Intrusion Detection Management

Bericht zum 2. praktischen Studiensemester

rainer.giedat@ncc-consulting.de

14. November 2003

Inhaltsverzeichnis

1 Die Firma

1.1 ncc network consulting GmbH

Die **ncc network consulting GmBH** besteht seit 1996 und hat ca. 20 Mitarbeiter auf zwei Standorte (München und Zürich) verteilt. Letztes Jahr ist die Firma **secu7** in **ncc** aufgegegangen, weil *Network Management und Security einfach zusammen gehört*.

Die **ncc** ist Micro$oft und HP Platin Partner. Sie verkauft unter anderem HP Software Produkte (vornehmlich aus er HP OpenView Reihe) und bietet dazu deren Integration, die Schulungen und den 1st Level support aus einer Hand an.

Seit einiger Zeit engagiert sich **ncc** intensiv im *Service Management* unter Verwendung der *ITIL*. Die IT Infrastructure Library (ITIL) ist der international anerkannte De-facto-Standard für ein professionelles IT Service-Management. Das IT Service-Management nach ITIL ermöglicht eine zielgerichtete, geschäftsprozessorientierte, benutzerfreundliche und kostenoptimierte Überwachung des IT-Service.

Die IT Infrastructure Library (ITIL) ist ein Regelwerk (Standard- bzw. IT-Bibliothek). Dieser Standard setzt sich aus einer Reihe von Modulen bzw. Disziplinen zusammen, welche den Unternehmen eine bessere Nutzung ihrer IT-Ressourcen erlauben und eine bessere Qualität des IT-Service gewährleistet. Wichtig ist hier der Bezug auf die Kerngeschäfte der Unternehmen bzw. der Organisation und die Konzentration der IT-Service auf die bestmögliche Unterstützung dieser Geschäftsprozesse.

2 Intrusion Detection

Intrusion Detection Systeme (IDS) dürfen in einer modernen Sicherheitsarchitektur nicht fehlen und sollten mitlerweile so selbstverständlich wie Virenscaner eingesetzt werden. Sie geben nicht nur Aufschluß über bereits kompromitierte Systeme und geben damit den Sicherheitsbeauftragten eines Unternehmens die Möglichkeit größeren Schaden durch Hintertüren, installierte Sniffer oder trojanische Pferde zu vermeiden, sondern erkennen meist auch Einbruchsversuche und geben dadurch Einblick in die Techniken des Angreifers. Mit diesen Informationen können die zuständigen Administratoren die Sicherheit ihres Netzes sukzessive verbessern, da sie die vom Angreifer ausgenutzten Schwachstellen vor Augen geführt bekommen und auch neue Techniken der Cracker kennen lernen.

80% der Angriffe auf ein Netzwerk kommen laut statistischen Schätzungen von Innen. Wegen dieser hohen Zahl ist es oft für die Verantwortlichen interessant ob die betriebsinternen Sicherheitsregeln von den Mitarbeitern eingehalten werden. (siehe: 3.3 Rechtliche Bedingungen) So können verschiedene IDS das unerlaubte Installieren von Software, das Verändern der Systenkonfiguration eines Rechners, das Benutzen „verbotener“ Netzwerkprotokolle und Surfen auf Internetseiten mit zweifelhaftem Inhalt als Angriff werten und Alarm schlagen, bzw. die Daten zur späteren Kenntnisnahme speichern.

Im Prinzip ist es auch möglich Angriffsversuche abzublocken. Dieses Feature sollte aber sehr gut durchdacht werden, da so ein Schuß leicht nach hinten losgehen kann (besonders bei fehlerhaft konfigurierten Systemen).

2.1 Verschiedene Ansätze

Diese vielfältigen Funktionalitäten können fast selbstverständlicher Weise nicht von einem einzigen IDS abgedeckt werden. Es läuft also bei der Planung immer auf mehrere verschieden geartete Systeme hinaus, die sich möglichst sinnvoll ergänzen, statt sich in der Funktionalität zu überschneiden.

Üblicherweise unterscheidet man drei verschiedene Ansätze.

2.1.1 Application-Based

AIDS sollen den unter Sicherheitsaspekten fehlerhaften oder gefährlichen Umgang mit einzelnen Softwarekomponenten erkennen. Der zu treibenden Aufwand ist im Allgemeinen nur für Sicherheitskritische Server zu rechtfertigen. Aufgrund der grundverschiedenen Softwarearchitekturen ist es beinahe unmöglich ein universell einsetzbares AIDS zu entwickeln. Im Prinzip enthält doch beinahe jede ernst zu nehmende Serversoftware ein rudimentäres IDS: Ihre LOG-Files. In ihnen werden merkwürdige Vorkommnisse (Authentifizierungsfehler...) festgehalten.

Die Aufgabe des Sicherheitsadministrators ist es dann, diese LOG-Files möglichst zentral zu sammeln und evtl. automatisch vorzubehandeln um ihre Verwertung möglichst einfach und damit effektiv zu gestalten.

Ein interessanter Ansatz unter Linux sind die so genannten *VServer*. Hier kann ein Prozess in einer auf Betriebsystemebene völlig abgeschotteten Umgebung (virtuelle Server) laufen. Durch ein Regelwerk ist genau definiert auf welche Systemresourcen der Prozess Zugriff für seine korrekte Ausführung benötigt. Alles andere ist schlichtweg verboten. Ein Cracker der die Software überlistet hat, hat also keinen Zugriff auf das restliche System und die Meldungen über die Regelverletzungen geben Aufschluß über seltsame Vorkommnisse.

Die Verantwortlichkeit des Administrators liegt hier in der Erstellung der Regelwerke, sofern sie nicht schon vorhanden sind.

