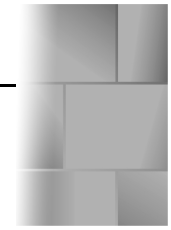
Kapitel 3



Was kann/soll ein IDS leisten?

Dieses Kapitel versucht die verschiedenen Wünsche und Erwartungen an ein IDS zu benennen. Nicht alle IDS sind in der Lage, diese Funktionen zu erfüllen. Einige Funktionen mögen nur von kommerziellen IDS zu Verfügung gestellt werden. Dennoch ist es für die Auswahl und die Implementierung eines Intrusion Detection-Systems wichtig, sämtliche Anforderungen im Vorfeld zu kennen und abwägen zu können.

3.1 Anforderungen an ein Intrusion Detection-System

Die wichtigste und am einfachsten zu formulierende Anforderung an ein Intrusion Detection-System ist:

Das Intrusion Detection-System soll Angriffe und Einbrüche auf zu schützenden Systemen erkennen und bei der Verteidigung dieser Systeme eine Unterstützung bieten.

Die Ziele und Anforderungen können jedoch weiter aufgeschlüsselt werden. Bei Beurteilung der eingesetzten oder projektierten Systeme ist eine detaillierte Liste der Anforderungen hilfreich. Die folgende Liste ist eine ausführliche Zusammenstellung der möglichen Ansprüche an ein IDS. Sie mag in dem einen oder anderen Umfeld eine Modifikation verlangen.

Allgemeine Ziele des IDS-Systems sind:

- Verfügbarkeit der bereitgestellten Dienste
- Integrität und Verfügbarkeit der bereitgestellten Daten
- Unterstützung in der Reaktion auf einen Angriff
- Fehlertoleranz
- Kontinuierlicher Lauf ohne wesentliche Administration
- Erzeugung minimalen Overheads
- Leichte Integrierung in die vorhandene Struktur
- Hohe Intelligenz, die ein Umgehen möglichst unmöglich macht

Um diese Ziele zu erreichen, sollte ein IDS folgende Funktionalitäten bieten:

- Die Integrität der Daten, die das IDS nutzt, muss gewährleistet sein. Das IDS muss die Herkunft und Integrität der Daten überprüfen. Des Weiteren sollte das IDS diese Daten im Rohformat für eine spätere Überprüfung abspeichern können und hierbei eine digitale Signatur der Daten ermöglichen. Damit sollte einer späteren Verfälschung der Daten vorgebeugt werden.
- Die Analyse der Daten muss nachvollziehbar erfolgen. Eine spätere erneute Analyse der identischen Rohdaten muss zu einem identischen Ergebnis führen. Nur so können die Ergebnisse später einer Überprüfung standhalten.

- Die Analyse der Daten muss selbstverständlich sämtliche möglichen Angriffe und Einbrüche, die im Vorfeld beschrieben wurden und die noch in Zukunft entwickelt werden, erkennen können.
- Die Erzeugung der Ergebnisse durch das IDS muss einen derartigen Rückschluss auf die Rohdaten erlauben, dass die Analyse nachvollzogen werden kann. Am einfachsten bietet das IDS die direkte Darstellung der Rohdaten in Zusammenhang mit der Alarmmeldung. Hierbei muss das IDS unterschiedliche Darstellungsformate für den Bericht erlauben und eine automatische Weiterverarbeitung unterstützen.
- Das IDS muss eine Konfiguration der Alarmierungsschwelle erlauben. Es muss die Möglichkeit bestehen, den verschiedenen Ereignissen Prioritäten zuzuweisen und nur bei Erreichen einer bestimmten Priorität den Alarm auszulösen.
- Das IDS sollte für die Speicherung der Daten und des Berichtsformates und der Alarmmeldungen ein allgemein anerkanntes Format verwenden. Dies erlaubt den einfachen Austausch der Informationen und die standardisierte Weiterverarbeitung der Daten mit weiteren Werkzeugen. Das IDS sollte auch in der Lage sein, Rohdaten in derartigen Formaten einlesen zu können.
- Das IDS muss Funktionen bieten, die die Vertraulichkeit der Daten garantieren können. In vielen Umgebungen ist die Speicherung von personenbezogenen Informationen nicht erlaubt. Hier ist die automatische Anonymisierung wünschenswert. Wünschenswert ist auch eine automatische Löschung der personenbezogenen Daten nach der Nutzung.
- Ein IDS sollte eine automatische Reaktion auf einen Angriff unterstützen. Diese Unterstützung sollte so implementiert worden sein, dass ein Angreifer diese Funktionalität nicht für einen Denial of Service ausnutzen kann.

Dies sind eine ganze Reihe von Anforderungen, die an ein Intrusion Detection-System gestellt werden müssen. Diese Anforderungen sind gültig unabhängig davon, ob es sich um ein Host-IDS oder ein Network-IDS handelt. Im nächsten Abschnitt sollen diese Anforderungen mit der Wirklichkeit verglichen und Beispiele für ihre Implementierung aufgezeigt werden.

