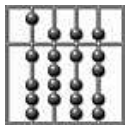

TECHNISCHE UNIVERSITÄT MÜNCHEN

FAKULTÄT FÜR INFORMATIK

Diplomarbeit

Aufbau, Betrieb und Analyse eines Honeynet

Gereon Volker

Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Dr. Helmut Reiser Andreas Völkl
Abgabedatum:	15. November 2003

Ich versichere, dass ich diese Diplomarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

....

Datum Unterschrift

Kurzfassung

Der Zweck eines Honeynet ist, das Verhalten sowie die Methoden von Angreifern zu erforschen. Ein Honeynet bildet dazu eine gewöhnliche Produktionsumgebung nach, allerdings werden die Rechner eines Honeynet nicht produktiv eingesetzt. Die Rechner sollen den potentiellen Angreifern eine „Spielwiese“ bieten, ohne dass der Angreifer dies erkennen kann bzw. sollte.

In dem Honeynet werden die einzelnen Rechensysteme mit unterschiedlichen Betriebssystemen installiert und betrieben. Wenn im Honeynet Netzwerkverkehr auftritt, so kann man davon ausgehen, dass ein Angriff bzw. ein Angriffsversuch erfolgt, da ja in diesem Netz nicht produktiv gearbeitet wird und somit auch kein Netzwerkverkehr stattfinden sollte.

Um Angriffe vom Honeynet auf andere Rechner im Internet zu verhindern muss es (nach außen) stark gesichert werden.

Eine kontinuierliche Überwachung während des Betriebes ist bei einem Honeynet unerlässlich.

Nach der Betriebsphase müssen die gesammelten Daten genau analysiert und ausgewertet werden. Anhand dieser Daten kann z.B. das Gefährdungspotential abgeschätzt und dadurch geeignete Gegenmaßnahmen ergriffen werden.

Inhaltsverzeichnis

Abbildungsverzeichnis ..... *vii*

Diagrammverzeichnis ..... *viii*

Tabellenverzeichnis ..... *ix*

1 Einleitung *1*

1.1 Aufgabenstellung **2**

1.2 Aufbau der Diplomarbeit **3**

1.3 Typographische Konventionen **3**

2 Grundlagen *5*

2.1 Honeypot **5**

2.1.1 Klassifikation von Honeypots 6

2.1.2 Positionierung eines Honeypots 8

2.2 Honeynet **9**

2.2.1 Anforderungen 9

2.2.2 Motivation 10

3 Konzeption des Honeynets *13*

3.1 Entwicklung von Honeynets **13**

3.2 Design der Architektur **15**

3.2.1 Topologie 15

3.2.2 Sicherheitskonzepte (Data Control) 18

3.2.3 Überwachungskonzepte (Data Capture) 20

3.2.3.1 Überwachung auf Netzwerkebene 20

3.2.3.2 Überwachung auf Kommando-Ebene (Shell-Ebene) 20

3.2.3.3 Sicherung der System Log-Dateien 24

3.2.4 Integritätschecker 24

3.2.5 Angebotene Dienste und zur Verfügung gestellte Daten 25

3.3 Anforderungen an die Software **26**

3.3.1 Betriebssysteme 26

3.3.2 Zusatzsoftware 27

3.4 Benachrichtigungsmechanismen **28**

3.4.1 Benachrichtigung über E-Mail 28

3.4.2 Benachrichtigung über Short Message Service (SMS) 28

3.5 Absicherung des Gateway- und Analyse-Rechners **29**

4 Aufbau und Betrieb *31*

4.1 Aufbau des Honeynets **31**

4.2 Verwendete Hardware **32**

4.3 Installation **32**

4.3.1 Installation des Gateways (*gway.lrz-muenchen.de*) ... 33
4.3.2 Installation des Linux-Honeypots (*internal.lrz-muenchen.de*) ... 38
4.3.3 Installation des Windows™-Honeypots (*tivoli.lrz-muenchen.de*) ... 40
4.3.4 Installation des Log-Rechners (*logging.lrz-muenchen.de*) ... 42

4.4 Sichern der Konfigurationen ... 44

4.5 Konfigurationsänderungen während des Betriebes ... 44
4.5.1 Update und Ergänzung der *snort*-Regeln ... 44
4.5.2 Änderungen der Firewall-Regeln ... 46
4.5.3 Vorbereitung für Auswertungen ... 46

5 Analyse des Honeynets ... 47

5.1 Vorgehensweise ... 47
5.1.1 Online-Analyse ... 47
5.1.2 Offline-Analyse ... 47
5.1.3 Nützliche Informationsquellen für die Analyse ... 48

5.2 Verwendete Analyse-Werkzeuge ... 49
5.2.1 *Iptables log* ... 49
5.2.1.1 Funktionsweise ... 49
5.2.1.2 Installation ... 50
5.2.1.3 Vorgenommene Veränderungen ... 51
5.2.1.4 Fazit ... 52
5.2.2 *Snort* in Verbindung mit *ACID* ... 52
5.2.2.1 Installation ... 53
5.2.2.2 Fazit ... 55
5.2.3 Sicherheitsaspekte bei webbasierten Analyse- und Auswertungswerkzeugen ... 56
5.2.4 *Ethereal* und *Packetyzer* ... 57
5.2.4.1 Installation ... 59
5.2.4.2 Fazit ... 59
5.2.5 *P0f* ... 60
5.2.5.1 Installation ... 61
5.2.5.2 Fazit ... 62
5.2.6 *traceroute* und *VisualRoute* ... 62
5.2.6.1 Installation ... 63
5.2.6.2 Fazit ... 63
5.2.7 Zusammenfassung ... 64

5.3 Entwicklung von zusätzlichen Werkzeugen ... 64

5.4 Ergebnisse ... 65
5.4.1 Klassifikation der Angreifer ... 65
5.4.1.1 Hacker ... 65
5.4.1.2 Cracker ... 65
5.4.1.3 Script Kiddie ... 66
5.4.1.4 Gruppenmitglieder ... 66
5.4.1.5 Professionelle Hacker ... 67
5.4.1.6 Vermutete Angreifer im Honeynet ... 67
5.4.2 Bedrohungen und durchgeführte Angriffe ... 67
5.4.2.1 Angriffe auf Web-Server ... 68
5.4.2.1.1 CodeRed2 (nach [cert 01]) ... 68
5.4.2.1.2 FX Scanner ... 70
5.4.2.1.3 BufferOverflows ... 72

5.4.2.2 Trojaner ... 72
5.4.2.3 „Mysterium 55808“ (vgl. [hei 03b]) ... 74
5.4.2.4 Viren und Würmer ... 75
5.4.2.4.1 W32.Slammer (SQLSlammer) ... 76
5.4.2.4.2 W32/Deloder-A ... 76
5.4.2.4.3 W32.Blaster (LovSan) ... 78
5.4.2.4.4 W32.HLLW.Gaobot.AA ... 81
5.4.2.5 Sonstige Beobachtungen ... 82
5.4.3 Erfolgreiche Kompromittierungen ... 85
5.5 Statistiken ... 86
5.5.1 Allgemeines ... 86
5.5.2 Häufigkeits-Statistiken ... 88
5.5.3 „Mysterium 55808“ ... 91
5.5.4 Vergleich der Aktivitäten zwischen Windows™ und Linux ... 92
5.5.5 Zusammenfassung der Analyse ... 98
6 Zusammenfassung und Ausblick ... 99
7 Glossar ... 103
8 Literaturverzeichnis ... 105
A Anhang ... 111
A.1 Skripte für Honeynet-Rechner ... 111
A.1.1 Gateway ... 111
A.1.1.1 Firewall-Skript ... 111
A.1.1.2 Regeln für *snort* ... 118
A.1.1.3 Skript für *tcpdump* ... 121
A.1.1.4 Konfiguration von *swatch* ... 121
A.1.1.5 Skripte für den SMS-Versand ... 122
A.1.2 Linux-Honeypot ... 124
A.1.2.1 „fake-syslog-Daemon“ ... 124
A.1.2.2 Perl-Skripte für den Webauftritt ... 124
A.1.2.3 Tripwire-Policy ... 125
A.2 FX Scanner-Anfragen ... 134
A.3 Passwörter des Deloder-Wurms ... 135
A.4 Benutzer und Passwörter des Gaobot-Wurms ... 136
A.5 PHP-Skripte für die Auswertung ... 137
A.5.3 Bestimmung der „Mysterium-Pakete“ pro Tag ... 137
A.5.4 Bestimmung der Anzahl der Verbindungen pro Stunde ... 138
A.5.5 Bestimmung der Top Hosts ... 139

Abbildungsverzeichnis

Abbildung 1: Unterschiedliche Einsatzszenarien eines Honeypots (vgl. [Spi 02]) 8
Abbildung 2: Honeynet als „Aquarium“ 11
Abbildung 3: Aufbau eines typischen Gen I Honeynet 13
Abbildung 4: schematischer Aufbau eines Gen II Honeynets 14
Abbildung 5: Honeynet mit Gateway und zwei Honeypots 17
Abbildung 6: Honeynet mit zusätzlichem Analyse-Server 18
Abbildung 7: Arbeitsweise von *snort_inline* 19
Abbildung 8: Funktionsweise von *Sebek*, *sebeksniff* und *sbdump* (vgl. [Bal 03]) 22
Abbildung 9: Konzeptuelle Darstellung der Umleitung des *read*-Aufrufes [Bal 03] 23
Abbildung 10: Konzeptuelle Darstellung des Generierens und Versendens von *Sebek* UDP-Paketen [Bal 03] 23
Abbildung 11: Aufgebautes Honeynet 31
Abbildung 12: Konfigurierter Gateway-Rechner mit unterschiedlichen „Abhördiensten“ 38
Abbildung 13: Konfigurations-Dialog von *TightVNC* 42
Abbildung 14: Globale Sicht mit allen beteiligten Systemen 43
Abbildung 15: iptables log 50
Abbildung 16: Startseite von ACID während des Betriebes 53
Abbildung 17: Einsatz eines SSH-Tunnels zur Online-Analyse mit Hilfe von webbasierten Werkzeugen (sehr vereinfachte Darstellung) 57
Abbildung 18: Benutzeroberfläche von *ethereal* 58
Abbildung 19: Benutzeroberfläche von *Packetyzer* 59
Abbildung 20: Beispiel-Ausgabe von *p0f* 61
Abbildung 21: Benutzeroberfläche von *VisualRoute 7.3b* unter Windows™ 63
Abbildung 22: Benutzeroberfläche von *FX Scanner* 71
Abbildung 23: Angriff des Blaster-Wurms am 11. August um 22:51 Uhr (Analyse mit *Packetyzer*) 81
Abbildung 24: Popup-Nachricht auf dem Windows™-Honeypot 84
Abbildung 25: Honeynet mit Management-Netz (Erweiterungsmöglichkeit) 102

Diagrammverzeichnis

Diagramm 1: registrierte Vorfälle bei CERT [cert 03b] 1
Diagramm 2: registrierte Schwachstellen bei CERT [cert 03b] 2
Diagramm 3: Anzahl der W32.HLLW.Gaobot.AA-Angriffe (pro Tag) 82
Diagramm 4: Anzahl der Alarme in *ACID* pro Tag 87
Diagramm 5: Dateigrößen der mitprotokollierten *tcpdump*-Dateien in MByte 88
Diagramm 6: Ports zu denen mehr als 100 Verbindungen während des Betriebs aufgebaut wurden 89
Diagramm 7: Domainzuordnungen der eingehenden Verbindungen 90
Diagramm 8: Anzahl unterschiedlicher IP-Adressen einer Domain, die Verbindungen zum Honeynet aufbauten 90
Diagramm 9: Anzahl der eingegangenen Pakete mit der *WindowSize* 55808 (*Stumbler*) 91
Diagramm 10: Anzahl der registrierten Verbindungen pro Tag 93
Diagramm 11: Anzahl der auf dem Gateway registrierten Verbindungen pro Stunde zwischen dem 15. Juli und dem 12. September 94
Diagramm 12: Anzahl der unterschiedlichen IP-Adressen pro Tag 95
Diagramm 13: Anzahl der auf dem Gateway registrierten unterschiedlichen IP-Adressen pro Stunde zwischen dem 15. Juli und dem 12. September 95

Tabellenverzeichnis

Tabelle 1: Durch Viren verursachte Schäden [EW 01] 1
Tabelle 2: Klassifizierung von Honeypots 7
Tabelle 3: Zusätzliche ausgewählte Pakete bei der Installation des Gateways 33
Tabelle 4: Zusammenfassung und Bewertung der Analyse-Werkzeuge 64
Tabelle 5: Versuchter Verbindungsaufbau von *Stumbler*; Abbruch durch Linux-Honeypot 75
Tabelle 6: Anzahl der Anfragen und IP-Adressen auf Port 1433 und Port 1434 76
Tabelle 7: Login-Versuche für die MySQL-Datenbank auf dem Linux-Honeypot 83
Tabelle 8: Registrierte Verbindungen mit Quell-Port 0 84
Tabelle 9: verwendete Hardware für die Auswertung 86
Tabelle 10: *Stumbler* IP-Adressen, von denen mindestens 6 Pakete den Linux-Honeypot erreichten 92
Tabelle 11: Vergleich der Verteilung der Anfragen auf die beiden Honeypots 93
Tabelle 12: Top-10 Rechner, die die meisten Verbindungen zum Windows™-Honeypot aufbauten 96
Tabelle 13: Top-10 Rechner, die die meisten Verbindungen zum Linux-Honeypot aufbauten 97

1 Einleitung

Das Internet hält immer weiter Einzug in unser tägliches Leben. Leider steigen auch die Fälle von Computerkriminalität jährlich an [Bka 02, Bka 03], das heißt neben dem großen Nutzen des Internets ergeben sich auch immer wieder neue Gefahren.
Schäden in Milliardenhöhe häufen sich in den letzten Jahren durch Viren und Würmer. Tabelle 1 zeigt eine Übersicht der durch Viren verursachten Schäden.

Code Name	Jahr	geschätzter Schaden
Explorer	1999	1,02 Mrd. US-$
Melissa	1999	1,10 Mrd. US-$
Love Bug	2000	8,75 Mrd. US-$
CodeRed	2001	2,62 Mrd. US-$
Sir Cam	2001	1,05 Mrd. US-$
Nimda	2001	0,59 Mrd. US-$

Tabelle 1: Durch Viren verursachte Schäden [EW 01]

Nach Angaben des britischen Unternehmens mi2g[1] sind durch den Virus *LovSan* im August 2003 weltweit Schäden in Höhe von 29,7 Milliarden US-Dollar entstanden.
Neben den Virenbefällen, stiegen auch die Anzahl der Vorfälle (Incidents) (Diagramm 1) sowie die Anzahl der bekannt gewordenen Schwachstellen (Diagramm 2) rapide an. Dabei ist zu beachten, dass sich jeweils der letzte Wert nur auf die ersten drei Quartale des Jahres 2003 bezieht.

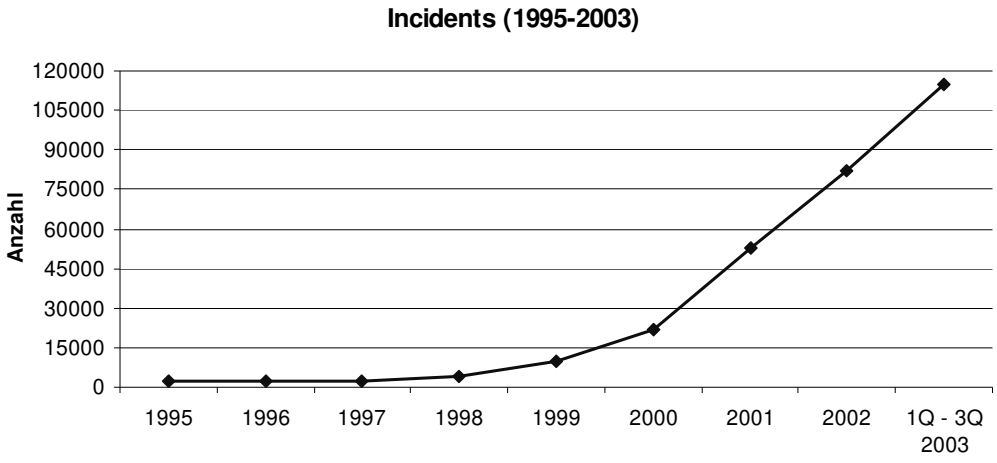

Diagramm 1: registrierte Vorfälle bei CERT [cert 03b]

[1] http://www.mi2g.com

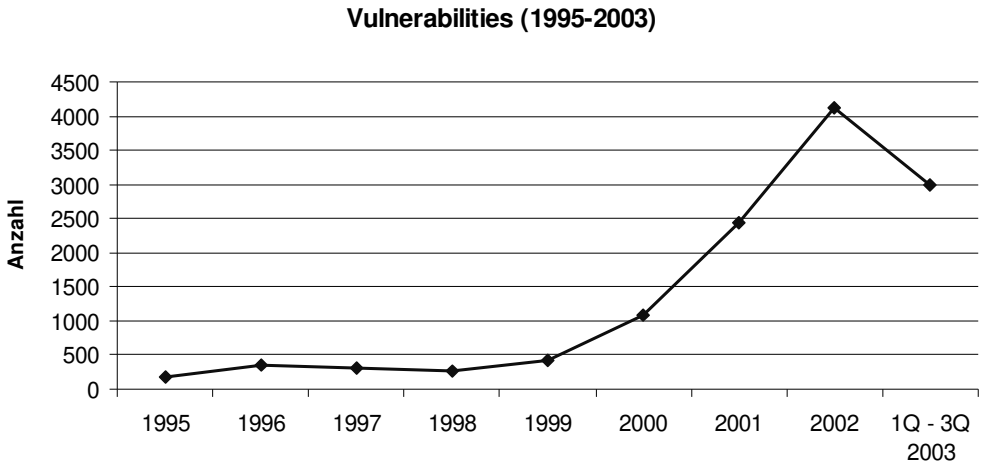

Diagramm 2: registrierte Schwachstellen bei CERT [cert 03b]

Diese beiden Diagramme zeigen, dass Computersysteme in den letzten Jahren in den Programmen immer öfter Schwachstellen gefunden wurden und somit die Möglichkeiten von Angriffen deutlich gestiegen und ausgenutzt wurden. Natürlich ist es nun interessant, wie Rechner angegriffen werden und ob es einen sicheren Schutz vor Angriffen geben kann.
Für die Erkennung von Angriffen können so genannte Honeynets hilfreich sein. Dies sind mehrere Rechner, bei denen bewusst auf Angriffe gewartet und gehofft wird, um z.B. daraus Folgerungen über das Angreiferverhalten ziehen zu können.

1.1 Aufgabenstellung

Die Diplomarbeit gliedert sich im Wesentlichen in drei Teile:

- **Konzeption eines Honeynet**: Hier gilt es die Anforderungen an ein Honeynet umzusetzen, die Topologie festzulegen und die Ziele zu definieren. Darüber hinaus müssen Software- und Hardwareanforderungen durchdacht werden.

- **Aufbau und Betrieb des Honeynet**: Das konzipierte Honeynet wird realisiert und in Betrieb genommen. Dazu müssen die Rechner geeignet präpariert werden.

- **Analyse des Honeynet**: Nach dem Betrieb müssen die gesammelten Daten analysiert ausgewertet werden. Dazu werden geeignete Werkzeuge vorgestellt. Dabei gilt herauszufinden, welche Rückschlüsse auf den Angreifer gezogen werden können.

1.2 Aufbau der Diplomarbeit

Die Gliederung der Diplomarbeit orientiert sich sehr an der Aufgabenstellung:

- Kapitel 2 beschäftigt sich mit den Grundlagen für den Betrieb eines Honeynets. Dazu gehören die Definitionen und Einsatzszenarien von Honeypots und Honeynets.

- Kapitel 3 beschreibt die Konzeption des Honeynets. Neben der Topologie des Honeynet stellt dieses Kaptitel verschiedene bereits vorhandene Software-Lösungen dar, um ein Honeynet zu betreiben und abzusichern.

- Kapitel 4 werden die Aufbau- und Betriebsphase des Honeynet aufgezeigt. Dazu zählen die Installation und Konfiguration der einzelnen Rechner als auch das Sichern der erstellten Konfigurationen.

- Kapitel 5 beschäftigt sich mit der Analyse und Auswertung der gewonnenen Daten. Neben der Klassifikation der einzelnen Angriffe erfolgen eine detaillierte Beschreibung der verwendeten Werkzeuge und die daraus resultierenden Ergebnisse. Zum Abschluss dieses Kapitels werden einzelne Statistiken präsentiert.

- Kapitel 6 enthält die Zusammenfassung der Arbeit und schließt mit einem Ausblick auf weitere Möglichkeiten für Einsatz von Honeynets ab.

Im Anhang befinden sich verschiedene Konfigurationsskripte, Perl-Skripte sowie Ergänzungen zu Internet Würmern.

1.3 Typographische Konventionen

Für die vorliegende Arbeit werden folgende typographischen Konventionen verwendet:

- Begriffe, die ausschlaggebend für die Diplomarbeit sind, werden **fett** geschrieben.

- Befehle, Variablen und Verzeichnisse werden mit der Schriftart `Courier New` dargestellt.

- Produkte und Markennamen werden mit *kursiver Schrift* gekennzeichnet. Software-Produkte werden, falls keine vorgegebene Schreibweise bekannt ist, für Linux klein geschrieben, bei Software-Produkten für Windows™ wird der erste Buchstabe groß geschrieben.

- Firmen-, Marken- und Produktnamen, Warenzeichen und eingetragene Warenzeichen werden anerkannt und gehören ihren rechtmäßigen Eigentümern.

- Bei Abbildungen, die auf keine Quellenangabe verweisen, handelt es sich um eigene Darstellungen.

2 Grundlagen

Im Folgenden werden zwei unterschiedliche Architekturen vorgestellt, die helfen können, potentielle Angriffe schneller zu erkennen und das Verhalten von Angreifern besser zu erforschen.

2.1 Honeypot

Das Honeynet Project definiert einen Honeypot wie folgt:

„A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource." [Spi 03a]

Somit ist ein Honeypot, nach dieser Definition des Honeynet Project, „eine Ressource eines Informationssystems, deren Wert darin besteht, dass ein unberechtigter oder verbotener Zugriff auf diese Ressource erfolgt".

Bei einem Honeypot handelt es sich also um irgendein IT-System, das öffentlich im Internet platziert wird um auf Angriffe zu warten. Dabei soll der Angreifer natürlich nicht merken, dass dieses System eine Falle („Honigtopf") ist. Sämtliche Aktivitäten (insbesondere der Netzverkehr), die auf diesem Computer stattfinden, werden mitprotokolliert und können anschließend analysiert werden.
Das Honeynet Project wurde zunächst im April 1999 als Mailingliste gegründet. Die offizielle Gründung erfolgte im Juni 2000. Es handelt sich hierbei um eine Non-Profit Organization, das heißt alle Mitglieder arbeiten auf freiwilliger Basis an diesem „Project" mit und alle Ergebnisse werden im Internet veröffentlicht. Zu den Zielen des Honeynet Project gehört das Ausfindigmachen und Aufdecken der Werkzeuge sowie das Erforschen der Taktiken und Motive der Blackhat-Community. Bei der Blackhat-Community handelt es sich um eine „Vereinigung", die aus unterschiedlichen Gründen versucht, IT-Systemen auf der Welt Schaden zuzufügen.

Es stellt sich natürlich die Frage wie ein Honeypot zur Sicherheit eines Systems beitragen kann. Der Themenbereich IT-Sicherheit lässt sich grob in drei Teilbereiche unterteilen: Prävention, Erkennung und Reaktion. Honeypots helfen hierbei nur indirekt IT-Systeme sicherer zu machen. Hauptsächlich helfen Honeypots bei der Erkennung von Angriffen und somit bei dem Aufdecken und Beseitigen von Schwachstellen, da der gesamte Netzverkehr bei diesen Rechnern überwacht und aufgezeichnet wird. Ebenso ist Prävention bzw. eine Verzögerung eines Angriffes auf die Produktivsysteme eines Unternehmens eine Aufgabe für Honeypots. Dies ist zum Beispiel dann möglich, wenn zuerst der Honeypot angegriffen wird. Die Verantwortlichen des Unternehmens können dadurch Gegenmaßnahmen ergreifen.

Die Vorteile von einem Honeypot lassen sich wie folgt zusammenfassen:

- **Kein Produktivsystem**: Da auf dem System nicht produktiv gearbeitet wird, ist jeglicher Datenverkehr verdächtig und somit als (versuchter) Angriff zu werten. Es entstehen keine Unmengen an gesammelten Daten, die erst gefiltert werden müssen.

- **Neue Werkzeuge und Taktiken**: Durch die Aufzeichnung aller Daten können neue Taktiken (z.B. die Vorgehensweise von Angreifern oder die Art des Angriffes) untersucht werden. Dies ist als Vorteil gegenüber einem IDS, das nur bereits bekannte Angriffe erkennt.

- **Minimale Hardware-Anforderungen**: Für einen Honeypot kann durchaus ein alter Pentium-Rechner verwendet werden. Die einzige Aufgabe besteht darin, Daten aufzuzeichnen.

Neben den gerade genannten Vorteilen bringen Honeypots natürlich auch Nachteile mit sich:

- **Beschränkter Einblick**: Es können lediglich Angriffe verfolgt und analysiert werden, die direkt auf das System verübt wurden.

- **Risiko**: Kein System ist 100-prozentig sicher. Es wird immer irgendwo Schwachstellen geben, z.B. in Firewalls. Durch eine „feindliche Übernahme“ des Systems durch den Angreifer besteht die Hauptgefahr darin, dass andere Systeme von diesem Honeypot aus angegriffen werden.

2.1.1 Klassifikation von Honeypots

Honeypots lassen sich nach unterschiedlichen Kriterien klassifizieren. Zum einen gibt es die Unterscheidung zwischen **Production-Honeypots** und **Research-Honeypots**. Production-Honeypots sind Honeypots die meistens in bzw. vor Unternehmens-Netzwerken installiert werden, um das Verhaltensmuster der Angreifer zu erforschen. Sie sollen mögliche Angreifer vom eigentlichen Netzwerk ablenken und somit der Prävention dienen. Leider kann es natürlich hier passieren, dass Angreifer somit erst auf das Unternehmen aufmerksam werden und eventuell ein größerer Schaden entstehen könnte.
Bei den Research-Honeypots (Forschungshoneypots) besteht das Hauptziel im Sammeln von Daten um das Angreiferverhalten zu studieren. Wie oben bereits erwähnt, ist hierbei der gesamte Datenaustausch zwischen dem Internet und dem Honeypot interessant.
Eine weitere Klassifizierungsmöglichkeit besteht darin, Honeypots nach dem Grad des Interaktionslevels zu charakterisieren, das heißt man unterscheidet wie viel „Spielraum“ einem Angreifer zur Verfügung gestellt wird.
Es wird unterschieden zwischen Honeypots mit niedriger und hoher Interaktionsebene. Bei Systemen mit niedriger Interaktionsebene werden „angreifbare“ Dienste durch spezielle Software auf den Rechnern lediglich simuliert. Hierbei gibt es in der Zwischenzeit eine große Auswahl an Diensten (z.B. *IIS* Web-Server von Microsoft, verschiedene DNS Server), die simuliert werden können. Alle Interaktionen des Angreifers mit dem System werden

mitprotokolliert. Der Angreifer hat keine Möglichkeit auf die Betriebssystem-Ebene des Rechners zu gelangen und diesen für seine eigenen Zwecke zu missbrauchen.
Honeypots mit hoher Interaktionsebene sind im Wesentlichen reale Systeme, die „nicht ganz auf dem neuesten Stand sind“, das heißt es sind nicht die neuesten Updates und Servicepacks installiert. Hier kann der Angreifer durchaus den Rechner kompromittieren und sich somit vollen Zugriff auf das System verschaffen. Auch hier werden alle Aktivitäten, die der Angreifer auf dem Rechner ausführt bzw. ausführen will mitprotokolliert z.B. durch versteckte Kernelmodule unter Linux.
In Tabelle 2 sind die einzelnen Merkmale der unterschiedlichen Honeypot-Ausprägungen zusammengefasst.

Im Prinzip entspricht ein Honeypot einem erweiterten IDS: Durch die Überwachung des Netzverkehrs sowie der Aktivitäten auf dem Honeypot selber lassen sich z.B. neue Angriffsmuster oder Angriffsmöglichkeiten erkennen. Ein IDS erkennt nur Angriffe anhand der verwendeten Signaturen. Bei einem Honeypot ist grundsätzlich jeder Verkehr interessant, da dieses System nicht produktiv benutzt wird. Es ist dabei egal, mit welchen Daten (z.B. IPv6-Netzverkehr) Angriffe auf einen Honeypot ausgeübt werden, alle Daten werden mitprotokolliert.

Honeypot mit niedriger Interaktionsebene	• Programm simuliert Dienste (z.B. http, ftp) • Keine Interaktionsmöglichkeit mit dem Betriebssystem • Angreifer bekommt keine bzw. nur gefälschte Antworten auf seine Anfragen • Geringes Risiko • Beispiel: BackOfficer Friendly [bof 03]
Honeypot mit hoher Interaktionsebene	• Reales System • Angreifer kann das System kompromittieren und kann sich vollen Zugang zum Betriebssystem verschaffen • Honeypot kann als Ausgangspunkt für weitere Angriffe dienen • Hohes Risiko, regelmäßige Überwachung besonders notwendig
Research Honeypot	Ziel: Sammeln von Informationen z.B. über Verwundbarkeiten, Angreiferverhalten (Werkzeuge, Taktiken, Motive)
Production Honeypot	Ziel: Erhöhung der Sicherheit und Unterstützung des Schutzes für ein Netz

Tabelle 2: Klassifizierung von Honeypots

2.1.2 Positionierung eines Honeypots

Für die Positionierung eines Honeypots gibt es verschiedene Möglichkeiten. Je nach Anforderung bzw. Aufgabe unterscheidet sich der Aufbau. Abbildung 1 zeigt verschiedene Szenarien für den Einsatz eines Honeypots (nach [Kro 03]).