2.1.2 Host-Based

HIDS überwachen - wie der Name vermuten lässt - die Sicherheit eines einzelnen Hosts. So gesehen sind AIDS ja auch HIDS nur mit dem Unterschied, daß HIDS sich um den gesamten Rechner anstatt nur um eine Teil kümmert. Erste Anlaufstelle sind hier natürlich ebenfalls die LOG-Files, die zur Auswertung zentral gesammelt werden sollten (was ja auf jedem vernünftigen Netzwerkbetriebssystem standardmäßig vorgesehen ist). Syslog unter U**x zum Beispiel.

Als HIDS werden jedoch meist speziell für die Sicherheitsüberwachung erstellten Programme bezeichnet. Das prominenteste Beispiel ist wahrscheinlich **Tripwire**. Dieses Programm errechnet mit einer Hash-Funktion Checksummen über einen konfigurierten Teil des Dateisystems. Wird jetzt eine der Dateien verändert stimmt die vorher errechnete Checksumme nichtmehr. Das funktioniert aber auch, wenn die Datei nur geöffnet oder gelesen (etwa kopoiert) wird, da dabei die I-Nodes im Dateisystem geändert werden. Die verwendeten Algorithmen sind Einwegfunktionen und damit extrem schwer (wenn überhaupt) zu fälschen. Einwegfunktionen gelten als praktisch eindeutig.

2.1.3 Network-Based

NIDS versuchen anhand des Netzwerkverkehrs Angriffe oder Angriffsversuche zu entdecken. Hier wird je nach Positionierung der gesamte Verkehr zu einem bestimmten Rechner oder der eines gesamten

LAN-Segments (pro Sensor) auf Auffäligkeiten hin untersucht und gegebenenfalls der Administrator alarmiert bzw. mit einer aktiven Abwehrreaktion reagiert.

In diesem Text beschränke ich mich auf die Betrachtung der Netzwerkbasierten Intrusion Detection Systeme.

3 Network-Based Intrusion Detection Systems (NIDS)

3.1 Ansätze zur ID

3.1.1 Pattern Matching

Pattern Matching ist die einfachste Form der ID und am Besten mit einem einfachen Virenscaner zu vergleichen. Jedes empfangene Netzwerkpaket wird für sich anhand seines Aufbaus und Inhalts untersucht und bewertet. Erkannt werden somit Angriffe die mit einem Paket eindeutig erkannt werden können, wie etwa gefährliche JavaScript Routinen in Webseiten oder auch Viren in E-Mails. Der Header der Datenpakete wird natürlich auch inspiziert, so daß z.B. die berühmte DoS Attacke in Form eines UDP-Pakets an den Port 0 Alarm schlägt. Der große Nachteil bei dieser Form der ID ist, daß meist nur Regeln für bekannte Sicherheitsprobleme erstellt werden können, die dann nur für ein spezielles Problem gültig sind.

Solche Sicherheitslücken, die durch diese Art speziefischer Mustererkennungsregeln erkannt werden können, sollten aber besser sofort nach deren Entdeckung durch Einspielen der Patches oder umkonfigurieren der Firewall[1] radikal ausgemerzt werden. Dieser Weg ist einer Einpflegung ins IDS selbstverständlich vorzuziehen[2].

Geschickt gewählt können solche Mustererkennungsregeln aber auch für ganze Klassen von Angriffen gültig sein.

Beispiel: Bufferoverflow Attacken

Ein Programmierfehler einer Software wird ausgenutzt um ein eigenes Programm auszuführen. Die Art und Position des Fehlers ist fast selbstverständlich immer verschieden. Ähnlich ist jedoch meist das Programm das ausgeführt werden soll: Eine Shell. Darum wird man durch Suchen nach der Zeichenkette ’/bin/sh’ oder dem häufig verwendeten Maschinencode auch auf die meisten dem Administrator unbekannten Sicherheitslücken aufmerksam. Im Idealfall sollte sich dieser daraufhin schleunigst auf die Suche nach einem Patch machen.

Die meisten IDS schalten der Mustererkennung einen Puffer vor der alle Fragmente einsammelt und zusammenfügt damit ein Angreifer der den Einsatz eines IDS ahnt nicht die Möglichkeit zu geben seinen Angriff durch „Zerstückeln“ zu verschleiern.

3.1.2 Statefull Pattern Matching

Statefull Pattern Matching wird verwendet wenn die einzelne Datenpakete eines Angriffsversuchs zwar unverdächtig, sie im Zusammenhang jedoch unmißverständlich sind. Ein Verbindungsaufbau zu

[1] Packetfilter ist gemeint. ID und Virenscaner gehören natürlich auch zu einer ordentlichen Firewall

[2] Außer man interessiert sich für bestimmte Angriffshäufigkeiten

Port 80 des Webservers ist im Allgemeinen nicht ungewöhnlich, mehrere Verbindungsversuche zu verschiedenen Ports ausgehend von einer Adresse jedoch mit Sicherheit ein Portscan und damit evtl. ein Hinweis auf einen bevorstehenden Angriff. Um möglichst alle Arten von Portscans zu entdecken, sollte nicht nur der zu schützende Host sondern das gesamte Netzwerk überwacht werden. Den Scan nach einem bestimmten Port über mehrere Hosts könnte sonst nämlich selbst ein IDS mit statefull pattern matching nicht erkennen.

3.1.3 Protocol Decode-Based Analysis

Unter **Protocol Decode-Based Analysis** versteht man die Überprüfung der Einhaltung eines Protokollstandards. Diese Art der ID ist sehr aufwändig und wird äußerst selten verwendet, denn Protokolldefinitionen sind meist sehr umfangreich und gelegentlich schwammig formuliert (unzählige MAY oder SHOULD in RFC's). Oft sind protokollspeziefische Probleme auf Programmierfehler zurückzuführen. Warum sollte denn eine IDS nicht auch Programmierfehler beinhalten? Protocol Decode-Based Analysis kann also nur zur Erkennung von durch Designprobleme des Protokolls verursachte Sicherheitslücken sinnvoll eingesetzt werden.