3.2 Tatsächlich gebotene Leistung

Die Eier legende Wollmilchsau des Intrusion Detection-Systems muss erst noch entwickelt werden. Sämtliche freie wie kommerzielle IDS-Systeme haben den einen oder anderen Nachteil. Hier sollen kurz die Anforderungen des letzten Abschnittes wieder aufgegriffen und ihre Implementierung erläutert werden. Dies soll insbesondere beim Einsatz freier Open Source-Software Anregungen dazu geben, wie die Anforderungen umgesetzt werden können.

- **Integrität der Eingangsdaten**. Ein IDS kann die Integrität und die Herkunft der Daten am besten überprüfen, wenn es diese Daten selbst erzeugt oder ermittelt. Im Falle eines NIDS sollte das IDS selbst einen eigenen Sniffer zur Verfügung stellen. Ein HIDS, welches einen Integritätstest der Systemdateien durchführt, sollte selbst auf diese Dateien zugreifen und die Vergleichsdatenbank selbst in der Vergangenheit erzeugt haben. Hierbei ist es erforderlich, dass das HIDS eine Unterwanderung der Dateizugriffe erkennen kann. Am sinnvollsten wird dieses HIDS direkt ins Betriebssystem eingebettet (siehe LIDS in Kapitel 8.3, »Linux Intrusion Detection System – LIDS« ab Seite 161 und die Kernel-basierten Rootkits in Anhang C, »Rootkits« ab Seite 573). Bei einer Protokoll-Analyse muss das HIDS die Quellen der Protokolle überprüfen können. Ist dies bei einer zentralen Protokollierung über das Netzwerk nicht möglich, so muss zumindest bei der Protokollierung über das Netzwerk eine Authentifizierung der Clients erfolgen, die diese Protokollierung nutzen. Der einfache BSD Syslogd bietet keinerlei Authentifizierung (siehe Kapitel 12, »Zentrale Protokollserver« ab Seite 369). Zusätzlich sollten alle Daten und Dateien, die als Rohdaten vom IDS gelesen werden, weiterhin abgespeichert werden und ihre Integrität mit krytographischen Prüfsummen (z. B. MD5, SHA, etc) oder Signaturen (GnuPG) garantiert werden.
- **Reproduzierbare Analyse.** Um eine reproduzierbare Analyse der Daten zu ermöglichen, ist es erforderlich, alle relevanten Rohdaten zu speichern. Sämtliche Daten, die einen Alarm ausgelöst haben, sollten so abgespeichert werden, dass dieser Alarm später nachvollziehbar ist. Dies beinhaltet auch das Vorhalten älterer Versionen der Konfigurationsdateien und der zur Analyse verwendeten Werkzeuge. Meist stellt sich nicht das Problem der Werkzeuge, da Alarmmeldungen innerhalb weniger Stunden oder Tage bearbeitet werden. Werden dennoch die Daten für eine Verfolgung durch amtliche Behörden gespeichert und verwendet, so kann es durchaus einige Monate und Jahre dauern, bis diese Daten nicht mehr benötigt werden. Die Konfigurationsdateien werden häufig angepasst und modifiziert. Hier sollten sämtliche Variationen vorgehalten werden, da leichte Konfigurationsänderungen wesentliche Änderungen der Ergebnisse erzeugen können. Unter Linux können hier am sinnvollsten entweder das Revision Control System (RCS) oder das Concurrent Version System (CVS) zur Versionsverwaltung eingesetzt werden. Eine zusätzliche Signatur der Daten ist sinnvoll.
- **Vollständige Angriffserkennung.** Dies ist sicherlich die problematischste Anforderung. Insbesondere die Erkennung neuartiger Angriffe stellt für NIDS eine große Herausforderung dar. Die statistische Anomalie-Analyse steckt immer noch in ihren Kinderschuhen und kann es noch lange nicht mit der Signatur-basierten Erkennung aufnehmen. Die beste Garantie für die Erkennung neuer Angriffe bieten also eine dauernde Pflege und Administration des Systems, ein wachsamer Administrator und geschultes Personal. Kommerzielle Systeme versuchen dies zu umgehen, indem sie automatische Updates der Signatur-Datenbanken in Form von Wartungsverträgen erlauben. Die Aktualisierungen erfolgen jedoch nur wöchentlich oder gar nur einmal

im Monat oder Quartal. Daher können diese Aktualisierungsabonnements nicht das geschulte Personal ersetzen. Ein weiteres Problem stellen die falsch-positiven Meldungen durch das IDS dar. Diese sind meist von der jeweiligen Umgebung abhängig und erfordern häufig langwierige Modifikationen und Tunings des IDS-Systems. Dies trifft sowohl auf HIDS als auch auf NIDS zu.