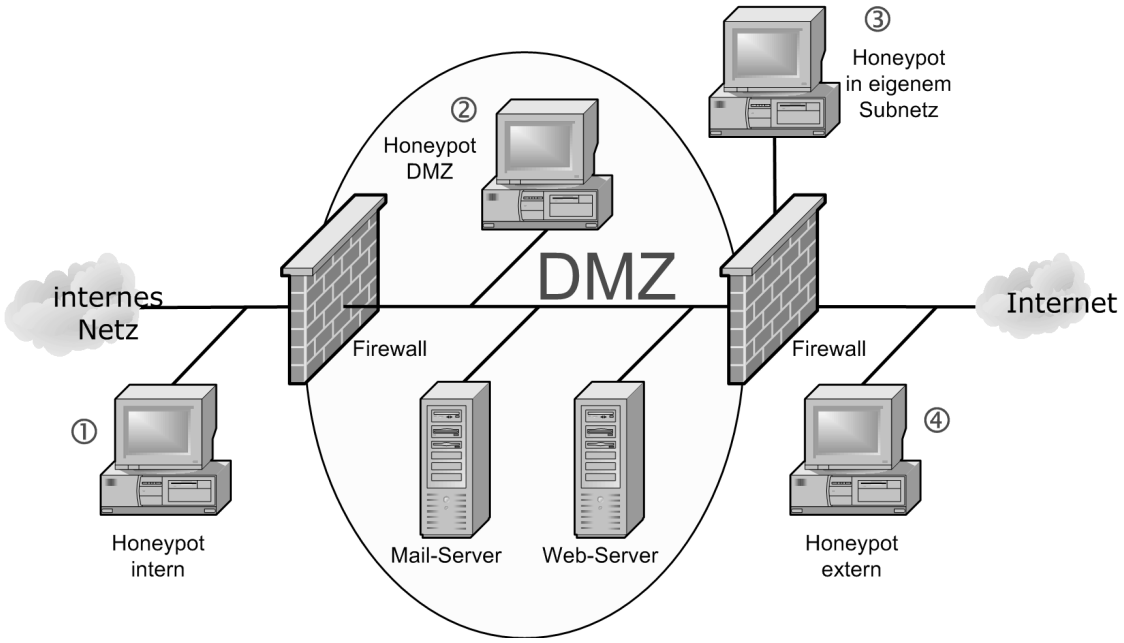

Abbildung 1: Unterschiedliche Einsatzszenarien eines Honeypots (vgl. [Spi 02])

Im internen Netz ①

Als erste Möglichkeit kann ein Honeypot im internen Netz eines Unternehmens installiert werden. Dieses Szenario kann zur Erkennung von Angriffen im internen Netz (durch autorisierte Nutzer) als auch aus dem externen Netz dienen. Wartet man auf Angreifer aus dem Internet, so ist diese Positionierung die Gefährlichste: Der oder die Angreifer erhalten somit Zugriff auf das interne Netz wenn der Honeypot kompromittiert wurde; die Kompromittierung ist ein Beweis für Schwachstellen in der Firewall.
Dieses Szenario erfordert eine ständige und genaue Überwachung der Vorgänge auf dem Honeypot.

In einer Demilitarisierten Zone (DMZ) ②

Rechner, die sich in der DMZ befinden bieten meistens Dienste an, die aus dem externen Netz genutzt werden können, z.B. Web-Server oder Mail-Server. Diese Rechner sind meistens durch eine Firewall vom Internet getrennt. Dabei eignet sich diese Positionierung für das Testen der in der DMZ befindlichen Dienste sowie die auf der Firewall installierten Regeln. Außerdem lassen sich Scans über den Adressbereich der DMZ feststellen. Der Honeypot sollte kein Produktivsystem sein, da sonst legitime Nutzer die Ergebnisse verfälschen können (durch Aufbau von Verbindungen aus dem Internet). Verschafft sich ein Angreifer Zugriff auf dem installierten Honeypot, so sollten die Regeln auf der Firewall und die Rechner in der DMZ überprüft werden. Der oder die Angreifer können nach einer Kompromittierung auch versuchen, Zugriff auf das interne Netz zu erlangen.

In einem eigenen Subnetz ③
Diese Platzierung ist hauptsächlich für einen Research Honeypot interessant. Nach einem erfolgten Angriff kann der Honeypot vom Netz getrennt und analysiert werden, ohne dass der Produktivbetrieb beeinträchtigt wird.

Vor einer Firewall ④
Der Honeypot steht aus Sicht des Betreibers im Internet und somit außerhalb seines Netzes. Mit dieser Positionierung können die Aktivitäten im Internet untersucht werden sowie das Verkehrsaufkommen außerhalb der Firewall. Es besteht nahezu keine Gefahr für das Netz des Betreibers, jedoch ist er für den installierten Honeypot verantwortlich und kann z.B. bei Angriffen, die von diesem Honeypot aus gestartet werden eventuell zur Rechenschaft gezogen werden.

Es wäre auch denkbar, die gerade vorgestellten Ansätze zu kombinieren, das heißt man installiert mehrere Honeypots an verschiedenen Stellen und kann so beobachten, wo Schwachstellen in den (eigenen) Systemen vorhanden sind.

2.2 Honeynet

Bei einem Honeynet handelt es sich nicht um ein fertiges Konzept oder um ein fertiges System, ein Honeynet ist eine Architektur. Im Gegensatz zu einem Honeypot besteht ein Honeynet nicht nur aus einem einzigen (angreifbaren) Rechner sondern aus einem Netz, das aufgebaut ist aus verschiedenen Systemen (Forschungshoneypots mit hoher Interaktionsebene) mit unterschiedlichen Betriebssystemen und Diensten. Das Honeynet soll nach außen wie eine Produktivumgebung wirken: Es soll dem Angreifer nicht den Eindruck vermitteln, dass es sich hier um eine Falle handelt bzw. dass seine Aktivitäten überwacht und aufgezeichnet werden.

2.2.1 Anforderungen

Es gibt im Wesentlichen drei Anforderungen an Honeynets (vgl. [HP 03]):

- **Data Control**
- **Data Capture**
- **Data Collection**

Die Aufgabe von **Data Control** ist die genaue Überwachung des ein- und ausgehenden Netzwerkverkehrs nachdem z.B. ein Honeypot kompromittiert wurde, um zu vermeiden, dass von diesem Honeypot eine Gefahr für andere Systeme im Internet ausgeht. Dabei soll der Angreifer natürlich nicht merken, dass z.B. Kontrollmechanismen zur Beschränkung des Netzverkehrs eingesetzt werden. Wichtig sind dabei folgende Anforderungen:

- Einsatz von mindestens zwei Instanzen von Data Control, um sich gegen Fehler zu schützen.

- Fehler in der Data Control-Ebene sollten das System nicht in einen offenen Zustand bringen.

- Die Konfiguration von Data Control muss jederzeit vom Administrator verändert werden können.

- Angreifer sollen es so nahezu unmöglich sein, Kontrollverbindungen zu erkennen.

- Automatische Benachrichtigung, falls das System kompromittiert wurde.

Bei **Data Capture** besteht die Aufgabe darin, dass sämtliche Aktivitäten, die im Honeynet auftreten mitprotokolliert werden, ohne dass auch hier der Angreifer davon etwas merken darf. Auch bei der Data Capture gibt es besondere Anforderungen:

- Die gesammelten Daten werden nicht lokal auf einem Honeypot gespeichert.

- Folgende Daten müssen gespeichert werden: Ein- und ausgehende Verbindungen, Netzverkehr und Aktivitäten auf dem System.

- Die Daten müssen in Echtzeit für einen Zugriff auf das System aus der Ferne zur Verfügung stehen.

- Die verwendeten Ressourcen zur Data Capture müssen gegen Kompromittierung geschützt werden. Die Integrität der Daten muss gewahrt werden.

Wird ein Honeynet in einer verteilten Umgebung installiert, so müssen z.B. die gesammelten Daten auf sicherem Wege zu einem zentralen System übermittelt werden. Dies ist Bestandteil der **Data Collection**. Somit wird die Auswertung deutlich vereinfacht. Folgende Anforderungen sind dabei notwendig:

- Die einzelnen Honeynets müssen untereinander synchron gehalten werden. Dazu eignet sich z.B. eine Zeitsynchronisierung mit NTP (Network Time Protocol).

- Die gewonnenen Daten müssen auf einem sicheren Weg zu einem zentralen Rechner übermittelt werden, um damit sicherzustellen, dass die Daten während der Übertragung nicht verändert wurden.

2.2.2 Motivation

Aber was will man nun mit einem Honeynet erreichen? Jeder möchte seine Ressourcen vor unbefugtem Eindringen oder Missbrauch schützen und installiert Systeme (z.B. Firewall oder Intrusion Detection Systeme (IDS)) um dies zu verhindern. Leider gibt es immer wieder Fehler, zum Teil auch Konfigurationsfehler, die Angreifern erlauben, die Schutzmechanismen zu umgehen oder diese auszuschalten [iss 03]. Mit Hilfe eines Honeynets lässt sich z.B. die Produktivumgebung eines Unternehmens in kleinerer Form nachbilden. Nun kann auf dem

Honeynet beobachtet werden, wie Angreifer sich Zugriff zu den Systemen verschaffen und somit ist eine Analyse der Fehler möglich.
Oft wird ein Honeynet mit einem Aquarium verglichen: Man kann bei einem Honeynet jederzeit beobachten, was im Inneren vor sich geht. In dieser Abstraktion stellen die „Fische“ die einzelnen Systeme des Honeynets dar.

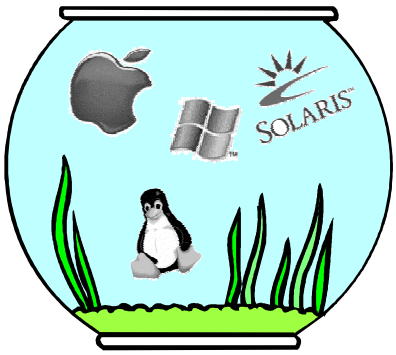

Abbildung 2: Honeynet als „Aquarium“

Durch die gewonnen Daten lassen sich Taktiken, Motive und neue Werkzeuge erforschen.

3 Konzeption des Honeynets

Der erste große Teil der Diplomarbeit beschäftigt sich mit der Konzeption des Honeynets. Dazu zählen z.B. Überlegungen der Topologie. Ebenso müssen Möglichkeiten der Überwachung für das Honeynet bestimmt und beurteilt werden.

3.1 Entwicklung von Honeynets

Im Zeitraum von 1999 bis 2001 wurde zunächst vom Honeynet Project eine erste Architektur für ein Honeynet entwickelt, das so genannte **Generation I Honeynet** („**Gen I Honeynet**“). Es handelt sich hierbei um eine möglichst einfache Implementierung eines Honeynets. Abbildung 3 zeigt den Aufbau eines typischen Gen I Honeynet. Der umrandete Bereich in der linken Hälfte des Bildes stellt das eigentliche Honeynet dar: Als Honeypots dienen ein Windows 2000 System und ein Linux-System. Der stärker gesicherte *log server* speichert alle Logfile-Einträge der Honeypots. Die Firewall überwacht und kontrolliert den gesamten Verkehr, der zwischen dem Internet und den Honeypots auftritt. Das Intrusion Detection System (IDS) überwacht ebenfalls den Netzwerkverkehr und erkennt Angriffe. Der Router hat die Aufgabe, den Verkehr z.B. auf gefälschte IP-Adressen oder DoS-Attacken zu überprüfen und zu filtern.

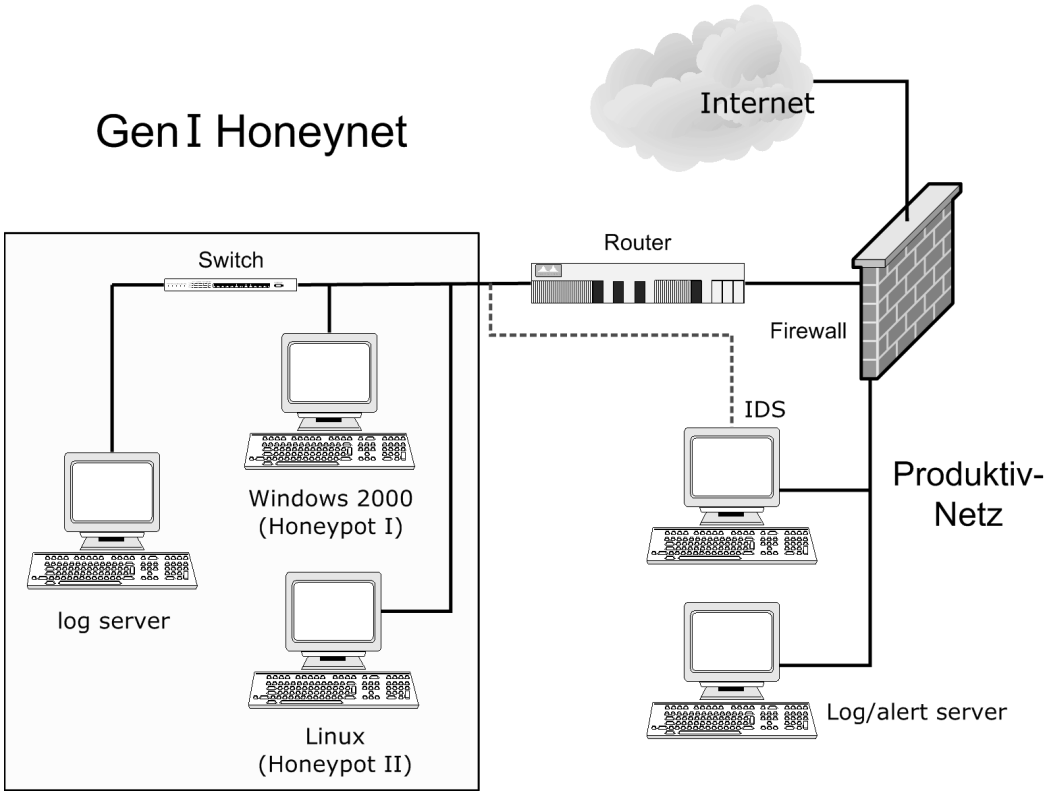

Abbildung 3: Aufbau eines typischen Gen I Honeynet

Seit dem Jahr 2002 wird an einer Weiterentwicklung des Gen I Honeynet gearbeitet, das so genannte **Gen II Honeynet**. Dabei werden die bereits gewonnenen Erkenntnisse aus dem

Gen I Honeynet verwendet und verbessert. Im Folgenden bezeichnet der Begriff Honeynet immer ein Gen II Honeynet. Abbildung 4 zeigt exemplarisch den Aufbau eines Gen II Honeynet.

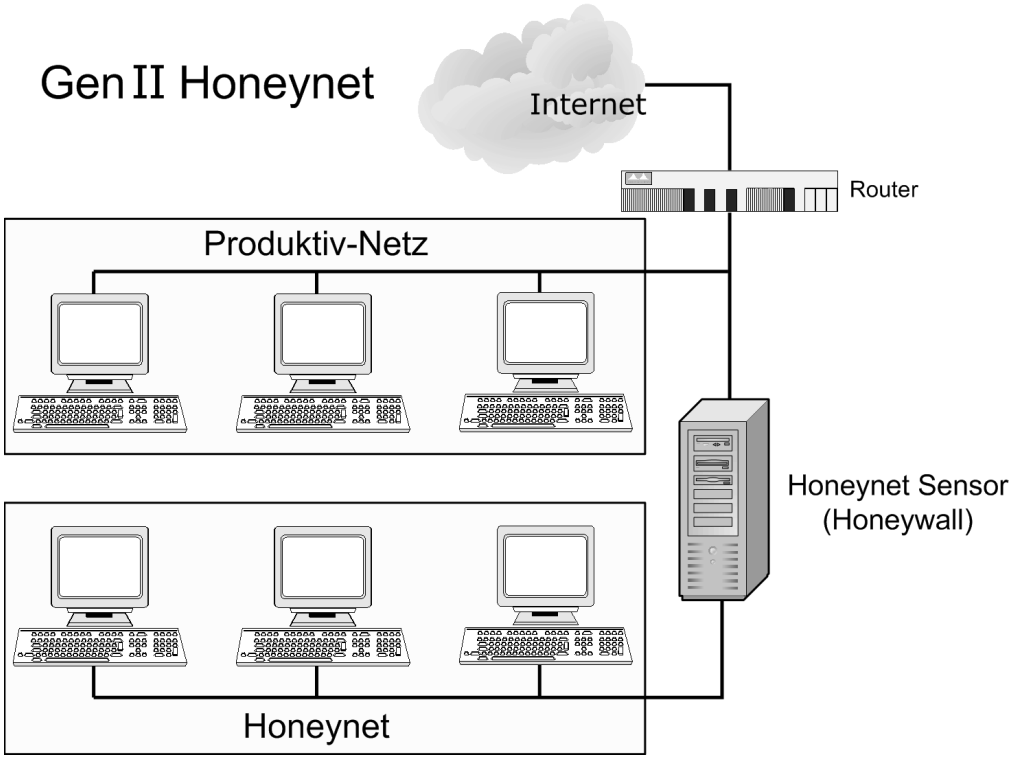

Abbildung 4: schematischer Aufbau eines Gen II Honeynets

Die Umgebung besteht aus zwei Teilnetzen: Dem Produktiv-Netz (oben im Bild) und dem Honeynet (unten im Bild). Das Trennelement zwischen den beiden Netzen ist der Honeynet Sensor (auch **Honeywall** oder Gateway genannt). Die Grundidee beim Gen II Honeynet ist, dass der gesamte Verkehr, der von und zu den Honeynet-Rechnern auftritt, über den Honeynet Sensor fließen muss. Somit vereint dieser Rechner die beiden Anforderungen Data Control und Data Capture (vgl. Abschnitt 2.2.1).

Virtuelle Honeynets

Die beiden vorgestellten Architekturen benötigen für jeden eingesetzten Honeypot sowie für das Gateway bzw. das IDS je einen eigenen physischen Rechner. Bei größeren Honeynets ist dafür eine große Anzahl an Ressourcen notwendig.

Seit Anfang des Jahres 2003 gibt es auch die Möglichkeit so genannte virtuelle Honeynets zu betreiben. Dabei simuliert ein physischer Rechner (Hostsystem) mit Hilfe eines Virtualisierungsprogramms mehrere Honeypots. Das Hostsystem übernimmt dabei die Aufgabe des Gateways. **Honeyd**[2] ist eine aktuelle Implementierung für virtuelle Honeynets.

Der Vorteil bei virtuellen Honeynets liegt darin, dass der Ressourcenbedarf zum Aufbau deutlich verringert wird. Allerdings haben virtuelle Honeynets auch Nachteile: Da das komplette Honeynet auf einem einzigen Rechner implementiert sein kann, existiert das Problem des „Single Point of Failure“, das heißt fällt dieser eine Rechner z.B. durch einen Hardwaredefekt aus, so ist das komplette Honeynet außer Betrieb. Außerdem besteht die

[2] http://www.honeyd.org/

Gefahr, dass bei einer Kompromittierung der Virtualisierungssoftware die gesammelten Daten vom Angreifer zerstört werden. Hinzu kommt, dass für den Einsatz des Host-Rechners ein leistungsstärkeres System von Nöten ist. Physische Honeynets können oft mit älteren Rechnern betrieben werden, die z.B. bereits ausgemustert sind.
Natürlich wäre auch denkbar ein Honeynet zu betreiben, das eine Mischform zwischen physischen und virtuellen Honeynet darstellt. Beispielsweise könnte ein Honeynet aus zwei physischen Rechnern aufgebaut werden: Ein Rechner dient als Gateway, auf dem anderen werden die einzelnen Honeypots virtualisiert. Das Problem des „Single Point of Failure“ wird dadurch vermieden.
Es wurde entschieden, dass in dieser Diplomarbeit ein physisches Honeynet konzipiert werden soll. Dafür waren folgende Gründe entscheidend:

- Software-Lösungen für Virtuelle Honeynets waren erst in der Entwicklung. Eventuelle Abstürze oder Fehler in der Virtualisierungs-Software hätten sich negativ auf den Betrieb ausgewirkt.

- Das Honeynet soll beim Angreifer den Eindruck erwecken, dass es sich um reale Systeme handelt.

- Es standen genügend System-Ressourcen für ein physisches Honeynet zur Verfügung.

3.2 Design der Architektur

Die Aufgabe der Diplomarbeit ist es, ein Honeynet zu konzipieren, zu betreiben und zu analysieren. Es soll also eine repräsentative „Spielwiese“ realisiert werden. Dazu werden ein Honeypot mit einem UNIX-Betriebssystem und ein Honeypot mit einem Microsoft Windows™-Betriebssystem verwendet.

3.2.1 Topologie

Das zentrale Element des Honeynet ist ein Gateway-Rechner, das sich zwischen den Honeypots und dem Internet befindet. Dadurch wird eine Möglichkeit geschaffen, sämtlichen Netzverkehr, der zwischen dem Internet und den verschiedenen Honeypotrechnern auftritt zu beobachten und mitzuprotokollieren, eine wichtige Voraussetzung um das Angreifer-Verhalten zu studieren.

Als Betriebssystem für den Gateway-Rechner wird das Betriebssystem Linux verwendet. Folgende Gründe sind dafür entscheidend:

- **Kosten/Verfügbarkeit**: Alle benötigten Komponenten sind Open-Source Software und somit frei verfügbar. Es fallen keine zusätzlichen Kosten für Software-Komponenten an.

- **Dokumentation**: Für die einzelnen Komponenten gibt es eine ausführliche Beschreibung (z.B. HowTo).

Der Gateway (Abbildung 5) verfügt über mehrere Netzwerkkarten: Ein externes (im Folgenden als *eth0* bezeichnet) und ein internes Netzwerk-Interface (im Folgenden als *eth1* bezeichnet) an dem die Honeypots über einen Hub angeschlossen sind. Da ein Hub keine Trennung von Kollisionsdomänen vornimmt, können auch Angriffe von einem Honeypot zu einem anderen Honeypot erkannt und aufgezeichnet werden.

Für das Gateway gibt es zwei Betriebsarten:

- **Mit Network Address Translation (NAT)**: Das Gateway übernimmt die Umsetzung der IP-Adressen, die Honeypot-Rechner erhalten somit private IP-Adressen. Steht z.B. nur eine externe IP-Adresse zur Verfügung, so kann diese Betriebsart gewählt werden. Der Nachteil an dieser Konfiguration ist, dass der Gateway als Router fungiert (der Gateway hat eine eigene IP-Adresse) und damit für den Angreifer „sichtbar“ ist (z.B. wird der „Time-to-live“-Wert dekrementiert).

- **Als Bridge**: Wird der Gateway als Bridge betrieben, ist er transparent für den Angreifer, das heißt, der Angreifer hat nahezu keine Möglichkeit diesen Rechner zu entdecken: Der Rechner hat in diesem Betriebsmodus keine IP-Adresse. Prinzipiell gibt es zwei Möglichkeiten für einen Angreifer eine Bridge zu erkennen: Über das Spanning Tree Protocol, das die Bridge z.B. zur Erkennung von Schleifen im Netz verwendet, kann auf das Vorhandensein einer Bridge geschlossen werden. Es besteht jedoch die Möglichkeit das Spanning Tree Protocol zu deaktivieren. Die zweite Möglichkeit besteht darin, dass der Angreifer den Netzverkehr auf MAC-Ebene (Schicht 2a) vor dem Gateway-Rechner (Internet) und auf dem Honeypot mitprotokolliert. In diesem Fall würde er z.B. bei der Überprüfung seiner IP-Adresse feststellen, dass sich die MAC-Adresse zwischen seinen beiden Protokollierungsspunkten verändert. Da der Angreifer bei diesem Szenario sich schon erfolgreich Zugriff auf den Honeypot verschafft hat, kann dieser Fall vernachlässigt werden.
  Da das Spanning Tree Protocol deaktiviert werden kann und beim zweiten „Nachteil“ die Kompromittierung bereits erfolgt ist wird der Gateway-Rechner in dieser Arbeit als Bridge betrieben.

Um einen Management-Zugriff auf den Gateway zu ermöglichen, wird eine zusätzliche Netzwerkkarte (*eth2*) mit einer IP-Adresse aus einem anderen Subnetz konfiguriert (*eth0* und *eth1* haben im Bridge-Modus keine IP-Adresse). Zweck des Management-Interfaces ist es, während des Betriebes das Honeynet überwachen zu können und bei Bedarf Änderungen an der Konfiguration (z.B. Konfiguration der Firewall) vorzunehmen oder den aktuellen Status des Honeynet abzufragen (Online-Analyse, siehe Kapitel 5.1.1). Der Gateway-Rechner ist somit nur noch über das Management-Interface angreifbar. Die Sicherung dieses Interfaces wird in Kapitel 3.5 beschrieben.

Um dem Angreifer den Eindruck zu vermitteln, dass es sich hier um ein „reales“ Netz handelt, werden mehrere Honeypots mit unterschiedlichen Betriebssystemen und konfigurierten Diensten installiert.

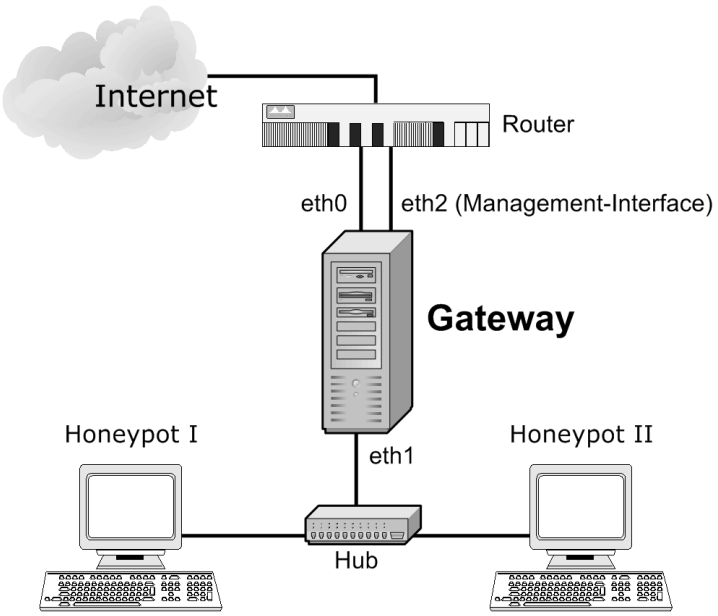

Abbildung 5: Honeynet mit Gateway und zwei Honeypots

Wird ein Rechner erfolgreich kompromittiert, so wird vom Angreifer oft als erstes versucht die Protokollierungsmechansimen des Betriebssystems (bei UNIX-Systemen: *syslog*, bei Windows™ Systemen *eventlog*) außer Kraft zu setzen und die bereits vorhandenen Daten zu löschen. Aus diesem Grund könnte zusätzlich am Hub, an dem die Honeypots angeschlossen sind, noch ein Log-Server installiert werden, der die Meldungen zusätzlich speichert. Allerdings besteht nun die Gefahr, dass auch dieser Rechner zu einem ungewollten Honeypot wird. Um dies zu verhindern wird dieser Rechner anstatt über den Hub über eine zusätzliche Netzwerkkarte (*eth3*) direkt am Gateway angeschlossen (Abbildung 6).

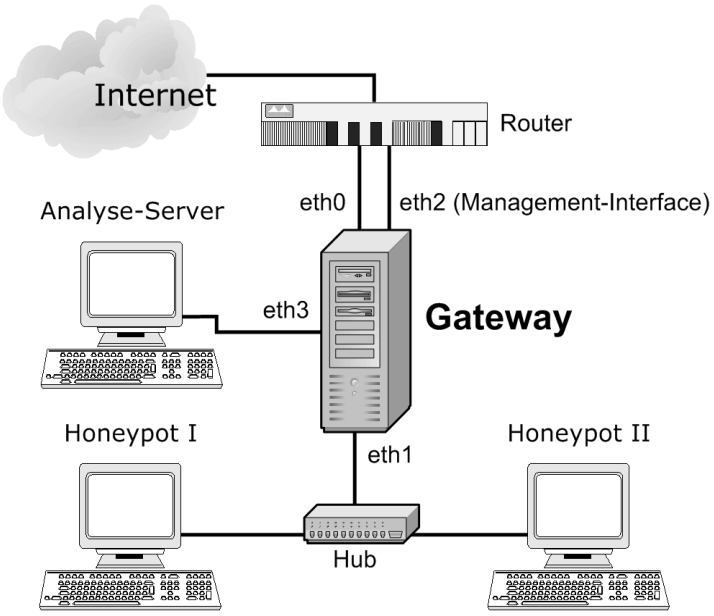

Abbildung 6: Honeynet mit zusätzlichem Analyse-Server

3.2.2 Sicherheitskonzepte (Data Control)

Beim Betreiben eines Honeynets besteht die große Gefahr, dass kompromittierte Rechner benutzt werden um von diesem aus andere Rechner im Internet anzugreifen. Der Angreifer möchte damit seine eigenen Spuren verwischen.
Um nun sicherzustellen, dass von den installierten Honeypots keine Gefahr für andere Rechner besteht, muss der ausgehende Netzverkehr überwacht werden. Eine geeignete Softwarelösung bietet hierbei eine vom Honeynet Project veränderte Version des frei verfügbaren Intrusion Detection System *snort*[3]. Vom Honeynet Project wird dabei eine spezielle Version zur Verfügung gestellt: *snort_inline*. Diese Version ermöglicht das Zusammenspiel zwischen der Firewall (*iptables*) und *snort*. Die Funktionsweise ist in Abbildung 7 dargestellt.
Alle ausgehenden IP-Pakete werden mit Hilfe des QUEUE-Targets von *iptables* an das Programm *snort_inline* vom Kernel Space in den User space weitergeleitet. *snort_inline* überprüft den Inhalt des Pakets mit den gespeicherten Signaturen (*rules*) und verwirft dieses bei einem Matching (*DROP*): Das IP-Paket erreicht den eigentlichen Empfänger somit nie. Ansonsten wird das Paket ins Internet (in diesem Fall über *eth0*) weitergeleitet.
Alle verworfenen Pakete werden in einer Log-Datei gespeichert.

[3] http://www.snort.org

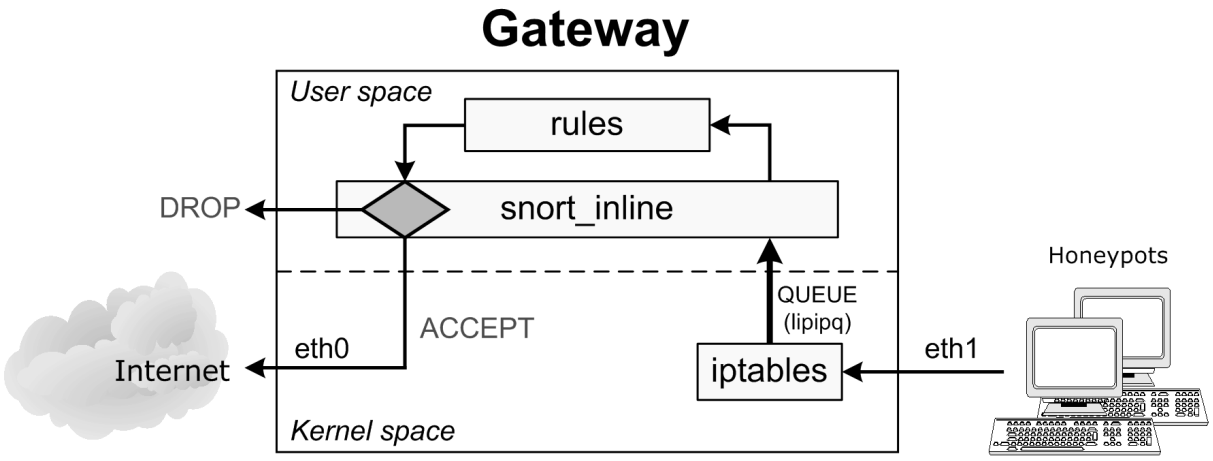

Abbildung 7: Arbeitsweise von *snort_inline*

Die Signaturen müssen natürlich während des Betriebes laufend aktualisiert werden, um auch zu gewährleisten, dass auch aktuelle Angriffe, für die bereits Signaturen zur Verfügung stehen, erkannt werden.