3.1.4 Anomaly-Based Analysis

Anomaly-Based Analysis ist seit einiger Zeit Gegenstand intensiver Forschungen. Durch dieses Verfahren soll ein IDS dazu befähigt werden Angriffe von sich aus, automatisch ohne genaue Kenntnisse über die Art der Angriffe zu erkennen.

In der Praxis werden von kommerziellen Systemen zwei Ansätze implementiert:

- **Parametrisierbar**

  Bei der Konfiguration des IDS wird parametrisiert, was normal ist. Beispielweise die im Netz verwendeten Protokolle und welche Hosts über diese miteinander kommunizieren. Dabei können zusätzlich statistische Werte als Grundlage zur Klassifizierung verwendet werden. Oft vorkommende Alarme deuten für gewöhnlich auf Fehlkonfiguration des IDS hin, während seltenere Alarme auf Bedienungsfehler der Anwender oder Angriffe zurückzuführen sind.

- **Statistisch**

  In diesem Fall wird dem IDS eine „Einarbeitungszeit" gewährt, in der es das Netzt beobachten und Statistiken zum Netzwerkverkehr ausarbeiten kann. Die so gewonnenen Erkenntnisse werden dann als Grundlage für die Überprüfungen (wie oben beschrieben) verwendet. Im Laufe des Betriebs werden die Statistiken ständig aktualisiert um sich an eventuelle Veränderungen anzupassen.

Bei beiden Varianten sind jedoch die Anzahl der Fehlalarme relativ hoch, denn auch Fehler in der Anwendersoftware, Bedienungsfehler der Anwender oder sogar nur selten Verwendete Anwendungen schlagen Alarm.

3.2 Positionierung

Um seine Aufgabe erfüllen zu können muß ein NIDS natürlich den gesamten für es relevanten Netzverkehr sehen können. In einem geswitchten Netzwerk ist das jedoch schwierig. Hier sollte man das NIDS an den Monitorport des Switches hängen. Genügend Bandbreite natürlich forausgesetzt, denn wie soll denn z.B. der gesamte Verkehr von 23 100mBit FastEthernet Ports über einen 100mBit Port zum NIDS übertragen werden?

Entscheidend für die Sichtbarkeit ist im Allgemeinen die Positionierung des so genannten Sensors im Netzwerk. Hier unterscheidet man:

- Host-Target Co-Location
- Host-Target Seperation

Soll der Sensor also auf einem zu überwachenden Netzwerkknoten oder seperat als unabhängiges System installiert werden? Beides hat Vor- und Nachteile:

- Vorteile Seperation
  - Manche Angriffe auf mehrere Ziele können erkannt werden (one-to-many portscan)
  - Keine zusätzliche Belastung der Endsysteme
  - NIDS können sich hier leichter verstecken, wenn sie keine eigene IP-Adresse haben und keine Daten aussenden.
  - Bei einer Installation auf Routern oder Firewalls ist die autom. Abwehr einfacher (keine Race-Conditions).
- Nachteile Seperation
  - Verschlüsselte Verbindungen problematisch
  - Evtl. Bandbreitenprobleme
  - Erkennt nur Angriffsversuche - nicht aber deren Erfolg
- Vorteile Co-Location
  - Weniger Performanceprobleme hinsichtlich Netzanbindung
  - Beobachtung des verschlüsselten Datenverkehrs möglich
- Nachteile Co-Location
  - Müssen auf jedem zu überwachenden System installiert und gepflegt werden.
  - Belastung des Systems
  - leichter zu entdecken und anzugreifen

3.3 Rechtliche Bedingungen

Um seiner Aufgabe nachzukommen muß ein NIDS den gesamten Netzwerkverkehr mitlesen, analysieren und die Ergebnisse protokollieren. Davon sind natürlich ebenfalls die personenbezogenen bzw. personenbeziehbaren Daten (Pseudonymisierung durch IP-Adressen) der Angestellten betroffen, insbesondere wenn private Internetnuztung am Arbeitsplatz erlaubt ist. Um rechtlichen Streitereien im Vornherein aus dem Weg zu gehen und um den Schutz der Daten im Interesse der Arbeitnehmer zu gewährleisten, ist der Einsatz eines NIDS in einer Firma während der Planung unbedingt mit dem Betriebsrat abzuklären, da dieser nach §87 BetrVG ein Mitbestimmungsrecht hat auf die Einführung von technischen Systemen die eine Überwachung des Verhaltens der Arbeitnehmer ermöglichen.

In diesem Zusammenhang kann grundlegend überlegt werden, ob die Überwachung des Übergangs vom internen Netz ins Internet genügen würde, da sich im Vergleich zur Überwachung des gesamten internen Netzes auf diese Weise viele personenbeziehbare Daten im Vornherein vermeiden lassen könnten (Grundsatz der Datenvermeidung, -sparsamkeit nach §3 TDSV). Ich persönlich würde von diesem Ansatz Abstand nehmen um die Sicherheit des Netzes durch interne Angreifer nicht unnötig zu gefährden.

Der zu bevorzugende Weg liegt in (§4 BDSG, §4 TDDSG) der expliziten Einwilligung der Arbeitnehmer. Trotzdem darf die Erhebung, Verarbeitung oder Nutzung nur streng zweckgebunden vor sich gehen (§28 BDSG) und die mit dieser Aufgabe betrauten Personen müssen in jedem Fall auf die Einhaltung des Datengeheimnisses verpflichtet werden (§5 BDSG). Stellt eine betroffene Person den Antrag auf Einsicht der über sie gespeicherten Daten und deren Zweck ist ihr dieser zu gewähren (§19 BDSG).