- **Berichtsformate.** Die Berichte des IDS müssen in unterschiedlichen Formaten mit unterschiedlichem Detailgrad erzeugt werden können. Dies ist häufig möglich. Dabei unterstützen viele Systeme auch einen direkten Export der Daten in eine Datenbank.
- **Prioritäten** Sämtliche verfügbaren IDS bieten die Funktion, unterschiedlich auf die verschiedenen Ereignisse zu reagieren. Durch eine Anbindung externer Dienste ist auch ein Versand von SNMP-Traps, E-Mails, Pager-Meldungen oder SMS möglich. Speziell der Versand von SMS ist unter Linux sehr einfach. Hierzu existieren Lösungen wie gsmlib (*http://www.pxh.de/fs/gsmlib/index.html*).
- **Standardformate.** Open Source IDS-Systeme sind häufig wesentlich progressiver in der Implementierung neuer Standardformate. Das Open Source-Netzwerk-IDS Snort bietet zum Beispiel die Möglichkeit, die Ergebnisse in XML abzuspeichern. Hierbei können zwei verschiedene Standard Document Type Definitions (DTD) zu Grunde gelegt werden, die öffentlich verfügbar sind. Hierbei handelt es sich um die Simple Network Markup Language (SNML, *http://www.cert.org/DTD/snml-0.2.dtd*) und das Intrusion Detection Message Exchange Format (IDMEF, *http://www.silicondefense.com/idwg/draft-ietf-idwg-idmef-xml-03.txt*). Die Rohdaten können in dem *libpcap*-Format abgespeichert werden. Die *libpcap*-Bibliothek stellt die Grundlage der meisten Programme unter Linux und inzwischen immer mehr auch unter Windows dar, die Pakete sniffen können. Die meisten Programme, die auf dieser Bibliothek aufbauen, können Dateien in diesem Format lesen und schreiben. Beispiele dieser Programme sind *tcpdump*, *snort*, *ethereal* etc. Häufig werden die Open Source-Produkte auch als Referenzimplementierung des Formates erzeugt. Kommerzielle Produkte tun sich hier meist schwerer. Häufig ist damit die Angst verbunden, den Kunden nicht an die eigene Produktreihe binden zu können.
- **Anonymisierung.** Beim Einsatz von IDS-Systemen darf das Thema Datenschutz nicht vernachlässigt werden. Das IDS sollte den Administrator unterstützen, indem es eine automatische Anonymisierung ermöglicht. Des Weiteren soll es möglich sein, den Zugriff auf die Daten unterschiedlich zu gestalten. Ein Zugriff auf personenbezogene Daten darf nur bestimmten Personen erlaubt werden. Die verfügbaren Systeme unterstützen dies teilweise. Unter Linux kann jedoch mit einfachen Bordmitteln (*sed*) eine Anonymisierung erreicht werden.

- **Intrusion Response.** Wenige Intrusion Detection-Systeme sind in der Lage, eine wirksame und schnelle automatische Reaktion zu ermöglichen. Dies hängt unter anderem damit zusammen, dass eine derartige Funktion schnell in ihr Gegenteil verkehrt werden kann. In Abhängigkeit von der Reaktion besteht die Möglichkeit, dass der Angreifer diese Funktion verwendet, um das System oder sämtliche Netzwerkfunktionalität außer Betrieb zu setzen. Diese Systeme müssen eine sehr intelligente und gezielte Antwort erlauben. Im Open Source-Bereich erlauben sowohl LIDS (Abschnitt 8.3, »Linux Intrusion Detection System – LIDS« ab Seite 161) als auch Snort (Abschnitt 9.2, »Snort« ab Seite 204) derartige Möglichkeiten.

Diese Betrachtung der Anforderungen im Lichte ihre Realisierung durch die verfügbaren Systeme soll zeigen, dass das ideale Intrusion Detection System noch nicht geschaffen wurde. Auf die Eier legende Wollmilchsau werden die Anwender noch einige Zeit warten müssen.

3.3 Aufwand und Kosten

Der Aufwand für die Installation und Pflege eines IDS-Systems und die damit verbundenen Kosten sind nur schwer abzuschätzen. Bei der Implementierung einer unternehmensweiten Sicherheitsstrategie sind aber Kosten und Nutzen abzuwägen. Hier werden häufig Begriffe wie *Total Cost of Ownership* (TCO) und *Return On Investment* (ROI) verwendet, um die Kosten und den Nutzen einer Strategie und ihrer Umsetzung abzuschätzen. Im Folgenden soll versucht werden, diese Begriffe auf die Umsetzung einer Intrusion Detection-Lösung anzuwenden.

Die folgenden Ausführungen können jedoch nur als eine grobe Richtung angesehen werden und sind sehr stark abhängig von der lokalen Umgebung im Unternehmen, dem vorhandenen Know-how und den verfolgten Zielen bei der Implementierung einer Intrusion Detection-Lösung. Die angegebenen Zahlen entsprechen den Erfahrungen der letzten Monate und Jahre bei der Implementierung von IDS Systemen in kleinen Umgebungen und werden sich sicherlich in nächster Zeit ändern. Auf der Kostenseite ist zum einen mit einer möglichen Reduktion zu rechnen, da die Konkurrenz unter den Firmen, die derartige Leistungen als externe Dienstleister anbieten, größer wird. Genauso können die Kosten in diesem Bereich aber auch steigen, da der Bedarf stark steigen wird. Eine IDS-Lösung von der Stange passt nur in den seltensten Fällen, sodass meist Beratungsleistungen erforderlich sind. Diese können intern zur Verfügung gestellt wie auch extern eingekauft werden. Dies wird entsprechend dem Gesetz von Angebot und Nachfrage möglicherweise auch die Kosten steigen lassen.