Ein Angreifer wird versuchen, sobald er sich Zugang zu einem Honeypot-System verschafft hat, Kontakt mit anderen Rechnern aufzunehmen. Um die Aktionsmöglichkeiten des Angreifers zu beschränken gibt es mehrere Möglichkeiten:

- Es ist kein ausgehender Verkehr vom Honeynet erlaubt. Mit dieser Restriktion wird sich der Angreifer schnell wieder zurückziehen, wenn er merkt, dass keine seiner Anfragen bzw. Angriffe beim Ziel-System ankommt.

- Jeder ausgehende Verkehr ist erlaubt. Dieser Ansatz ist etwas zu optimistisch und zu gefährlich, da dabei der Angreifer uneingeschränkte Möglichkeiten hat weitere Rechner anzugreifen. Es ist nicht das Ziel eines Honeynet, auch andere Rechner zu gefährden.

- Limitierung des ausgehenden Verkehrs. Hier wird die Anzahl der ausgehenden Verbindungen, unterschieden nach verschiedenen Protokollen, auf ein frei wählbares Limit gesetzt. Ist dieses Limit erreicht, so werden alle weiteren ausgehenden Verbindungen geblockt. Eingehende Verbindungen sind jedoch jederzeit möglich. Der Angreifer merkt nicht sofort, dass nur eine gewisse Anzahl an ausgehenden Verbindungen erlaubt ist.

Am besten eignet sich natürlich die letztgenannte Möglichkeit. Dazu wird auf dem Gateway-Rechner eine Firewall (*iptables*) installiert. Diese setzt auf die Bridge-Funktionalität auf. Das Besondere daran ist, dass die Firewall den Verkehr auf TCP/IP-Ebene (also OSI Schichten 3 und 4) filtert, während das Gateway selber auf OSI Schicht 2 arbeitet. Für diese Kombination aus Bridge und Firewall (auch **Bridge-Firewalling** genannt) muss allerdings ein modifizierter Kernel verwendet werden, die Installation wird in Kapitel 4.3.1 beschrieben.

3.2.3 Überwachungskonzepte (Data Capture)

Die Überwachung eines Honeynets gliedert sich in mehrere Ebenen. Diese unterschiedlichen Ebenen werden im Folgenden vorgestellt und erläutert.

3.2.3.1 Überwachung auf Netzwerkebene

Da der Angreifer keinen physischen Zugang zu den Honeypots hat, müssen alle möglichen Angriffe über das Netzwerk kommen.
Um diese Netzwerk-Daten aufzuzeichnen, gibt es so genannte (Netzwerk-)Sniffer. Im normalen Betriebszustand einer Netzwerkkarte nimmt diese nur Pakete entgegen, die für den Rechner bestimmt sind. Allerdings lassen sich Netzwerkkarten auch in den so genannten „promiscous mode“ setzen. In diesem Zustand werden alle Pakete von der Netzwerkkarte mitgelesen und können zur späteren Auswertung abgespeichert werden. Typische Programme sind hierfür *tcpdump* [tcp 03] oder *ngrep* [ngr 03].
Eine weitere Überwachungsmöglichkeit auf der Netzwerkebene (auf den ISO/OSI-Schichten 3 und 4) bietet ein Intrusion Detection System (IDS). Eine genaue Untersuchung von *snort* als IDS für das LRZ ist in [Fun 03] beschrieben. Für den Einsatz in Research-Honeynets eignet sich das IDS *snort*. Anhand von Signaturen wird der Netzverkehr überprüft und bei einer Übereinstimmung ein Alarm erzeugt. Diese Alarme werden in einer Datenbank (z.B. MySQL[4] oder Microsoft SQL Server[5]) gespeichert und können später ausgewertet werden. Wichtig ist dabei, dass die Signaturen regelmäßig erneuert werden, damit auch neue Angriffe erkannt werden. Durch den Einsatz eines IDS können (versuchte) Angriffe erkannt werden.
Auf dem Gateway-Rechner wird zur Speicherung und Auswertung der *snort*-Alarme eine MySQL-Datenbank installiert. Durch den Einsatz von *tcpdump* zur Protokollierung des Verkehrs lassen sich die gesammelten Daten mit dem Programm *ethereal*[6] und *Packetzyer*[7] (siehe Kapitel 5.2.4) leicht auswerten und analysieren.
Damit der gesamte Verkehr lückenlos ausgewertet werden kann, werden *snort* und *tcpdump* parallel eingesetzt.

3.2.3.2 Überwachung auf Kommando-Ebene (Shell-Ebene)

Leider reicht die alleinige Aufzeichnung des Netzwerkverkehrs für eine lückenlose Analyse nicht aus. Ein kleines Beispiel soll dies verdeutlichen:
Ein Angreifer hat sich Zugang zum System verschafft und kopiert sich per FTP (File Transfer Protocol) oder SSH (Secure Shell) zusätzliche Software auf das System. Der Vorgang des Kopierens lässt sich mittels Überwachung des Netzverkehrs nachvollziehen, die eigentliche Verwendung bzw. der Nutzen oder der Zweck der Software aber nicht. Im Gegensatz zu FTP, wo zumindest der Dateiname (durch die FTP Kommandos) im Klartext übertragen wird, erfolgt bei SSH die Übertragung der Daten komplett verschlüsselt, es ist nicht möglich zu erkennen, welche Daten übertragen wurden.

[4] http://www.mysql.com
[5] http://www.microsoft.com/germany/ms/sql/
[6] http://www.ethereal.com
[7] http://www.packetyzer.com

Der Angreifer wird versuchen, seine Spuren so gut wie möglich zu verwischen bzw. die Beweise für sein Eindringen aus den Log-Dateien zu entfernen. Es ist daher notwendig, die Daten (ausgeführte Befehle) an einem Ort zu speichern, auf den der Angreifer keinen Zugriff hat und sich auch keinen Zugriff verschaffen kann.
Um die Daten sicher aufzuzeichnen gibt es verschiedene Ansätze und Lösungen für unterschiedliche Betriebssysteme. Bei Windows™-Systemen lässt sich die Eingabeaufforderung durch ein vorgeschaltetes Perl-Programm (*ComLog* [com 03]) so verändern, dass alle eingegebenen Befehle in eine Datei geschrieben werden.
Bei UNIX-Systemen gibt es bereits mehrere Lösungen: Zum einen lässt sich der Kernel mit Hilfe eines Patches [lkp 03] verändern, so dass die read()-Methode des Betriebssystems alle Kommandos in die System Log-Datei (unter SuSE Linux `/var/log/messages`) schreibt. Eine andere Möglichkeit besteht darin, den Befehlsinterpreter so zu verändern, dass alle eingegebenen Kommandos zusätzlich in einer Datei oder in der System Log-Datei gespeichert werden. Wie oben bereits erwähnt, muss natürlich dafür gesorgt werden, dass der Angreifer diese nicht löscht bzw. dass sie zusätzlich auf einem anderen Medium oder Rechner gespeichert werden. Eine genaue Beschreibung der Sicherung der System Log-Datei sowie die Vermeidung der Erkennung der zusätzlichen Sicherung durch den Angreifer erfolgt in Kapitel 3.2.3.3. Selbst wenn der Angreifer die Protokollierung der von ihm eingegebenen Kommandos merkt, sind alle Informationen, die zum erfolgten Angriff führten, bereits gesichert.
Für den Betrieb des Honeynet wird ein Kernel-Patch verwendet: Dieser Patch ist leicht zu installieren und alle Benutzer-Eingaben werden sofort in die System Log-Datei geschrieben.
Vom Honeynet Project wird ein Programmpaket zur Verfügung gestellt, dass auch aus SSH-File-Transfers die übertragenen Dateien extrahieren kann: *Sebek*. Zum Zeitpunkt der Bearbeitung der Diplomarbeit stand *Sebek* nur für Linux, OpenBSD und Solaris zur Verfügung. Eine Version für Windows™-Betriebssysteme ist in Planung.
Das Programmpaket *Sebek* besteht aus drei Komponenten (Abbildung 8):

- Auf dem Honeypot läuft das Programm **Sebek** [seb 03a] als Kernel Modul. Als Grundlage für die Entwicklung diente dabei das *Adore Rootkit*[8]. Damit der Angreifer das geladene Modul nicht entdeckt bzw. entfernen kann, wird ein zusätzliches Kernel Modul geladen, das das *Sebek*-Modul aus der Liste der geladenen Module entfernt. Die Aktivitäten des Angreifers werden als UDP-Pakete (aus Gründen der Einfachheit des Protokolls) versendet.

- Die Komponente **sebeksniff** [seb 03b] hört auf dem Gateway-Rechner (*sebek server*) auf dem konfigurierten Port und Interface und speichert die ankommenden Daten in eine Log-Datei.

- Mit dem Perl-Programm **sbdump** lassen sich die eingegebenen Befehle aus der Log-Datei extrahieren.

[8] https://www.team-teso.net/releases/adore-0.39b4.tgz

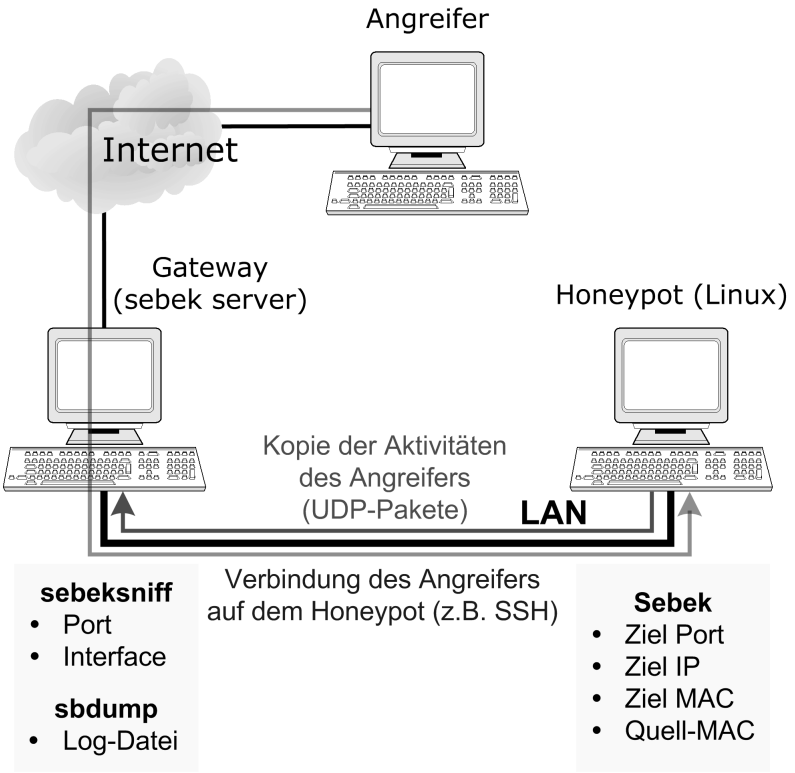

Abbildung 8: Funktionsweise von *Sebek*, *sebeksniff* und *sbdump* (vgl. [Bal 03])

Um auch die Vorgehensweisen auf dem Rechner zu überwachen, werden alle Kommandos, die der Angreifer auf dem Rechner eingegeben hat mitprotokolliert und gespeichert.

Abbildung 9 zeigt das zugrunde liegende Konzept:
Ruft ein Prozess die *read()*-Methode im User Space auf, so wird ein Systemaufruf gestartet. Da in der Syscall Table der Eintrag für die *read()*-Methode durch das Kernel-Modul verändert wurde, wird nun die *read()*-Methode von *Sebek* ausgeführt. *Sebek* hat nun Zugriff auf die Daten des Systemaufrufs. Die Eingaben werden einerseits an den „*Data logger*“ weitergereicht, andererseits zurück an die eigentliche *read()*-Methode des Kernels geführt. Durch das Abfangen der Eingaben im Kernel Space können alle Daten gesammelt werden, bevor die Eingaben ausgeführt werden. Dies funktioniert auch mit verschlüsselten Verbindungen, weil die Eingaben dekodiert werden müssen und somit auch ein Systemaufruf stattfindet.

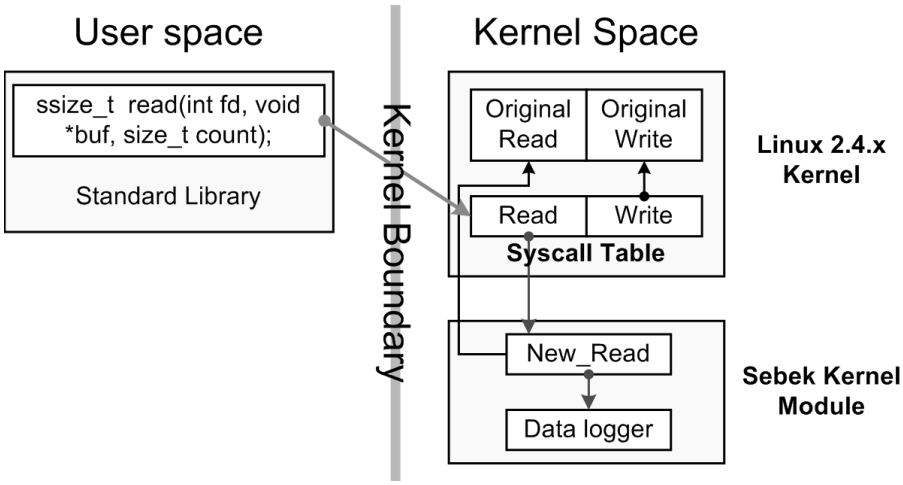

Abbildung 9: Konzeptuelle Darstellung der Umleitung des *read*-Aufrufes [Bal 03]

Nun müssen die im „*Data logger*“ gesammelten Daten über das Netz zum Gateway übertragen werden. Würden die Daten einfach über das Netz gesendet werden, könnte der Angreifer die Protokollierung durch einfaches Mithören des Netzverkehrs bemerken. Durch das *Sebek*-Modul wurde der Kernel so verändert, dass nur die von *Sebek* generierten Nachrichten für den Angreifer nicht zu erkennen sind. Außerdem wird verhindert, dass das System die Übertragung der *Sebek*-Pakete blockiert (z.B. durch eine vom Angreifer installierte Firewall) oder die Anzahl der übermittelten Pakete protokolliert. Abbildung 10 zeigt die konzeptuelle Idee für die Übermittlung. Das *Sebek* Kernel-Modul erzeugt und sendet UDP-Pakete direkt über den Netzwerkkartentreiber, ohne Verwendung des TCP/IP-Stacks des Kernels und der lokalen Filterung durch *netfilter/iptables*. Unter Linux basieren Netzwerk-Sniffer auf der Bibliothek *libpcap*[9], die über das *Socket Interface* die Netz-Daten sammelt. Aus diesem Grund kann der Angreifer diese Nachrichten nicht sehen oder mitprotokollieren. Eine genaue Beschreibung ist in [Bal 03] zu finden.

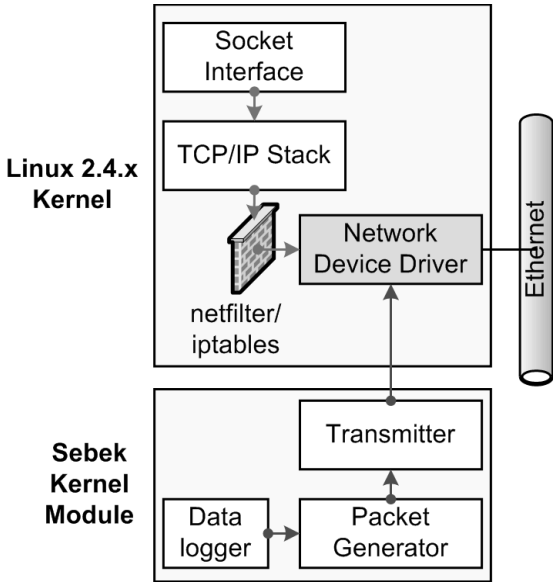

Abbildung 10: Konzeptuelle Darstellung des Generierens und Versendens von *Sebek* UDP-Paketen [Bal 03]

[9] Bibliothek für Netzwerk-Analyseprogramme

3.2.3.3 Sicherung der System Log-Dateien

Um zu verhindern, dass die System-Protokoll-Dateien gelöscht werden, muss ebenfalls ein Schutzmechanismus implementiert werden.
Zunächst werden alle Meldungen zusätzlich auf den Analyse-Server kopiert. Dies geschieht simultan beim Erzeugen einer Meldung. Wird der *syslog*-Daemon auf einem UNIX-System mit einem zusätzlichen Parameter gestartet und die Konfigurationsdatei angepasst, so werden die Meldungen auf den angegebenen Rechner weitergeleitet. Für Windows™-Systeme gibt es Zusatzsoftware (z.B. *eventlog to syslog utility* [smi 03]), die diese Aufgabe übernimmt.
Eine der ersten Aktivitäten nach einem erfolgreichen Eindringen auf einem UNIX-System ist meistens das Ausschalten des *syslog*-Daemon. Falls der Angreifer sich root-Rechte verschafft hat, lässt sich dies einfach mit „`kill -9 syslog`“ durchführen. Dies hat dann zur Folge, dass keine weiteren Einträge in die Log-Dateien geschrieben werden.
Um den Angreifer zu täuschen, kann man einen „Fake-*syslog*-Daemon“ installieren, der sich nach außen den Eindruck erweckt, als wäre es der *syslog*-Daemon-Prozess. Der eigentliche *syslog*-Daemon läuft unter einem anderen nicht verfänglichen Namen, die zugehörige Konfigurations-Datei befindet sich auch nicht an der für die Distribution gewohnten Stelle (z.B. bei Linux im Verzeichnis `/etc`), sondern z.B. in einem versteckten Ordner. In dieser Datei wird auch festgelegt, dass alle Einträge zusätzlich an einen anderen Rechner über das Netz weitergeleitet werden. Die Standard-Konfigurations-Datei aber existiert weiterhin, um den Angreifer nicht misstrauisch werden zu lassen.

3.2.4 Integritätschecker

Während des Betriebes eines Honeynets ist es unerlässlich, regelmäßig den Systemzustand der einzelnen Honeypots zu überprüfen. Zu diesem Zweck werden auf den Honeypots so genannte Integritätschecker installiert. Diese Programme kontrollieren z.B. den Inhalt von Verzeichnissen oder die Größe und Zugriffsrechte von Dateien. Die zu überwachenden Dateien und Verzeichnisse können gemäß den Anforderungen festgelegt werden. Bei der ersten Überprüfung (beim ersten Start) wird von den angegebenen Dateien eine Prüfsumme erstellt, die mit der in den künftigen Überprüfungen ermittelten Prüfsumme verglichen wird. Sind beide Prüfsummen identisch, so hat sich die Datei mit sehr hoher Wahrscheinlichkeit nicht verändert.
Vor allem nach Angriffen sollte ein Integritätscheck durchgeführt werden, damit festgestellt werden kann, welche Dateien der Angreifer verändert oder angelegt hat.
Auf dem Linux-Honeypot wird der frei verfügbare Integritätschecker *Tripwire* [tri 03a] eingesetzt, da dieser „als meist geschätztes Tool zum Erkennen von unerlaubten Änderungen an Betriebssystemen und kritischer Anwendungssoftware“ [HROG 03] gilt.
Für Windows™-Betriebssysteme ist *Tripwire* leider nicht frei verfügbar, sondern lediglich als 30-Tage Testversion, die auch nur den Funktionsumfang simuliert [tri 03b]. Ein frei verfügbarer Integritätschecker (*LANguard System Integrity Monitor*) wird von der Firma *GFI Network Security*[10] zur Verfügung gestellt. Dieses Programm hat zwar einen kleineren Funktionsumfang als *Tripwire*, reicht aber für die erforderlichen Anwendungszwecke aus: Es

[10] http://www.gfi.com/languard/

lässt sich einstellen, welche Dateien und Verzeichnisse überwacht werden sollen. Es kann konfiguriert werden, dass nach einer durchgeführten Überprüfung ein Statusbericht per Mail versendet wird.

3.2.5 Angebotene Dienste und zur Verfügung gestellte Daten

Damit der Angreifer auch eine Möglichkeit hat, sich Zugang zu den Honeypots zu verschaffen, sollten eine gewisse Anzahl an Diensten zur Verfügung gestellt werden.
Folgende Dienste werden angeboten:

- **FTP-Server**: Viele Angreifer versuchen mit Hilfe eines Anonymous-Accounts auf einem FTP-Server Daten wie z.B. Musikdateien oder Filme abzulegen. Auf dem Windows™-Honeypot kommt der FTP-Server, der bei dem *ISS* (*Internet Information Services*) Programmpaket mitgeliefert wird, zum Einsatz. Auf dem Linux-Honeypot wird der Standard FTP-Daemon installiert.

- **Web-Server**: Auch Web-Server sind beliebte Angriffsziele. Als Web-Server kommen der *ISS* von Microsoft und der *Apache* Web-Server zum Einsatz.

- **Datenbank-Server**: Auf dem Linux-Honeypot wird ein MySQL-Server installiert, der zum einen als Angriffsziel zum anderen als Datenquelle für Webseiten dient.

- **SSH-Server**: Auf dem Linux-Honeypot ist ein SSH-Server per Default installiert.

- **Mail-Server**: Der im *ISS*-Paket enthaltene Mail-Server wird auf dem Windows™-Honeypot installiert. Der Mail-Server auf dem Linux-Honeypot ist aus dem Internet nicht ansprechbar (Defaulteinstellung von SuSE Linux). Nachdem auf den Honeypot-Systemen keine E-Mails abgerufen werden, spielt der Mail-Server eine untergeordnete Rolle.

Neben den zur Verfügung gestellten Diensten ist es auch notwendig, dass diese Dienste auch Daten ausliefern, um die Attraktivität der Honeypots zu steigern. Dabei dürfen natürlich keine sensiblen Daten öffentlich zur Verfügung gestellt werden.

Daten für einen Web-Server
Auf einem Server im LRZ sind alle Daten über das Verkehrsaufkommen innerhalb des Münchner-Wissenschaftsnetz (MWN) und am Übergang zum Gigabit-Wissenschaftsnetz (G-WiN) gespeichert. Es handelt sich hierbei um so genannte Accounting Daten (dazu gehören Quell- und Ziel-IP-Adressen, deren DNS-Namen, das übertragene Datenvolumen, etc.), die für Außenstehende natürlich interessante Daten darstellen.
Innerhalb des MWN können aktuelle Statistiken über das Verkehrsaufkommen des MWN unter der Internetadresse *http://wwwmwn.lrz-muenchen.de* abgerufen werden.
Diese Daten sollen als „interessant wirkende Daten“ in verfälschter Form mit einer History-Funktion auf dem Linux-Honeypot zur Verfügung gestellt werden. Dazu werden mit Hilfe eines Perl-Skriptes die IP-Adressen mit Hilfe eines Zufallsgenerators verändert, die DNS-Namen entfernt und anschließend das Erstellungsdatum der „gefälschten“ Statistik in einer

MySQL-Datenbank gespeichert. Mit Hilfe einer Skriptsprache (in diesem Fall PHP[11]) werden die Daten ausgelesen (LAMP[12]-Architektur). Für jeden Tag gibt es zwei Accouting-Statistiken.

Daten für einen FTP-Server
Auf dem FTP-Server des Windows™-Honeypot befindet sich ein ISO-Image des Live-Linux-Systems Knoppix (Version 3.2[13]), sowie der MD5-Hashwert und einige Readme-Dateien.

Schlechte Konfiguration von Diensten
Andererseits kommt es auch öfters vor, dass Systemadministratoren Dienste (evtl. sogar unbeabsichtigt) installieren, die vergessen wurden zu konfigurieren und abzusichern oder Schwachstellen bekannt wurden und nicht behoben wurden (z.B. *IIS* von Microsoft). Der FTP-Server auf dem Linux-Honeypot und der Web-Server auf dem Windows™-Honeypot bleiben ohne Daten: Hier wurde „vergessen“, diese Dienste zu konfigurieren.

3.3 Anforderungen an die Software

Um ein Honeynet zu betreiben, ist es wichtig sich Gedanken über die verwendeten Betriebssysteme und die eventuell benötigte Zusatzsoftware zu machen.

3.3.1 Betriebssysteme

Die Auswahl der zu installierten Betriebssysteme auf den Honeypots ist eine wichtige Aufgabe. Es gilt abzuwägen, wie viel Spielraum man einem Angreifer lässt. Bei der Installation eines nicht mehr ganz aktuellen Betriebssystem (z.B. Microsoft Windows™ 2000) existieren mehrere Sicherheitslöcher und damit potentielle Angriffsmöglichkeiten als bei einem sehr aktuellen Betriebssystem (z.B. Microsoft Windows™ 2003 Server). Ähnliches gilt für UNIX-Betriebssysteme.
Im Rahmen der Diplomarbeit soll ein kleines Honeynet mit zwei Honeypots realisiert werden. Als Betriebssysteme werden Microsoft Windows™ 2000 und ein SuSE Linux 8.0. Für Windows™ 2000 gibt es in der Zwischenzeit vier so genannte Servicepacks, das sind größere Updates von Microsoft um Sicherheitslöcher zu beseitigen. Von diesen Servicepacks wird allerdings keines installiert, so bestehen genügend Angriffsmöglichkeiten für diesen Rechner. Auch bei SuSE Linux 8.0, erschienen im Frühjahr 2001, gibt es in der Zwischenzeit eine Vielzahl an bekannt gewordenen Angriffsmöglichkeiten: Die installierten Programmpakete sind also nicht mehr auf dem neuesten Stand.
Neben den Betriebssystemen für die Honeypots müssen auch Entscheidungen über die Betriebssysteme für das Gateway und den Analyse-Server getroffen werden. Diese Betriebssysteme sollten natürlich auf dem neuesten Stand sein. Hier wird die zum damaligen Zeitpunkt (Mai 2003) neueste Version von SuSE Linux (Version 8.2) verwendet.

[11] Skriptsprache zur Erstellung von dynamischen Web-Seiten (http://www.php.net)
[12] Abkürzung für eine Web-Server Umgebung die aus den Komponenten **L**inux, **A**pache, **M**ySQL, **P**HP aufgebaut ist.
[13] Stand: Mai 2003

3.3.2 Zusatzsoftware

Leider sind bei den Betriebssystemen nicht immer alle benötigten Programme mitgeliefert, die für ein erfolgreiches Betreiben eines Honeynets notwendig sind.

Zusatzsoftware für Windows™-Systeme
Folgende Zusatzprogramme werden benötigt und installiert:

- Es fehlt die Möglichkeit, System-Log-Einträge zusätzlich auf einen anderen Rechner (vorzugsweise Linux-Rechner) zu speichern. Hierzu wird das Programm *eventlog to syslog* [smi 03] benötigt.

- Um während des Betriebes den Honeypot überwachen zu können und evtl. den Rechner neu zu starten, ist eine Software zur Fernsteuerung (Remote-Zugriff) notwendig. Hierbei eignet sich die frei verfügbare Software *TightVNC*[14]. Diese überträgt die Windows™-Benutzeroberfläche über das Netz ähnlich wie bei einem Terminal Server.

Zusatzsoftware für Linux-Systeme (Software, die nicht in der SuSE 8.0/8.2 Linux Distribution enthalten ist)

- *swatch*[15]: Bei dem Programm *swatch* handelt es sich um ein Perl-Skript, das Log-Dateien mit Hilfe von regulären Ausdrücken überwacht (in diesem Fall `/var/log/messages`) und bei einer Übereinstimmung eine definierte Aktion ausführt. Dieses Programm ist notwendig für die Benachrichtigung bei Aktivitäten im Honeynet.

- Bridging-Firewall: Die Bridge-Funktionalität ist bereits in den Distributionen enthalten, es fehlt die Erweiterung von *iptables*, die eine Firewall auf Bridge-Ebene erlaubt (siehe Kapitel 3.2.2).

- *snort*: Hier sollte immer die aktuellste Version verwendet werden, um zu verhindern, dass *snort* durch bekannt gewordene Schwachstellen angreifbar ist (siehe Kapitel 3.2.3.1).

- *snort_inline*: Dient zur Beschränkung des ausgehenden Verkehrs der Honeypots (siehe Kapitel 3.2.2).

- *Sebek*: Zur Überwachung der Angreifer-Aktivitäten auf dem Honeypot (siehe Kapitel 3.2.3.2)

[14] http://www.tightvnc.org
[15] http://swatch.sourceforge.net/

3.4 Benachrichtigungsmechanismen

Während des Betriebs eines Honeynets ist es natürlich notwendig, dass der Administrator des Netzes über die Vorkommnisse informiert wird, um eventuelle Gegenmaßnahmen einleiten zu können. Außerdem ist eine direkte Überwachung 24 Stunden am Tag, um auf Angriffe zu warten, nicht praktikabel.
Bei den Benachrichtigungsmechanismen gibt es verschiedene Möglichkeiten, wie der Administrator über Vorkommnisse informiert werden kann.

3.4.1 Benachrichtigung über E-Mail

Eine relativ einfache Möglichkeit den Betreuer über Vorkommnisse im Honeynet zu informieren, ist per E-Mail.
Primär sind dabei folgende Daten interessant:

- Verbindungstyp: ein- oder ausgehend
- Von bzw. zu welchem Honeypot
- Quell-Adresse der Verbindung
- Quell-Port der Verbindung
- Ziel-Port der Verbindung

Wenn bei jeder auftretenden Verbindung eine Mail versendet wird, dann kann die Mailbox des Betreuers relativ schnell anwachsen. Hier lassen sich Filter bzw. die Anzahl an versendeten E-Mails pro Stunde beschränken.
Da das Internet ein unsicheres und nicht zeitnahes Transportmedium ist, ist nicht garantiert, dass auch alle E-Mails den Empfänger erreichen bzw. die E-Mails sofort an den Empfänger weitergeleitet werden. Außerdem kann es zu Ausfällen bei Netzkomponenten (z.B. durch Stromausfälle) und Mail-Servern kommen.

3.4.2 Benachrichtigung über Short Message Service (SMS)

Neben der E-Mail-Benachrichtigung sollte zusätzlich eine Benachrichtigungsmöglichkeit existieren, die den Betreuer über Vorkommnisse in Kenntnis setzen kann, da der Betreuer wahrscheinlich nicht permanent vor einem Rechner sitzt und auf E-Mails wartet. Hierzu eignet sich der Short Message Service (SMS) für Mobiltelefone (Handys). Eine Nachricht mit bis zu 160 Zeichen kann den Betreuer, auch wenn er unterwegs ist, informieren. Da es sich wie auch beim E-Mail-Versand um einen nicht zeitnahen und unsicheren Dienst handelt, ist nicht sichergestellt, dass die Nachrichten auch beim Empfänger ankommen.
Leider ist das Versenden von SMS-Nachrichten nicht umsonst, z.B. wird beim Mobilfunkanbieter O_2[16] für den Versand einer SMS über einen Dialin Zugang per Modem oder ISDN bei Benutzung der Deutschen Telekom als Festnetzgesellschaft bis zu 29,2 Cent/Minute[17] berechnet (die Abrechnung ist nicht pauschal für eine SMS). Daher sollte

[16] http://www.o2-online.de
[17] Stand: Mai 2003, Quelle: Deutsche Telekom

diese Möglichkeit der Benachrichtigung nur für besondere Vorkommnisse und auf eine bestimmte Anzahl für einen bestimmten Zeitraum beschränkt werden. Ein Dialin Zugang für den SMS-Versand ist am LRZ eingerichtet.