Grundsätzlich steht dem Einsatz eines NIDS rechtlich nichts im Wege, denn nach §14(2), §14(4) und §31 BDSG ist der Betreiber oder Inhaber eines Netzes befugt, personenbezogene Daten zu erheben, verändern oder zu nutzen, wenn dies zur Verfolgung von Straftaten oder Ordnungswidrigkeiten erforderlich ist. Außderdem dürfen diese Daten ebenfalls zur Sicherstellung des ordungsgemäßen Betriebs einer DV-Anlage verwendet werden. Im Falle einer Straftat oder ordnungswidrigen Verletzung von Sicherheitsrichtlinien dürfen diese Daten sogar zur Verfolgung der Täter genutzt werden (§14(2) BDSG).

3.4 False-Positives-Negatives

Der Sicherheitsgewinn durch Intrusion Detection Systeme steigt und fällt mit Anzahl und Qualität der Alarmmeldungen.

Daß auf nicht erkannte Angriffe nicht reagiert werden kann ist selbstverständlich und das ein IDS das die meisten Angriffe nicht erkennt sinnlos ist, auch. Um diese False-Negatives zu vermeiden muß die Signaturdatenbank - wie bei einem Vierenscanner - immer auf dem neuesten Stand gehalten werden. Geschickte Bösewichte versuchen ihr Vorhaben durch Manipulation der Datenpackete, die den Angriff ausmachen, unbemerkt am NIDS vorbei zu mogeln. Ein verbreitetes Verfahren liegt in der Fragmentierung. Die meisten modernen IDS sammeln jedoch Fragmente, setzen sie zusammen und untersuchen sie erst, wenn sie komplett sind. Die Vereitelung solcher Verdunklungsversuche liegen meist in der Hand der NIDS selbst und selten beim Administrator.

Leider am Anfang viel zu leicht unterschätzt, aber mindestens genauso gefährlich sind die False-Positives - Fehlalarme. Ein frisch installiertes und schlampig designtes NIDS überschüttet den Administrator regelrecht mit Alarmmeldungen. Die Gefahr dabei ist, daß die Übersichtlichkeit darunter

so stark leidet, daß die wirklich relevanten Informationen beinahe unauffindbar untergehen. Ein erster Schritt zur Vermeidung von nutzlosen Alarmen ist das Platzieren hinter der Firewall, also in freundlichem Gebiet. Was nutzen uns denn Alarme über Angriffe, die sowieso von der Firewall geblockt werden? Viel interessanter ist doch, welche Angriffe die Firewall durchdringen. Auf jeden Fall sollte der Administrator das standardmäßig eingestellte Regelwerk mit den Angriffssignaturen genauestens unter die Lupe nehmen und mit seine Anforderungen abgleichen. Snort zum Beispiel hält SNMP-Traps für gefährlich und schlägt alarm. Das kann sehr unangenehm werden, wenn das Netz mit SNMP überwacht wird, ist aber anderenfalls wirklich ein starkes Sicherheitsrisiko. Die Aufgabe das gesamte Regelwerk durchzusehen ist sehr mühsam, kann aber auch lehrreich in Bezug auf mögliche Attacken sein und ist der sinnvollste und effektivste Schritt um aus einer beinahe sinnlosen Datenschleuder ein mächtiges Werkzeug zur Sicherheitsüberwachung eines Netzes zu machen.

Selbst nach Durchführung dieser beiden Maßnahmen wird das NIDS bei der Inbetriebnahme wahrscheinlich viele Alarme melden. Das liegt meisten daran, daß vielen Netzwerkadministratoren bis zum Einsatz eines NIDS nicht hundertprozentig klar ist, welche Protokolle genau in ihren Netzen eingestzt werden und wie genau das geschieht. Ein Feintuning der Regeln des NIDS ist angebracht. Dazu benutzt man am besten seinen Lieblingssniffer und geht den Alarmen der Reihe nach auf den Grund. Nicht selten werden einem dabei bis Dato unbekannte Netzprobleme oder Fehlkonfigurationen vor Auge geführt (wo kommen denn all die “Source Quenches” her?). Das ist eine gute Gelegenheit das Netz auf Fordermann zu bringen.

4 Snort

Als NIDS haben wir Snort gewählt, und zwar aus mehreren Gründen.

Der wohl ausschlaggebende Grund war dessen weite Verbreitung. Dadurch wird es von vielen freien Entwicklern ständig weiterentwickelt und vor allem um viele Plugins und Zusätze bereichert.

Mit der Entdeckung neuer Angriffsmethoden erscheinen immer recht schnell auch Regeln für Snort. Das ist sehr wichtig um mit dem IDS immer gewappnet zu sein. Die Regelsprache ist äußerst mächtig, aber auch relativ einfach zu erlernen, so daß es für Interessierte ein Einfaches sein müsste selbst eigene Regeln zu verfassen.

Auch die Möglichkeiten für die Meldung der Alarme sind sehr vielseitig und lassen sich beliebig kombinieren oder selbst programmieren und in Form von Plugins einbinden.

Snort läuft auf beinahe jedem beliebigen Betriebsystem und läßt sich damit sowohl auf den zu schützenden Rechnern selbst (Host-Target Co-Location) oder als autonomer Sensor im Netz (Host-Target Seperation) betreiben. Solch ein Sensor kann auf einem gehärteten Linuxsystem und ohne IP-Adresse auf der Sensorkarte sehr sicher und beinahe unsichtbar installiert werden.

