabelle 1: Die vier Fälle bei der Bewertung von Netzwerktraffic durch ein IDS (nach [7] Seite 11).

Damit ist die *Genauigkeit* (nach [7] Seite 11) eines IDS als Anzahl korrekter Alarme dividiert durch die Anzahl korrekter Alarme plus falsche Alarme festgelegt. Je genauer ein IDS ist, desto weniger Fehlalarme werden produziert.

$$\text{Genauigkeit} = \frac{|korrekteAlarme|}{|korrekteAlarme|+|falscheAlarme|}$$

Die *Vollständigkeit* (nach [7] Seite 11) ist definiert durch Anzahl korrekter Alarme dividiert durch die Anzahl korrekter Alarme plus fehlende Alarme. Je vollständiger ein IDS ist, desto weniger Einbrüche werden übersehen.

$$\text{Vollständigkeit} = \frac{|korrekteAlarme|}{|korrekteAlarme|+|fehlendeAlarme|}$$

Im Idealfall sollten beide Werte 100% betragen. Dies ist in der Praxis jedoch nicht zu erreichen. Daher wird auf die Genauigkeit am meisten Wert gelegt, da zu viele Fehlalarme das Vertrauen der Administratoren in das IDS schwächt.

3.1 Erkennen von Missbrauch

Analyseprogramme die Missbrauch erkennen arbeiten mit dem Vergleichen von Signaturen oder Mustern. Diese sind als Teile bekannter Angriffsszenarien meist in einer Datenbank gespeichert und werden mit anfallenden Daten im Betrieb abgeglichen. Ist ein solches Muster erkannt, wird ein Alarm erzeugt. Unbekannte Angriffstechniken kann ein derartiges System nicht erkennen. Aufgedeckte Einbrüche können jedoch die ausgenutzte Schwachstelle im System sehr genau aufzeigen.

Praktisch ist es nicht realistisch, alle bekannten Angriffsmuster in das System einzupflegen und in allen Feinheiten anzupassen. Damit ist diese Technik alleine kein sehr guter Schutz. Durch Aufteilen der Signaturen in kleinere Stücke können schon Angriffsversuche oder Bestandteile von alten Mustern in neuen Angriffsstrategien erkannt werden.

In heutigen Systemen ist diese Technik am weitesten verbreitet. Dabei wird zwischen *smart* und *raw* Vergleichen unterschieden:

Smart: Der Datenstrom wird mit dem Wissen des zugehörigen Protokolls ausgewertet. Damit ist ein verbessertes Patternmatching (Vergleich von Mustern) möglich, da zum Beispiel zwischen Kommandos und Transportdaten unterschieden werden kann. Derartige Systeme müssen sich nach außen

hin so verhalten, wie es ein Client von einem echten Dienstanbieter erwarten würde. Der Vorgang der Überprüfung darf also nicht erkennbar sein. Nachteilig ist der hohe Zeitaufwand. Der Netzwerkverkehr muss an der Sensorstelle bis in die oberen Schichten des ISO/OSI Modells entpackt werden. Verwendung findet die Smart-Methode beispielsweise in RealSecure.

Raw: Bei dieser Methode entfällt das Auswerten nach Protokollen. Es wird nur der Inhalt eines beliebigen Paketes analysiert, ohne Interpretation von protokollspezifischen Besonderheiten. Der Hauptvorteil ist der geringe Geschwindigkeitsverlust und damit die Option höhere Datenaufkommen als bei der Smart-Methode zu überwachen.

Beispiel: Wird mittels der Smart-Methode in einem FTP GET⁷ Kommando der String /etc/passwd gefunden, so wird sehr wahrscheinlich ein Angriff vorbereitet. Steht dieser String hingegen in einer E-Mail die per POP⁸ übertragen wird, so muss kein Alarm ausgelöst werden. Bei der Raw-Methode hingegen kann die Analyseeinheit nicht unterscheiden ob dieser String tatsächlich als gefährlich eingestuft werden muss. Dazu sind weitere Informationen, oder die Begutachtung durch den Administrator notwendig.

3.2 Erkennen von anomalem Verhalten

Analysetechniken für anomales Verhalten basieren auf der Kenntnis von normalem Verhalten eines Nutzers oder Dienstes. Weicht eine Aktivität über einen bestimmten Schwellwert ab, so wird ein Angriff erkannt, obwohl es sich nicht zwingend um einen solchen handeln muss.

Beispiel: Am Arbeitsplatz einer Bürokräft loggt sich morgens um 9 Uhr stets die gleiche Person ein und nutzt ausschließlich ein Officeprogramm und den Webbrowser. Das Ausloggen findet zwischen 16 und 20 Uhr statt. Wird auf diesem System um drei Uhr morgens eine Session per SSH gestartet, so wird Alarm ausgelöst.

Dabei ergeben sich zwei Probleme. Zum einen das „Beschreibungsproblem“: Wie kann das Verhalten eines Nutzers oder Netzwerkdienstes kompakt und maschinenauglich beschrieben werden? Zum anderen das „Vergleichsproblem“: Falls ersteres Problem gelöst und bei einem realen Vergleich eine Abweichung erkannt wird, wie hoch muss diese sein, um einen Alarm zu rechtfertigen?

4 Management

Die einzelnen Komponenten wie Sensoren und Analysestellen in einem IDS können zentral, oder verteilt verwaltet werden. Dabei ist auf eine möglichst unkomplizierte Verwaltung zu achten. Dies übernimmt die Management-Einheit (siehe Abbildung 1). Zum Management gehören folgende Aufgaben:

⁷Befehl zum Laden einer Datei vom Zielsystem auf den eigenen Rechner.