3.5 Absicherung des Gateway- und Analyse-Rechners

Durch die Bereitstellung eines Management-Zugriffes für den Gateway-Rechner ist dieser über diesen Weg angreifbar. Es ist also notwendig, diesen Zugang geeignet abzusichern.

Benötige Dienste nur an das Management-Interface binden
Der Zugang zum Gateway-Rechner erfolgt über SSH. Dabei wird der Dienst nicht auf dem Standard-Port 22 gestartet, sondern auf einem Port größer 30000. Somit ist die Gefahr, dass dieser offene Port entdeckt wird geringer, da die meisten Portscanner lediglich die Ports 1 bis 1024, sowie einige ausgewählte höhere Ports überprüfen. Im konkreten Fall wird der SSH-Server an Port 36987[18] gebunden.
Der für die späteren Analysezwecke eingesetzte *Apache* Web-Server wird ebenfalls von Port 80 auf einen höheren Port (9004) gebunden.
Darüber hinaus werden diese Dienste lediglich an das Management-Interface gebunden. Durch die verwendete Bridge-Architektur erhält der Gateway-Rechner nur eine IP-Adresse nach außen, die aus einem anderen Subnetz ist.

Zugangsbeschränkung
Damit nicht jeder Rechner aus dem Internet auf den Gateway-Rechner einloggen kann, wird der Zugriff auf wenige Rechner mit *iptables* beschränkt. Damit auch bei Ausfällen von Netzkomponenten ein Zugriff möglich ist, sollten Rechner aus unterschiedlichen (Sub-)Netzen eingetragen werden, z.B. zwei Rechner vom Lehrstuhl für Kommunikationssysteme und Systemprogrammierung der LMU München sowie ein Rechner des Instituts für Informatik der TU München.
Damit selbst im Falle eines Findens des SSH-Zugangs kein Einloggen per Passwort möglich ist, wird die Authentisierungsmethode des SSH-Server auf das Public-Key-Authentication-Verfahren umgestellt, das heißt, es muss jeweils ein öffentlicher Schlüssel der erlaubten Rechner auf dem Gateway-Rechner hinterlegt sein.

Für den Analyse-Rechner müssen ähnliche Sicherheitsvorkehrungen getroffen werden: Ein Zugriff auf diesen Rechner soll nur vom Gateway-Rechner aus möglich sein. Auch hier wird der SSH-Serverdienst auf einen anderen Port gebunden, für den Fall, dass es einem Angreifer gelingen sollte, den Rechner ausfindig zu machen.
Weitere Schutzmechanismen werden auf dem Gateway-Rechner mit Hilfe von Firewall-Regeln definiert. So ist der einzige Kommunikationsweg zwischen den Honeypot-Rechnern und dem Analyse-Rechner das Senden von *syslog*- oder *eventlog*-Meldungen über den UDP-Port 514.

[18] Unbenutzer Port gemäß [ian 03]

4 Aufbau und Betrieb

Das folgende Kapitel beschreibt den Aufbau, die Inbetriebnahme sowie den eigentlichen Betrieb des Honeynet.

4.1 Aufbau des Honeynets

Der Aufbau des Honeynets erfolgt im Leibniz-Rechenzentrum im vierten Obergeschoss. Abbildung 11 zeigt das aufgebaute Honeynet: Der schwarze Rechner auf der linken Seite ist der Gateway, die drei Rechner auf der rechten Seite sind von oben nach unten der Linux-Honeypot, der Windows™-Honeypot und der Analyse-Server. Zum einfachen Umschalten zwischen den Rechnern wird ein Keyboard-Video-Mouse (KVM) Switch installiert (auf dem Tisch). Die beiden Honeypots sind über einen Hub (auf dem KVM-Switch) miteinander verbunden.

Abbildung 11: Aufgebautes Honeynet

4.2 Verwendete Hardware

Für die Diplomarbeit wurden folgende Rechner zur Verfügung gestellt:

- Dell Optiplex GXi 5200 M
  - Intel Pentium® I 200 MHz Prozessor
  - 64 MB Arbeitsspeicher
  - 4,55 GB Seagate SCSI Festplatte (Medialist Pro 9140) und 2,15 GB Seagate SCSI Festplatte (Barracuda 2 LP)
  - 3Com OnBoard Netzwerkkarte von 3Com (3c905, Bommerang)
  - Rechnername: *internal.lrz-muenchen.de*

- Dell Optiplex GXi 5200 M
  - Intel Pentium® I 200 MHz Prozessor
  - 256 MB Arbeitsspeicher
  - 2,15 GB Seagate SCSI Festplatte (Barracuda 2 LP)
  - 3Com OnBoard Netzwerkkarte von 3Com (3c905, Bommerang)
  - Rechnername: *tivoli.lrz-muenchen.de*

- Dell Optiplex GXi 5200 M
  - Intel Pentium® I 200 MHz Prozessor
  - 128 MB Arbeitsspeicher
  - 4,55 GB Seagate SCSI Festplatte (Medialist Pro 9140) und 2,15 GB Seagate SCSI Festplatte (Barracuda 2 LP)
  - 3Com OnBoard Netzwerkkarte von 3Com (3c905, Bommerang)
  - Rechnername: *logging.lrz-muenchen.de*

- Compaq Prosignia 720
  - Intel Pentium® III 500 MHz Prozessor
  - 128 MB RAM Arbeitsspeicher
  - 9,1 GB SCSI Festplatte
  - 4 Netzwerkkarten
    - DLink DGE-500T (10/100/1000 MBit, *eth0*)
    - Intel Pro 100/S (10/100 MBit, *eth1*)
    - Intel Pro 100 OnBoard (10/100 MBit, *eth2*)
    - 3c905C-TX/TX-M (Tornado, 10/100 MBit, *eth3*)
  - Rechnername: *gway.lrz-muenchen.de*

4.3 Installation

Der folgende Abschnitt beschreibt die Installation und weitere Konfiguration der einzelnen Rechner. Zum Abschluss dieses Kapitels zeigt Abbildung 14 auf Seite 43 eine Gesamtübersicht aller beteiligten Systeme und Transitsysteme.
Für die Auswahl der Rechnernamen der Honeypots liegen folgende Überlegungen zugrunde: Der Name sollte „interessant“ wirken, so dass der Angreifer denkt, es handle sich um ein wichtiges System bzw. ein System, auf das er keinen Zugriff erhalten sollte. Für den Linux-

Honeypot wurde der Name „*internal.lrz-muenchen.de*“ gewählt (Rechner mit Daten, die nur für interne Zwecke bestimmt sind), der Windows™-Honeypot erhält den Namen „*tivoli.lrz-muenchen.de*“ (Rechner, zum Management von Netzkomponenten[19]).
Die IP-Adressen des Honeynet wurden von den Verantwortlichen des LRZ aus dem Subnetz *129.187.19.0* festgelegt. Dieses Netz wird im LRZ als Transportnetz benutzt.

4.3.1 Installation des Gateways (*gway.lrz-muenchen.de*)

Für die Grundinstallation wurde eine Minimal-Installation der Linux-Distribution SuSE 8.2 ausgewählt. Während der Installation wurden folgende Pakete (unterteilt in drei Kategorien) hinzugefügt (Tabelle 3):

System	Web-Server	Datenbank
tpcpdump	apache (1.3.27)	mysql (3.23.55)
libpcap	mod_php_core (4.3.1)	mysqlclient (3.23.55)
ncurses-devel	mod_php4 (4.3.1)	mysql-devel (3.23.55)
kernel_source (2.4.20)	mesasoft	mysql-shared (3.23.55)
make	mesaglu	perl-DBI
patch	gd (2.0.11)	perl-Data-ShowTable
patchutils	libjpg	perl Msql-Mysql-Modules
gcc (3.3)	libpng	perl-TimeDate
glibc-devel	freetype2	libodbc
bindutil	XFree86-libs	postgresql-libs
binutils	3ddiag	
libtool	libmcrypt	
ucdsnmp	libmcal	
imap-lib	libclms	
gmp	libmng	
wget	sablot	
	expat	
	t1lib	
	qt3	

Tabelle 3: Zusätzliche ausgewählte Pakete bei der Installation des Gateways

Installation des Bridge-Firewalling

Damit die Firewallregeln von *iptables* auch bei einem als Bridge konfigurierten Rechner funktionieren, muss zusätzlich noch der Kernel verändert werden. Der erforderliche Patch steht unter [bfw 02] zum Download bereit.
Die Installation gliedert sich in folgende Schritte:

1. Die Pfade in der Patch-Datei müssen noch angepasst werden:
   von `--- linux-2.4.19/` in `--- linux/`
   von `+++ linux-2.4.19-brnf0.0.7/` in `+++ linux/`
2. Patch-Datei ins Verzeichnis `/usr/src` kopieren

[19] *Tivoli* ist eine Netzmanagement-Software der Firma IBM, http://www.tivoli.com

3. In das Verzeichnis `/usr/src` wechseln
4. Mit `patch p0 < bridge-nf-0.0.7-against-2.4.19.diff` werden die Kernel Quelldateien gepatcht.
5. In das Verzeichnis `/usr/src/linux` wechseln
6. Kernel-Konfigurationsmenü mit `make menuconfig` starten
7. Unter `Networking options --->`, `802.1d Ethernet Bridging` und den neuen Menüpunkt `netfilter (firewalling) support` mit einem „*“ markieren.
8. Beide Module werden nun in den Kernel einkompiliert (gekennzeichnet mit `[*]`). Mit `Exit` das Konfigurationsmenü verlassen und die Konfiguration speichern.
9. Abhängigkeiten überprüfen mit `make dep`.
10. Kernel Image mit `make bzImage` erzeugen (Dauer bei der eingesetzten Maschine 30 min.). Das erzeugte Image wird in `arch/i386/boot` abgelegt.
11. Module mit `make modules` neu übersetzen (Dauer ca. 4 Stunden) und mit `make modules_install` installieren.
12. Kernelimage in `/boot`-Verzeichnis kopieren und Eintrag im Bootmanager erzeugen.

Nach einem Reboot erscheinen während des Startens des Rechners folgende Zeilen:

```
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
NET4: Ethernet Bridge 008 for NET4.0
Bridge firewalling registered
```

Leider sind bei der SuSE Linux 8.2 Distribution keine Programme zum Managen einer Bridge enthalten. Zum Managen werden folgende Funktionen benötigt:

- Initialisieren und Löschen einer Bridge:
  - `brctl addbr <name>` legt eine Bridge mit der Bezeichnung *<name>* an.
  - `brctl delbr <name>` löscht die Bridge.

- Hinzufügen bzw. Löschen von Interfaces zu einer Bridge:
  - `brctl addif <brname> <ifname>` fügt das Netzwerkinterface *<ifname>* der Bridge *<brname>* hinzu (Port einer Bridge).
  - `brctl delif <brname> <ifname>` entfernt das Netzwerkinterface *<ifname>* aus der Bridge *<brname>*.

- Festlegung von Metriken
  `brctl setpathcost <bridge> <port> <cost>` setzt die Kosten *<cost>* für den Port *<port>* der Bridge *<bridge>*.

- Ein- bzw. Ausschalten des Spanning Tree Protocols
  `brctl stp <bridge> off` deaktiviert das Spanning Tree Protocol der Bridge *<bridge>*. Die Bridge versendet keine Bridge Protocol Data Units, die dem Angreifer die Existenz einer Bridge informieren würden.

- Anzeigen der aktuellen Konfiguration
  - `brctl show` zeigt Informationen über die Bridge und die zugehörigen Netzwerkinterfaces an.

- `brctl showmacs <brname>` zeigt die von der Bridge *<brname>* gelernten MAC-Adressen an

Unter [bri 03] kann das benötigte Programmpaket herunter geladen und mittels `configure`, `make` und `make install` übersetzt und installiert werden.

Für das Management-Interface wird eine IP-Adresse aus dem Subnetz *129.187.18.0* verwendet.

Installation der Aufzeichnungsprogramme

1. *snort*
   Es wurde die aktuellste Version im Quelltext (zum Zeitpunkt der Installation Version 2.0.0) von der Homepage[20] herunter geladen. Um eine MySQL-Datenbank zum Speichern der Alarme benutzen zu können, muss die Konfiguration mit `./configure --with-mysql` erfolgen (dabei werden die Header-Dateien von MySQL benötigt). Das Programmpaket wird mit `make` übersetzt und mit `make install` in `/usr/local/bin` installiert. Für die Log-Dateien muss noch das Verzeichnis `/var/log/snort` angelegt werden. Das Regelset wird nach `/etc/snort` kopiert. Dort befindet sich auch die Konfigurationsdatei `snort.conf`.
   Vom Honeynet Project wird bereits ein Start-Skript für *snort* unter [hsn 03] zur Verfügung gestellt.

2. *snort_inline*
   Das im Kapitel 3.2.2 vorgestellte Programm liegt als vorkompiliertes Programmpaket auf der Internetseite des Honeynet Projects [si 03] bereit. Mit Hilfe eines Konvertierungs-Skriptes [con 03] lassen sich die Regeln in „drop-Regeln“ verwandeln, damit keine (bekannten) Angriffe vom Honeynet aus auf andere Rechner im Internet verübt werden. Auch hier wird auf der Honeynet Project Internet-Seite bereits ein Start-Skript [hsi 03] zur Verfügung gestellt.

3. *tcpdump*
   Bei der Installation wurde bereits dieses Paket ausgewählt und installiert. Damit nicht alle Daten in einer einzigen Datei gespeichert werden, erfolgt die Speicherung tageweise. Das entsprechende Skript befindet sich im Anhang (A.1.1.3).

4. *sebeksniff*
   Das im Abschnitt 3.2.3.2 vorgestellte Programm wird entpackt und mit `./configure`, `make` und `make install` installiert.

[20] http://www.snort.org

Installation und Konfiguration der Benachrichtigung

1. *swatch*
   Das Programm wird entpackt und mit dem mitgelieferten Perl-Skript installiert. Es wird eine Konfigurationsdatei `.swatch.conf` erstellt, in der die zu überwachenden regulären Ausdrücke sowie die durchzuführenden Aktionen eingetragen werden. (Konfigurationsdatei befindet sich im Kapitel A.1.1.4)

2. E-Mail
   Als SMTP-Server wird in *YaST2* der Mail-Server des LRZ eingetragen: *mailout.lrz-muenchen.de*.
   Bei jeder ein- oder ausgehenden Verbindung zu den Honeypot-Rechnern wird an eine angegebene Mail-Adresse eine Nachricht versendet, solange das eingestellte Limit nicht überschritten wird.

3. SMS
   Der SMS-Versand erfolgt über den Rechner *nm1.lrz-muenchen.de*. Dort wurde eine Kennung eingerichtet, die Authentisierung erfolgt über das Public-Key-Authentication-Verfahren. Die Beschränkung des Versandes erfolgt über Semaphoren. Die verwendeten Skripte (Shell-Skripte) befinden sich im Anhang (A.1.1.5).
   In Absprache mit den Verantwortlichen des LRZ wurde der Versand auf maximal drei SMS-Nachrichten pro Stunde beschränkt:

   - Zwei Nachrichten für die Benachrichtigung bei ausgehenden Verbindungen von den Honeypots.

   - Eine Nachricht, falls in einer Stunde mehr als 25 eingehende Verbindungen registriert wurden. Dies dient als Indikator für z.B. skript- oder toolgesteuerte Angriffe auf Web-Server, bei denen viele unterschiedliche Verbindungen zum Server aufgebaut werden.

Benutzerkonten

Außer dem Benutzer root sind keine Benutzerkonten vorhanden.

Einrichtung des Zuganges

Die Datei `/etc/ssh/sshd_config` wird gemäß den Anforderungen aus Abschnitt 3.5 angepasst: Der SSH-Server wird an Port 36987 gebunden. Das Sperren des Logins mit Passwort wird erst aktiviert, wenn die benötigen Schlüssel der Rechner in die Dateien `/root/.ssh/known_hosts` und `/root/.ssh/authorized_keys` eingetragen wurden.

Konfiguration der Bridge und Firewall

Als Vorlage dient das vom Honeynet Project zur Verfügung gestellte Shell-Skript [mil 03]. Dabei werden folgende Änderungen vorgenommen bzw. hinzugefügt:

- Erweiterung der Bridge: Hinzunahme des Interfaces für den Log-Rechner (*eth3*). Es wird eine Variable `LOG_IFACE` und `LOG_IP` definiert.

- Alle Anfragen aus dem Internet zum Analyse-Server werden verworfen (Ausnahme: SSH-Zugriffe vom Gateway zum Analyse-Server).

- NTP-Anfragen von allen Rechnern zu einem Server des LRZ (*ntp1.lrz-muenchen.de*) werden akzeptiert und nicht protokolliert.

- Zugriff auf den Windows™-Honeypot mittels *TightVNC* (siehe Kapitel 3.3.2), der Standardport wird von 5900 auf 5978 abgeändert (zufällige Auswahl). Ein nicht protokollierter Zugriff kann nur über eine IP-Adresse des TUM-VPN-Zuganges (Netzadresse *129.187.51.0*) erfolgen.

- DNS-Anfragen (UDP und TCP) der beiden Honeypots an die LRZ-DNS-Server *129.187.10.25* und *129.187.16.1* werden ohne Logging akzeptiert.

- Der *Sebek*-Support wird aktiviert: `SEBEK="yes"`. Für `SEBEK_DST_IP` wird die IP-Adresse *129.187.19.108* (freie IP-Adresse im Subnetz), für `SEBEK_DST_PORT` der Port 3141 verwendet.

- SSH-Zugriffe vom Gateway-Rechner auf den Analyse-Server (Port 36999) werden ohne Logging akzeptiert.

- SSH-Zugriffe auf den Gateway-Rechner (Port 36987) sind nur von folgenden IP-Adressen erlaubt (Variable `MANAGER`):
  - *141.84.218.33* (pcheger3.nm.informatik.uni-muenchen.de)
  - *141.84.218.1* (sunheger1.nm.informatik.uni-muenchen.de)
  - *131.159.4.197* (athalle5.informatik.tu-muenchen.de)
  - *129.187.15.176* (Laptop Herr Völkl)

- SSH-Zugriffe auf den Linux-Honeypot vom LRZ-Accounting-Server (*10.155.10.35*) werden ohne Protokollierung akzeptiert.

- Syslog-Meldungen von den Honeypots zum Analyse-Server werden ohne Protokollierung akzeptiert.

Das angepasste Skript befindet sich im Anhang (Abschnitt A.1.1.1).

Abbildung 12 zeigt den konfigurierten Gateway-Rechner. An der Netzwerkkarte *eth1* sind vier bzw. fünf Programme gebunden. Das Programm *p0f* muss nicht notwendigerweise

während des Betriebes aktiv sein, es wird ausführlich in Kapitel 5.2.5 beschrieben. Die Bridge *br0* ist über die drei Netzwerkkarten definiert (*eth0*, *eth1* und *eth3*), die Firewall mit *iptables* überwacht alle Netzwerkkarten.

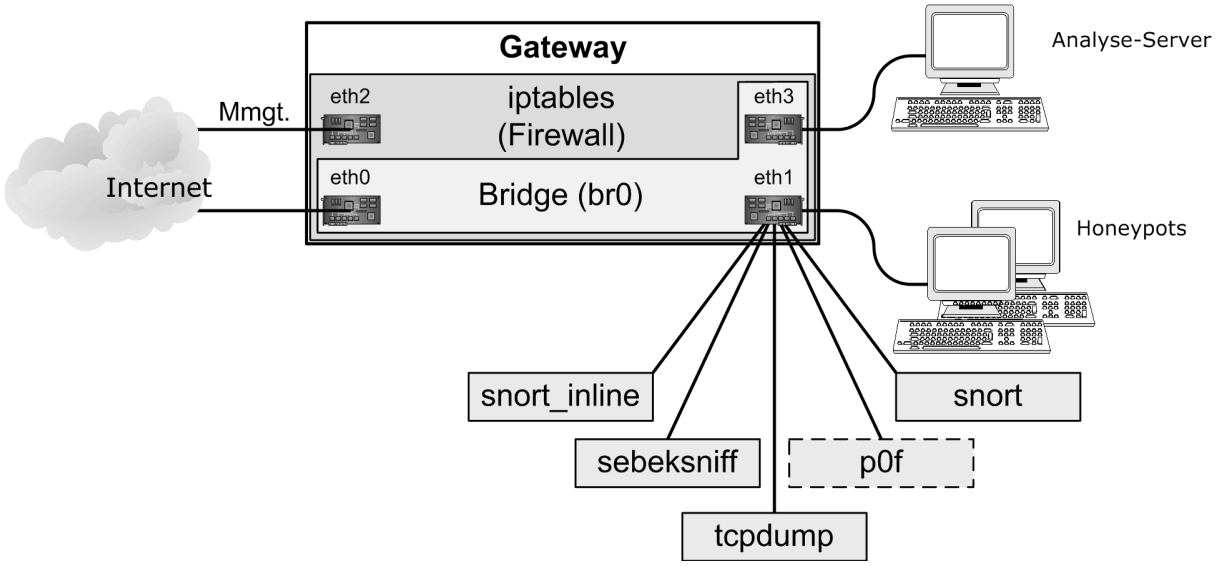

Abbildung 12: Konfigurierter Gateway-Rechner mit unterschiedlichen „Abhördiensten“

4.3.2 Installation des Linux-Honeypots (*internal.lrz-muenchen.de*)

Grundinstallation

Im Installationsprogramm *YaST2* von SuSE Linux 8.0 wird die Option „*Standard System (ohne Office)*“ ausgewählt. Zusätzlich werden in der erweiterten Auswahl der Teilast „*Netzwerk/Server*“ selektiert. Für die spätere Datenbankanbindung mit Perl werden noch folgende Pakete ausgewählt:

- perl-DBI
- perl-Data-ShowTable
- perl Msql-Mysql-Modules
- perl-TimeDate

Benutzerkonten

Neben dem Administrator (`root`) existieren noch zwei weitere Benutzer: `thaler` (Passwort: `traffic`) und `mueller` (Passwort: `muel1972`). Für den Benutzer `root` wird das Passwort `h1ynet8!` eingerichtet.

Installation der Protokollierungsmechanismen

Key-Logging
Um die Eingaben auf der Kommandozeile mitprotokollieren zu können wird ein Kernel-Patch installiert (siehe Kapitel 3.2.3.2).

Sebek
Das Programmpaket *Sebek* [seb 03a] wird entpackt und mit `./configure` vorbereitet. Im `Makefile` müssen ein paar Änderungen vorgenommen werden:

- Pfad für den Befehl `tar`:
  von `TAR = gtar` in `TAR = tar`
- Verzeichnis der „include“-Dateien
  von `INCLUDE=-I/usr/src/linux-2.4/include`
  in `INCLUDE=-l/usr/src/linux/include`

Anschließend kann mit `make` das Programm kompiliert werden.

Syslog
Es wird ein fake-*syslog*-Daemon erzeugt. Dabei handelt es sich um ein kleines C-Programm, das in einer Endlosschleife läuft und jede Stunde kurz „aufwacht“. Der Quelltext befindet sich im Anhang (A.1.2.1). Dieses C-Programm soll dem Angreifer den *syslog*-Daemon vortäuschen. Die ursprüngliche Konfigurationsdatei `/etc/syslog.conf` bleibt unverändert. Der eigentliche *syslog*-Daemon wird neu übersetzt. Der Pfad für die Konfigurations-Datei `syslog.conf`, die sich standardmäßig in `/etc/` befindet, wird in der Datei `syslogd.c` auf `/usr/local/etc/.status/sys.log` abgeändert. In dieser Datei wird die zusätzliche Weiterleitung aller *syslog*-Meldungen an den Analyse-Server eingetragen:

```
*.crit                      @129.187.19.107
*.*;mail.none;news.none     @129.187.19.107
```

Auf dem Honeypot selber werden die *syslog*-Meldungen ebenfalls gespeichert.

Dieser *syslog*-Daemon wird im System als „init service“ (ein unverfänglicher Name, bei dem sich nicht auf die eigentliche Aufgabe des Programms schließen lässt) geführt. Das Programm wird in `initd` umbenannt und nach `/sbin` kopiert. Das Startskript vom ursprünglichen *syslog*-Daemon wird kopiert und entsprechend abgeändert. Beim Starten des Systems werden dann beide „Dienste“ gestartet:

```
Starting syslog services        done
Starting init service           done
```

Installation des Webservers

Die Daten, die vom Accounting-Server des LRZ kommen, müssen verfälscht werden. Die Daten werden zweimal pro Tag auf den Honeypot kopiert. Ein Perl-Skript entfernt die Mail-Adressen sowie die DNS-Namen aus den übertragenen HTML-Dateien und erzeugt zufällige Adresseinträge im dritten und vierten Block der IP-Adresse. Anschließend wird diese Datei ins Webserver-Verzeichnis kopiert.
Da sich der „Webauftritt" auf eine Datenbank abstützt (in diesem Fall MySQL) erzeugt das Perl-Skript (siehe Kapitel A.1.2.2) auch einen Datenbankeintrag für die erzeugte bzw. veränderte Seite.

Installation des Integritätscheckers

Wie in Abschnitt 3.2.4 beschrieben, wird für den Linux-Honeypot der Integritätschecker *tripwire* installiert. Damit bei einem erfolgreichen Angriff, der Angreifer dieses Programm nicht sofort entdeckt und den Honeypot wieder verlässt, erfolgt die Installation nicht in die im `install.cfg` angegebenen Standardverzeichnisse, sondern in ein verstecktes Verzeichnis. Die Datei `install.cfg` wird wie folgt angepasst:

- `TWBIN="/.. /usr/sbin"`
- `TWPOLICY="/.. /etc/tripwire"`
- `TWMAN="/.. /usr/man"`
- `TWDB="/var/lib/…/tripwire"`
- `TWDOCS="/usr/doc/…/tripwire"`

Mit dem Installationsskript `install.sh` wird das Programm in die angegebenen Pfade installiert. Als *LocalKey* wird `HGH@LRZ`, als *SiteKey* `HoneynetLRZ` verwendet.
Die verwendete Policy-Datei befindet sich im Anhang (Abschnitt A.1.2.3).

4.3.3 Installation des Windows™-Honeypots (*tivoli.lrz-muenchen.de*)

Es erfolgt eine Standard-Installation von Windows™ 2000 (ohne Servicepacks oder Hotfixes). Zusätzlich werden die auf der Installations-CD befindlichen „*Internet-Informationsdienste*" (Web-Server, FTP-Server, Mail-Server) nachinstalliert.
Auf dem Web-Server werden keine Daten hinterlegt. Dies soll eine Unachtsamkeit beim Konfigurieren eines Rechners zeigen: Es wurde eine Software installiert, aber vergessen zu konfigurieren bzw. abzusichern. In das Root-Verzeichnis des FTP-Servers (`C:\Inetpub\ftproot`) wird das aktuelle ISO-Image von Knoppix[21] (Version 3.2) kopiert.

[21] http://www.knoppix.org

Benutzerkonten

Neben dem Administrator existieren noch zwei weitere Benutzer: `Thaler` (Passwort: `traffic`) und `Müller` (Passwort: `muel1972`). Diese sind analog zu dem Linux-Honeypot. Als Administrator-Passwort wird `h1ynetwin@lrz` verwendet.

Installation der Protokollierungsmechanismen

Die in Abschnitt 3.2.3.2 und 3.2.3.3 erklärten Programme (*ComLog* und *eventlog to syslog*) werden installiert. Als Ziel-Server für die Nachrichten wird bei *eventlog to syslog* der Analyse-Server definiert.
Damit das Programm *ComLog* funktioniert, muss der Schutz für das Überschreiben von System-Dateien ausgeschaltet werden. Dazu ist folgender Registry-Änderungen unter `HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon` notwendig:
Der Wert `SFCDisable` muss auf den Wert „`ffffff9d`“ (REG_DWORD) abgeändert werden. Nach einem Neustart ist der Dateischutz für die Systemdateien von Windows™ deaktiviert. Es erscheint auch ein Eintrag in der Ereignisanzeige
Nach der Installation werden alle Eingaben, die pro Aufruf in der „*Eingabeaufforderung*“ eingegeben werden, in einer eigenen Datei unter `C:\WINNT\Help\Tutor` (auf dem Honeypot selbst) gespeichert. Dies ist der Standardpfad für die Speicherung dieser Log-Dateien.

Installation des Integritätscheckers

Der in Abschnitt 3.2.4 beschriebene Integritätschecker *LANguard System Integrity Monitor* wird installiert. Die zu überwachenden Verzeichnisse sowie die Mail-Benachrichtigung werden definiert. Der Integritätscheck wird auf dem Rechner jede Nacht um 2 Uhr durchgeführt.

Installation des NTP-Clients

Damit die Zeit von einem NTP-Server synchronisiert wird, sind einige wenige Einstellungen notwendig. Diese sind auf [lrzntp 03] beschrieben.

Installation der Fernsteuerungssoftware *TightVNC*

Zur Remote-Steuerung des Rechners (z.B. von dem PC des Betreuers) wird das Programm *TightVNC* benutzt. Es handelt sich hierbei um eine Freewarelösung. Das Programm lässt sich einfach durch Ausführen der `setup.exe` installieren. Damit der Dienst automatisch beim Starten des Systems aktiviert wird, sind ein paar Registry-Einträge notwendig: Nach der Installation befindet sich dafür unter *Programme – TightVNC* ein entsprechender Menüpunkt. Außerdem muss *TightVNC* als Dienst registriert werden, da sonst *TightVNC* erst

gestartet wird, wenn sich ein Benutzer angemeldet hat. Bei einem Remote-Neustart wäre sonst kein Zugriff mehr möglich. Einen entsprechenden Menüpunkt für die Installation des Dienstes befindet sich im Startmenü.
Anschließend erfolgt die Festlegung des Portes und des Passwortes im Konfigurations-Dialog (siehe Abbildung 13). Als Offset (*display*) wird 78 (entspricht Port 5978) und als Passwort `8kgEfsiW` festgelegt.

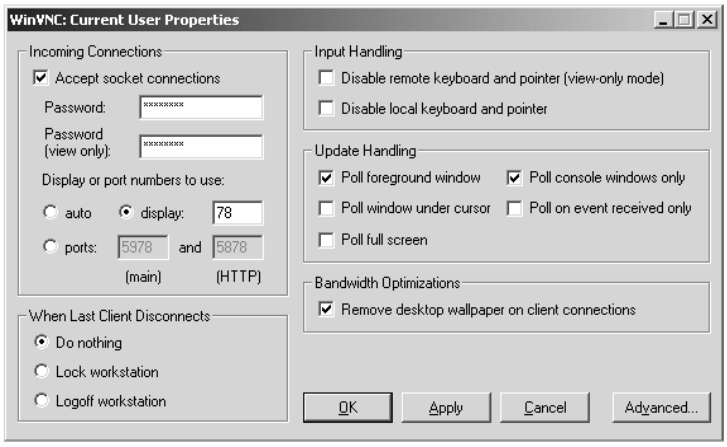

Abbildung 13: Konfigurations-Dialog von *TightVNC*

Der Zugriff erfolgt über das Programm `vncviewer.exe`. Die Einträge im Startmenü werden nach der Installation entfernt, um den Angreifer nicht auf diese Software aufmerksam zu machen.