Eine Begründung für die Einführung eines IDS und eine Berechtigung für die damit verbundenen Kosten sind häufig nur sehr schwer zu finden. Meist ist die Begründung »FUD«. FUD steht für Fear (Angst), Uncertainty (Unsicherheit) und Doubt (Zweifel).

Dies überzeugt jedoch kaum das Management einer Firma geschweige denn die Aktionäre in einer Aktiengesellschaft, für die jede Ausgabe einen Verlust darstellt. Hier sind harte Zahlen gefragt. Bis zum letzten Jahr (2001) waren keine Zahlen verfügbar, die den Wert eines IDS messen konnten. Erforderlich sind Gutachten, die es ermöglichen, den ROI (Return on Investment) oder besser den Return On Security Investment (ROSI) zu berechnen. Eine Messung des Sicherheitsstandards ist jedoch sehr schwer oder unmöglich. Eine Quantifizierung der Sicherheit erfolgt immer in Anteilen subjektiv. Ich sehe diese Sichtweise sehr kritisch, aber sie wird sich auf Dauer durchsetzen.

Die Ausgabe 02/2002 des CIO-Magazins (*http://www.cio.com*) fasst die Probleme recht gut zusammen und stellt erste Forschungsergebnisse in diesem Bereich vor. Hier wird die Anschaffung eines Intrusion Detection-Systems mit der Anschaffung von Sprinkleranlagen in Baumwollspinnereien im 19. Jahrhundert verglichen. Im Weiteren wird in diesem Artikel die Forschungsarbeit von HuaQuiang Wei beschrieben, der, bei der Entwicklung des IDS Hummer (*http://www.csds.uidaho.edu/~hummer/*), erstmalig versuchte, die Kosten eines Angriffes zu quantifizieren und mit den Kosten zu vergleichen, die für den Aufbau des IDS erforderlich sind. Diese Daten wurden leider nicht veröffentlicht. Jedoch behauptet der zitierte Artikel, dass ein IDS-System, welches Kosten von $ 40.000 verursacht, in der Lage war, zu 85% Angriffe zu erkennen, die ansonsten Kosten von $ 100.000 erzeugt hätten. Damit kann das IDS $ 85.000 einsparen. Abzüglich den $ 40.000 Investition bleibt ein Nettogewinn von $ 45.000.

Hierbei handelt es sich jedoch weiterhin um Zahlen, die scheinbar aus der Luft gegriffen sind. Im August 2002 wurden auf der Webseite *http://www.security focus.com* zwei Artikel veröffentlicht, in denen David Kinn und Kevin Timm neue Rechenverfahren zur Diskussion stellten. Diese Berechnungsmethoden stellten zum Zeitpunkt der Verfassung dieses Buches die letzten und fortgeschrittensten Methoden zur Berechnung des ROI eines IDS dar.

In diesen beiden Artikeln wird der Return on Investment als Formel entwickelt:

```
ROI = R - ALE
Return on Investment = Recovery Cost - Annual Loss Expectancy
```

Der Return on Investment ist der zu erwartende Nettogewinn durch die Investition. Die Recovery Cost sind die aufzuwendenden Kosten bei den Einbrüchen, wenn kein IDS installiert ist. Die Annual Loss Expectancy bezeichnet die zu erwartenden Verluste, wenn ein IDS installiert ist. Die ALE kann weiter aufgeschlüsselt werden als:

```
ALE = (R - E) + T
Annual Loss Expectancy = (Recovery Cost - Annual Savings) + Annual Cost of IDS
```

Diese Artikel entwickeln die Formel weiter und versuchen die Einsparungen zu quantifizieren. Hierzu werden zunächst die zu schützenden Werte als Asset Value (AV) quantifiziert. Diese enthalten sämtliche Werte des möglicherweise kompro-

mittierten Systems: Hardware, Software und Daten. Anschließend wird zusätzlich ein *Exposure Factor* (EF) eingeführt. Hiermit soll die Verwundbarkeit und Erreichbarkeit des initial kompromittierten Systems in Prozent gemessen werden. Zusätzlich werden die *Underlying Exposed Assets* (UEA) definiert. Dies sind die Systeme, die in einem zweiten Schritt bei einem Angriff vom Angreifer erreicht werden können. Hierfür wird auch ein Secondary Exposure Factor (EFs) definiert, der die prozentuale Verwundbarkeit dieser Systeme beschreibt. Anschließend wird aus diesen Daten ein *Cascading Threat Multiplier* (CTM) berechnet. Dieser stellt die eigentliche Neuerung bei dieser Methode dar. Der Angriff wird nicht singulär betrachtet, sondern die Berechnung bezieht weitere Systeme, die anschließend von dem Angreifer leichter attackiert werden können, mit ein. Aus den Werten der betroffenen Systeme lassen sich dann die CTM und die *Single Loss Expactancy* (SLE) berechnen. Die SLE bezeichnet die voraussichtlichen Kosten bei einem Einbruch auf einem System mit den erwarteten zusätzlichen Angriffen auf sekundären Systemen. Die CTM und die SLE werden berechnet als:

```
CTM = 1 + (( UEA x EFs ) / AV )
SLE = EF x AV x CTM
```