⁸Post Office Protocol

Erkennungsmanagement beinhaltet die Kommunikation mit dem IDS über Konfigurations- und Logdateien, oder besser einem grafischen Interface. Dabei kann auch eine Nachkontrolle der anfallenden Daten durch den Menschen erfolgen.

Updatemanagement: Dauernd werden neuartige Angriffsstrategien erdacht und bei Attacken eingesetzt. Sobald eine solche Strategie als bestätigt gilt, muss bei signatur-basierten Systemen eine Signatur erworben oder selbst erstellt werden. Dann ist diese in das laufende IDS einzupflegen. Wie in den folgenden Abschnitten noch ersichtlich wird, ist dies nicht immer automatisierbar und kann den laufenden Betrieb beeinträchtigen.

Verfügbarkeitsmanagement behandelt die Problematik, das IDS permanent funktionsfähig zu halten. Dabei sind sowohl die Software als auch die Hardware kritisch. Es ist zwischen geplanten Abschaltungen auf Grund von Wartung und ungewollten Ausfällen auf Grund von DoS-Attacken zu unterscheiden.

Schutz des Management: Damit ist die Management-Einheit selbst in einem IDS gemeint. Es ist sicherzustellen, dass die Kommunikation zwischen IDS-Komponenten und dem Management nicht von Angreifern manipuliert oder unterbrochen werden kann. Dazu ist nicht nur die verschlüsselte Authentifizierung, sondern nach Möglichkeit auch die Verschlüsselung der Kommunikation zwischen den einzelnen Komponenten empfehlenswert.

Skalierbarkeit ist meist einfacher, wenn ein IDS nicht zentral verwaltet wird. Zusätzlicher Bedarf an Hardware kann punktgenau installiert werden. Dabei führt die Wartung und Erneuerung des Systems jedoch zu hohen Kosten. Ebenso steigt die Anfälligkeit für Fehler, da einzelne Komponenten leicht übersehen werden können. Somit hängt die Skalierbarkeit eng mit den anderen Aufgaben zusammen.

5 Performance

Neben den immer höheren Geschwindigkeiten, die in Netzwerken erzielt werden, betrifft die Performance auch Bereiche wie Verfügbarkeit, welche durch gezielte Attacken auf das IDS selbst beeinträchtigt wird. Diese und weitere Performanzkriterien werden in diesem Abschnitt vorgestellt.

5.1 Geschwindigkeit

Die Geschwindigkeit in heutigen Netzen beträgt an Knotenpunkten 1 GBit/s oder bereits 10 GBit/s. Ein Entwurf des Standardisierungsgremiums IEEE geht gar von marktreifen Produkten mit 100 GBit/s bis zum Jahre 2010 aus (IEEE 802.3 Higher Speed Study Group [6]). Zudem liegt diese Geschwindigkeit nicht nur auf einer Leitung, sondern in besagten Knotenpunkten mehrfach an. Systeme, die nicht nur die IP-Header inspizieren, sondern auf Anwendungsprotokollebene arbeiten, können hier nicht eingesetzt werden. Derzeit verfügt kein Router über die entsprechende Rechenleistung. Daher werden zumindest Stichproben analysiert. Bei den angesprochenen Geschwindigkeiten ist dies jedoch

nicht praktikabel. Das eigentlich zum Schutz des Netzwerks eingesetzte System übersieht die Mehrheit des eingehenden Traffics.

5.2 Verfügbarkeit

Ein weiteres wichtiges Kriterium ist die Verfügbarkeit. Diese kann nicht nur durch Attacken auf das IDS selbst, zum Beispiel durch DoS, verringert werden, sondern auch durch benötigte Drittsysteme. Wenn in einem System beispielsweise Logdateien als Komponente zur Auswertung des Traffics verwendet werden, so muss sichergestellt sein, dass der Server mit diesen Daten verfügbar ist. Und zwar nicht nur permanent für die Auswertung selbst, sondern ebenfalls für die Clients, die eventuell einem Angriff ausgesetzt sind und anfallende Informationen auf den Logserver schreiben wollen. Das bereits angesprochene Problem, dass bei Überlastung des Netzes Signale von Sensoren verloren gehen, muss berücksichtigt werden. Da selbst eine spezielle QoS Regel für IDS-Traffic nicht absolut verlässlich ist, kann ein zweites autonomes Netz notwendig sein.

5.3 Anti-IDS-Techniken

Eine Möglichkeit, ein IDS zu überlisten, besteht in der Einbettung spezieller, harmloser Befehle in ein Angriffsmuster. Ein derart getarnter Angriff kann einer Überprüfung nicht stand halten, da das IDS nicht weiß, wie verschiedene Betriebssysteme diese Pakete auswerten. Ein solcher Exploit funktioniert meist auch nur auf einem speziellen System. Hier gibt es zwei Lösungsansätze. Entweder es wird auf jedem Host ein IDS installiert, um Attacken auf betriebssystem- und versionsspezifische Eigenheiten zu erkennen. Oder die zentrale Analyseeinheit muss in der Lage sein anfallende Daten aus der Sicht von unterschiedlichen Systemen auszuwerten.

Zudem kann das Ziel eines Angriffs oft auf mehrere Arten erreicht werden. Die Analyseeinheit braucht daher für eine Angriffstechnik mehrere Signaturen, oder einen leistungsfähigen Präprozessor, der in der Lage ist, die unterschiedliche Abfolge von Befehlen auf immer gleiche Art anzuordnen. Ein IDS müsste im Idealfall zusätzliche Informationen über das Betriebssystem des Ziels eines Netzwerkpaketes haben. Nur dann kann individuell auf mögliche Angriffstechniken reagiert werden. Eine praktikable Möglichkeit, dies zu implementieren, ist das in Kapitel 8 beschriebene passive OS-Fingerprinting.