4.3.4 Installation des Log-Rechners (*logging.lrz-muenchen.de*)

Der Rechner wird mit einem SuSE Linux 8.2 System mit minimaler X-Oberfläche installiert. Zusätzlich wird das Paket *ethereal* zum späteren Auswerten der Daten ausgewählt. Damit *syslog*-Meldungen von anderen Rechnern entgegennimmt, muss der *syslog*-daemon mit der Option `-r` gestartet werden (Eintrag in `/etc/sysconfig/syslog`). Außerdem wird die DNS-Auflösung deaktiviert, da sonst bei jedem Eintrag versucht wird, den DNS-Namen des Rechners zu bestimmen.

Benutzerkonten

Außer dem Benutzer `root` sind keine Benutzerkonten vorhanden.

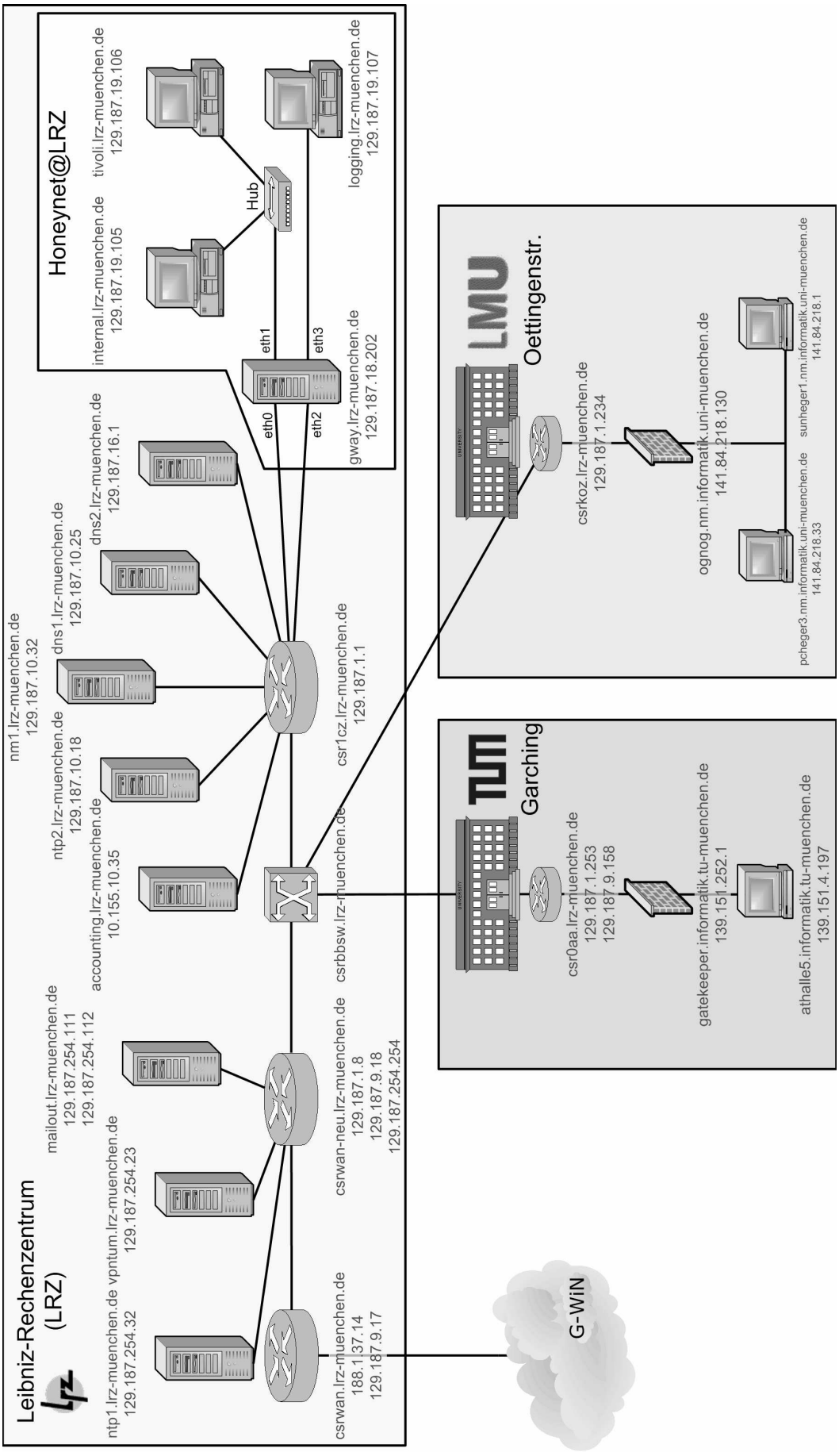

Abbildung 14: Globale Sicht mit allen beteiligten Systemen

4.4 Sichern der Konfigurationen

Damit die Konfiguration wiederhergestellt werden kann, falls ein Angreifer einen Rechner so kompromittiert hat, dass er nicht mehr hochfährt, muss die Konfiguration gesichert werden.
Dazu wird das Programm *partimage* [par 03] verwendet.
Es erlaubt das Sichern von Images für ReiserFS- als auch für NTFS-Partitionen über das Netzwerk per NFS. Auf der aktuellen Knoppix-CD[22] ist dieses Programm enthalten. Um die NFS-Funktionalität nutzen zu können muss der NFS-Daemon gestartet werden:

```
/etc/init.d/nfs-common start.
```

Anschließend kann ein Verzeichnis auf einem NFS-Server (in diesem Fall war der NFS-Server ein Notebook) gemountet werden und die Partition gesichert werden.
Diese Vorgehensweise wird für alle installierten Rechner wiederholt.

4.5 Konfigurationsänderungen während des Betriebes

Während des Betriebes war es notwendig einige Veränderungen an den Konfigurationen vorzunehmen, z.B. ein Updaten sowie eine Verfeinerung der *snort*-Regeln, da sehr viele erzeugte „Alarme“ in eine einzige Kategorie nämlich „Snort Unmatched“ abgelegt wurden.
Die einzelnen Änderungen werden im Folgenden erläutert.

4.5.1 Update und Ergänzung der *snort*-Regeln

Während des Betriebes sind die Regeln (Signaturen) regelmäßig aktualisiert worden (meistens jede Woche). Für *snort_inline* wurden diese Regeln mit dem convert-Skript in „drop-Regeln“ umgewandelt.
Da durch die Freischaltung aller Ports für die Honeypot-Rechner ab dem 12. August deutlich mehr Verbindungen aufgebaut wurden, sind folgende Signaturen ergänzt worden, um die Alarme übersichtlich darzustellen und die anschließende Auswertung nach genutzten Diensten und Ports zu erleichtern. Alle Verbindungen, die keiner Signatur zugeordnet sind, werden im Alarm „Snort Unmatched“ gespeichert. Die Signaturen sind aus Beobachtungen während des Betriebes entstanden. Eine komplette Liste befindet sich im Anhang (A.1.1.2).

- Routing-Informationen werden nicht protokolliert: Ein Router versendet periodisch RIP-Nachrichten, die nicht als Angriffe zu werten sind.

- *TightVNC*-Verbindungen von und zum Windows™-Honeypot auf Port 5978 aus dem VPN-TUM Netz werden nicht erfasst. Auch hier handelt es sich um gewollte Verbindungen. Verbindungen, die nicht vom VPN-TUM Netz kommen, werden natürlich protokolliert.

[22] Eine von CD startbare GNU/Linux-Distribution (Live-System)

- Verbindungen zu den Mail-Servern werden in einem eigenen Alarm gespeichert.

- Für FTP-Verbindungen wird eine Regel erstellt.

- DNS-Anfragen werden mit einer Regel erfasst.

- Pakete, deren *WindowSize* 55808 beträgt werden in einem Alarm gespeichert (*Stumbler*, siehe 5.4.2.3).

- Für den Transfer der „Rohdaten“ für den Webserver auf dem Linux-Honeypot existiert eine eigene Regel.

- *Sebek*-Übertragungen werden mit einer Regel erfasst. Dies ermöglicht eine schnelle Überprüfung auf Aktivitäten auf dem Linux-Honeypot.

- NetBIOS-Ports (137, 139, 445) werden jeweils in einem Alarm gespeichert. Dabei wird zwischen den Zieladressen sowie zwischen UDP und TCP-Verbindungen unterschieden.

- *Syslog*-Meldungen vom Linux-Honeypot zum Analyse-Server werden nicht erfasst. Hierbei handelt es sich um beabsichtigten Verkehr, der für Angriffe keine Rolle spielt.

- *Syslog*-Meldungen vom Windows™-Honeypot zum Analyse-Server werden mit einem Alarm erfasst. Aus diesen Alarmen können Rückschlüsse auf Aktivitäten auf dem Honeypot gezogen werden (z.B. die Beendigung des RPC-Dienstes).

- Diverse Trojaner-/Viren-Ports und Scanner-Tools erhalten eigene Alarme (siehe Kapitel 5.4.2.2):
    - 57 (FXScanner)
    - 4000 (SkyDance)
    - 4444 (RPC-Exploit Shell)
    - 4899 (RAdmin)
    - 12345 (NetBus)
    - 17300 (Kuang2)
    - 22226 (W32/HLLW.Gaobot-AA)
    - 22227 (W32/HLLW.Gaobot-AA)
    - 27374 (SubSeven)

- Für Anfragen auf den Port 554 (Real-Server) wird aufgrund einer Schwachstelle [sf 03] eine eigene Regel definiert.

- Zeitsynchronisation mit NTP wird in einem Alarm gespeichert. Hierbei handelt es sich um beabsichtigten Verkehr.

- Bei HTTP-Verbindungen zu Port 80 werden nur Pakete gespeichert, die HTTP-Kommandos enthalten (gesetztes PSH-Flag). Der Verbindungsaufbau wird verworfen.

- Für ICMP-Antworten und ausgehende ICMP-Anfragen werden eigene Regeln erzeugt.

- Anfragen in Bezug auf Microsoft SQL Server Exploits (Port 1433 und Port 1434) werden in eigenen Alarmen gespeichert.

4.5.2 Änderungen der Firewall-Regeln

Durch das Auftreten des Blaster-Wurms wurde auf dem Windows™-Honeypot innerhalb kürzester Zeit der RPC-Dienst beendet was einen Neustart des Rechners mit sich zog. Aus diesem Grund wurde am 12. August der Port 135 für eingehende Verbindungen gesperrt, allerdings werden die Verbindungswünsche protokolliert. Der genaue Ablauf des Blaster-Wurms ist in Kapitel 5.4.2.4.3 beschrieben.

4.5.3 Vorbereitung für Auswertungen

Um auch schon während der Betriebsphase die Vorgänge auf dem Honeynet beobachten zu können, wurden zwei Werkzeuge zur Online-Analyse installiert: *Iptables log* und *ACID*. Beide Programme werden in Kapitel 5.2 vorgestellt.
Für diese Werkzeuge sind ein Web-Server mit PHP-Unterstützung sowie eine MySQL-Datenbank notwendig. Da sich die notwendigen Daten auf dem Gateway-Rechner befinden, werden diese Werkzeuge auch auf dem Gateway installiert.
Die Bezeichnungen für die Log-Einträge des Firewall-Skriptes werden für die Auswertung mit *iptables log* angepasst (siehe Kapitel 5.2.1.3).

5 Analyse des Honeynets

Neben der Konzeption und dem Betrieb eines Honeynets ist natürlich das Auswerten und die Analyse der Daten die zentrale Aufgabe.

5.1 Vorgehensweise

Im nachfolgenden Abschnitt wird auf die Vorgehensweise für die Analyse und Auswertung der Daten eingegangen. Dabei muss zwischen der Online-Analyse und der nachträglichen Analyse (Offline-Analyse) der gespeicherten Daten (Log-Dateien, Netzwerkdumps) unterschieden werden.

5.1.1 Online-Analyse

Wie in Kapitel 3.4.1 und 3.4.2 werden zwei Mittel zur Benachrichtigung bei Alarmen eingesetzt: SMS und E-Mail. Dies ist die Grundlage für weitere Erkenntnisse. Durch diese Benachrichtigung wird der Betreuer des Honeynet sofort über die Vorkommnisse informiert. SMS und E-Mail liefern jedoch nur einen Anhaltspunkt (es werden ja nur Einträge aus der Firewall Log-Datei versendet). Außerdem wird der Betreuer nicht 24 Stunden am Tag vor seinem PC verbringen und auf E-Mails warten bzw. diese sofort lesen.
Deshalb erfolgt als nächster Schritt die Analyse mit Hilfe von *iptables log* (siehe Kapitel 5.2.1). Es handelt sich dabei zwar auch nur um Einträge aus der Firewall Log-Datei, jedoch wird zum einen der zeitliche Ablauf dargestellt, zum anderen erfolgt eine DNS-Auflösung des anfragenden Rechners (falls dieser registriert ist). Durch eine selektive Abfrage kann bestimmt werden, welche Ports angefragt wurden. Was noch nicht bestimmt werden kann, ist, ob es sich bei dieser Anfrage um einen Angriff gemäß einer Signatur von *snort* gehandelt hat.
Dies kann im nächsten Schritt mit Hilfe von *ACID* (siehe Kapitel 5.2.2) herausgefunden werden. Handelte es sich bei der Verbindung um eine von *snort* bekannte Signatur, so wird dies angezeigt.

5.1.2 Offline-Analyse

Nach einem erfolgten Angriff ist es oft notwendig, die genaue Vorgehensweise des Angreifers zu erforschen. Dies ist vor allem dann notwendig, wenn im Angriff kein bisher bekanntes Muster erkennbar ist. Hier kommt nun das in Kapitel 5.2.4 beschriebene Werkzeug *ethereal* zum Einsatz. Damit können die einzelnen gesendeten und empfangenen IP-Pakete genau analysiert werden und somit neue Angriffsmuster erkannt werden.
Die Analyse der System Log-Dateien der einzelnen Honeypots spielt ebenfalls eine wichtige Rolle für die Rekonstruktion der erfolgten Angriffe. Hier erfolgt die Auswertung der gesammelten Daten auf dem Analyse-Server.

Wurde ein Honeypot-System erfolgreich kompromittiert, so sind auch die Kommandozeilen-Eingaben des Angreifers von Bedeutung. Die gesammelten Daten von *ComLog* und *Sebek* (siehe Kapitel 3.2.3.2) dienen als Grundlage für die Analyse.
Iptables log, *snort*, *p0f* (siehe Kapitel 5.2.5) erlauben auch ein erneutes nachträgliches Einspielen der Protokolldateien. Damit können auch die Daten auf einem leistungsfähigeren Rechner ausgewertet werden, sowie größere Datenmengen schnell analysiert werden.

5.1.3 Nützliche Informationsquellen für die Analyse

Neben den eigenen „Analyse-Fähigkeiten“ gibt es auch die Möglichkeit im Internet Hinweise über aktuelle Angriffe oder Bedrohungen zu erfahren. Nachstehend werden einige Informationsquellen kurz vorgestellt.

Mailing-Listen und Newsgroups
In Mailing-Listen werden Probleme diskutiert und versucht Lösungen, Erklärungen und/oder Antworten zu finden.
Auf der Homepage von *SecurityFocus*[23] gibt es die Möglichkeit, sich kostenlos in verschiedene Mailing-Listen unter anderem zu den Themen entdeckte Schwachstellen (*bugtraq*), Vorfälle (*incidents*) oder Honeypots (*honeypots*) einzuschreiben. Die Anzahl der empfangenen Mails schwankt stark. Auch sind nicht alle Beiträge für den Betreuer eines Honeynets interessant. Jede Woche erscheint auch ein Newsletter (jeweils für Windows™ und Linux) mit einer Zusammenfassung aller wichtigen Ereignisse der jeweils letzten Woche.

Hersteller von Antiviren-Software
Auch die Hersteller von Antiviren-Software können als Informationsquelle für Analysen verwendet werden. Beim Auftreten von neuen Viren oder Würmern werden auf deren Internet-Seiten Beschreibungen veröffentlicht. Als Beispiele sind hier die Unternehmen Symantec[24] und Sophos[25] zu nennen.

Sonstige Quellen
Neben den eben genannten Quellen existieren im Internet noch eine Vielzahl an Seiten, die über aktuelle Sicherheitslöcher, neue Vorfälle, etc. informieren. Exemplarisch werden einige kurz vorgestellt:

- **heise Security** [heis 03]: Eine spezielle Seite des heise-Verlages, die über alle aktuelle Bedrohungen und Schwachstellen informiert und Tipps zum sicheren Konfigurieren von Systemen gibt. Neben unterschiedlichen Foren werden auch verschiedene Werkzeuge vorgestellt.

- **BSI** [bsi 03]: Das Bundesamt für Sicherheit in der Informationstechnik (BSI) enthält interessante Informationen zum Thema Internet-Sicherheit.

[23] http://www.securityfocus.com
[24] http://securityresponse.symantec.com/avcenter/vinfodb.html#threat_list
[25] http://www.sophos.de/downloads/ide/

- **The CERT® Coordination Center** [cert 03a]: Umfassende Informationen über Sicherheitsprobleme im Internet. Die Sicherheitsprobleme (und deren Beseitungsmöglichekiten) werden in so genannten CERT® Advisories (z.B. CERT® Advisory CA-2003-20 W32/Blaster worm [certa20 03]) verfasst. Auch in Deutschland existieren diverse CERTs, z.B. RUS-CERT[26] oder DFN-CERT[27]

- **Internet Storm Center** [isc 03]: Informationen über die aktuellen Angriffe rund um den Globus. Dabei werden hauptsächlich Statistiken über die meist angegriffenen Ports geführt.

- **SANS** [sans 03]: Informationsquelle für aktuelle Sicherheitsprobleme.

5.2 Verwendete Analyse-Werkzeuge

Im Internet wird bereits eine Vielzahl an Programmen zur Auswertung von Netzwerkdumps, Firewall Logdateien, etc. zur Verfügung gestellt. Im folgenden Abschnitt werden die für die Diplomarbeit verwendeten Werkzeuge kurz vorgestellt. Für die Online-Analyse eignen sich webbasierte Werkzeuge, da mit diesen unabhängig von der Software-Umgebung des Betreuers, das Honeynet überwachen kann. Aus diesem Grund werden für die Online-Analyse die beiden Werkzeuge *Iptables log* und *ACID* auf dem Gateway-Rechner installiert. Beide Werkzeuge sind einfach zu bedienen und erfüllen den gewünschten Zweck. *Ethereal* bzw. *Packetyzer* werden auf einem zusätzlichen Rechner für die Offline-Analyse installiert.

5.2.1 *Iptables log*

Iptables log [ipt 02] ist ein Open-Source Tool für Linux auf Basis von Perl und PHP zur Online-Auswertung der Log-Einträge von *iptables* sowie Archivierung der Daten in einer MySQL-Datenbank. Zum Zeitpunkt der Bearbeitung der Diplomarbeit war die Beta-Version 0.4 verfügbar.

5.2.1.1 Funktionsweise

Ein Perl-Skript (`feed_db.pl`) überwacht mit `tail` die Log-Datei `/var/log/messages`. Handelt es sich dabei um einen *iptables*-Eintrag (dieser wird mit einem bestimmten Schlüsselwort erkannt), so wird der Eintrag geparst und die die MySQL-Datenbank geschrieben. Dabei erfolgt eine Einordnung in die definierten Ketten (*chains*) von *iptables*. Durch ein PHP-Skript können die aktuellen Einträge über einen Browser eingesehen werden. Abbildung 15 zeigt einen Screenshot der Oberfläche von *iptables log*. Auf der rechten Seite kann ein Filter für die einzelnen Ketten, eine Datumsauswahl sowie die Anzahl der anzuzeigenden Datensätze pro Seite gesetzt werden, im unteren Teil wird anhand der Daten in der Datenbank eine Top-Ten Statistik erstellt. Dazu zählen die *Top Hosts* (welche Hosts

[26] http://cert.uni-stuttgart.de/
[27] http://www.cert.dfn.de/

am meisten Anfragen gestellt haben), *Top Proto* (Verteilung der verwendeten Protokolle), *Top Ports* (welche Ports wurden am häufigsten angefragt) und *Top Domains*. *Top Ports* und *Top Domains* sind in der Abbildung nicht zu sehen.

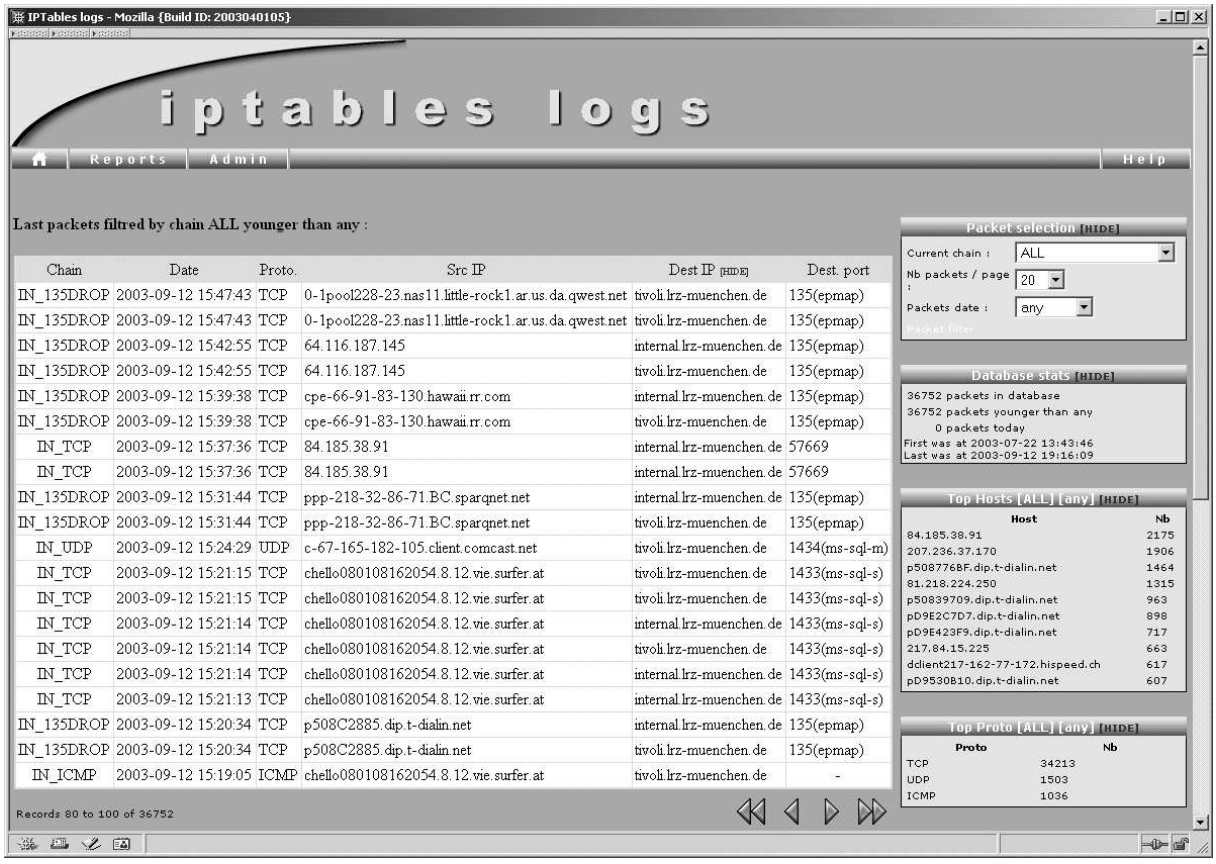

Abbildung 15: iptables log

5.2.1.2 Installation

System-Voraussetzungen

- *netfilter/iptables*-Firewall
- Web-Server (vorzugsweise *Apache*) mit PHP-Modul
- MySQL-Datenbank
- Perl-Interpreter mit MySQL-Unterstützung

Installationsvorgang

1. Quellen entpacken

2. Datenbank vorbereiten
   a) Einloggen auf dem Datenbank-Server (in diesem Fall als Benutzer root)
   ```
   > mysql –u root –p
   ```

b) Neue Datenbank für die Einträge erzeugen (in diesem Fall iptables)
```
mysql> create database iptables;
```
c) Benutzer und zugehörige Rechte setzen (Benutzer: „iptables_admin“, Passwort: „*<passwort>*“)
```
mysql> grant create,select,insert on iptables.* to↵
iptables_admin@localhost identified by '<passwort>';
```
d) Datenbank-Client beenden und Datenbank vorbereiten (Datenbankskript `db.sql` ist im Archiv enthalten)
```
> cat db.sql | mysql -u iptables_admin -p iptables
```

3. Web-Server vorbereiten: Datenbank und Web-Server müssen dabei nicht notwendigerweise auf demselben System installiert werden. Es ist allerdings aus Sicherheits- und Performancegründen empfehlenswert beide Komponenten auf einem System zu installieren.
   a) Kopieren des Verzeichnisses `web` (ebenfalls im Archiv enthalten) in ein Unterverzeichnis des `DocumentRoot` des Web-Servers (z.B. iptables)
   ```
   > cp -R web /srv/www/htdocs/iptables
   ```
   b) Anpassen der Variablen in der Konfigurations-Datei `config.php`. Das Passwort wird im Klartext gespeichert.
   ```
   $db_host="localhost";
   $db_user="iptables_admin";
   $db_password="<passwort>";
   $db_name="iptables";
   ```

4. Skript (`feed_db.pl`) für die Datenbank anpassen und starten
   a) Setzen der Werte für die Variablen
   ```
   my $dsn = "DBI:mysql:iptables:localhost";
   ```
   (DSN-Name für Perl bestehend aus Perl-Schnittstelle für Datenbanken (DBI), Datenbanktyp, Datenbankname und Rechner, auf dem die Datenbank installiert ist)
   ```
   my $db_user_name = "iptables_admin";
   my $db_password = "<passwort>";
   my $log_file = "/var/log/messages";
   ```
   b) Startskript `iptablelog` nach `/etc/init.d/` kopieren und starten

5. Die Startseite kann nun aufgerufen werden.

5.2.1.3 Vorgenommene Veränderungen

Wie in Kapitel 5.2.1.1 beschrieben, verwendet *iptables log* ein bestimmtes Schlüsselwort zur Erkennung von *iptables* Log-Einträgen (Variable `$log_tag` in `feed_db.pl`). Dieses wurde an die Honeynet Umgebung angepasst: Von „*IPTABLES*“ in „*HW*“ („*HW*“ steht dabei für Honeywall). Ebenso wurde das Firewall-Skript angepasst: Alle „*LOG*-Regeln“ erhalten den Präfix „*HW*“ gefolgt von einem Ausdruck ohne Leerzeichen (z.B. „*IN_TCP*“). Dieser Ausdruck

wird von *iptables log* als Ketten-Name (*chain*) interpretiert. Somit ist eine Auswertung nach den unterschiedlichen Ketten einfach realisiert.

5.2.1.4 Fazit

Iptables log eignet sich gut, um einen groben Überblick über die Vorgänge im Honeynet zu bekommen. Leider wird zu Zeit an diesem Programm nicht weiterentwickelt, deshalb sind viele Funktionen und Erweiterungen vom Entwickler geplant, aber noch nicht implementiert. Von Richard Stevens[28] wurde speziell für die Firewall-Log Dateien der Project Honeynet Firewall ein Werkzeug erstellt, das eine grafische Auswertung ermöglicht. Leider ist dieses Werkzeug nur zur Offline-Analyse geeignet, d.h. es müssen die Daten erst eingelesen werden. Ein weiteres Werkzeug zur Auswertung von Firewall-Log Dateien ist *fw-analog*[29]: *iptables log* war aber für die Anforderungen ausreichend. Ansonsten wurden keine weiteren Alternativen gefunden. Mit wachsender Größe der Datenbank verzögert sich der Aufbau der Seiten. Dies ist aber auf die Rechenleistung, Geschwindigkeit der eingesetzten Festplatte als auch auf die Größe des Arbeitsspeichers des verwendeten Rechners zurückzuführen. Die Statistiken helfen vor allem bei der Wiedererkennung von Angreifern.

5.2.2 *Snort* in Verbindung mit *ACID*

ACID (**A**nalysis **C**onsole for **I**ntrusion **D**atabases) [aci 03] ist ein webbasiertes Werkzeug zur Analyse der erzeugten Alarme von *snort*. Wie bei *iptables log* basiert *ACID* auf der Skriptsprache PHP sowie auf einer Datenbank (vorzugsweise MySQL). Im Rahmen der Diplomarbeit wurde die Version v0.9.6b23 eingesetzt.
Folgende Funktionen sind realisiert:

- **Erstellung benutzerdefinierter Anfragen**: *ACID* erlaubt Anfragen anhand verschiedener Kriterien. Dabei kann es sich um Meta-Informationen (z.B. Name der Signatur, Zeitraum) als auch um TCP/IP-Informationen (z.B. Quell- und/oder Ziel-Adresse, Quell- und/oder Ziel-Port, gesetzte TCP-Flags) handeln.

- **Anzeige des Payloades durch Paket Decoder**: Der Payload der gespeicherten Alarme kann decodiert und ausgegeben werden.

- **Management der erzeugten Alarme**: Die in der Datenbank gespeicherten Alarme können logisch gruppiert, gelöscht oder in eine Archiv-Datenbank verschoben werden. Eine E-Mail-Benachrichtigung ist ebenfalls möglich.

- **Diagramm-Erstellung**: Es können Diagramme anhand verschiedenere Kriterien erstellt werden.

Abbildung 16 zeigt die Startseite von *ACID*. Im oberen Teil wird die globale Statistik für alle in der Datenbank gespeicherten Alarme angezeigt. Dabei wird zwischen den Protokollen TCP,

[28] http://richardstevens.de/logreader/
[29] http://tud.at/programm/fwanalog/

UDP und ICMP unterschieden. Im unteren Teil werden bereits vordefinierte Anfragen zur Auswahl angeboten.