4.1 Regelsprache

Eine Snort-Regel besteht aus einem Rule-Header und den Rule-Options. Der Header enthält (fast analog zum IP-Header gehalten) das Protokoll, die Quell- & Ziel-Adressen mit Netzmasken, die Ports und am Anfang die auszuführende Aktion. Zur Verfügung stehen standardmäßig folgende Aktionen:

- alert: Alarm schlagen (Def. Methode)
- log: ins Logfile schreiben

- pass: Paket ignorieren
- activate: Alarm und aktivieren einer “dynamic rule”
- dynamic: auf Aktivierung warten und loggen

Mit den Standardaktionen `activate` und `dynamic` kann das Statefull-Pattern-Matching Modul verwendet werden um zusammengehörige Datenpackete in ihrem Kontext zur späteren Untersuchung abzuspeichern.

Man kann Aktionen (oder Rule-Types) auch selbst definieren:

```
ruletype sql_log
{
type alert output
alert_syslog: LOG_AUTH LOG_ALERT
output database: log, mysql, user=snort dbname=snort host=localhost
}
```

Obige Konstruktion definiert eine neue Ausgabe für Alarme (`type alert output`). Es wird festgelegt, wie sie vom Syslogdaemon behandelt werden sollen (`alert_syslog...`) und das die Alarme einerseits an das Systemlog übergeben, andererseits in eine lokale mySQL Datenbank mit dem Namen Snort geschrieben werden sollen.

Die Rule-Options enthalten Alarmmeldung und Angriffsdefinition, meistens in Form der Beschaffenheit des Payload oder spezieller Bits im IP-Header.

Angriffsdefinition:

- **IP-Paket-spezifisch:**
  - **ttl** Time to Live
  - **tos** Type of Service
  - **ipoption** IP-Options (z.B.: Source-Routing!!)
  - **flags** TCP flags
  - **itype** ICMP Type (z.B.: Echo, SourceQuench, Redirect!!...)
  - **dsize** datagram size
  - **fragbit** IP fragmentation bit
  - **seq und ack** Stati einer TCP-Verbindung
  - **ip_proto** IP Protokoll (z.B.: tcp, udp, ospf, ipsec...)
- **Payload-spezifisch:**
  - **content, content-list** Bytefolge oder eine Liste von Bytefolgen (Wörter)
  - **offset** Stelle im Payload
  - **depth** Suchtiefe

  - **uricontent** Inhalt des URI-Teils
  - **regex** Verwendung regulärer Ausdrücke
  - **nocase** keine Groß-Kleinschreibung

- **Information:**
  - **msg** Die anzuzeigende Alarmmeldung
  - **logto** Dateiname in den geschr. werden soll
  - **reference** weiter Informationen über den Angriff (WWW-URL)
  - **session** TCP-Reassembly, ganze TCP-Session
  - **priority** Gewicht des Angriffs (severity)
  - **classtype** Klassifizierung (z.B.: attempted_admin)

- **Snort Intern:**
  - **rev** Revision (Versionsverwaltung der Regeln)
  - **sid** eindeutige SnortID
  - **tag** markiert weitere Pakete um den Kontext zu loggen

4.2 LOG-Format

Standardmäßig werden Teile des IP-Headers zusammen mit der definierten Meldung, “classification” und “priority” in die Datei **/var/log/snort/alert** geschrieben.

Beispiel:

```
[**] [1:1310:5] PORN free XXX [**]
[Classification: SCORE! Get the lotion!] [Priority: 1]
12/07-15:49:20.082609 127.0.0.1:80 -> 127.0.0.1:33010
TCP TTL:64 TOS:0x2 ID:25894 IpLen:20 DgmLen:846 DF
***AP*** Seq: 0xAF345837  Ack: 0xAF4505C6  Win: 0x7FFF  TcpLen: 32
TCP Options (3) => NOP NOP TS: 2242111 2242111
```

In diesem Beispiel ist jemand beim Betrachten einer Seite mit unerwünschtem Inhalt auf seinem eigenen Rechner (auf dem auch snort läuft) erwischt worden. (Um solche sinnlosen Alarme zu vermeiden siehe **False-Positives-Negatives**.)

Im obigen Beispiel fällt auf, daß Teile des IP-Headers mitprotokolliert werden. Der Übrsichtlichkeit wegen und um hohes Datenaufkommen und starke CPU-Belastung zu vermeiden, könnte man snort konfigurieren diese Informationen für sich zu behalten oder die Alarme gleich an die Managementstation zu übertragen oder beides.

Dazu wird ein Eintrag in **snort.conf** geändert:

```
output <name>: <options>
```

Die oben genannte Änderung würde sich durch ersetzen von `alert_syslog` in `alert_fast` ergeben. In der Standardinstallation sind folgende Output-Module im “Lieferumfang” bereits enthalten:

- alert_syslog: <facility> <priority> <options>
- alert_fast: <file> (keine Header mehr)
- alert_full: <file> (komplette Pakete)
- log_tcpdump: <file> (zur Untersuchung mit tcpdump oder ethereal)
- alert_smb: <list of hosts> (WinPopUp, benötigt “smbclient”)
- alert_unixsock (experimentell)
- xml (praktisch für graphische Darstellung)
- database <log — alert>, <datbase_type>, <parameter_list>
- CSV (Comma Seperated Value, für Datenbankimport)
- unified: sollte schnellste Methode sein, bald Standard, zwei Dateien: Paket-Header, Payload. Binär
- snmp_trap (verschickt SNMP Traps)
- log_null (nur Alarm, kein log (für eigene Def.))

Damit läßt sich jetzt schon eine ganze Menge anfangen. Ein gut durchdachtes Logging am Besten mit einer zentralen Anlaufstelle im Netz als Managementstation erleichtert das Reagieren auf ungewollte Ereignisse enorm.