5.4 Fehllalarmattaken

Ein Angreifer kann in Vorbereitung des eigentlichen Ziels eine große Anzahl von Fehllarmen produzieren. Dabei handelt es sich nicht um Fehllarme im eigentlichen Sinn. Diese Meldungen können durchaus einem echten Angriff vorausgehen. Jedoch wird der echte Angriff in vielen verteilten Scheinangriffen versteckt. Das IDS wird mit falschen Angriffsversuchen geflutet. Trotzdem müssen die verantwortlichen Administratoren jeden Alarm untersuchen, da genau einer davon wirklich Schaden anrichten wird. Diesen einen Vorfall in der Menge zu erkennen, ist jedoch äußerst schwierig. Entweder die Kapazität zur Auswertung ist bereits erschöpft, oder die Auswertung liegt weit hinter den anfallenden Alarmen zurück. In derzeitigen Systemen ist, soweit bekannt, keine Vorrichtung

für derartige Attacken enthalten. Meist bleibt nur die Abschaltung betroffener Netzwerkteile, um den eigentlichen Angriff nicht doch zu übersehen.

5.5 Datenschutz

Ein IDS überwacht im Idealfall den gesamten Netzwerkverkehr und somit jede Aktivität von Nutzern. Dies steht in Konflikt mit Datenschutzrichtlinien. Es ist darauf zu achten, dass sensible Informationen nicht gespeichert und nicht zur Analyse von Nutzerverhalten, oder Fehlverhalten verwendet werden. Ein zusätzliches Problem sind die sehr unterschiedlichen gesetzlichen Regelungen in verschiedenen Ländern. Da die bekanntesten Hersteller, wie zum Beispiel Cisco, in den Vereinigten Staaten von Amerika angesiedelt sind, kann ein IDS beim Einsatz in Deutschland erhebliche rechtliche Bedenken hervorrufen. Daher schließen deutsche Unternehmen meist gesonderte Vereinbarungen mit dem Betriebsrat, die die Auswertung von Nutzerdaten zum Zwecke der Intrusion Detection zulassen, jede andere Anwendung jedoch explizit ausschließen.

6 Maschinelles Lernen

Definition „Regel“: Eine Regel für ein IDS beschreibt, wie ein bestimmtes Muster für anfallende Daten, wie zum Beispiel HTTP-Traffic, aussehen, bzw. nicht aussehen darf. Zudem wird die Reaktion auf ein erkanntes Ereignis festgelegt.

Definition „Maschinelles Lernen“: Ein Verfahren lernt maschinell in Bezug auf eine Aufgabe und ein Qualitätsmaß, wenn mit zunehmender Erfahrung die Aufgabe mit höherer Qualität gelöst wird (nach [11], Seite 13).

Die bisher vorgestellten Methoden zum Erkennen von Einbrüchen müssen maschinentauglich vorliegen und mit vertretbarem Aufwand in dieses Format konvertiert werden. Alternativ kann die Maschine selbstlernend programmiert werden. Hier gibt es zwei Arten des Lernens. Zum einen das überwachte Lernen. Dabei steht dem Programm während der Lernphase ein „Lehrer“ zur Verfügung, der bei der Bewertung behilflich ist. Dieser kann in Form des Menschen, oder als zusätzliche Software auftreten. Zum anderen das unüberwachte Lernen. Hier bewertet der lernende Algorithmus selbst. Beim maschinellen Lernen zählt die Lesbarkeit der erzeugten Regeln durch den Menschen nicht zu den wichtigsten Kriterien. Jedoch kann eine lesbare Regel bei der Entwicklung und dem Debugging sehr helfen. Viel wichtiger sind die Effizienz unter den Aspekten der Geschwindigkeit und des Speicherverbrauchs, sowie die Anpassungsfähigkeit an neue Daten und Angriffsszenarien. Generell kann man die meisten Vorgänge des maschinellen Lernens in diese Phasen einteilen:

Erzeugen von Testszenarien: Diese werden später zur Überprüfung der Güte der in der Lernphase erzeugten Regeln verwendet.

Sammeln von Rohdaten: Diese liegen normalerweise als Bytestrom, ohne zusätzliche Information vor.

Aufbereiten der Rohdaten: Der Bytestrom wird entsprechend des Protokolls aufbereitet. Zum Beispiel kann er in die einzelnen Pakete zerlegt werden. Zudem wird die Bedeutung für die Bewertung einzelner Abschnitte in den Teilstücken festgelegt.

Klassifizierung: Die aufbereiteten Daten werden beispielsweise in erwünschte und unerwünschte Pakete eingeteilt. Das System muss während der Lernphase wissen, ob ein Datensatz für den jeweiligen Zweck als gut oder schlecht eingestuft werden soll.

Lernphase: Der Lernalgorithmus generiert aus den klassifizierten Daten Regeln. Dies kann mehrmals wiederholt werden, um bessere Ergebnisse zu erzielen.

Testphase: Mit Hilfe der Testdaten wird die Erkennungsrate gemessen. Ist diese nicht ausreichend, so wird die Lernphase erneut durchlaufen. Dabei können neue bzw. andere Eingabedaten verwendet werden. Ist die Qualität den Anforderungen entsprechend, so kann das System im realen Betrieb eingesetzt werden.

Einige der aktuellen Forschungsansätze und bereits umgesetzten Techniken sollen im Folgenden kurz vorgestellt werden.