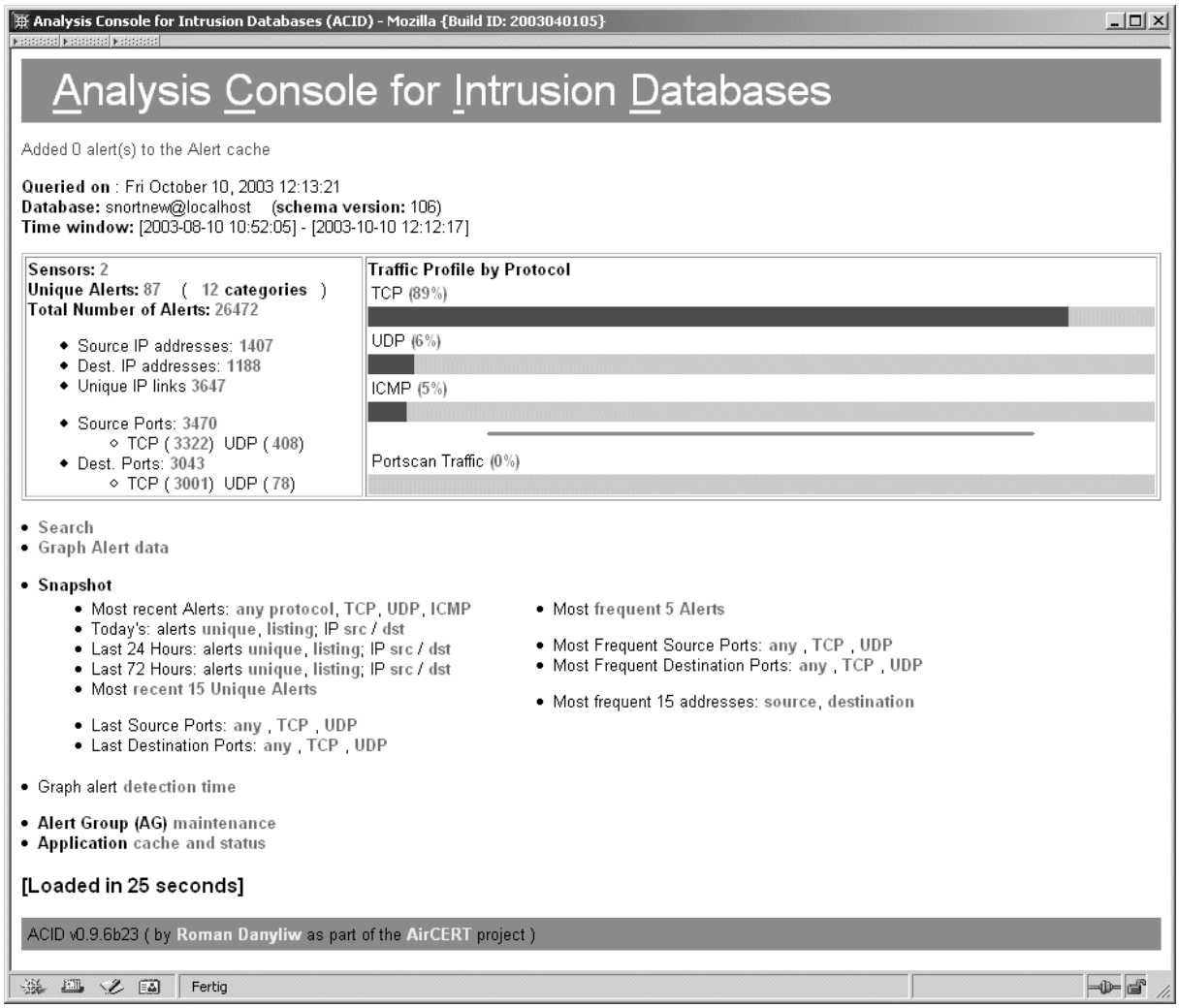

Abbildung 16: Startseite von ACID während des Betriebes

5.2.2.1 Installation

System-Voraussetzungen

- Web-Server (vorzugsweise *Apache*) mit PHP-Modul (PHP Version 4.3 oder neuer)
- Datenbank (vorzugsweise MySQL) für die Speicherung und Auswertung der Alarme
- *snort*

Notwendige Zusatzprogramme

1. *ADOdb* [ado 03]: *ADOdb* (**A**ctive **D**ata **O**bjects **D**ata **B**ase) ist eine Sammlung von PHP-Methoden für einen „standardisierten“ Zugriff auf verschiedene Datenbanken (z.B. MySQL, PostgreSQL, Interbase, Oracle, Microsoft SQL Server). Dabei können die

zugrunde liegenden Datenbanken ohne großen Aufwand gewechselt werden; die ursprünglichen Anfragen bleiben bestehen. Dieses Produkt ist Open-Source-Software.

2. *JpGraph* [jpg 03]: *JpGraph* ist eine Klassenbibliothek für PHP, die auf der *GD*[30] Bibliothek aufsetzt. Diese Klassenbibliothek ist unter der QPL 1.0[31] veröffentlicht: Für nicht kommerzielle, Open-Source oder Forschungszwecke ist die Nutzung kostenlos. Mit Hilfe dieser Bibliothek lassen sich einfach Diagramme (z.B. Balkendiagramme) als Grafiken in Echtzeit erzeugen. Falls von *ACID* keine Grafiken erzeugt werden sollen ist diese Bibliothek nicht notwendig.

Installationsvorgang

1. Vorbereitung des Web-Servers
   a) Das *ADOdb*-Programmpaket in ein Unterverzeichnis (z.B. `adodb`) unterhalb des `DocumentRoot` des Web-Servers entpacken.
   b) Das *JpGraph*-Programmpaket in ein Unterverzeichnis (z.B. `jpgraph`) unterhalb des `DocumentRoot` des Web-Servers entpacken.
   c) Es sind keine weiteren Konfigurationen an diesen Paketen mehr nötig.

2. Vorbereitung von *snort*
   Damit *ACID* Alarme aus der Datenbank anzeigen kann, muss *snort* die Alarme in einer Datenbank speichern.
   a) Quellen von *snort* entpacken
   b) Vorbereiten der *snort*-Datenbank
      - Einloggen auf dem Datenbank-Server (in diesem Fall als Benutzer root).
        `> mysql -u root -p`
      - Neue Datenbank für die Einträge erzeugen (in diesem Fall snort).
        `mysql> create database snort;`
      - Datenbank-Client beenden und Datenbank vorbereiten (Skript *create_mysql* im Programmpaket enthalten).
        `> cat ./contrib/create_mysql | mysql -u root -p snort`
      - Zusatz-Tabellen für *snort* installieren (Archiv *snortdb-extra.gz* im Programmpaket enthalten).
        `> zcat ./contrib/snortdb-extra.gz | mysql -u root -p snort`
   c) In der Konfigurationsdatei von *snort* (`snort.conf`) muss das „database logging“ aktiviert werden
      `output database: log, mysql, user=root password=<passwort>↵`
      `dbname=snort host=localhost detail=full`

3. Installation von *ACID*: Datenbank und Web-Server müssen dabei nicht notwendigerweise auf demselben System installiert werden. Es ist allerdings aus Sicherheits- und Performancegründen empfehlenswert beide Komponenten auf einem System zu installieren.
   a) Das *ACID*-Programmpaket in ein Unterverzeichnis (z.B. `acid`) unterhalb des `DocumentRoot` des Web-Servers entpacken.

[30] GD ist eine ANSI C Bibliothek zum dynamischen Erzeugen von Grafiken (z.B. PNG oder JPEG Grafiken), http://www.boutell.com/gd/
[31] Qt Free Licensee, http://www.trolltech.com/licenses/qpl.html

b) Anpassung in der Konfigurationsdatei `acid_conf.php`. Das Passwort wird im Klartext gespeichert.
   - Pfad für *ADOdb*: `$DBlib_path = "/srv/www/htdocs/adodb";`
   - Datenbank-Typ: `$DBtype = "mysql";`
   - Datenbank-Name: `$alert_dbname = "snort";`
   - Datenbank-Server: `$alert_host = "localhost";`
   - Benutzerkennung: `$alert_user = "root";`
   - Benutzerpasswort: `$alert_password = "<passwort>";`
   - Pfad für *JpGraph*: `$ChartLib_path = "/srv/www/htdocs/jpgraph/src";`
   - Maximale Ausführungszeit der Skripte von *ACID* (von 180 Sekunden auf 1800 Sekunden erhöht, da manche Anfragen, vor allem das Löschen von Alarmen, deutlich längere Zeit benötigen, abhängig vom verwendeten Rechner): `$max_script_runtime = 1800;`

   Falls eine Archiv-Datenbank verwendet wird, sind die ebenfalls die Variablen für die Archiv-Datenbank analog zu setzen.

c) Erstmaliges Starten von *ACID*
   Wird die Startseite zum ersten Mal aufgerufen, so müssen noch Ergänzungen in der Datenbank vorgenommen werden. Durch Klicken auf den Link „*Setup page*“ sowie auf den Button „*Create ACID AG*“ werden die Änderungen ausgeführt.

5.2.2.2 Fazit

Snort in Verbindung mit *ACID* eignet sich gut, um während des Betriebes bereits einen Überblick zu bekommen, welche Vorgänge sich im Honeynet abspielen. Durch eine Verfeinerung bzw. Ergänzung der *snort* Alarmregeln (siehe 4.5.1) kann man schnell die Angriffe kategorisieren. Die Installation verläuft ohne Probleme. *ACID* erlaubt auch das Decodieren des Inhalts der TCP/IP-Pakete. Dadurch lassen sich bereits einfache Angriffe bzw. Angriffsmuster erkennen. Leider wachsen der Datenbestand und somit auch die Datenbank während des Betriebes kontinuierlich an. Dadurch benötigen die Anfragen und die Ausgabe der Ergebnisse deutlich mehr Zeit. Die Bedienung von *ACID* ist etwas gewöhnungsbedürftig, da teilweise noch Fehler enthalten sind (es handelt sich um eine Betaversion), aber nach einer kurzen Einarbeitungsphase ist die Bedienung kein Problem mehr; man hat sich an die Eigenheiten gewöhnt: Nach dem Löschen von Alarmen können nicht unmittelbar im Anschluss wieder Alarme gelöscht werden. Durch den erneuten Abruf der Startseite von *ACID* kann dieses Problem umgangen werden. Die Hardware des verwendeten Rechners spielt eine große Rolle bei der Arbeit mit *ACID*. Hier sollte ein leistungsfähiger Rechner der neueren Generation mit genügend Hauptspeicher verwendet werden.

5.2.3 Sicherheitsaspekte bei webbasierten Analyse- und Auswertungswerkzeugen

Wie bereits in Kapitel 3.5 beschrieben, ist der Gateway-Rechner und der Analyse-Rechner besonders gegen Angriffe geschützt worden. Nichtsdestotrotz werden die HTML-Seiten über das Internet unverschlüsselt übertragen und der Verkehr könnte somit abgehört werden und eventuell in falsche Hände geraten, denn es soll der Angreifer z.B. nicht merken, dass alle seine Aktivitäten überwacht werden.

Für eine sichere Übertragung gibt es mehrere Möglichkeiten:

- **Web-Server mit SSL/TLS-gesicherter Übertragung**: Die Secure Socket Layer (SSL) befindet sich nach dem ISO/OSI-Modell zwischen der Anwendungs- und Transportschicht und wurde ursprünglich von der Firma Netscape im Jahre 1994 als proprietäre Lösung für den sicheren Austausch zwischen Web-Server und Client entwickelt. Für die Normierung diente SSL bereits 1995 als Grundlage der Transport Layer Security (TLS) Working Group der Internet Engineering Task Force (IETF) [HROG 03, RFC 2246]. Mit Hilfe von kryptographischen Verfahren werden die Daten der Anwendung verschlüsselt und gesichert übertragen. Für den *Apache* Web-Server ist diese Funktionalität im Modul mod_ssl[32] implementiert.

- **Tunneling des HTTP-Verkehrs**: Mit Hilfe eines SSH-Tunnels können verschiedene Protokolle (z.B. HTTP) verschlüsselt übertragen werden. Dazu muss zum Zielsystem eine SSH-Verbindung bestehen. Wie bereits in Kapitel 4.3.1 beschrieben, sind durch die Konfiguration der Firewall nur Zugriffe von vier Rechnern erlaubt. Für den Zugriff über einen Dialin-Zugang (z.B. ISDN oder T-DSL) mit jeweils neuer IP-Adresse bei der Einwahl ist diese Lösung nicht geeignet. Abbildung 17 zeigt beispielhaft eine Lösung für dieses Problem: Der Betreuer des Honeynets baut eine Verbindung zu einem Rechner mit fester IP-Adresse auf (*athalle5.informatik.tu-muenchen.de*). Zusätzlich wird ein Port-Forwarding definiert. Wird auf dem lokalen Rechner der Port 9005 angesprochen, so wird dieser über die SSH-Verbindung zum Rechner *gway.lrz-muenchen.de* (*129.187.18.202*) auf den Port 9004 umgeleitet. Auf diesem Port ist der Web-Server installiert. Auf den Webserver des *gway.lrz-muenchen.de* kann nun in einem Webbrowser über *http://localhost:9005/* zugegriffen werden, dabei ist die Verbindung zwischen dem lokalen Rechner und dem Rechner *athalle5.informatik.tu-muenchen.de* verschlüsselt, zwischen diesem Rechner und dem Gateway-Rechner wird der Verkehr unverschlüsselt übertragen. Wird der Zwischenrechner nahe an dem Gateway-Rechner platziert, so kann das Sicherheitsrisiko minimiert werden.

[32] http://www.modssl.org/

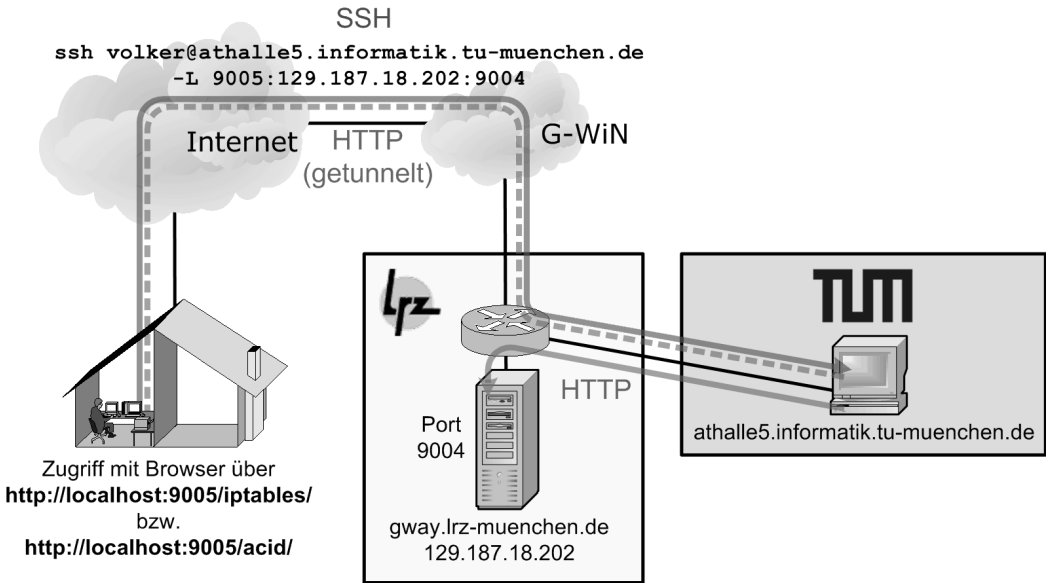

Abbildung 17: Einsatz eines SSH-Tunnels zur Online-Analyse mit Hilfe von webbasierten Werkzeugen (sehr vereinfachte Darstellung)

Zusätzlich kann auch der Port des Web-Servers vom Standard-Port 80 auf einen anderen Port verändert werden. Damit ist dieser Server nicht so leicht zu entdecken.

Für den Einsatz in dieser Diplomarbeit wurde die Möglichkeit mit dem Tunneling realisiert, sowie der Port des Web-Servers auf den Port 9004[33] abgeändert. Ausschlaggebend hierfür war die Beschränkung der Rechner, die Zugriff zum Gateway-Rechner haben.

5.2.4 *Ethereal* und *Packetyzer*

Ethereal [eth 03] ist ein frei verfügbarer Netzwerk Protokoll Analysator für UNIX und Windows™ Betriebssysteme mit grafischer Benutzeroberfläche. Mit Hilfe dieses Programms lässt sich eine Vielzahl von Protokollen untersuchen. Abbildung 18 zeigt die Benutzeroberfläche von *ethereal*. Im oberen Fenster ist der zeitliche Verlauf der aufgezeichneten Verbindungen zu sehen. Hier werden nur Informationen über die Uhrzeit, Quell- und Zieladresse, das verwendete Protokoll sowie keine kurze Charakterisierung angezeigt. Nach dem Markieren eines Eintrages werden im mittleren Fenster die Einzelinformationen des Frames aufgelistet. Dabei wird der Frame in die einzelnen Schichten gemäß dem ISO/OSI-Modell zerlegt (anhand des Beispiels erläutert):

- *Ethernet II* (Schicht 2): Quell- bzw. Ziel-MAC-Adresse, Hersteller der verwendeten Netzwerkkarte

- *Internet Protocol* (Schicht 3): Informationen des IP-Protokolls: z.B. Quell- und Ziel-Adresse, gesetzte Flags, Länge, TTL-Wert

[33] Unbenutzer Port gemäß [ian 03]

- *Transmission Control Protocol* (Schicht 4): Informationen des TCP-Protokolls: Quell- und Ziel-Port, Sequenznummer, Flags und Optionen

Im unteren Fenster wird der Ethernet-Frame als Datenpaket in hexadezimaler Notation angezeigt.

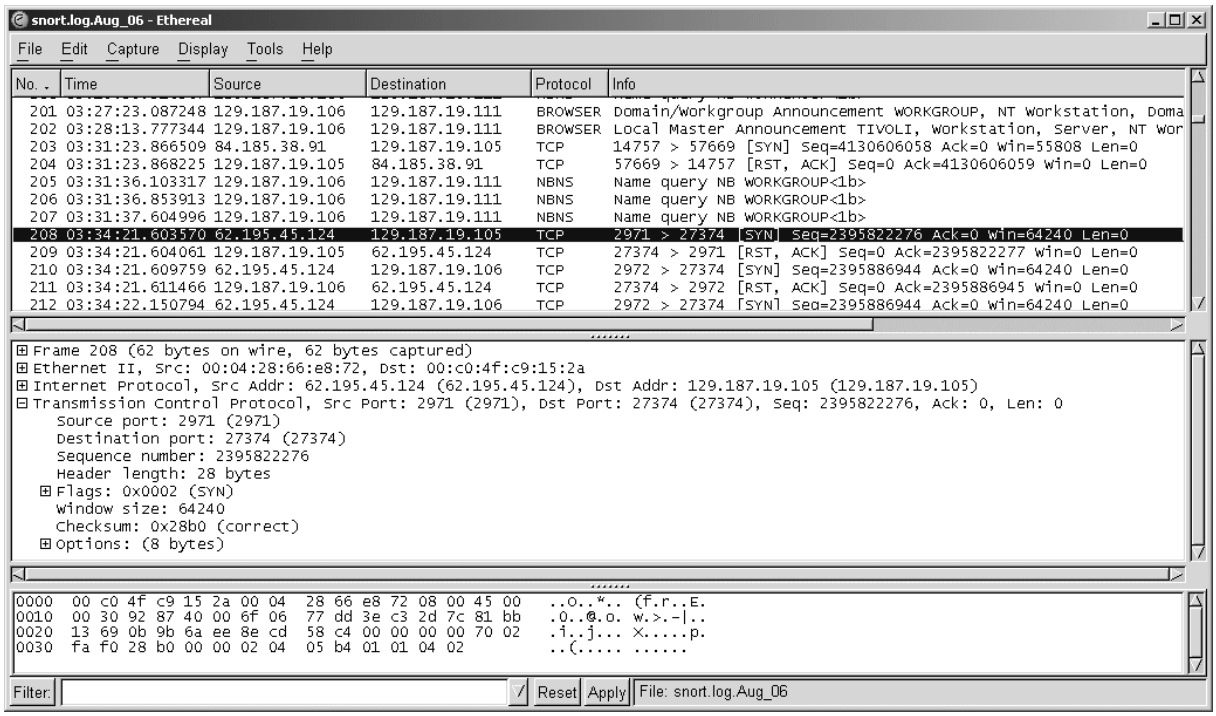

Abbildung 18: Benutzeroberfläche von *ethereal*

Das Programm ist Open-Source und unter GNU[34] GPL veröffentlicht. Zum Zeitpunkt der Bearbeitung der Diplomarbeit war die Version 0.9.15 aktuell.

Die in Abbildung 18 gezeigte GTK-Oberfläche ist für Windows™-Systeme ungewohnt. Die Firma Network Chemistry[35] bietet mit dem Programm *Packetyzer* [pac 03] ein Windows™ konformes Aussehen sowie eine erweiterte Funktionalität von *ethereal*. Es ist z.B. möglich in den Paket-Inhalten entweder nach ASCII oder Byte-Mustern zu suchen. Die Bedienung wurde zum Teil wesentlich vereinfacht. Abbildung 19 zeigt die Benutzeroberfläche des Programms, die Dreiteilung ist wie auch bei *ethereal* vorhanden. Dabei werden die einzelnen Protokolle unterschiedlich gefärbt.

[34] http://www.gnu.org
[35] http://www.networkchemistry.com

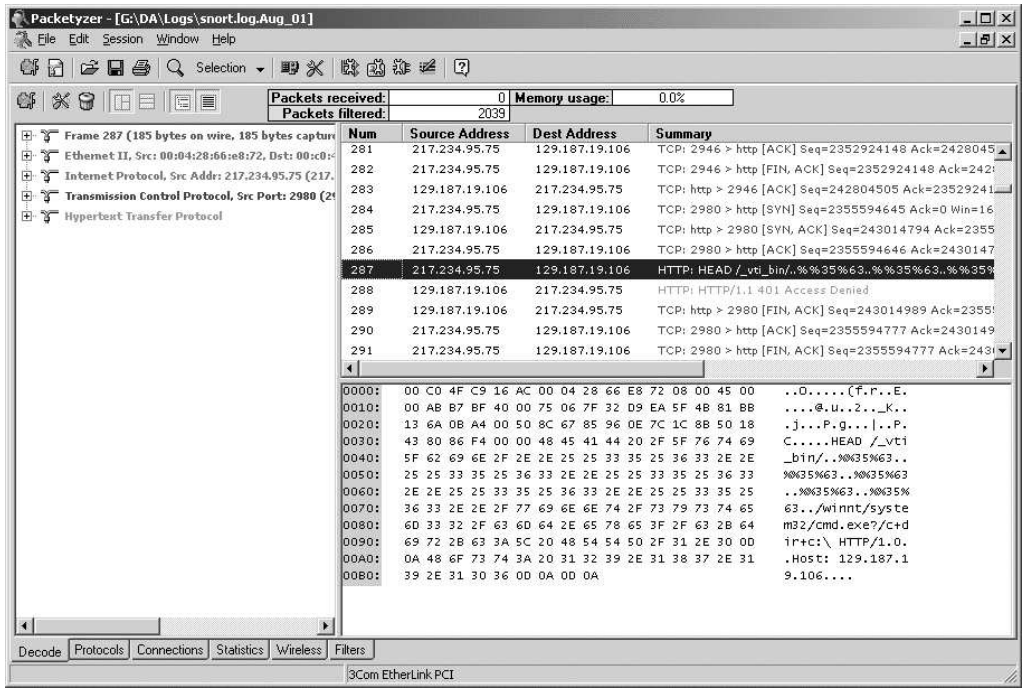

Abbildung 19: Benutzeroberfläche von *Packetyzer*

5.2.4.1 Installation

In den meisten Linux-Distributionen ist *ethereal* bereits enthalten. Für Windows™ Betriebssysteme kann das Programm auf der Homepage heruntergeladen werden. Mit Hilfe einer Setup-Routine lässt sich das Programm leicht installieren. Für das Programm *Packetyzer* steht ebenfalls auf der Homepage ein Setup-Programm zum Download zur Verfügung.

5.2.4.2 Fazit

Für die Online-Analyse ist *ethereal* nicht besonders gut geeignet, da auf dem Gateway-Rechner keine X-Oberfläche installiert ist und deshalb das GUI-Fenster auf den Ziel-Rechner umgeleitet werden müsste. Die Stärken von *ethereal* liegen in der genauen Aufspaltung der gespeicherten Frames. Dadurch ist das Programm für die genaue Untersuchung des Ablaufes eines Angriffes bzw. von Ereignissen zweckdienlich. Es setzt gewisse Grundkenntnisse im Bereich der Protokolle voraus. Das Einlesen und Verarbeiten von einer großen Datenmenge (größer 50 MB) nimmt eine gewisse Zeit in Anspruch, daher ist ein leistungsfähiger Rechner für die Analyse zu empfehlen.
Für Windows-Systeme ist das Programm *Packetyzer* besser geeignet: Die Bedienung fällt damit deutlich leichter: Die Erstellung von Filtern ist einfach. Leider befindet sich dieses Programm noch in einem frühen Entwicklungsstadium: Ab und zu kommt es zu Abstürzen. Das Einlesen von großen Dateien kann, wie auch bei *ethereal*, etwas länger dauern.

5.2.5 *P0f*

Alle bisher vorgestellten Werkzeuge beschränken sich auf das Auswerten und Analysieren der Daten, die den Gateway-Rechner passiert haben. *P0f*[36] (sprich: pof) [p0f 03] ist ein Programm mit dem sich auch Informationen über den Rechner des Angreifers herausfinden lassen. Dazu wird die Technik des **Passive Fingerprinting** benützt. Anhand des Aufbaus der am Gateway durchlaufenen Pakete kann festgestellt werden, welches Betriebssystem (und welche Version) der Angreifer benützt. Dazu werden die einzelnen Pakete mit gespeicherten Signaturen verglichen. Als Grundlage für Passive Fingerprinting dienen folgende Werte in einem TCP/IP-Paket (vgl. [HP 02]):

- **TTL** (Time-to-Live): Der Time-to-Live Wert unterscheidet sich bei verschiedenen Betriebssystemen (seit Windows™ NT: 128, Linux/FreeBSD: 64 oder 255)

- **WindowSize**: Die WindowSize kann ebenfalls Aufschluss über das verwendete Betriebssystem geben (Linux: feste Größe (32120), Windows™: variable Größe)

- **Don't Fragment Bit**: Bei vielen Betriebssystemen wird dieses Bit gesetzt, aber z.B. bei OpenBSD nicht.

- **TOS** (Type-of-Service): Ist dieser Wert gesetzt und falls ja, auf welchen?

P0f identifiziert dabei den Angreifer anhand folgender Verbindungsrichtungen:

- Rechner, die zu einem Honeypot eine Verbindung aufbauen wollen („SYN-Modus")

- Rechner, zu denen eine Verbindungsaufbau erfolgt („SYN-ACK-Modus")

- Rechner, zu denen keine Verbindung aufgebaut werden kann („RST(+ACK)-Modus")

Außerdem unterstützt *p0f* noch einige zusätzliche Funktionen:

- Erkennung von Firewalls und Masquerading (z.B. Verwendung eines Routers). *P0f* arbeitet dabei mit einer Heuristik. Werden mehrere verschiedene Betriebssysteme mit einer einzigen IP-Adresse registriert, so kann angenommen werden, dass diese Rechner z.B. über einen gemeinsamen NAT-Router sich den Internetzugang teilen. Eine genaue Beschreibung der verwendeten Heuristik ist in [pof 03] beschrieben.

- Bestimmung der Entfernung zum Angriffssystem (Hops: TTL_{OS} - TTL_{Paket}) und die Betriebszeit. Bei vielen UNIX-Betriebssystemen wird im TCP-Header ein Zeitstempel übermittelt, aus dem die Betriebszeit berechnet werden kann[37].

- Art der verwendeten Verbindung (z.B. DSL) und der Internet Service Provider (ISP) des Angreifers (durch den gesetzten MTU-Wert).

[36] POF: **P**assive **O**perating System **F**ingerprinting. Der Autor dieser Software verwendet anstatt des Buchstaben „O" die Zahl „0".
[37] http://www.securiteam.com/securitynews/5NP0C153PI.html

Im Gegensatz zum Active Fingerprinting, wo gezielte Anfragen an ein System gestellt werden müssen (z.B. mit *nmap*[38]) ist dies beim Passive Fingerprinting nicht notwendig. Der Angreifer bemerkt also die „Identifizierung“ seines Rechners nicht.
P0f ist ein kommandozeilen-basiertes Werkzeug, das für viele Betriebssysteme frei zur Verfügung steht. Das Programm kann während des Betriebes zur Online-Überwachung eingesetzt werden, aber es ist auch möglich gespeicherte Netzwerkdumps wieder einzulesen. Ausführliche Informationen über die verschiedenen Parameter von *p0f* erhält man durch den Aufruf von `./p0f -h`. Während der Bearbeitung der Diplomarbeit stand die Version 2.0.2 zur Verfügung.
Abbildung 20 zeigt die Ausgabe beim Lesen eines gespeicherten Netzwerkdumps im *tcpdump*-Format. Die Ausgabe beinhaltet Quell- und Ziel-Adresse, Quell- und Ziel-Port, sowie das ermittelte Betriebssystem und die ermittelten Hops.

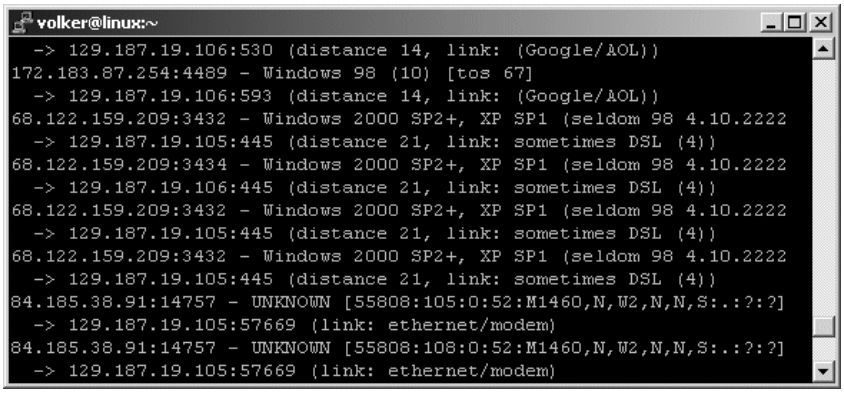

Abbildung 20: Beispiel-Ausgabe von *p0f*

5.2.5.1 Installation

Für Linux stehen die Quellen auf der Homepage zur Verfügung, für Windows™ existiert eine bereits vorkompilierte Version. Für Windows™ gibt es keine System-Voraussetzungen.
Voraussetzungen für Linux:

- *libpcap* 0.4 oder neuer
- GNU *cc* 2.7.x oder neuer
- GNU *make* 3.7.x oder neuer
- GNU *bash/awk/grep/sed/textutils* (nur für *p0frep* benötigt)

Die Linux-Version muss entpackt und mit `make` übersetzt werden, die Windows™-Version muss lediglich in ein beliebiges Verzeichnis extrahiert werden.

[38] *nmap* ist ein Portscanner, der auch Active Fingerprinting benützt, http://www.insecure.org/nmap/

5.2.5.2 Fazit

Mit *p0f* lassen sich die Informationen über den Rechner des Angreifers gewinnen. Dabei ist zu beobachten, dass der Angriff durchaus über einen Proxy erfolgen kann. Es ist geplant, dass es in zukünftigen Versionen auch die Möglichkeit gibt, die Daten in einer Datenbank zu speichern. Das Programm befindet sich noch in einem frühen Entwicklungsstadium, kann aber z.B. für statistische Zwecke oder zur Erkennung „wiederkehrender Besucher“ benutzt werden.