4.3 Preprozessoren

Snort erlaubt das dynamische Laden von Modulen, welche die Pakete durchleuchten können, noch bevor sie durch das Regelwerk überprüft werden. Diese Module wurden von den Entwicklern von Snort **Preprocessor** getauft. Sie können zur Reduzierung von False-Positives sehr hilfreich sein und bauen Snort von einem rein auf Pattern-Matching basierten IDS zu einem mit allen Schickanen aus. Snort untertützt somit sogar mehr als alle gängigen Ansätze zur Intrusion Detection, wie Protocoll-Decoding, Statefull-Pattern-Matching und Anomaly-Detection.

4.3.1 Protocol-Decode

Für Snort sind im Augenblick vier Protocol-Decode Module vorhanden. Sie sind leider (wär ja auch zu viel verlangt) keine vollständigen Protocol-Decoder. Vielmehr parsen sie die einzelnen Pakete diverser Protokolle und “normalisieren” den Inhalt. Damit ist gemeint, daß sie zu lange oder verstümmelte Anfragen erkennen und korrigieren, bevor sie an zur Hauptuntersuchung weitergegeben werden.

Folgende Protokolle werden unterstützt:

- **HTTP** entfernt auch Unicodes in URLs und erkennt CGI-Null Attacken
- **RPC** Remote Procedure Call - bringt Anfragen in das standard 4-Byte format
- **Back Orifice** (kann sogar Brute Force Attacken auf den vewendeten Schlüssel durchführen - LANGSAM!)
- **Telnet (und FTP)** bringt den Inhalt von Telnet und FTP-Paketen in ein Format, so daß Snort den Inhalt ohne Modifizierungen mit `content` testen kann.

4.3.2 Portscan Detector

Ein Portscan wird von diesem Plugin definiert als **P** oder mehr TCP-Anfragen/UDP-Pakete in **T** Sekunden. Die Werte **P** und **T** lassen sich selbstverständlich konfigurieren. Die aktuelle Version erkennt ebenfalls Stealth, FIN und XMas scans.

Bei einem Portscan muß nicht zwingend ein Rechner (IP) alle Ports eines Anderen abklappern. Es kann auch vorkommen, daß ein Rechner mehrere Andere nach einem bestimmten Port absucht (zur Zeit ist offensichtlich MS-SQL modern). Der Portscan Detector erkennt nicht nur beide dieser Möglichleiten sondern verät oft auch das Programm, mit dem der Scan durchgeführt wurde.

Um False-Posotives zu minimieren gibt es das Zusatzmodul **Portscan Ignorehost**. Damit können Adressen definiert werden, von denen aus "Portscans" erlaubt sind. In der Regel werden aber selten von bestimmten Adressen Portscans initiiert (obwohl es diesen Security-Ansatz gibt). Hier werden bei Bedarf Adressen von Servern eingetragen, die dazu neigen Portscans zu simulieren. Denken wir dazu einmal an einen NTP-Server. Dieser wird oft in kurzer Zeit zu vielen Rechnern Kontakt aufnehmen um die Systemuhr zu synchronisieren. Dieser Vorgang würde für den Portscan Detector wie ein Portscan aussehen.

4.3.3 Frag2

Dieses Plugin kümmert sich um die IP-Defragmentierung. Natürlich lassen sich auc hier alle wichtigen Parameter wie Zeitlimits und Speicherverbrauch einstellen.

4.3.4 Stream4

Stream4 bringt die einzelnen Pakete eines TCP-Streams miteinander in Verbindung und ermöglicht damit Statefull Analysis. Auf diese Weise kann Snort auch Fehler im Verbindungsaufbau (Drei-Wege-Handschlag) und Portscans zuverlässig erkennen. Standardmäßig arbeitet dieses Plugin aus Performancegründen nur für die Ports 21, 23, 25, 53, 80, 110, 111, 143 und 513, das läßt sich aber selbstverständlich umkonfigurieren. Apropos Performance: Dieses Plugin fügt Snort einen neuen Kommadozeilenparameter hinzu (-k) der die Überprüfung aller Checksummen abstellt. Dieses Feature ist interessant, wenn in einem Netz die Router Pakete mit fehlerhafter Checksumme sowieso verwerfen und eine eigene Überprüfung somit nur unnütze Zeitverschwendung wäre.

4.3.5 Spade

Spade ist ein von SiliconDefense entwickeltes Plugin und ergänzt Snort um die Möglichkeit der Anomaly Detection. Spade ist die Abkürzung von *Statistical Packet Anomaly Detection Engine*. Spade

berechnet für jedes Paket die Unwahrscheinlichkeit ($-log_2(P(X))$) mit der dieses Packet auftritt und vergleicht diese mit dem konfigurierten Wert. Ist das Auftreten unwahrscheinlich genug, schlägt Spade alarm. Standardmäßig wird Spade anfangs keine Alarme senden, da der Default-Threshold negativ ist, kann aber von Anfang an auf eine Wert gesetzt werden. Spade lernt im laufenden Betrieb und baut sich langsam eine Wahrscheinlichkeitstabelle (Propability-Table) auf. Diese wird regelmäßig in eine Datei gesichert und bleibt damit bei einem Neustart von Snort erhalten.

Das Schwierigste is wohl den besten Threshold zu finden um eine möglichst nur die Interessante Pakete zu Gesicht zu bekommen, aber keine zu verpassen. Spade kann so konfiguriert werden, daß es eine vorgegebene Zeit still läuft und nach Ablauf der Zeit einen Wert vorschlägt, mit dem eine vorher bestimmte Zahl von Alarmen entstanden wäre.