6.1 Lernen durch Example Classification

Bei der Klassifizierung von Netzwerkverkehr wird meist auf das positive, also normale Verhalten eines Nutzers trainiert. Zudem wird dies in vielen Systemen mit Informationen zu Fehlverhalten kombiniert. Daher ist diese Technik mit der in Kapitel 3.2 vorgestellten zu vergleichen, wobei die Muster nicht vom Menschen erstellt werden, sondern erlernt und im laufenden Betrieb selbständig angepasst werden.

Ein Ansatz für den Lernvorgang ist an die Biologie, genauer an das menschliche Immunsystem angelehnt. Dieser basiert auf der Verteilung, Unterschiedlichkeit, Unabhängigkeit und Selbstheilung der verteidigenden Komponenten. Da es sich nie verhindern lässt, alle Eventualitäten mit Hilfe von Regeln und Sicherheitsbestimmungen abzudecken, muss das System in der Lage sein, autonom auf unbekannte Ereignisse zu reagieren und Schäden zu beheben. Dies wird durch möglichst kurze „gutartige“ Sequenzen trainiert.

Vom Einsatz in realen Systemen wird diese Technik noch durch folgende Probleme abgehalten:

- Für maximale Effizienz ist viel Feintuning nötig. Zudem ist nicht immer ersichtlich, wie die optimalen Parameter aussehen müssen.
- Der Lernvorgang ist ressourcenaufwändig. Meist ist es nicht möglich, diesen in mehrere Phasen aufzuteilen. Wird eine Regel beispielsweise als veraltet, unnötig oder gar falsch identifiziert muss das System diese „vergessen“. Dies wird nur durch einen vollständigen Neudurchlauf des Trainings erreicht.
- Um gute Ergebnisse zu erzielen, muss die Datenmenge, die zu Trainingszwecken zur Verfügung steht, sehr groß sein. Noch dazu muss sie fehlerfrei sein, um Angriffe nicht fälschlicherweise als harmlos einzustufen.

6.2 Neuronale Netze

Definition: Ein neuronales Netz ist die Verschaltung von gleichartigen, relativ einfachen Bausteinen zu einem Netzwerk und die Informationsweiterleitung in

diesem Netz nach Prinzipien, wie sie bei biologischen Nervenzellen (Neuronen) vorliegen (nach [13] Seite 439).

Ursache für die Versuche, neuronale Netze zur Bewertung von Netzwerkaktivität zu verwenden, ist die bisher fehlende Möglichkeit, neuartige Angriffsmethoden zuverlässig zu erkennen. Im Forschungsbereich hat diese Technik gezeigt, dass sie prinzipiell Angriffsmuster erkennen kann. Zudem können verrauschte Daten ein solches System nicht sonderlich beeinflussen und die Bewertung erfolgt nach der Trainingsphase sehr schnell. Somit ist eine Echtzeitanalyse möglich.

Die Liste der Nachteile ist jedoch noch sehr lang:

- Geringe Mengen von Trainingsdaten generieren gute Ergebnisse. Datenmengen in echten Systemen sind um ein vielfaches größer und können den Lösungsraum somit exponentiell erhöhen. Dies kann zu einer enormen Fehlerrate führen. Schlimmstenfalls konvergiert das Netz bei bestimmten Daten nicht.
- Selbst wenn Konvergenz gegeben ist, so sind die notwendigen Trainingszeiten äußerst hoch. Heute können dank steigender Prozessorleistung und effizienteren Verfahren immer mehr Daten in weniger Zeit verarbeitet werden. Jedoch bleibt dieser Vorgang zeitkritisch.
- Die Regeln in einem neuronalen Netz sind meist nicht vom Menschen les- und interpretierbar. Im Falle einer Fehlfunktion macht dies das Finden der Ursache praktisch unmöglich.
- Zudem ist nicht immer vorhersehbar, wie das Netz auf veränderte Parameter reagiert. Der Prozess des Feintunings ist oft mehr Kunst als Wissenschaft. Zudem machen die langen Trainingszeiten ein häufiges Ausprobieren und Ändern des Algorithmus schwierig.
- Um die Systeme möglichst einfach zu halten, beschränken sich aktuelle Forschungsarbeiten hauptsächlich auf Raw-Vergleiche. Eine Vielzahl von Angriffen ist jedoch nur mit der „smarten“ Methode zu erkennen.

Ein Ansatz bei dem neuronale Netze Verwendung finden können, weicht stark von allen hier vorgestellten Methoden ab. Jeder Nutzer hat eine charakteristische Tippsignatur. Das bedeutet, dass sich anhand von Tastenanschlags, Schnelligkeit und Pausen während der Nutzung eines Keyboards ein Mensch eindeutig erkennen lässt. Bisher wird dies nur bei der Authentifizierung durch Passwörter als zusätzliche biometrische Komponente verwendet. Nachteil dessen ist, dass sich Angriffe von außen oder von manipulierten Netzwerkdiensten nicht erkennen lassen.

6.3 Genetische Algorithmen

Definition: Algorithmus, der Strategien aus der Evolutionstheorie verwendet, um zu einem Problem eine möglichst gute Lösung zu finden (nach [13] Seite 255).

Bei diesem Verfahren werden kurze Sequenzen aus bekannten Angriffsmustern untereinander kombiniert. Zu Beginn des Trainings ist die Trefferrate sehr gering. Erst durch die Rekombination von „erfolgreichen“ Strings wird eine Verbesserung erzielt. In diesem iterativen Verfahren müssen jene neu erzeugten Strings

Angriffe erkennen. Die mit den besten Trefferraten werden für den nächsten Rekombinationsschritt verwendet. Ist keine Verbesserung mehr erkennbar, so wird der letzte Satz an *Mutationstrings* im Produktivsystem eingesetzt. Bisher ist die Anzahl an Fehlalarmen jedoch viel zu hoch und damit ein Einsatz in realen Systemen nicht geplant. Ein kleiner Vorteil gegenüber den neuronalen Netzen ist, zumindest während der Entwicklungsarbeit, die für den Menschen verständliche Darstellung der Regeln.