Um nun die jährlichen Verluste durch Angriffe zu berechnen, fehlt eine Angabe der Häufigkeit der Angriffe. Dies wird in der Variable *Annual Rate of Occurance* (ARO) ausgedrückt. Der jährliche Verlust durch einen Angriff kann dann berechnet werden als:

```
ALE = SLE x ARO
```

Nun sind weiterhin Annahmen erforderlich, die sich teilweise auf die Studien von HuaQuiang Wei stützen lassen und die Wirksamkeit eines IDS beschreiben. Hier wird davon ausgegangen, dass ein IDS im Schnitt in der Lage ist, Angriffe zu 80% zu erkennen. Es sind jedoch sicherlich nicht alle Angriffe gefährlich und erfolgreich. Daher soll hier von etwa 50% erfolgreichen Angriffen ausgegangen werden.

Um nun auf die Formel vom Beginn zurückzukommen, ist der ROI dann:

```
ROI = ALE - (ALE - 50% x ALE + T)
ROI = 50% ALE - T
```

Nun bleibten noch die Werte ALE und T zu bestimmen. Im Folgenden soll von der Firma *nohup.info* ausgegangen werden, die über eine Zentrale mit drei Filialen verfügt. Diese Firma verdient ihr Geld mit der Bereitstellung von Webdiensten. Da der Name Programm ist, garantiert diese Firma die ununterbrochene Bereitstellung dieser Dienste. Sie besitzt momentan 1500 Kunden und die Tendenz ist stark steigend. Sie garantiert jedem Kunden die ununterbrochene Verfügbarkeit der Dienste. Hierfür zahlt jeder Kunde mindestens EUR 250 pro Monat. Insgesamt erhält die Firma nohup.info pro Monat Einnahmen in der Höhe von EUR 450.000 für ihre Internetdienste. Ein Ausfall ihrer Internetverbindung würde diese Firma zahlungsunfähig machen. Ein Einbruch und die Offenlegung von internen Daten würde sie in den Ruin treiben.

Die Firma nohup.info wird über einen Router mit dem Internet verbunden. Intern verwendet sie momentan einen Apache-Webserver. Auf dringenden Kundenwunsch wird aber demnächst auch ein Windows 2000-Rechner mit IIS-Webserver eingerichtet werden. Auf beiden Plattformen werden Java-Applikationsserver eingesetzt. Der Wert jedes einzelnen Systems (Asset Value, AV) beträgt etwa EUR 3.000). Da die Systeme leidlich gut gewartet werden, liegt der Exposure Factor bei 50%. Die Administratoren dieser Firma sind wie bei allen Start-Ups hoffnungslos überlastet. Hinter diesen Systemen befindet sich das gesamte Know-how und die einzige Einnahmequelle der Firma. Ein Einbruch hätte katastrophale Folgen. Daher kann der monatliche Umsatz der Firma als Grundlage für die Underlying Exposure Assets herangezogen werden: EUR 450.000. Die entsprechende Verwundbarkeit dieser sekundären Rechner ist recht hoch, wenn der Angreifer sich bereits derartig nah an das Netzwerk heranarbeiten und einen primären Einbruch erfolgreich durchführen konnte. Daher ist der EFs (Secondary Exposure Factor) mit 50 % recht großzügig bemessen. Dieser Faktor muss nach oben angepasst werden, wenn zwischen den primären und den sekundären Systemen Vertrauensstellungen existieren. Hierbei kann es sich um Microsoft Windows NT- oder 2000-Domänen handeln. Jedoch genügen bereits gemeinsame Kennworte der Administratorkonten auf den Systemen. Dann ist dieser Faktor auf etwa 70-80 % anzupassen. Schließlich ist die Wahrscheinlichkeit eines Angriffes zu bewerten. Die Erfahrung zeigt, dass etwa vier bis fünf neue erfolgreiche Angriffe gegen gut gewartete Microsoft-Betriebssysteme und ein bis zwei erfolgreiche Angriffe gegen gut gewartete Linux-Distributionen pro Jahr veröffentlicht werden. Dies setzt bereits eine sehr stringente Administration der Systeme voraus. Diese Systeme bieten keine unnötigen Dienste an und die vorhandenen Dienste sind optimal konfiguriert. Ist dies nicht der Fall, so ist bei beiden Betriebssystemen die Zahl mit mindestens 2 zu multiplizieren. In der Vergangenheit durfte bei Linux- oder UNIX-Betriebssystemen in diesem Fall eine Multiplikation mit 4 oder 5 durchgeführt werden. UNIX- wie Linux-Betriebssysteme aktivierten in der Vergangenheit alle installierten Dienste. Dies ist heute (beginnend mit den Linux-Distributionen SuSE 8.0 und Red Hat 7) nicht mehr der Fall.