5.2.6 *traceroute* und *VisualRoute*

Mit dem in Kapitel 5.2.5 vorgestellten Werkzeug *p0f* können Rückschlüsse auf den Angreifer gezogen werden, aber leider nicht über seine Herkunft bzw. von welchem Ort er seinen Angriff gestartet hat. Das liegt daran, dass die IP-Adresse keinerlei Auskunft über den Standort des Rechners gibt. Mit *traceroute* [tra 99] (für UNIX) bzw. *tracert* (für Windows™) lässt sich der Weg zum Zielhost zurückverfolgen und dadurch evtl. Erkenntnisse über den Standort des Angreifers, unter der Annahme, dass diese IP-Adresse nicht gefälscht ist, herausfinden. Folgendes Beispiel zeigt eine kurze Analyse anhand einer T-Online DSL IP-Adresse:

```
:> traceroute pD9E423F9.dip.t-dialin.net
traceroute to pD9E423F9.dip.t-dialin.net (217.228.35.249), 30 hops
max, 40 byte packets
[...]
10  KR-EB1.KR.DE.net.dtag.de (62.154.121.38)  21 ms  20 ms  20 ms
```

Der letzte angezeigte Hop ist der Einwahlknoten des DSL-Kunden. Die Bezeichnung verrät etwas über den Standort dieses Transitsystems. Man könnte vermuten, dass „KR“ ein Autokennzeichen ist. Mit Hilfe einer Datenbank für Autokennzeichen[39] stellt man fest, dass sich der Einwahlknoten des Angreifers wahrscheinlich in Krefeld (Kennzeichen KR) befindet. Bei einem Gespräch mit einer T-Systems Mitarbeiterin bestätigte sich diese Vermutung.

Mit Hilfe von *VisualRoute* [vis 03] ist ebenfalls eine Lokalisierung der IP-Adresse möglich. Abbildung 21 zeigt die Oberfläche von *VisualRoute*. Im Feld „Adresse“ wird der Name oder die IP-Adresse des gefragten Rechners eingegeben (in diesem Fall *http://www.mit.edu*). Anschließend erfolgt die Bestimmung des Weges, die Ausgabe erfolgt zum einen grafisch, anhand einer Weltkarte mit Zoom-Funktion, zum anderen als Liste (unterer Teil in der Abbildung). In der Liste werden verschiedene Informationen unter anderem die Bezeichnung der Zwischenstationen, die geographische Lage oder die Netzwerk-Bezeichnung ausgegeben. Die geographische Lage und die Netzwerk-Bezeichnung werden mit Hilfe von `whois`-Abfragen bestimmt.

[39] z.B. http://www.autokennzeichen.info/

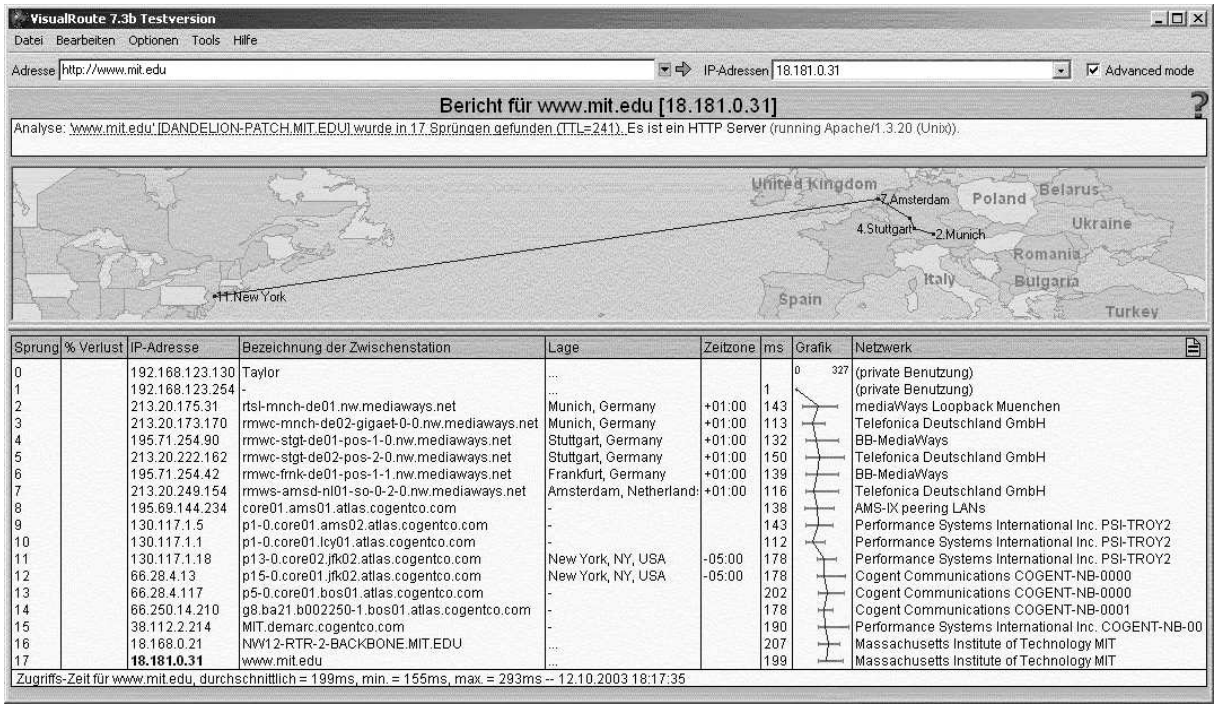

Abbildung 21: Benutzeroberfläche von *VisualRoute 7.3b* unter Windows™

5.2.6.1 Installation

Als Voraussetzung für *VisualRoute* muss eine Java-Laufzeitumgebung installiert sein. Auf der Homepage kann eine 15-tägige Testversion heruntergeladen werden. Dabei stehen Versionen für Windows™- und Mac/UNIX-Betriebssysteme zur Verfügung. Das Programm wird mit Hilfe einer Setup-Routine installiert.
Alternativ dazu kann ein *VisualRoute*-Server im Internet als Informationsquelle dienen. Im Unterschied zur Programm-Version ist der Startrechner fest vorgegeben (Rechner, der diesen Dienst zur Verfügung stellt). Unter z.B. folgenden Adressen kann der *VisualRoute*-Dienst in Anspruch genommen werden (die Liste ist nur eine Auswahl):

- *ViusalRoute* Server Frankfurt/Deutschland (*http://www.webhits.de/visualroute*)
- *VisualRoute* Server Bern/Schweiz (*http://visualroute.bboxbbs.ch*)
- *VisualRoute* Server Surrey/England (*http://visualroute.visualware.co.uk*)

5.2.6.2 Fazit

Visualroute dient als Ergänzung zu den bisher vorgestellten Werkzeugen. Für statistische Zwecke kann es sicher interessant sein, aus welchen Ländern die einzelnen Angriffe gestartet wurden. Auch hier, wie auch bei *p0f*, dass die IP-Adressen durchaus gefälscht sein können oder der Angreifer einen Proxy verwendet hat. Der im Internet auf den *VisualRoute*-Servern angebotene Dienst reicht für diese Zwecke vollkommen aus.

5.2.7 Zusammenfassung

In Tabelle 4 ist eine Zusammenfassung der beiden Analysearten und der verschiedenen Werkzeuge und ihre Bewertung dargestellt.

Online-Analyse			Offline-Analyse		
Werkzeug	**Informationsgewinn**	**Zeitaufwand[40]**	**Werkzeug**	**Informationsgewinn**	**Zeitaufwand[40]**
SMS	sehr gering	gering	*ethereal, Packetyzer*	hoch – sehr hoch	hoch
E-Mail	gering	gering	Internet	gering – sehr hoch	hoch
p0f, *traceroute*, *VisualRoute*	gering – mäßig	gering	„Handarbeit“	gering – sehr hoch	sehr hoch
iptables log	mäßig	mäßig			
ACID	hoch	mäßig			

Tabelle 4: Zusammenfassung und Bewertung der Analyse-Werkzeuge

Da sich die verwendeten Werkzeuge noch fast alle im Beta-Stadium befinden, kann nicht ausgeschlossen werden, dass durch die Verwendung evtl. zusätzliche Sicherheitslücken entstehen könnten.

5.3 Entwicklung von zusätzlichen Werkzeugen

Leider existiert keine so genannte „All-in-One“-Lösung zur Auswertung aller gesammelten Daten für beliebige Honeynets. Allein durch die Vielfalt an Möglichkeiten ein Honeynet zu betreiben, ist es nahezu unmöglich mit einem einzigen Werkzeug alle Daten auf einmal auszuwerten.
Eine integrierte Lösung für Honeynets wurde unter dem Namen *Honeybread* bereits in [BP 02] vorgestellt. Bei dieser Lösung wird das Honeynet auf einem einzigen Rechner implementiert, die Honeypots sind als virtuelle Rechner mit Hilfe von *VMWare*[41] realisiert. Die Management-Oberfläche ist webbasiert implementiert und sollte in Kürze[42] auf der Homepage[43] des Autors zur Verfügung gestellt werden.
Für ein Honeynet in einer verteilten Umgebung bzw. mit physischen Honeypots kann man sich dennoch überlegen, bestimmte Auswerte-Aufgaben in einem Analyse-Werkzeug zusammenzufassen. Die Hauptaufgabe liegt darin, alle protokollierten Daten auf einem System zu sammeln und dort korreliert auswerten zu können.
Die System Log-Dateien können mit Hilfe des Log-Servers bereits auf einem Rechner analysiert werden. Durch die Verwendung unterschiedlicher Betriebssysteme für die einzelnen Honeypots wird das zentrale Speichern aller Log-Dateien erschwert. Für die Protokolldateien der Kommandozeileneingaben gibt es z.B. keine Möglichkeit, diese einfach

[40] Zeitaufwand für die Analyse der Daten, z.B. ist bei einer SMS dies mit dem Lesen der Nachricht erledigt.
[41] Virtualisierungssoftware für UNIX- und Windows™-Betriebssysteme, *http://vmware.com*
[42] Stand: September 2003
[43] http://security.rbaumann.net/

auf einem Rechner zu sammeln. Ebenso treten Probleme bei den Log-Dateien der einzelnen Web-Server oder FTP-Server auf.

Für die nachträgliche Auswertung der Log-Dateien (z.B. System Log) gab es leider kein geeignetes Werkzeug, um diese schnell auswerten zu können. Das Werkzeug *analog*[44] ist primär nur für Log-Dateien von Web-Servern geeignet, aber leider nicht für syslog-Dateien. Es wäre jedoch leicht möglich, z.B. die System Log-Dateien mit Hilfe eines Perl-Skriptes in einer Datenbank abzulegen und dann ähnlich zu *iptables log* auszuwerten.

5.4 Ergebnisse

Während und nach dem Betrieb eines Honeynets ist natürlich interessant, welche Erkenntnisse bzw. Ergebnisse man daraus ableiten bzw. ziehen kann. Die Ergebnisse werden nun in den folgenden Abschnitten beschrieben.

5.4.1 Klassifikation der Angreifer

Angreifer lassen sich in verschiedene Gruppen bzw. Profile einteilen. Dabei lassen sich auch die Ziele der jeweiligen Gruppe definieren (vgl. [Ano 03, Zan 03]).

5.4.1.1 Hacker

„*Ein Hacker ist ein Individuum, das sich in hohem Maße für die geheimnisvollen und verborgenen Arbeitsabläufe von Computerbetriebssystemen interessiert. Hacker sind oft Programmierer und verfügen insofern über fortgeschrittenes Wissen über Betriebssysteme und Programmiersprachen*“ [Ano 03]. Dabei stoßen sie ab und zu auf Schwachstellen in Programmen bzw. entwerfen Programme, die diese Schwachstellen aufzeigen. Hacker geben ihr Wissen weiter und haben nicht die Absicht Daten auf fremden Computersystemen zu zerstören oder Rechner unbenutzbar für andere zu machen.

5.4.1.2 Cracker

Ein Cracker ist die „böse“ Variante des Hackers. Er versucht in Computersysteme einzudringen, um dann dort Daten zu zerstören, zu verfälschen oder den Rechner für andere unbenutzbar zu machen. Cracker installieren oft auf den Zielsystemen versteckte Tools (z.B. Rootkits[45]), die ihnen erlauben, evtl. weitere Angriffe von diesem „gecrackten“ System aus zu starten.

[44] http://www.analog.cx/
[45] Ein Rootkit ist eine Sammlung von verschiedenen speziell präparierten Programmen, die dem Angreifer ermöglichen, auf diesen Rechner zurückzukehren und als Superuser (`root`) die volle Kontrolle über den Rechner zu haben.

5.4.1.3 Script Kiddie

Der wahrscheinlich am meisten verbreitete und am häufigsten anzutreffende Hacker, sind Script Kiddies. Script Kiddies sind meist Jugendliche (Kiddie; Kind) ohne großes Hintergrundwissen, die sich im Internet fertige Skripte herunterladen und diese ausprobieren, teilweise ohne die Auswirkungen zu kennen oder abschätzen zu können.
Ein Script Kiddie ist zum Beispiel daran zu erkennen, dass alle Rechner bzw. IP-Adressen eines Subnetzes nach einem bestimmten Merkmal von einer einzigen Adresse aus überprüft („gescannt“) wurden. Ebenso hinterlassen Script Kiddies oft Spuren, die auf die verwendeten Werkzeuge schließen lassen (vgl. [HP 01]).
Viele versuchte Angriffe, die von Script Kiddies aus gestartet werden, erzeugen keinen großen Schaden, da viele Systeme die entsprechende Schwachstelle nicht besitzen oder diese Schwachstellen bereits von dem verantwortlichen Administrator behoben wurden. Allerdings finden Script Kiddies durch ihre „unbarmherzige Ausdauer“ [Ano 03] doch immer wieder einzelne Systeme, die verwundbar sind. Auf diesen Systemen verursachen die Angreifer Schäden und greifen von diesem Rechner neue Systeme an. Dabei spielen die Daten eine untergeordnete Rolle, das Hauptaugenmerk liegt darin, möglichst eine große Anzahl an Systemen erfolgreich zu kompromittieren (vgl. [HP 00]).
Die Gefahren, die von Script Kiddies ausgehen sind nicht zu unterschätzen. Durch groß angelegtes Scannen von Netzen ist die Wahrscheinlichkeit sehr hoch, von einem Script Kiddie angegriffen zu werden. Dies trifft natürlich auch für Honeynets zu, so dass die meisten Angriffe, die auftreten, von Script Kiddies zu erwarten sind, die zufällig auf die Schwachstellen aufmerksam wurden.
Angriffe von Script Kiddies sind selten am Vormittag zu beobachten: Zu dieser Zeit sind sie in der Schule. Daher registriert man Angriffe dieser Art vor allem am späteren Nachmittag bis Abend bzw. in der Nacht. In den Sommerferien reduzieren sich die Aktivitäten, da hier die Haupturlaubszeit ist.

5.4.1.4 Gruppenmitglieder

Gruppenmitglieder lassen sich in zwei weitere Gruppen unterteilen: Warez-Gruppen und Hacker-Gruppen.
Das Ziel von Warez-Gruppen ist es, kommerzielle Softwareprodukte zu cracken (d.h. z.B. den Kopierschutz oder die Abfrage nach einer Registrierungsnummer zu entfernen) und diese Produkte für andere im Internet zur Verfügung zu stellen. Dafür ist Speicherplatz (Webspace) im Internet notwendig, da die Mitglieder dieser Gruppen natürlich im Hintergrund bleiben möchten. Meistens durchsuchen Warez-Gruppen das Internet nach FTP-Servern, die Schreibzugriffe mit dem so genannten „Anonymous-Account“ erlauben. Für diesen Account ist kein Passwort notwendig. Für den Betreuer eines Honeynet sind Warez-Gruppen uninteressant, da keine neuen Techniken bekannt werden. Durch einen starken Anstieg des Verkehrsaufkommens sind solche „Angriffe“ erkennbar.
Hacker-Gruppen beschäftigen sich meistens mit der Programmierung neuer Programme um Systeme anzugreifen, die bekannt gewordene Schwachstellen ausnutzen. Ein Angriff ist oft ein Test der entwickelten Programme. Geraten diese Programme in die Hände von Script Kiddies besteht eine große Gefahr. Der Nutzen der Angriffe durch Hacker-Gruppen ist hoch:

Die Angriffe können vor allem in Bezug auf die Angriffswerkzeuge aufschlussreich sein (vgl. [BP 02]).

5.4.1.5 Professionelle Hacker

Professionelle Hacker sind die Gruppe, die wahrscheinlich am seltensten in einem Honeynet beobachtet werden. Sie haben sehr gutes Fachwissen und sind sich oft dessen bewusst was sie tun. Angriffe von professionellen Hackern sind nicht zufällig und gut vorbereitet. Sie arbeiten oft in einem Sicherheits-Unternehmen oder für sich selbst.
Manche professionellen Hacker arbeiten auch im Auftrag von Firmen und Unternehmen („Hired Hacker“). Je nach Auftraggeber müssen sie für ihre Tätigkeiten eventuell keine juristischen Konsequenzen befürchten.
Für professionelle Hacker sind Honeynets kein attraktives Ziel, da diese Rechner z.B. nicht besonders geschützt sind: Sie werden unter Umständen ziemlich schnell merken, dass sie sich in einem Honeynet aufhalten. Es gäbe natürlich viel zu Lernen, falls sich doch ein professioneller Hacker auf einem Honeypot „verirrt“ hätte.

5.4.1.6 Vermutete Angreifer im Honeynet

Natürlich ist es nun interessant, welcher Typ von Angreifer sich hauptsächlich im installierten Honeynet aufgehalten hat. Dazu lässt sich sagen, dass wie vermutet hauptsächlich Script Kiddies sich an den Honeypots versucht haben. Es gab z.B. viele Scan-Vorgänge auf beiden Honeypots ungeachtet dessen, welches Betriebssystem oder Dienst installiert ist: Angriffe auf einen *IIS* Web-Server sind bei dem Linux-Honeypot sicher nicht erfolgreich. Ähnliches war auch bei verschiedenen Trojaner-Werkzeugen zu beobachten.
Einige Angreifer versuchten auf den FTP-Servern Verzeichnisse zu erstellen, allerdings ohne Erfolg. Hier kann nicht genau beurteilt werden, ob es sich auch nur um Script Kiddies oder um Mitglieder einer Warez-Gruppe gehandelt hat.

5.4.2 Bedrohungen und durchgeführte Angriffe

Im Folgenden werden verschiedene Bedrohungen vorgestellt und wie sich diese in der Betriebsphase des Honeynets ausgewirkt haben.
Im Internet existieren viele Bedrohungen und somit eine Vielzahl von Angriffsmöglichkeiten. Bedrohungen, die durch Aktionen eines Benutzers am Rechner durchgeführt werden müssen (z.B. Öffnen eines Anhangs von einer E-Mail mit enthaltenem Virus) sind für Honeynets nicht interessant, da ja auf diesen Rechnern z.B. kein Benutzer seine E-Mails abruft. Aus diesem Grund beschränkt sich die Analyse auf Angriffe, die ohne Benutzerinteraktion durchführbar sind.

5.4.2.1 Angriffe auf Web-Server

Angriffe auf Web-Server zählen mit zu den häufigsten Versuchen, Zugriff auf ein fremdes System zu erlangen. Hinzu kommt, dass für den Web-Server von Microsoft (in diesem Fall *IIS* 5.0) eine Vielzahl an Schwachstellen (bei der Kombination Windows™ 2000 mit *IIS* 5.0: 22 Schwachstellen [DFN 01]) und somit Angriffsmöglichkeiten bekannt wurden [cert 01].
Auch die eingesetzte Version des *Apache* Web-Servers (1.3.23) enthält Schwachstellen [cert 02]. Während auf den *IIS* Web-Server mehrere Angriffe durchgeführt wurden, fanden auf dem *Apache* Web-Server nur „Script-Kiddie Angriffe“ und Überprüfungen auf Existenz von PHP-Testdateien (Zeile 4 - 6) statt, wie am folgendes Beispiel zu sehen ist (Ausschnitt aus `/var/log/httpd/access_log`):

```
1  80.142.223.83 - - [22/Aug/2003:00:43:56 +0200]
   "GET /scripts/..%255c../winnt/system32/cmd.exe?/c+dir+c:\ HTTP/1.0"
   404 313
2  80.142.223.83 - - [22/Aug/2003:05:36:48 +0200] "GET
   /_vti_bin/..%255c..%255c..%255c..%255c..%255c../winnt/system32/
   cmd.exe?/c+dir+c:\  HTTP/1.0" 404 334
3  80.142.223.83 - - [22/Aug/2003:08:05:31 +0200] "GET
   /cgi-bin/%5C..%5C..%5C..%5C..%5Cwinnt%5Csystem32%5Ccmd.exe?/c+dir
   HTTP/1.0" 404 318
4  217.228.139.32 - - [04/Sep/2003:04:39:26 +0200]
   "GET /phpinfo.php HTTP/1.0" 404 286
5  217.228.139.32 - - [04/Sep/2003:04:39:26 +0200]
   "GET /test.php HTTP/1.0" 404 283
6  217.228.139.32 - - [04/Sep/2003:04:39:26 +0200]
   "GET /index.php3 HTTP/1.0" 404 285
```

Ungeachtet einer Überprüfung des Zielsystems werden Angriffsmöglichkeiten des *IIS* Web-Servers auf einem *Apache* Web-Server durchgeführt (Zeile 1 - 3): Die verwendeten Pfadangaben (z.B. `/winnt/system32/cmd.exe`) sind nur auf Windows™-Systemen zu finden. Bei beiden IP-Adressen handelt es sich übrigens um T-Online DSL Kennungen (wie eine einfache Überprüfung mittels `nslookup` zeigt). Auch der Windows™-Honeypot wurde von diesen IP-Adressen überprüft.

5.4.2.1.1 CodeRed2 (nach [cert 01])

CodeRed2 ist bereits länger bekannt und der Nachfolger von CodeRed. Das erste Auftreten war am 5. August 2001. Der Wurm nutzt eine BufferOverflow-Schwachstelle in der Datei `idq.dll` unter Windows™ NT 4.0 und 2000 aus sowie eine Schwachstelle in Windows: „Relative Shell Path Vulnerability“ (MS00-052) [msb52 00].

Der Ablauf gliedert sich in folgende Abschnitte:

1. Injektion ins *IIS*-System (an den Web-Server gesendete URL)
   Diese URL ist in drei Teile gegliedert:
   a) HTTP-Request: `GET /default.ida`

b) BufferOverflow: Zeichenfolge mit 224 „X“
c) Wurm-Code: Eine Folge von 22 Unicode-Zeichen, die als Programm-Code interpretiert werden. Das letzte Zeichen „%u00=a“ ist eine ungültige Unicode Kodierung.

Durch den BufferOverflow-Code wird auf dem Stack die Rücksprungadresse des URL-Parsers der Datei `idq.dll` überschrieben und an dieser Stelle der Wurm-Code platziert.

```
GET /default.ida?XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXX%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u780
1%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53f
f%u0078%u0000%u00=a HTTP/1.0
```

2. Einrichtung von Backdoors
   - Wurm kopiert `%windir%/cmd.exe` nach
     `<Laufwerk>:\inetpub\scripts\root.exe`
     `<Laufwerk>:\progra~1\common~1\system\MSDAC\root.exe`
     (`<Laufwerk>` = `C:` und `D:`)

   - Wurm richtet virtuelle Verzeichnisse „/c“ und „/d“ durch trojanisierte Shell `explorer.exe` ein (zusätzliche Backdoor, falls `root.exe` gelöscht wird).

3. Trojanisierung der Shell `explorer.exe`
   - Wurm kopiert eigene `explorer.exe` nach `C:\explorer.exe` und `D:\explorer.exe`. Durch die oben genannte Windows™-Schwachstelle wird die trojanisierte Shell vor der echten Shell aufgerufen.

   - Deaktivierung der Windows™ File Protection [mkb 03], dadurch werden Systemdateien nicht automatisch wiederhergestellt.

   - Veränderung der Berechtigungen (in der Registry) von „/scripts“ und „/MSDAC“. Damit erhalten die angelegten virtuellen Verzeichnisse Skript-, Ausführen-, Schreib- und Lese-Berechtigung.

   - Anlegen der virtuellen Verzeichnisse „/c“ und „/d“, die ebenfalls Skript-, Ausführen-, Schreib- und Lese-Berechtigungen erhalten (in der Registry).

   - Die trojanisierte Shell überprüft Registry alle 10 Minuten, ob die vorgenommenen Änderungen noch vorhanden sind und trägt diese, falls sie gelöscht wurden, wieder ein.

4. Weiterverbreitung: Auswahl der Ziel IP-Adresse

   - 12,5% zufällig
   - 50% aus demselben Class-A-Netzwerk
   - 37,5% aus demselben Class-B-Netzwerk

- Ausnahme: *127.x.x.x*, *224.x.x.x* und *255.255.255.255*

5. Seiteneffekt: DoS durch ARP-Request-Floods
   87,5% der Zielrechner stammen aus demselben Netzwerk
   Auflösung der IP-Adresse zu MAC-Adressen durch ARP-Broadcast
   → „Flut“ von ARP-Requests

Beobachtungen im Honeynet

- Der erste CodeRed2-Angriff erfolgte unmittelbar nach dem Beginn der Betriebsphase, insgesamt kam es zu 11 Angriffen, die genau nach dem obigen Ablauf durchgeführt wurden.

- Nach der Ausführung des Wurms wurden maximal 15 Rechner abgefragt, anschließend blockierte die Firewall am Gateway alle weiteren Versuche.

- Es erfolgte kein erfolgreicher Zugriff auf die trojanisierte Shell von außen. Mögliche Anfragen wären z.B.
  - `http://129.187.19.106/scripts/root.exe?/c+<Befehl>`
  - `http://129.187.19.106/c/winnt/system32/cmd.exe?/c+<Befehl>`

  Durch die Weiterverbreitungs-Phase wurde das Limit für die erlaubten ausgehenden Verbindungen schnell erreicht. Dadurch waren keine Zugriffe auf den Honeypot möglich.

- Auch auf dem Linux-Honeypot war die GET-Anfrage zu sehen. Hier hatte es keinerlei Auswirkungen auf den *Apache* Web-Server.

- Durch die große Anzahl an ausgehenden Anfragen (durch die Weiterverbreitungs-Phase des Wurmes) stieg die Größe der Netzwerkdump-Dateien erheblich (die Aufzeichnung des Verkehrs wurde am *eth1*-Interface durchgeführt, also vor der Firewall): 6 Stunden HTTP-Anfragen verursachen ca. 120 MB Netzwerkdump.

- Die Anfragen sind nicht in der Log-Datei zu finden (der *IIS* Web-Server interpretiert nur die Dateiendung). *Snort* erkennt diesen Angriff mit der Signatur „WEB-IIS ISAPI .ida attempt“.

- Nach der Entfernung der durch CodeRed2 vorgenommenen Änderungen und einem Reboot war der Honeypot wieder zurückgesetzt.

5.4.2.1.2 FX Scanner

Bereits in der Anfangsphase des Betriebes war auffällig, dass des Öfteren ein Verbindungsaufbau zu Port 57 zu beobachten war. Auf Port 57 ist kein Dienst registriert („any private terminal access“ [ian 03]). Es folgten weitere Anfragen derselben IP-Adresse auf Port 80 und 21. Mit Hilfe der Suchmaschine „Google“ und den Suchbegriffen „Port 57“

konnte ein Hinweis für diese Anfragen gefunden werden[46]: Der Angreifer verwendet das Werkzeug „FX Scanner“.
Mit dem einfach zu bedienenden Werkzeug *FX Scanner*[47] lassen sich *IIS* Web- und FTP-Server auf ihre Angriffsmöglichkeiten untersuchen. Abbildung 22 zeigt die Benutzeroberfläche des Programms. Es können mehrere IP-Adressen gleichzeitig überprüft werden. Die ermittelten Schwachstellen sind unter dem Punkt „Results“ bzw. „Debug“ zu aufgelistet.

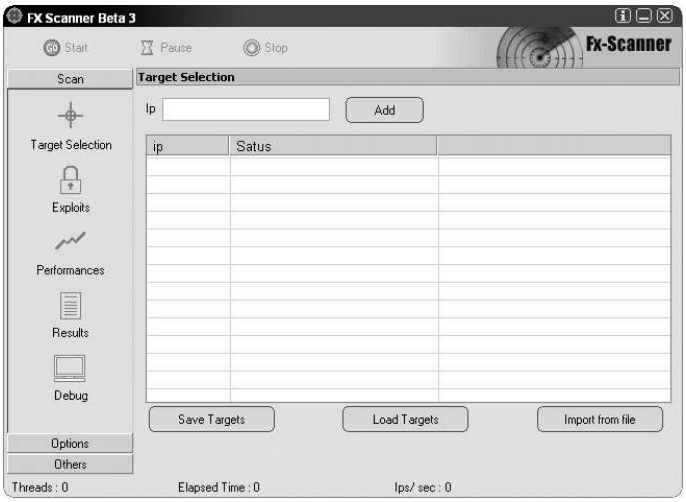

Abbildung 22: Benutzeroberfläche von *FX Scanner*

Das Charakteristische an diesem Werkzeug ist, dass zunächst auf Port 57 die Existenz eines Rechners geprüft wird.

Die Vorgehensweise von *FX Scanner* wird nun an einem Angriff vom 1. August erläutert:

1. Angreifer überprüft mit `ping`, ob der Ziel-Rechner erreichbar ist.

2. Anfrage auf Port 80: `HEAD / HTTP/1.0`. Anhand der Antwort kann bestimmt werden, ob es sich um einen *IIS* Web-Server handelt.

3. Ist auf dem Zielsystem ein *IIS* Web-Server installiert, folgen nun 77 Anfragen, die unter anderem die Existenz von einem CodeRed-Wurm überprüfen (es werden die im Abschnitt 5.4.2.1.1 beschriebenen Anfragen ausgeführt). Ist kein *IIS* Web-Server vorhanden, wird mit Schritt 4 fortgefahren. Alle durchgeführten Anfragen sind im Anhang (Abschnitt A.2) aufgelistet.

4. Versuchter TCP-Verbindungsaufbau zu Port 57. Dieser wird mit gesetztem RST- und ACK-Flag beantwortet.

[46] http://cert.uni-stuttgart.de/archive/intrusions/2002/11/msg00250.html. Es handelte sich dabei um den zweiten Treffer der Anfrage.
[47] Auf der eigentlichen Homepage von FX Scanner (http://www.fx-tools.net) ist das Werkzeug nicht mehr erhältlich, aber z.B. unter http://www.der-klan.de/tools/scanner.html wird es zum Download angeboten.

Beobachtungen im Honeynet

- Viele der Port 57 Anfragen kamen von T-Online DSL Accounts (21 der 53 anfragenden IP-Adressen). Auch bei den anderen Anfragen kann man erkennen, dass sie von Dialin-Zugängen aus gestartet wurden.

- Bei der Anzahl der Anfragen auf den beiden Honeypots war nur ein geringer Unterschied festzustellen (Windows™-Honeypot: 181, Linux-Honeypot: 176). Dies deutet auch darauf hin, dass es sich bei den Angreifern um Script Kiddies handelt.