6.4 Datenfusion

Aktuelle kommerzielle IDS Produkte arbeiten hauptsächlich mit Signaturvergleichen auf IP-Headern (laut [8] Seite 87). Zwar verwenden manche nicht nur Informationen aus dem Netzwerk, sondern auch von Endsystemen, jedoch werden erhaltene Informationen nicht ausreichend miteinander verglichen. Dies führt zu häufigen Fehlalarmen. Besser wäre es also, gesammelte Daten nicht einzeln auszuwerten, sondern miteinander zu verknüpfen, um aussagekräftigere Analyseergebnisse zu erhalten.

Wie die Daten aus Quellen wie Logfiles, Netzwerkdaten aus dem eigenen, oder auch aus fremden Netzen und Statistikanalysen am besten kombiniert werden sollen, ist Gegenstand der aktuellen Forschung. Selbst das amerikanische Verteidigungsministerium (Department of Defense) beteiligt sich an dieser. Wegen der großen Anstrengungen wurde in diesem Bereich bereits viel erreicht. Technologien wie statistische Schätzverfahren, digitale Signalverarbeitung, Kontrolltheorie und künstliche Intelligenz wurden erfolgreich zusammengeführt.

In Kombination mit anderen Möglichkeiten wie dem Erkennen von Portscans und Information von der Firewall, wird der Schutz durch ein IDS noch weiter erhöht. Ebenso können Informationen aus Virenscannern mit in die Bewertung einfließen.

Im einfachsten Fall werden die Daten mit einem Bayesschen⁹-Ansatz verknüpft. Spamfilter verwenden ähnliche Techniken. Dabei ist auf die unterschiedliche Vertrauenswürdigkeit und Relevanz, sowie Zuverlässigkeit der einzelnen Quellen zu achten. Wegen der hohen Komplexität dieses Prozesses kann es sinnvoll sein, eine Bewertung durch den Menschen ebenfalls mit einfließen zu lassen.

6.5 Entscheidungsbäume

Entscheidungsbäume werden für eine Vielzahl von Anwendungsgebieten verwendet. Darunter sind Stochastik, Geschäfts- und Prozessregeln, sowie Data-Mining. Da Data-Mining mit Mustererkennung arbeitet, kann man Entscheidungsbäume für IDS verwenden. Jedoch finden sie meist nicht alleine, sondern in Kombination mit neuronalen Netzen Anwendung.

Entscheidungsbäume enthalten sogenannte Klassifikationsregeln. Diese bestehen aus Vorhersageattributen, die einen booleschen Term bilden. Ist das Ergebnis für eine bestimmte Eingabe true, so wird die Eingabe mit dem abhängigen Attribut versehen. Beispiel für eine solche Regel (aus [12] Seite 5):

$$(\text{Alter} < 35) \wedge (\text{Geschlecht} = \text{m}) \wedge (\text{Autotyp} = \text{Sportwagen}) \Rightarrow (\text{Risiko} = \text{hoch})$$

⁹nach dem englischen Mathematiker Thomas Bayes

Day	Outlook	Temp.	Humidity	Wind	Play Golf
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Abbildung 2: Vom Menschen aufgestellte Tabelle (aus [10] Seite 12).

Ein Entscheidungsbaum besteht aus inneren Knoten und Blattknoten, sowie der Wurzel. Erstere definieren eine Testbedingung für jedes einzelne Attribut. Zu jedem möglichen Ergebnis des Tests verläuft eine Kante vom aktuellen Knoten zu seinem Unterbaum. Die Attribute dieser Kanten bezeichnet man als Vorhersageattribute. Zweitere markieren das Ende eines Zweiges im Baum. Sie stellen den ermittelten Wert des Zielattributs dar und sind die oben beschriebenen abhängigen Attribute. Um einen Entscheidungsbaum zu generieren, muss erst eine verlässliche Datenbasis gesammelt werden. Diese kann beispielsweise durch Experimente oder aus bereits bekannten Sachverhalten extrahiert werden. Im Folgenden soll ein Baum generiert werden, der abhängig von Wetterdaten entscheidet, ob Golf gespielt werden kann, oder nicht. Dazu wird die in Abbildung 2 gezeigte Tabelle zu Trainingszwecken genutzt.

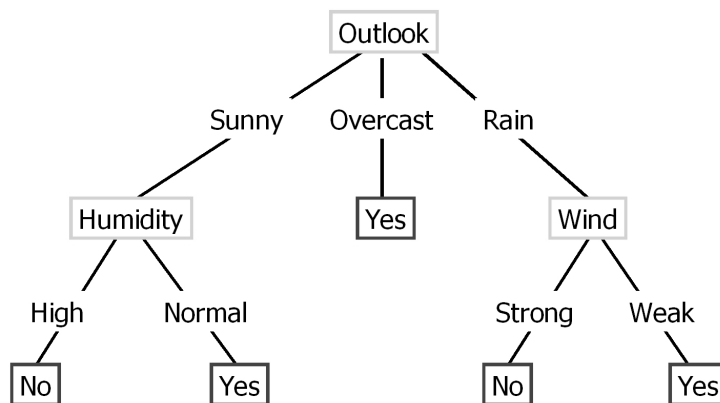


Abbildung 3: Entscheidungsbaum für Golfspielen (aus [10] Seite 13).