Nun kann die SLE berechnet werden:

```
CTM = 1 + (( UEA x EFs ) / AV ) = 1 + (( 450.000 x 50 % ) / 3.000 = 76
SLE = EF x AV x CTM = 50 % * 3.000 * 76 = 114000
```

Diese Rechnung zeigt bereits die Besonderheit bei der Berechnung nach Kinn und Timm. Die sekundär verwundbaren Systemen gehen stark in die Bewertung mit ein. Dies entspricht auch der Erfahrung in einem echten Angriff.

Um nun die Annual Loss Expectancy (ALE) zu berechnen, ist es erforderlich, diese Zahl mit der erwarteten Häufigkeit eines derartigen Angriffes zu multiplizieren. Wird hier von 1,5 statistisch erfolgreichen Angriffen pro Jahr ausgegangen, so ergibt sich eine ALE von:

```
ALE = ARO * SLE = 1,5 x 114.000 = 171.000
```

Wenn nun angenommen wird, dass ein IDS in der Lage ist, diese Angriffe in 50% der Fälle so zu entdecken und zu reagieren, dass diese Angriffe leicht behoben werden können, ohne dass der Angreifer auf weitere Systeme übergreifen kann, so berechnet sich die ROI als:

```
ROI = 50 % ALE - T
```

Nun fehlen noch die Kosten des Intrusion Detection-Systems (T). Handelt es sich hierbei um Open Source, so sind keine Lizenzgebühren fällig. Es bleiben die Administration und die Wartungskosten, Kosten für die Hardware usw.

Ein Administrator, der 50% seiner Arbeitszeit dem IDS widmet, kostet etwa EUR 40.000 – 50.000 pro Jahr. Es ergibt sich eine ROI von:

```
ROI = 50 % x 171.000 - 50.000 = 35.500
```

Dieses Beispiel sollte im Kleinen die Berechnung des ROI eines Intrusion Detection-Systems mit Zahlen nach der Methode von Kinn und Timm (*http://online.securityfocus.com/infocus/1621*) zeigen. Für größere Firmen stellen sich selbstverständlich andere Zahlen dar, die den ROI wesentlich deutlicher zeigen. Dennoch, das Beispiel der Firma nohup.info zeigt, wie das IDS einen ROI von etwa 35.000 jährlich erwirtschaften kann. Wenn befürchtet werden muss, dass die Firma nohup.info bei einem erfolgreichen Angriff alle Kunden verliert, so kann als UEA der Jahresumsatz von ca. EUR 5.000.000 eingesetzt werden. Dies wird einen wesentlich deutlicheren ROI aufzeigen.

Die Anwendung dieser Methode und die wirtschaftliche Bewertung von Intrusion Detection-Systemen wird in der Zukunft weiter zunehmen. Hierzu werden weitere Methoden entwickelt werden. Die treibende Kraft werden die Versicherungsunternehmen sein. In naher Zukunft werden diese den IT-Markt wohl in großem Stil erschließen. Hierzu werden vergleichbare Berechnungsmethoden benötigt, die es erlauben, den jährlichen Verlust, die Exposition und die Sicherheitsstandards berechenbar zu machen. Das ist sicherlich nicht so einfach, wie bei Hausrat- oder KFZ-Versicherungen und wird noch einige Forschung erfordern. Dieses Kapitel soll eine momentan verfügbare Berechnungsmethode vorstellen, um die Kosten möglichst objektiv berechnen zu können.

3.4 Kommerzielle und freie IDS

Dieses Kapitel soll verschiedene kommerzielle und frei verfügbare Intrusion Detection-Systeme vorstellen. Die Funktionalitäten der unterschiedlichen Lösungen sollen kurz stichpunktartig aufgeführt werden. Leider bestand nicht die Möglichkeit, sämtliche Funktionalitäten der kommerziellen Systeme zu testen, da der Autor nicht in allen Fällen die entsprechende Testumgebung besaß.

Es handelt sich, wie bereits erwähnt, nicht um einen kompletten Vergleich aller verfügbaren Systeme. Sollte ein Leser eine Ungenauigkeit in diesem Vergleich

entdecken oder ein IDS-System fehlen, welches unbedingt aufgenommen werden muss, so würde ich mich über eine E-Mail freuen.

Ein im Internet verfügbarer Vergleich, der jährlich durchgeführt wird, wird von der unabhängigen *The NSS Group* veröffentlicht. Die Dokumente sind nach einer Registrierung online verfügbar oder können als Acrobat-PDF erstanden werden (*http://www.nss.co.uk/*).

Die folgenden IDS-Systeme sind teilweise Stand-Alone-Lösungen und werden in Form einer Appliance verkauft. Eine Appliance stellt eine Kombination aus Hardware und Software dar, die komplett verkauft wird. Häufig kommt hierbei speziell angepasste Hard- oder Software zum Einsatz. Meistens sind diese Appliances in der Lage, eine bessere Leistung zu bieten, als wenn die Software auf handelsüblicher Hardware eingesetzt wird. Jedoch besteht in vielen Fällen die Möglichkeit, die Software einzeln zu erwerben und selbst die Hardware (und möglicherweise das Betriebssystem zur Verfügung zu stellen).