Anomaly Detection sollte mit Vorsicht genossen werden, da dieses System erst am Anfang der Entwicklung steht. Trotzdem lohnt es sich auf jeden Fall einmal einen Blick darauf zu werfen, denn es ist wohl der Einzigste Weg einem selbst unbekannte Angriffe zu erkennen und kennen zu lernen.

Aktuelle Informationen zu Spade:

`http://www.silicondefense.com/software/spice`

5 Management

Für die meisten nicht in der IT-Branchen angesiedelten Firmen stellt es für gewöhnlich eine große Aufgabe dar ihr Netzwerk zu entwerfen und zu Verwalten. Insbesondere, da das nötige Know-How erst eingekauft oder entwickelt werden muß. Solche Unternehmen sollten ein Outsourcing in Betracht ziehen, weil ein darauf spezialisierter Partner das nötige Wissen und die Erfahrung hat um diese Aufgabe für gewöhnlich schneller und kostengünstiger zu realisieren.

Die Firma **ncc GmBH** versteht sich als so ein Partner. Daraus entwächst natürlich der Wunsch **Intrusion Detection** und **Intrusion Reaction** als Dienstleistung dem Kunden günstig anbieten zu können. Dafür ist jedoch ein zentrales Management und ein garantiertes und schnelles Einschreiten der Spezialisten erforderlich. Für diese Aufgaben werden bei **ncc** im Allgemeinen zwei Softwarepackete verwendet:

- **Network Node Manager (NNM)**
- **Service Desk**

5.1 Network Node Manager

Der **HP OpenView Network Node Manager** ist der Quasistandard unter der Software für professionelles Netzwerkmanagement. Es arbeitet auf Basis von SNMP und ist damit in der Lage die Struktur großer Netze automatisch zu erfassen und graphisch, der Übersichtlichkeit halber in mehreren Ebenen (Teilnetze), darzustellen. Der NNM ermittelt per SNMP neben Namen, Hersteller und Netzwerkkonfiguration auch die Art der Geräte im Netz und kann ihnen in der Darstellung somit ein ihren Aufgaben angepasstes Icon zuweisen und erkennt dessen Rolle im Netz. Mit dem NNM hat man immer einen guten Überblick über den Zustand des gesamten (auch noch so großen) Netzes. Von Trafficanalysen nach ausgelesenen Statistiken bis hin zur zentralen Konfiguration der Netzwerkhardware auf SNMP-Basis deckt der NNM fast alle Anforderungen eines Administrators ab.

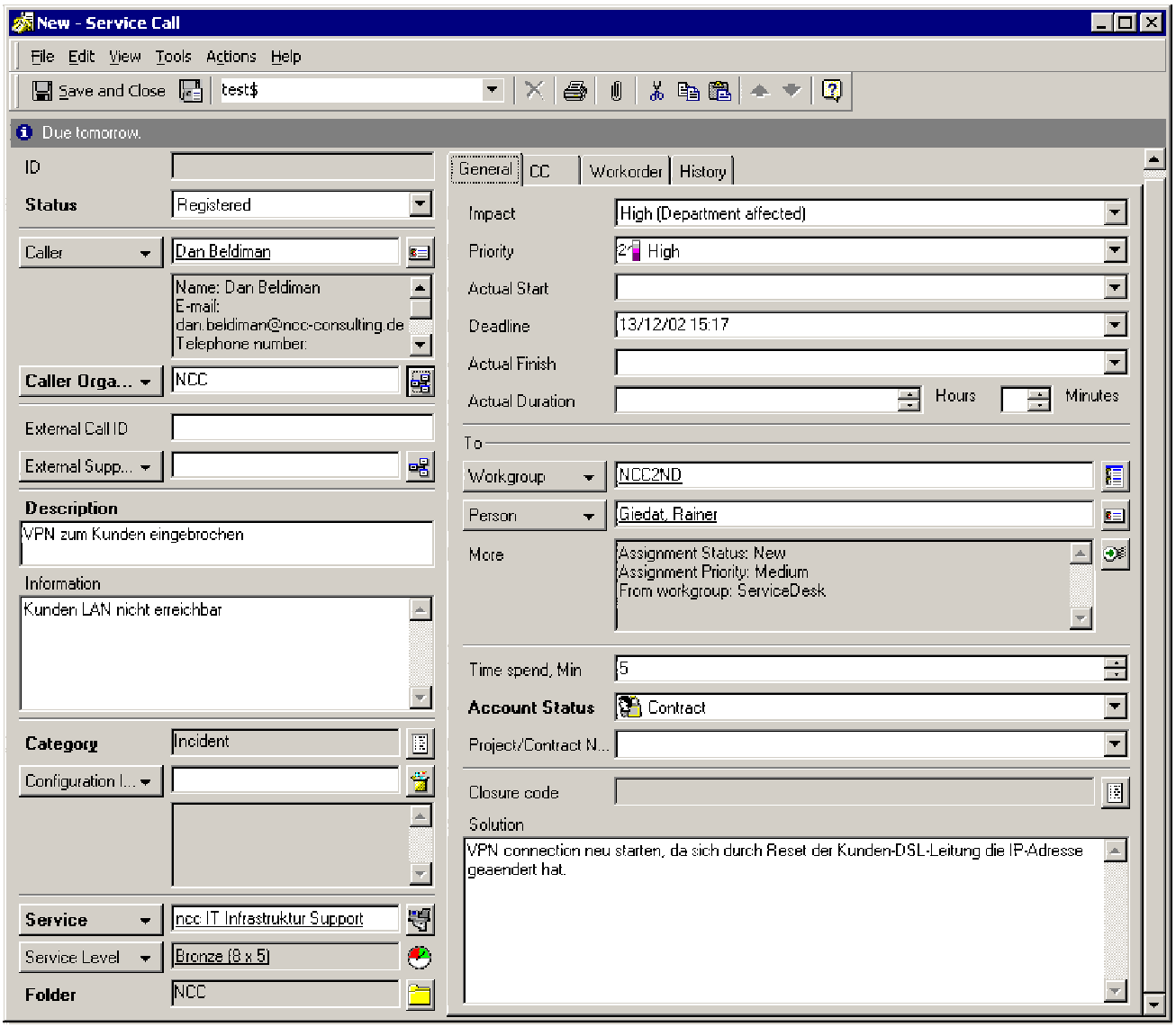
Der NNM enthält als Softwarekomponente einen Alarmbrowser. Dieser Meldet die ihm von Netzwerkknoten per SNMP-Trap zugesandten Alarme dem Adminitrator und speichert diese für die evtl. spätere Auswertung in einer Datenbank. Der NNM bietet sich dank dieser Einrichtung als Managementstation für ein snortbasiertes NIDS geradezu an, denn Snort ist bekanntlich in der Lage Alarme per SNMP-Trap zu versenden. Somit hat der verantwortliche Administrator den Zustand seines Netzes stets “im Auge” und kann auf Angriffe von Innen oder Außen schnell reagieren.