Diese Daten werden nun mit Hilfe des ID3-Algorithmus¹⁰ aufbereitet. Der

¹⁰ID3 steht für Interactive Dichotomizer Version 3. Entwickelt 1975 von J. Ross Quinlan an

Algorithmus soll an dieser Stelle nicht weiter erläutert werden, da er den Rahmen dieser Arbeit sprengen würde. In [12] Kapitel 2.1 wird er jedoch ausführlich dargestellt. Als Ergebnis erhält man den in Abbildung 3 gezeigten Entscheidungsbaum. Aus diesem Baum können nun Klassifikationsregeln erstellt werden. Und zwar genau so viele wie die Anzahl Blattknoten. Hier fünf Stück. Ein Beispiel für diesen Baum sieht folgendermaßen aus:

$$(\text{Outlook} = \text{Sunny}) \wedge (\text{Humidity} = \text{Normal}) \Rightarrow (\text{PlayGolf} = \text{yes})$$

Zuletzt noch ein Entscheidungsbaum für ein anomalie-basiertes IDS (Grafik 4). Dabei wird auf das Beispiel in Abschnitt 3.2 eingegangen. Wird an diesem speziellen Computer ein Vorgang als anomal erkannt, so wird das Benutzerkonto gesperrt. Eine Regel kann so aussehen:

$$(\text{Passwordeingaben} < 5) \wedge (\text{Loginzeit} \neq 7\text{-}20 \text{ Uhr}) \wedge (\text{Remotelogin} = \text{fremdes Netz}) \wedge (\text{genutzte Software} = \text{nicht erlaubt}) \Rightarrow (\text{Kontosperre} = \text{ja})$$

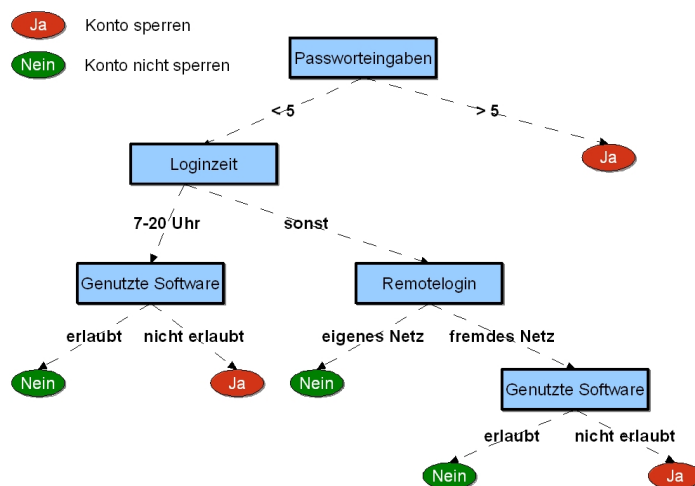


Abbildung 4: Entscheidungsbaum für ein anomalie-basiertes IDS.

6.6 Zustandsübergänge und Petrinetz-Modellierung

Zustandsübergangsdiagramme bestehen aus Pfeilen und Knoten, die durch diese verbunden sind. Dabei stellt ein Knoten einen Zustand innerhalb des Systems und der Pfeil einen Wechsel von einem Zustand in einen anderen dar. Im Kontext von IDS können die Pfeile als Sequenz von Kommandos, die ein Angreifer durchlaufen muss, um vom Ausgangszustand in einen „Angriffszustand“ zu gelangen, verstanden werden. Von diesem Ansatz verspricht man sich ein besseres Erkennen von neuartigen, oder zumindest dem System unbekanntem, Angriffsmethoden. Dabei ist nicht mehr der Zustandsübergang, der bei anderen Systemen als unerwünschtes Muster erkannt wird, sondern der neue Zustand

der Universität von Sydney.

kritisch für die Bewertung. Ungelöst ist die Problematik der unterschiedlichen Reihenfolge, in der die Übergänge stattfinden können. Der Graph für ein solches Diagramm ist zwar nicht vollständig, da manche Übergänge nicht unmittelbar aufeinander folgen können, bzw. andere Befehle als Voraussetzung haben, jedoch wird er sehr viele Kanten enthalten. Dies verlängert sowohl die Bewertung von Netzwerkverkehr, als auch den Entwurf, da möglichst alle Eventualitäten eines Angriffs vorhergesehen werden müssen.

Theoretisch bietet dieser Ansatz eine bessere Erkennungsrate, was praktisch jedoch noch nicht belegt ist. Zudem macht die Komplexität dieses Verfahrens eine Auswertung in Echtzeit wahrscheinlich nicht möglich.

6.7 Graphische Darstellung

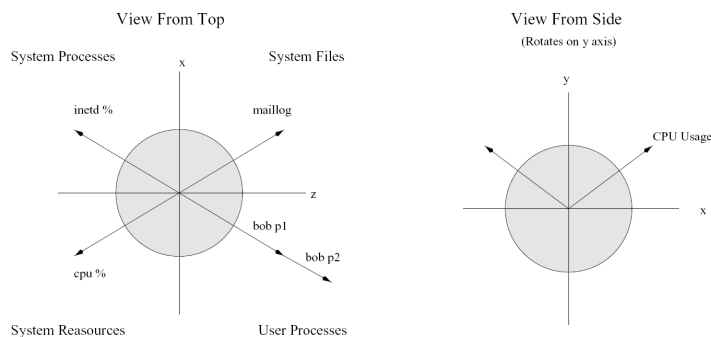


Abbildung 5: Seitenansicht und Draufsicht eines Spicule (aus [9] Seite 3).