3.4.1 NetRanger/Cisco Secure IDS

Dieses NIDS (*http://www.cisco.com*) wird als Stand-Alone Appliance-Lösung von Cisco vertrieben. Es setzt auf einer stark modifizierten Variante des Sun Solaris UNIX-Betriebssystems auf. Dieses IDS wird in mehreren verschiedenen Ausbaustufen angeboten, die sich in erster Linie in ihrer Leistungsfähigkeit unterscheiden. Als besonderes Feature existiert das Cisco Secure IDS auch als Plug-In für einen Cisco Catalyst 6000 Switch. Dies ermöglicht direkt den Einsatz des IDS im Switch.

- Ausbaustufen:
  - IDS-4210: 45 Mbit/s ca. EUR 7.000
  - IDS-4235: 200 Mbit/s ca. EUR 11.000
  - IDS-4250: 500 Mbit/s ca. EUR 29.000
- Signatur-Analyse:
  - Vordefinierte Signaturen: >300
  - Erkennung von benutzerdefinierten Zeichenketten möglich
  - Updates etwa alle 60 Tage
  - Defragmentierung, Zustandsüberwachung, aber keine Protokolldekodierung
- Reaktion:
  - TCP Reset möglich
- Grafischer Client
  - Microsoft Windows NT
  - Solaris

3.4.2 RealSecure

RealSecure (*http://www.realsecure.com*) ist eines der ersten kommerziellen IDS-Systeme. Die aktuelle Version ist 7.0. Dieses NIDS kann auf verschiedenen Betriebssystemen eingesetzt werden. Zusätzlich wurde ein HIDS-Modul entwickelt: RealSecure Server Sensor.

- Preis
  - Network-Sensor: ca. EUR 9.500
  - Management: ca. EUR 500 pro Sensor
- Betriebssysteme:
  - Microsoft Windows NT/2000
  - Sun Solaris
  - Red Hat Linux
- Signatur-Analyse:
  - Vordefinierte Signaturen: >1.200
  - Unterstützung der Snort-Syntax (Trons Snort)
  - Defragmentierung, Protokolldekodierung, Zustandsüberwachung
- Reaktion
  - Konfiguration der Checkpoint FW-1
- Grafischer Client
  - Microsoft Windows NT/2000

3.4.3 Snort

Snort (*http://www.snort.org*) ist ein freies Open Source NIDS-System. Die aktuelle Version ist 1.8.8. In der Version 1.8.6 ist es von den ISCAlabs (*http://www.iscalabs.com*) zertifiziert worden. Dieses NIDS wurde zu Beginn von Marty Roesch entwickelt und kann auch in Form einer Appliance von verschiedenen Firmen (z. B. Sourcefire *http://www.sourcefire.com*) erworben werden. Inzwischen arbeiten an der Weiterentwicklung von Snort verschiedenste Gruppen mit. So wurden auch grafische Clients für dieses System entwickelt, deren Einsatz später in diesem Buch beschrieben wird. Snort ist ein echtes Open Source-Projekt.

- Preis
  - Open Source, daher keine Lizenzkosten
  - Als Appliance: Sourcefire NS 3000 für Gigabit-Netzwerke: $ 9.995
- Betriebsysteme:
  - UNIX (*BSD, Linux etc.)
  - Microsoft Windows NT

- Signatur-Analyse:
  - Vordefinierte Signaturen: 1881 (Stand 02.09.2002)
  - Tägliche Updates verfügbar durch verschiedenste Websites
  - Einfache Definition neuer Signaturen durch den Benutzer
  - Defragmentierung, Protokolldekodierung, Zustandsüberwachung
- Reaktion
  - TCP Reset, ICMP Unreachable
  - Konfiguration einer Firewall: Checkpoint FW-1, Cisco PIX, Cisco Router (mit SnortSam-Plug-In)
- Grafischer Client für Verwaltung und Analyse
  - Microsoft Windows NT/2000
  - UNIX

3.4.4 Network Flight Recorder NID

NFR (*http://www.nfr.com*) produziert sowohl ein NIDS als auch ein HIDS. Hier soll zunächst das NIDS besprochen werden. Network Flight Recorder ist von Marcus Ranum gegründet worden. Marcus Ranum hat mit der DEC Seal die wahrscheinlich erste Firewall entwickelt. Die NIDS-Sensoren werden als vorkonfigurierte Appliances zur Verfügung gestellt. Es existiert jedoch auch die Möglichkeit, die Software auf üblicher PC-Hardware zu installieren, wenn nur geringe Netzwerkdurchsätze erwartet werden. Die Verwaltung erfolgt von einem Desktop-Rechner.

- Preis
  - NID 310 100 Mbit/s
    - Nur Software: ca $ 7.000
    - Appliance: ca. $ 10.000
  - NID 315 Appliance 3x100 Mbit/s: ca. $ 16.000
  - NID 325 Appliance 1x1 Gbit/s: $ 18.000
- Signatur-Analyse:
  - NCode – Signatur-Sprache, viele Websites bieten Updates in NCode
  - Defragmentierung, Protokolldekodierung, Zustandsüberwachung
- Reaktion
  - TCP Reset
  - Konfiguration einer Firewall

- Grafischer Client für Verwaltung und Analyse
  - Microsoft Windows NT/2000
  - UNIX

3.4.5 Dragon

Dragon (*http://www.enterasys.com*) produziert sowohl ein NIDS als auch ein HIDS. Hier soll das NIDS besprochen werden. Der Dragon Sensor wird als konfigurierte Appliance oder als Software zur Verfügung gestellt.