Um die NIDS für Angreifer möglichst unsichtbar zu halten und das Produktivnetz nicht unnötig zu belasten sollte man überlegen die Senoren mit zwei Netzwerkschnittstellen auszurüsten. Eine ohne IP-Adresse zum “schnüffeln” und eine, die mit den anderen Senoren und der Managementstation ein eigenes Managementnetz bilden.

5.2 Service Desk 4.5

Service Desk ist eine Help Desk Applikation aus der OpenView Produktreihe von Hewlett Packard. SD ist dazu gedacht die Aufgaben in einem Service- oder Supportzentrum zu organisieren. Wird an ein Servicezentrum eine Supportanfrage gestellt - Telefonisch oder per eMail - trägt der Mitarbeiter, der die Anfrage entgegen genommen hat diese mit einem Client in die ServiceCall Datenbank ein und weist sie dem dafür zuständigen Team zu. Die Mitglieder dieses Teams werden dann per eMail benachrichtigt. Je nach festgelegter Priorität und Vertrag mit dem Kunden hat das Team eine gewisse Zeit um sich dem Problem anzunehmen. Jeder Schritt wird im SD erfasst. So setzt der bearbeitende Angestellte z.B. den Status des ServiceCalls auf “In Progress”, wenn er sich mit dem Problem beschäftigt und auf “Finished”, wenn das Problem gelöst ist (es gibt dabei eine große Menge von Zwischenschritten, die verwendet werden können). Weil im SD die Kundenbeziehungen und vor allem die mit den Kunden abgeschlossenen Verträge sehr gut abgebildet werden können und die zur Bearbeitung notwendig gewesene Zeit erfasst wird, kann SD auch gut zur Abrechnung der Serviceleistungen benutzt werden. SD ist ein sehr umfangreiches Softwarepacket mit einer enormen Komplexität. Es beinhaltet neben dem Task-Management (natürlich mit Aufteilung in Workorders, CheckListen...) auch ein komplettes Benutzer- und Gruppenmanagement um die einzelnen Arbeitschritte möglichst einfach automatisch auf bestimmte Spezialisten/Teams verteilen zu können und den Kunden die Möglihkeit zu geben selbst direkt Supportanfragen über das System zu stellen oder den Status ihrer Anfragen abzurufen ohne Einblick in die Daten anderer Kunden zu bekommen.

Nachfolgend eine Abbildung der Maske zur Bearbeitung eines ServiceCalls:



Natürlich ist Supportanfrage nicht gleich Supportanfrage. So unterscheidet man neben anderen z.B. RFC (Request for Change) oder *Incidents*. In unserem Zusammenhang hinsichtlich des Intrusion Detection Management hätten wir also gerne bei schwer wiegenden Angriffsversuchen die automatische Erstellung eines *Incident* im ServiceDesk, der dann dem dafür zuständigen Sicherheitsspezialisten zugeordnet werden kann.

5.3 Hand in Hand zum Ziel

Zur schnellen Reaktion auf einen Angriff sollen die von Snort per SNMP-Traps an den NNM gesendeten Alarme sofort - abhängig von der im Trap mitgesendeten Priority - in das ServiceDesk eingepflegt werden. Zum automatischen generieren eines Incidents gibt es im SD das Kommandozeilentool `sd_event`.

Für die Integration ServiceDesk-NNM müssen im Node Manager erst in der Datei `trapd.conf` die Traps eingeführt werden um sie dann weiterreichen zu können. Anschließend muß im Menü `options->Event`

`Configuration` aufgerufen und der passende Event ausgewählt werden. Im `Actions`-Menü des `Modify Event`-Dialogs kann das bei Empfangen des Traps auszuführenden Kommando angegeben werden. In unserem Fall etwa so:

```
sd_event.exe -f sd_event.ini -v event_id=\"$s $2 $X $x\" \
description=\"Snort alarm from $2: $3\"
```

Dieses Kommando wird einen Incident mit der ID "<Snort-Adresse> <Priorität> <Datum> <Uhrzeit>" erstellen, der als Bechreibung den Text `Snort alarm from <Snort-Adresse>: <msg>` enthält, wobei $3 das 3te Argument des SNMP-Traps darstellt, im Allgemeinen also die Beschreibung des Angriffs. Mit diesen Informationen versorgt kann der Administrator jetzt schnell und hart durchgreifen und ...

...alles wird gut....

...oder?[3]

[3] Paranoia does not mean that they aren't right behind you