Für den Menschen ist eine visuelle Aufbereitung von Datenbeständen schneller zu erfassen als die Darstellung in Tabellen oder das Lesen von Logfiles. Dies wird von vielen IDS völlig außer Acht gelassen. Eine Visualisierung kann von den eigentlich angefallenen Daten abstrahieren und es kann eine sehr schnelle Bewertung durch den Menschen erfolgen. Beispielsweise könnte bei einer Anomalie, welche dem System unbekannt ist, alleine auf Grund der veränderten Daten der betroffene Bereich auf einem Bildschirm farblich hervorgehoben dargestellt werden. Durch solche graphischen Änderungen kann zudem die Fehlerrate bei der Bewertung durch den Menschen massiv gesenkt werden. Beim Lesen von immer gleichen Listen werden eher Fehler übersehen oder Unbekanntes als normal eingestuft.

Ein Ansatz mit dreidimensionaler Darstellung stammt von Greg Vert et al. aus [9]. Die Grafik in Abbildung 5 stellt ein sogenanntes Spicule dar. Bei einem Spicule werden relevante Messgrößen als Vektoren aufgetragen. Deren Winkel, oder deren Breite sowie Länge ändern sich mit wechselnden Systemzuständen. Bereits durch diese einfache Grafik lässt sich eine Veränderung – unabhängig von Zahlen – sofort visuell erfassen.

Jedes Spicule repräsentiert einen einzelnen Rechner im Netzwerk. Die Größe des Kreises um den Mittelpunkt des Koordinatensystems steht für die „Security Fitness“. Diese ist die Summe von Faktoren, die die Verwundbarkeit des jeweiligen Systems erhöhen oder erniedrigen. Das Spicule eines gerade attackiertes System hat also einen größeren Durchmesser als normal. Normalisierbare Werte

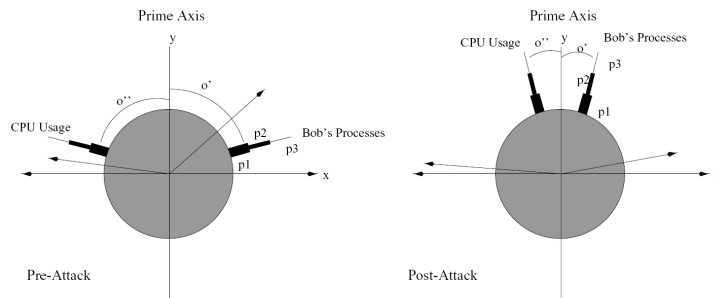


Abbildung 6: Spicule Darstellung vor und nach einer Attacke (aus [9] Seite 7).

wie die CPU-Auslastung – sie liegt immer zwischen 0% und 100% – werden als Vektoren einheitlicher Länge dargestellt. Diese wandern mit steigenden Werten von der x-Achse in Richtung der y-Achse. Unbegrenzte Werte, wie zum Beispiel die Anzahl geforkter Kindprozesse¹¹ eines Prozesses, werden als Vektoren variabler Länge und Dicke gezeichnet. Jeder Vektor liegt zudem in einem der vier Quadranten. Diese werden als Systemdateien, Benutzerprozesse, Systemressourcen und Systemprozesse bezeichnet. Somit kann eine Veränderung der Vektoren nach den zugehörigen Bereichen unterschieden werden.

Abbildung 6 zeigt, wie sich die veränderten Systemzustände vor und nach einer Attacke auf die Vektoren auswirken. Während Werte wie die CPU-Auslastung und die Anzahl an bestimmten Prozessen steigen, sinkt die Auslastung unbeteiligter Prozesse. Andere können davon völlig unberührt bleiben, wie zum Beispiel die Größe von Logdateien.

Mit dieser visuellen Technik wird, im Gegensatz zu einfachen Logdateien oder Tabellen, die Erkennungsleistung durch den Menschen drastisch erhöht. Zudem kann eine Maschine, basierend auf typischen Werten für die einzelnen Vektoren während eines Angriffs, selbst entscheiden, ob eine Attacke vorliegt.

7 Derzeit erhältliche IDS

Anbieter von kommerziellen und freien IDS gibt es viele. Im Rahmen dieser Arbeit sollen jedoch nur drei der bekanntesten kurz vorgestellt werden.

7.1 Snort

Snort [1] ist nach eigenen Angaben das am weitesten verbreitete IDS. Dies liegt mit Sicherheit an der Lizenz. Snort wurde unter der GPL erstmals 1998 veröffentlicht. Es unterstützt eine Vielzahl an Scanmethoden und erlaubt die Analyse in Echtzeit. Jedoch basiert die Erkennung auf Signaturen bekannter Angriffe. Diese müssen öffentlich verfügbar und zur GPL kompatibel sein. Auf der Webseite von Snort stehen laufend aktualisierte Regeln zum Download bereit. Die wichtigsten Funktionalitäten umfassen:

- Echtzeitanalyse des Netzwerkverkehrs

¹¹Mit der C-Funktion `fork()` kann sich ein Unix-Prozess selbst kopieren. Diese Kopie wird Kindprozess genannt.

- Protokollieren von Paketen in Netzwerken, welche auf IP basieren
- Protokollanalyse
- Untersuchen von Datenpaketen nach schädlichen Mustern
- Erkennen von versteckten Portscans und Bufferoverflows

Aktive Komponenten enthält Snort nicht. Jedoch kann es erkannte unerwünschte Pakete verwerfen und somit den Angriff verhindern. Da Snort selbst lediglich kommandozeilen-orientiert arbeitet, existieren zahlreiche graphische Frontends wie `sguil`¹² oder `Snort Center`¹³ (wird nicht mehr weiterentwickelt).

Kommerziell wird Snort von der Firma Sourcefire vertrieben. Das Unternehmen integriert Snort in Hardware und verkauft diese mit Supportverträgen.