- Preis
  - Dragon-Sensor
    - Nur Software: ca $ 3.000 – $ 15.000
    - Appliance: ca. $ 7.500 – $ 22.000
- Signatur-Analyse:
  - Vordefinierte Signaturen: >1300
  - Benutzerdefinierte Signaturen mit Scriptsprache
  - Wöchentliche Updates
  - Defragmentierung, Protokolldekodierung, Zustandsüberwachung
- Reaktion
  - Benutzerdefinierte Befehle
- Grafischer Client für Verwaltung und Analyse
  - UNIX

3.4.6 Entercept

Entercept (*http://www.entercept.com*) ist ein HIDS. Der Hersteller bezeichnet es als ein Verteiltes Host Intrusion Prevention-System. Es überwacht die Systemaufrufe im Betriebssystem und verhindert so einen Einbruch. Hierzu erlaubt es die Definition von Richtlinien in einer Regeldatenbank. Die Agents überwachen zentral gesteuert die einzelnen Rechner. Der Enteract Agent ist verfügbar für Microsoft Windows NT und 2000 und Sun Solaris. Die Verwaltungskonsole ist nur für Microsoft-Betriebssysteme verfügbar.

- Preis
  - Enteract Sensor: ca. $ 1.300
  - Enteract Console: ca. $ 5.000
- Überwachung:
  - Sämliche Prozessaufrufe
  - Zugriffe/Modifikation der Registry (Nur WinNT/2k)

  - Protokolldateien
  - Installation von Hintertüren
- Reaktion
  - Verhindert die Ausführung des Angriffes
- Grafischer Client für Verwaltung und Analyse
  - Microsoft Windows NT und 2000

3.4.7 NFR HID

Das NFR HID (*http://www.nfr.com*) ist ein HIDS der Firma NFR (s.o.). Es handelt sich ebenfalls um ein Verteiltes Host Intrusion Detection-System. Zusätzlich bietet es als Intrusion Management-System (IMS) auch NIDS-Funktionalitäten. Die Agents überwachen zentral gesteuert die einzelnen Rechner. Das NFR HIDS basiert auf der Centrax-Technologie von Cybersafe (*http://www.cybersafe*) und wurde stark modifiziert und erweitert. Der Agent ist verfügbar für Microsoft Windows NT4.0, Windows 2000, Sun Solaris 2.5, 2.6, 7.0 und 8.0, HP-UX 10.2 und 11.0 und IBM AIX 4.2.1, 4.3.2 und 4.3.3.

- Preis
  - StarterPack (Konsole + 10 Sensoren) ca. $ 13.000
  - Weitere Sensoren: ca. $ 800
- Überwachung:
  - Benutzerverhalten
  - Netzwerkverkehr des Rechners
  - Protokolldateien
  - Integritätstests mit *Tripwire*
- Reaktion
  - Abmeldung des Benutzers
  - Deaktivierung des Benutzerkontos
  - Aktivierung von *Tripwire*
- Grafischer Client für Verwaltung und Analyse
  - Microsoft Windows NT und 2000

3.4.8 LIDS

LIDS (*http://www.lids.org*) ist ein freies Open Source HIDS. Es arbeitet ähnlich wie das Werkzeug Enteract. Das Linux Intrusion Detection-System überwacht sämtliche Zugriffe auf Ressourcen im Kernel und kann den Zugriff entsprechend

definierter Richtlinien erlauben oder verweigern. LIDS steht nur als Patch für Linux-Kernel 2.2 und 2.4 zur Verfügung.

- Preis
  - Open Source
- Überwachung:
  - Benutzerverhalten
  - Portscans
  - Protokolldateien
  - Verhalten der einzelnen Prozesse
  - Kernelprozesse/Module
- Reaktion
  - Abmeldung des Benutzers
  - Verweigerung des Zugriffes
- Grafischer Client für Verwaltung und Analyse
  - Nein

3.4.9 Tripwire

Tripwire (*http://www.tripwire.org*) ist ein freies Open Source HIDS für Linux. Varianten für andere Betriebssysteme sind kommerziell erhältlich (*http://www.tripwire.com*). Es handelt sich um ein Werkzeug, welches die Integrität der Systemdateien in regelmäßigen Abständen überprüft.

- Preis
  - Open Source
- Überwachung:
  - Dateien und Datei-Inhalte
- Reaktion
  - Lediglich Alarmierung per E-Mail
- Grafischer Client für Verwaltung und Analyse
  - Nein, der Tripwire Manager ist lediglich kommerziell verfügbar für Microsoft Windows NT4.0 und 2000 und Sun Solaris (SPARC) 7 und 8. Der Tripwire-Manager kann dann die Tripwire-Installation auf den folgenden Plattformen verwalten und überwachen: Windows NT4.0 und 2000, Solaris (SPARC), IBM AIX; HP-UX, Compaq Tru64, FreeBSD und Linux.