7.2 Dragon

Dragon [2] ein Produkt der Firma Enterasys, liegt derzeit in Version 7 vor. Es kombiniert Ereignisse im Netzwerk, auf PCs, Firewalls und Routern, sogar in Gigabit Ethernet Geschwindigkeit. Dragon analysiert TCP/IP-Traffic per Mustererkennung, Protokollüberwachung und Erkennung von anormalen Ereignissen. Darüberhinaus sind aktive Komponenten zum Eingreifen in den Netzwerkverkehr vorhanden.

7.3 Cisco Intrusion Prevention System

Der größte Netzwerkausrüster, Cisco [3] bietet eine Komplettlösung für große Firmennetzwerke, verbunden mit eigener Hardware und proprietären Standards. Die Sensoren basieren auf Regeln und überwachen IP-Traffic sowie Syslog-Dateien von Cisco Routern. Sie enthalten ebenso wie Dragon aktive Komponenten. Die Einheit zum Auswerten wird Director genannt. Cisco ist bei der graphischen Aufbereitung und dem Management der Komponenten durch GUIs mit führend in der graphischen Darstellung in IDS. Zudem nennt es sich nicht nur Detection, sondern Prevention System. Dies ist jedoch eher ein Marketingbegriff. Ein Prevention System ist in der Lage erkannte Angriffe zu neutralisieren, indem Traffic umgeleitet wird, oder Einstellungen in Firewalls dynamisch verändert werden. Dies leisten jedoch genau die in Abschnitt 2.3 beschriebenen aktiven Komponenten, über welche Dragon ebenfalls verfügt.

8 Ausblick

Was in Zukunft in IDS integriert, oder wo die Integration verbessert werden muss wird in [4] ausführlich dargestellt. Hier sollen nur kurz die wichtigsten Punkte erläutert werden:

Interoperativität: Die bereits bestehende Verknüpfung von Netzwerkkomponenten wie Firewalls und Routern mit dem IDS muss verbessert werden. Zudem wäre ein offener Standard wünschenswert, um nicht vollständig auf die Hardware mit proprietären Protokollen eines einzigen Herstellers angewiesen zu sein.

¹²<http://sguil.sourceforge.net/>

¹³<http://users.pandora.be/larc/>

Reaktionsgeschwindigkeit: Die per Datenfusion kooperierenden Systeme müssen schneller untereinander kommunizieren, um Entscheidungen zuverlässig und autonom in Echtzeit zu treffen.

Passives OS-Fingerprinting: Dabei handelt es sich um einen Mechanismus, der anhand betriebssystemspezifischer Merkmale, die sich im Netzwerkverkehr feststellen lassen, den Hersteller und oft auch die Version des Systems erkennen kann. Kennt man also das Ziel einer Attacke und weiß durch das Fingerprinting, dass es sich bei dem System um ein für diesen Angriff unanfälliges handelt, so kann die Reaktion ausbleiben, oder mit einer niedrigeren Priorität versehen werden. Mit diesem Verfahren ließe sich das Auftreten von Fehlalarmen effektiv verringern.

9 Fazit

Es wurde der generelle Aufbau eines Intrusion Detection Systems vorgestellt, sowie die unterschiedlichen Methoden zur Analyse von anfallenden Daten. Desweiteren wurden die Unterschiede zwischen theoretischen Ansätzen und tatsächlichen Implementierungen in im echten Einsatz befindlichen Systemen erläutert. Insbesondere wurde gezeigt, wo Lücken und Probleme im Performance-, Sicherheits- und Rechtsbereich liegen. Ebenso ungelöst ist die Überwachung von verschlüsselter Kommunikation. All diese Punkte werden derzeit intensiv erforscht und es wird versucht, Ergebnisse davon in den Livebetrieb zu übertragen. Wie dargestellt ist dies jedoch aus Gründen wie zu geringer Rechenleistung oder zu hohen Kosten nicht immer praktikabel.

Literatur

- [1] <http://www.snort.org> [15.12.2006]
- [2] <http://www.enterasys.com/products/ids/> [15.12.2006]
- [3] <http://www.cisco.com/> [15.12.2006]
- [4] <http://www.infosecwriters.com/texts.php?op=display&id=115>
[14.01.2007]
- [5] <http://www.bsi.bund.de/literat/studien/ids02/gr1.htm> [10.01.2007]
- [6] <http://ieee802.org/3/hssg/> [10.01.2007]
- [7] Coolen, R.; Luijuf, H.A.M. (TNO Physics and Electronics Laboratory, The Netherlands). "Intrusion detection: Generics and State-of-the-Art", 2002
- [8] Allen, J.; Christie, A.; Fithen, W.; McHugh, J; Pickel, J.; Stoner, E.; "State of the practice of intrusion detection technologies"; CarnegieMellon Software Engineering Institute, Pittsburgh, 2000
- [9] Vert, G; Frinke, D. A.; & McConnell, J. C. (University of Idaho). A Visual Mathematical Model for Intrusion Detection
- [10] Schafer, B. (University of Northern Iowa). Machine Learning Methods. Cedar Falls, IA, 2006
- [11] Rieck, K. (Freie Universität Berlin, Diplomarbeit). Maschinelles Lernen in hostbasierten Intrusion-Detection-Systemen. Berlin, 2004
- [12] Mörwald, T. (Universität Passau, Bachelorarbeit). Intrusion Detection mittels Entscheidungsbäumen. Passau, 2004
- [13] Duden, Informatik: ein Fachlexikon für Studium und Praxis / hrsg. von der Red. Studium und Beruf. Bearb. von Volker C. und Andreas S. - [Taschenbuchausg.]. - Mannheim; Leipzig; Wien; Zürich: Dudenverl., 2003.