5.4.2.1.3 BufferOverflows

Durch einen BufferOverflow will der Angreifer erreichen, dass ein von ihm gewähltes Programm ausgeführt wird. Dazu versucht der Angreifer den internen Programmpuffer einer Variablen zu überschreiben, z.B. durch eine überlange Eingabe (vgl. [HR 02]). Das Programm des Angreifers wird mit den Rechten des ursprünglichen Programms ausgeführt (z.B. `root`).
Bei Web-Servern kann dies zum Absturz des Dienstes führen.

Typische Anfrage (gekürzt, die gesamte Zeichenfolge enthält 4095 „A“s):

```
SEARCH /AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Beobachtungen im Honeynet

- Auf beiden Honeypots waren solche Anfragen zu sehen.

- Der *Apache* Web-Server lieferte als Error-Code *414 Request-URL too long*, der *IIS* Web-Server *500 Internal Server Error*. Auswirkungen hatten diese Anfragen keine.

- Ein Angreifer versuchte es vermutlich sogar zweimal den *IIS* Web-Server zum Absturz zu bringen: Am 22. August wurden 28 Versuche, am 8. September 59 Versuche registriert. Beide Male handelte es sich um einen DSL Account aus Italien (*ppp-217-133-220-84.cust-adsl.tiscali.it* und *ppp-217-133-220-110.cust-adsl.tiscali.it*). Außer diesen BufferOverflow Angriffen erfolgten keine weiteren Angriffsversuche.

5.4.2.2 Trojaner

„*Ein Trojanisches Pferd kann ein Programm sein, das etwas Nützliches oder auch nur etwas Interessantes tut. Es tut immer etwas Unerwartetes, wie beispielsweise ohne Ihr Wissen Kennwörter stehlen oder Dateien kopieren*“. [Ano 03, RFC 1244]

Für Trojanische Pferde gibt es weitere Klassifizierungen (nach [Ano 03]):

- **Destruktive Trojaner**: Dieser Trojanertyp kann unter Umständen große Datenmengen zerstören, z.B. durch Löschen der Festplatte. Im Gegensatz zu Würmern repliziert sich ein Trojaner nicht von selber. Ein bekanntes Beispiel für einen destruktiven Trojaner ist der *PC-Cyborg-Trojaner* [ciac 89] (auch bekannt unter dem Namen *AIDS-Trojaner*): 1989 wurden 10000 Disketten mit einem Programm mit AIDS-Informationen an viele medizinische Einrichtungen weltweit verteilt. Nach der Ausführung des Programms, wurden nach einer festgelegten Anzahl von Neustarts die Daten der Festplatte verschlüsselt. Um wieder Zugriff auf die Daten zu bekommen, sollte eine „Lizenzgebühr“ an die Adresse von PC Cyborg in Panama überwiesen werden. Die Verschlüsselung wurde von einem Virenspezialisten in Großbritannien gebrochen.

- **Trojaner, die die Privatsphäre verletzen**: Bei dieser Trojanerart werden meistens Funktionen ausgeführt, die dem Programmierer interessante Daten über den Benutzer liefern (z.B. Passwörter oder Kreditkartennummern). Im Jahre 1998 entwarfen zwei Schüler die so genannten *T-Online Power Tools*, die an die Schüler die Zugangsdaten zu T-Online von ihren Opfern übermittelten.

- **Netzwerktrojaner**/**Backdoors**/**Remotezugriffs-Tools**: Zu dieser Gruppe zählen Programme wie *NetBus* oder *BackOrifice*. Dabei wird ein bestimmter Port für den Angreifer geöffnet, über den er Aktivitäten wie z.B. Remotezugriffe, Modifikation von Dateien oder Aufzeichnen von Tastatureingaben durchführen kann. Für reine Remotezugriffe unter Windows™ sind *RAdmin*[48] oder VNC-Dienste zu nennen.

- **Rootkits**: Ein Rootkit ist eine Sammlung von modifizierten Systemprogrammen, die die Aktivitäten des Angreifers verstecken. So werden z.B. Prozesse des Angreifers in der Prozesstabelle nicht angezeigt. Ein Beispiel hierfür ist das *KNARK*[49] Rootkit.

- **DDoS-Agents**: Programme, die mit Hilfe von koordinierten Paketüberflutungen versuchen, Server zum Absturz zu bringen werden DDoS-Tools genannt. Beispiele hierfür sind *Stacheldraht*[50] oder *TrinOO*[51].

Während der Laufzeit des Honeynet wurden hauptsächlich bekannte Trojaner-Ports überprüft, ob bereits einer dieser Ports geöffnet wurde. Im Einzelnen waren dies:

- *Skydance* (Port 4000) [sky 00]: Remote-Zugriffs-Tool für Windows™ 95/98/Me/NT

- *RAdmin* (Port 4899) [dsh 03]: Remote-Zugriffs-Tool für Windows™ 95/98/Me sowie Windows™ NT/2000/XP

[48] http://www.famatech.com/
[49] http://packetstormsecurity.nl/UNIX/penetration/rootkits/knark-0.59.tar.gz
[50] http://packetstormsecurity.org/distributed/stachel.tgz
[51] http://packetstormsecurity.org/distributed/trinoo.tgz

- *NetBus* (Port 12345) [ant 03]: Remote-Zugriffs-Tool für Windows™ 95/98 sowie Windows™ NT/2000/XP

- *Kuang2* (Port 17300) [isc 03]: Backdoor für Windows™-Betriebssysteme

- *SubSeven* (Port 27374) [isc 03]: Remote-Zugriffs-Tool für Windows™ mit sehr vielen Funktionen

5.4.2.3 „Mysterium 55808“ (vgl. [hei 03b])

Mitte Mai 2003 tauchten im Internet TCP-Pakete auf, deren *WindowSize* den Wert 55808 hatten. Woher die Pakete kommen ist unklar, ebenso, ob es sich dabei „um ein trojanisches Pferd, eine Backdoor oder einen Wurm handelt“ [hei 03b].
Die Sicherheitsunternehmen ISS[52] und Intrusec[53], die diesen Paketen den Namen *Stumbler* gaben, haben diese genau analysiert und herausgefunden, dass es ein Linux-Binary ist, das Netzwerkkarten in den „promiscous mode“ setzt und somit alle Pakete aus dem Netz an das System weiterleitet. Ab diesem Zeitpunkt versendet dieses System an anscheinend zufällige IP-Adressen TCP-Pakete (diese erreichten den Linux-Honeypot) mit der *WindowSize* 55808 (Weiterverbreitung). Die Absenderadressen werden dabei verfälscht (gespooft). Über die eigentliche Absicht von *Stumbler* ist man sich noch nicht klar. Die oben genannten Unternehmen denken, dass es sich um einen noch nicht ganz ausgereiften Distributed Portscanner[54] handelt. Es gibt aber auch die Vermutung, dass es ein neues Werkzeug für DDoS-Attacken sein könnte.
Etwa zur selben Zeit ist auch eine Variante für Windows™ Betriebssysteme aufgetaucht, die von dem Unternehmen Network Associates *Randex* [na 03] benannt wurde.
Während der Betriebsphase tauchten von Anfang an Pakete dieser Art auf. Zu Beginn war das Auftreten der Pakete deutlich häufiger als am Ende der Betriebsphase. Hauptsächlich kamen die Pakete von einer bestimmten (gefälschten) IP-Adresse. Bei allen Paketen war der Ziel-Port auf 57669 und die Ziel-Adresse auf die des Linux-Honeypot gesetzt, den Windows™-Honeypot erreichte kein einziges Paket dieser Form. Da dieser Port auf dem Linux-Honeypot nicht geöffnet war, sendete dieser auf die SYN-Anfrage ein TCP-Paket mit gesetztem RST-Flag (Tabelle 5). In Kapitel 5.5.3 sind verschiedene Statistiken zu diesen Paketen zusammengestellt.

[52] Internet Security Systems, http://www.iss.net
[53] http://www.intrusec.com
[54] Beim Distributed Portscanning wird ein Zielsystem von mehreren Rechnern gleichzeitig gescannt.

84.185.38.91 → 129.187.19.105	129.187.19.105 → 84.185.38.91
Internet Protocol Src Addr: 84.185.38.91 Dst Addr: 129.187.19.105 Transmission Control Protocol Src Port: 14757 Dst Port: 57669 Seq: 4130606058 Ack: 0 Len: 0 Flags: 0x0002 (SYN) Window size: 55808	Internet Protocol Src Addr: 129.187.19.105 Dst Addr: 84.185.38.91 Transmission Control Protocol, Src Port: 57669 Dst Port: 14757 Seq: 0 Ack: 4130606059 Len: 0 Flags: 0x0014 (RST, ACK)

Tabelle 5: Versuchter Verbindungsaufbau von *Stumbler*; Abbruch durch Linux-Honeypot

5.4.2.4 Viren und Würmer

„*Ein Computervirus ist eine Befehlsfolge, die ein Wirtsprogramm zur Ausführung benötigt*“ [Eck 01]. Meistens muss der Virus von einem Benutzer aktiv aktiviert werden, z.B. durch Öffnen eines Anhangs von einer E-Mail oder durch Benutzen einer Diskette, die einen Virus enthält. Für Honeynets und Honeypots sind Viren weniger von Interesse, da diese Systeme nicht produktiv benutzt werden und z.B. keine E-Mails auf einem Honeypot abgerufen werden.
Im Unterschied zu einem Virus ist ein Wurm ein eigenständiges, selbst ablauffähiges Programm das sich vervielfältigt. Es wird also kein Wirtsprogramm zur Ausführung benötigt.
Zu den bekanntesten Würmern zählt der ILOVEYOU-Wurm aus dem Jahr 2000.

Im Folgenden werden die Würmer vorgestellt, die während der Betriebsphase im Honeynet auftraten. Alle Würmer konnten erst nach der Entfernung der Portsperren [lrzport 03] am G-WiN-Zugang ab dem 12. August das Honeynet erreichen (ausgenommen sind Angriffe innerhalb des MWN):

„**Sperrung der Ports 135, 137, 138, 139, 445 und 593 (NetBIOS über TCP/IP, Microsoft-Netzwerk)**
Am Zugang des G-WiN sowie über die Modem/ISDN- und VPN-Zugänge sind die Ports

- 135 (Microsoft RPC, wird für SMB Datei/Drucker-Sharing benutzt)
- 137 (NetBIOS Name Service)
- 138 (NetBIOS Datagram Service)
- 139 (NetBIOS Session Service, wird für SMB Datei/Drucker-Sharing benutzt)
- 445 (Microsoft Domain Service, Windows 2000/XP Common Internet File Sharing)
- 593 (Microsoft RPC via *IIS*)

(TCP und UDP) gesperrt. Dies verhindert den Betrieb der Druck- und Dateifreigabe des Microsoft-Netzwerks über das Internet.

Grund:
Sicherheit der Rechner im MWN. (Über NetBIOS sind diverse DoS-Angriffe (Denial of Service) und das Ausspähen von Daten möglich).
Insbesondere verbreiten sich auch Internet-Würmer wie W32.Blaster über diese Ports.

Sperrung der Ports 1433, 1434 (Microsoft SQL-Server)
Am Zugang des G-WiN sind die Ports 1433 und 1434 (TCP und UDP) gesperrt. Dies verhindert den Betrieb des Microsoft SQL-Serverdienstes über das Internet.

Grund:
Sicherheit der Rechner im MWN und Schutz vor Überlastung des Netzes durch Wurm-Angriffe wie z.B. SQLSlammer."

5.4.2.4.1 W32.Slammer (SQLSlammer)

Der im Januar 2003 aufgetretene Wurm nutzt eine Schwachstelle des Microsoft™ SQL Servers. Speziell präparierte UDP-Pakete (Port 1434) erzeugen im Server ein BufferOverflow und damit die Infizierung. Anschließend versucht der Wurm weitere Rechner zu infizieren, in dem er zufällige IP-Adressen auswählt und an diese ebenfalls die UDP-Pakete sendet. Durch diese Verbreitungsmethode erzeugte der Wurm eine hohe Netzlast. Ein Sicherheitspatch von Microsoft stand bereits seit dem 24. Juli 2002 zur Verfügung (vgl. [hei 03a]). Auch auf dem Port 1433 kann dieser Dienst angegriffen werden.
Auch 7 Monate nach dem Auftreten des Wurms, waren doch eine beträchtliche Anzahl an Anfragen auf diese Ports zu beobachten. Auf beiden Honeypots stand allerdings kein Dienst auf diesen Ports zur Verfügung. Die Ergebnisse sind in Tabelle 6 dargestellt. Auffällig dabei ist, dass deutlich mehr Anfragen an den Linux-Honeypot auf Port 1434 gestellt wurden.

	Linux-Honeypot	Windows™-Honeypot
Anfragen auf Port 1433	289	323
unterschiedliche IP-Adressen	93	97
Anfragen auf Port 1434	382	311
unterschiedliche IP-Adressen	210	192

Tabelle 6: Anzahl der Anfragen und IP-Adressen auf Port 1433 und Port 1434

5.4.2.4.2 W32/Deloder-A

Dieser Wurm ist bekannt seit Anfang März 2003 und nutzt dazu den Port 445 bei den Betriebssystemen Windows™ 2000 und XP. Er versucht eine Verbindung zum Zielrechner aufzubauen und versucht anhand einer Liste von einfachen Passwörtern das Administrator-Passwort zu erraten (alle Passwörter sind in Kapitel A.3 aufgelistet). Gelingt dieser Versuch, so installiert der Wurm eine Backdoor und entfernt die Standard Netzwerkfreigaben.

Es ist auch möglich durch diesen Angriff mit Hilfe von ungültigen Anfragen auf diesem Port die CPU-Auslastung des Zielrechners auf 100% ansteigen zu lassen[55].
Seit Anfang März sind verschiedene Varianten dieses Wurmes aufgetaucht [Sto 03].

Beobachtungen im Honeynet

- Durch die Wahl des Administrator-Passwortes des Windows™-Honeypot (siehe Kapitel 4.3.3) konnte der Wurm diesen Rechner nicht infizieren.

- Auf beiden Honeypots trafen Anfragen ein, auf dem Windows-Honeypot deutlich mehr (Anfragen auf Port 445 wurden vom Linux-Honeypot mit gesetztem RST-Flag beantwortet).

Der folgende Ablauf (am Beispiel eines Angriffes auf den Windows™-Honeypot vom 18. August 2003) gliedert sich in die folgenden Abschnitte. Dabei entspricht der Verlauf genau der Beschreibung von [Sto 03].

1. Drei-Wege-Handshake:

Zeit	Quelle	Ziel	Protokoll	Information
17:32:43	67.30.46.88	129.187.19.106	TCP	1713 → 445 [SYN]
17:32:43	129.187.19.106	67.30.46.88	TCP	445 → 1713 [SYN, ACK]
17:32:43	67.30.46.88	129.187.19.106	TCP	1694 → 445 [ACK]

2. SMB Protokollvereinbarung:

Zeit	Quelle	Ziel	Protokoll	Information
17:32:44	67.30.46.88	129.187.19.106	SMB	Negotiate Protocol Request
17:32:44	129.187.19.106	67.30.46.88	SMB	Negotiate Protocol Response

3. Angreifer möchte Ressource benutzen, der Windows™-Honeypot erfragt weitere Informationen:

Zeit	Quelle	Ziel	Protokoll	Information
17:32:44	67.30.46.88	129.187.19.106	SMB	Session Setup AndX Request, NTLMSSP_NEGOTIATE
17:32:44	129.187.19.106	67.30.46.88	SMB	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_ PROCESSING_REQUIRED

4. Angreifer sendet Login-Informationen und wird geblockt (verwendetes Passwort war falsch):

Zeit	Quelle	Ziel	Protokoll	Information
17:32:44	67.30.46.88	129.187.19.106	SMB	Session Setup AndX Request, NTLMSSP_AUTH
17:32:44	129.187.19.106	67.30.46.88	SMB	Session Setup AndX Response, Error: STATUS_LOGON_FAILURE

[55] http://marc.theaimsgroup.com/?l=bugtraq&m=101408718030099&w=2

5. Angreifer beendet die Session, Honeypot meldet Fehler zurück:

Zeit	Quelle	Ziel	Protokoll	Information
17:32:45	67.30.46.88	129.187.19.106	SMB	Logoff AndX Request
17:32:45	129.187.19.106	67.30.46.88	SMB	Logoff AndX Response, Error: Bad userid

Die Aktionen 3. bis 5. werden solange durchgeführt, bis alle Passwörter aus der Liste des Wurmes probiert wurden.

6. Zum Abschluss erfolgt der Verbindungsabbau:

Zeit	Quelle	Ziel	Protokoll	Information
17:33:12	67.30.46.88	129.187.19.106	TCP	1713 → 445 [FIN, ACK]
17:33:12	129.187.19.106	67.30.46.88	TCP	445 → 1713 [FIN, ACK]
17:33:12	67.30.46.88	129.187.19.106	TCP	1713 → 445 [RST]

5.4.2.4.3 W32.Blaster (LovSan)

W32.Blaster ist ein Wurm, der eine Sicherheitslücke im DCOM/RPC-Dienst [msb26 03] vom Microsoft ausnutzt. Microsoft stellte bereits am 16. Juli 2003 einen Patch bereit, der diese Schwachstelle behebt. Seit dem 11. August 2003 breitet sich der Wurm im Internet aus. Er verbindet sich mit dem Zielsystem auf Port 135, erzeugt einen BufferOverflow und führt den eingeschleusten Code aus: Es wird auf dem Zielsystem der Port 4444 geöffnet, per TFTP (Trivial File Transfer Protocol) die Datei `msblast.exe` vom Rechner, auf dem der Wurm ausgeführt wurde, im Verzeichnis `%Windir%\System32` gespeichert und gestartet. Nun versucht auch das Zielsystem weitere Rechner auf dieselbe Weise zu infizieren. Bei Windows™ 2000 wird der RPC-Dienst beendet, es erscheinen Einträge in der *Ereignisanzeige*, bei Windows™ XP wird der Rechner nach einer Minute heruntergefahren und neu gestartet.
Zusätzlich sollte am 15. August dadurch ein DDoS-Angriff gegen den Windows™ Update Server *windowsupdate.com* gestartet werden, allerdings ohne Erfolg, da Microsoft die entsprechenden Server absicherte.

Während der Betriebsphase waren jedoch auch schon Angriffe vor dem 12. August festzustellen. Im Folgenden werden die einzelnen Abschnitte zwischen dem 16. Juli und 12. August genauer beschrieben:

16. Juli 2003
Microsoft gibt eine Schwachstelle im DCOM/RPC-Dienst bekannt und veröffentlicht ein Update, das diese Schwachstelle behebt.

27. Juli 2003
In der Mailingliste *Incidents* von *SecurityFocus* wird bekannt gegeben, dass das Verkehraufkommen auf Port 135 drastisch ansteigt und dass ein Programm (Exploit) zur Ausnutzung dieser Schwachstelle im Internet existiert.

29. Juli 2003, 19:34 Uhr
Ein Rechner des Lehrstuhls für Wirtschaftsinformatik der TU München (*atkrcmardom3.informatik.tu-muenchen.de*) verbindet sich zum Windows™-Honeypot auf Port 135 und beendet den RPC-Dienst. Der Versuch eine Shell auf Port 4444 zu öffnen schlägt jedoch fehl: Zu diesem Zeitpunkt existieren bereits unterschiedliche Exploits[56] für Windows™ 2000 und Windows™ XP. Wahrscheinlich wurde ein Exploit für Windows™ XP verwendet. Trotz der Port-Beschränkungen des LRZ war diese Verbindung möglich, da innerhalb des MWN keine Port-Beschränkungen vorhanden sind. Es blieb bei diesem einmaligen Angriff.
Folgende Einträge waren in der Ereignisanzeige zu sehen:

- *Service Control Manager: N/A: Der Dienst "Remoteprozeduraufruf (RPC)" wurde unerwartet beendet. Dies ist bereits 0 Mal vorgekommen. Folgende Korrekturmaßnahmen werden in 0 Millisekunden durchgeführt: Kein Vorgang.*
- *EventSystem: N/A: Das COM+-Ereignissystem entdeckte einen fehlerhaften Rückgabecode während der internen Verarbeitung. HRESULT war 800706BF von Zeile 42 von .\eventsystemobj.cpp. Wenden Sie sich an den Microsoft Software Service, um diesen Fehler zu melden.*

Die letzte Meldung wurde bis zu einem Neustart des Rechners ca. alle 2 Minuten in die Ereignisanzeige geschrieben.

11. August 22:50:07 Uhr
Ein LRZ Dialin-Account stellt eine Verbindung zum Windows™-Honeypot auf Port 135 her. Diese wird sofort, ohne weitere Aktivitäten, von dem initiierenden Rechner beendet.

11. August 22:51:51 Uhr
Es erfolgt ein Verbindungsaufbau auf Port 135 vom LRZ-Rechner *infovista.lrz-muenchen.de* (Frame 1240 - 1243 in Abbildung 23). Anschließend wird ein RPC-Aufruf gestartet, der die Schwachstelle ausnützt und den Port 4444 öffnet (Frame 1248 - 1257). Über diesen Port wird die *Eingabeaufforderung* gestartet und der TFTP-Befehl `tftp -i 129.187.10.31 GET msblast.exe` übertragen.

Frame 1261
129.187.10.31:1733 > 129.187.19.106:4444 TCP [PSH,ACK] Len=38

```
0000:  00 C0 4F C9 16 AC 00 04 28 66 E8 72 08 00 45 00  ..O.....(f.r..E.
0010:  00 4E 49 F5 40 00 7F 06 90 B5 81 BB 0A 1F 81 BB  .NI.@...........
0020:  13 6A 06 C5 11 5C 07 B3 AC 4A 08 E6 63 05 50 18  .j...\...J..c.P.
0030:  FA F0 7A D3 00 00 74 66 74 70 20 2D 69 20 31 32  ..z...tftp -i 12
0040:  39 2E 31 38 37 2E 31 30 2E 33 31 20 47 45 54 20  9.187.10.31 GET
0050:  6D 73 62 6C 61 73 74 2E 65 78 65 0A              msblast.exe.
```

Dieser Aufruf wird von *snort_inline* blockiert und somit das weitere Verbreiten des Wurms durch den Honeypot-Rechner verhindert. Insgesamt wird 8-mal versucht, diese Datei auf den Honeypot Rechner zu kopieren, danach wird mit einer Fehlermeldung abgebrochen (Orange

[56] „Ein Exploit wird oft zur Demonstration einer Sicherheitslücke geschrieben und veröffentlicht. Dadurch soll erreicht werden, dass Hersteller von Software gezwungen werden auf bekannt gewordene Sicherheitslücken zu reagieren.“ [net 03]

markierte Frames). Außerdem wird der RPC-Dienst beendet: Es werden in der *Ereignisanzeige* die oben erwähnten Meldungen erzeugt (Frame 1295 - 1296).
Hätte *snort_inline* den Download nicht blockiert, so wäre die Datei `msblast.exe` gestartet worden (Frame 1298 und Frame 1300).

Frame 1298

129.187.10.31:1733 > 129.187.19.106:4444 TCP [PSH,ACK] Len=18

```
0000:  00 C0 4F C9 16 AC 00 04 28 66 E8 72 08 00 45 00  ..O.....(f.r..E.
0010:  00 3A 4A 7D 40 00 7F 06 90 41 81 BB 0A 1F 81 BB  .:J}@....A......
0020:  13 6A 06 C5 11 5C 07 B3 AC 70 08 E6 63 6E 50 18  .j...\...p..cnP.
0030:  FA 87 A2 8E 00 00 73 74 61 72 74 20 6D 73 62 6C  ......start msbl
0040:  61 73 74 2E 65 78 65 0A                          ast.exe.
```

Frame 1300

129.187.10.31:1733 > 129.187.19.106:4444 TCP [PSH,ACK] Len=12

```
0010:  00 34 4A 9E 40 00 7F 06 90 26 81 BB 0A 1F 81 BB  .4J.@....&......
0020:  13 6A 06 C5 11 5C 07 B3 AC 82 08 E6 63 6E 50 18  .j...\......cnP.
0030:  FA 87 EB 89 00 00 6D 73 62 6C 61 73 74 2E 65 78  ......msblast.ex
0040:  65 0A                                            e.
```

Mit dem in Kapitel 5.2.4 vorgestellten Werkzeug *Packetyzer* wurde der Netzwerkdump von *snort* analysiert. Der gesamte Verlauf ist in Abbildung 23 dargestellt.

11. August 22:51:57 Uhr
Auch aus dem Mitarbeiternetz des LRZ (*129.187.15.0*) erfolgt ein RPC-Aufruf auf dem Windows™-Honeypot, ebenso der Versuch die Datei `msblast.exe` zu speichern. Durch den in der Zwischenzeit beendeten RPC-Dienst können keine weiteren Anfragen mehr bearbeitet werden.

12. August 12:20 Uhr
In der Zwischenzeit wurde der Windows™-Honeypot noch von weiteren 16 IP-Adressen (meistens LRZ Dialin Adressen) angegriffen und 3-mal neu gestartet. Zu diesem Zeitpunkt wurden auch die LRZ-Portbeschränkungen für die beiden Honeypot-Rechner aufgehoben.

12. August 12:26 Uhr
Es erfolgt der erste Blaster-Angriff außerhalb des G-WiN. Der RPC-Dienst wird wieder beendet.

12. August 13:11 Uhr
Nach weiteren vier Angriffen wird eine Regel in der Firewall des Gateway-Rechners hinzugefügt: Alle eingehenden Verbindungen auf Port 135 werden ab sofort geblockt, um zu verhindern, dass nicht permanent der Rechner neu gestartet werden muss. Im weiteren Verlauf des Betriebes erfolgen von weiteren 2710 IP-Adressen Anfragen auf Port 135. Ob es sich hierbei immer um Blaster-Angriffe gehandelt hat, kann nicht festgestellt werden.

Num	Source Address	Dest Address	Summary	AbsTime
1240	129.187.10.31	129.187.19.106	TCP: 1722 > 135 [SYN] Seq=128105698 Ack=0 Win=64240 L...	22:51:51.525.489
1242	129.187.19.106	129.187.10.31	TCP: 135 > 1722 [SYN, ACK] Seq=148728816 Ack=1281056...	22:51:51.526.211
1243	129.187.10.31	129.187.19.106	TCP: 1722 > 135 [ACK] Seq=128105699 Ack=148728817 Wi...	22:51:51.547.209
1248	129.187.10.31	129.187.19.106	DCERPC: Bind: call_id: 127 UUID: 000001a0-0000-0000-c00...	22:51:53.326.537
1249	129.187.10.31	129.187.19.106	DCERPC: Request: call_id: 229 opnum: 4 ctx_id: 1	22:51:53.327.056
1250	129.187.10.31	129.187.19.106	TCP: 1722 > 135 [PSH, ACK] Seq=128107231 Ack=1487288...	22:51:53.327.298
1251	129.187.10.31	129.187.19.106	TCP: 1722 > 135 [FIN, ACK] Seq=128107475 Ack=14872881...	22:51:53.327.553
1252	129.187.19.106	129.187.10.31	TCP: 135 > 1722 [ACK] Seq=148728817 Ack=128107475 Wi...	22:51:53.328.031
1253	129.187.19.106	129.187.10.31	TCP: 135 > 1722 [ACK] Seq=148728817 Ack=128107476 Wi...	22:51:53.328.036
1254	129.187.19.106	129.187.10.31	DCERPC: Bind_ack: call_id: 127 accept max_xmit: 5840 max_re...	22:51:53.328.780
1255	129.187.19.106	129.187.10.31	TCP: 135 > 1722 [FIN, ACK] Seq=148728877 Ack=12810747...	22:51:53.338.644
1256	129.187.10.31	129.187.19.106	TCP: 1722 > 135 [RST] Seq=128107476 Ack=148728817 Wi...	22:51:53.343.580
1257	129.187.10.31	129.187.19.106	TCP: 1722 > 135 [RST] Seq=128107476 Ack=128107476 Wi...	22:51:53.344.569
1258	129.187.10.31	129.187.19.106	TCP: 1733 > 4444 [SYN] Seq=129215561 Ack=0 Win=64240...	22:51:53.732.582
1259	129.187.19.106	129.187.10.31	TCP: 4444 > 1733 [SYN, ACK] Seq=149316356 Ack=129215...	22:51:53.733.316
1260	129.187.10.31	129.187.19.106	TCP: 1733 > 4444 [ACK] Seq=129215562 Ack=149316357 W...	22:51:53.740.728
1261	129.187.10.31	129.187.19.106	TCP: 1733 > 4444 [PSH, ACK] Seq=129215562 Ack=149316...	22:51:53.825.372
1262	129.187.19.106	129.187.10.31	TCP: 4444 > 1733 [ACK] Seq=149316357 Ack=129215600 W...	22:51:53.942.383
1263	129.187.19.106	129.187.10.31	TCP: 4444 > 1733 [PSH, ACK] Seq=149316357 Ack=129215...	22:51:55.694.591
1264	129.187.10.31	129.187.19.106	TCP: 1733 > 4444 [ACK] Seq=129215600 Ack=149316442 W...	22:51:55.856.489
1265	129.187.19.106	129.187.10.31	TCP: 4444 > 1733 [PSH, ACK] Seq=149316442 Ack=129215...	22:51:55.856.994
1266	129.187.19.106	129.187.10.31	TFTP: Read Request, File: msblast.exe, Transfer type: octet	22:51:55.988.061
1267	129.187.10.31	129.187.19.106	TCP: 1733 > 4444 [ACK] Seq=129215600 Ack=149316462 W...	22:51:56.075.441
1268	129.187.19.106	129.187.10.31	TFTP: Read Request, File: msblast.exe, Transfer type: octet	22:51:57.006.949
1288	129.187.19.106	129.187.10.31	TFTP: Read Request, File: msblast.exe, Transfer type: octet	22:51:59.010.185
1294	129.187.19.106	129.187.10.31	TFTP: Read Request, File: msblast.exe, Transfer type: octet	22:52:03.015.892
1295	129.187.19.106	129.187.19.107	Syslog: DAEMON.WARNING: Eventlog message size too la...	22:52:06.147.844
1296	129.187.19.106	129.187.19.107	Syslog: DAEMON.ERR: Service Control Manager: N/A...	22:52:06.199.682
1297	129.187.19.106	129.187.10.31	TFTP: Read Request, File: msblast.exe, Transfer type: octet	22:52:11.017.694
1298	129.187.10.31	129.187.19.106	TCP: 1733 > 4444 [PSH, ACK] Seq=129215600 Ack=149316...	22:52:14.825.710
1299	129.187.19.106	129.187.10.31	TCP: 4444 > 1733 [ACK] Seq=149316462 Ack=129215618 W...	22:52:14.973.063
1300	129.187.10.31	129.187.19.106	TCP: 1733 > 4444 [PSH, ACK] Seq=129215618 Ack=149316...	22:52:16.825.732
1301	129.187.19.106	129.187.10.31	TCP: 4444 > 1733 [ACK] Seq=149316462 Ack=129215630 W...	22:52:16.976.039
1302	129.187.10.31	129.187.19.106	TCP: 1733 > 4444 [RST] Seq=129215630 Ack=149316462 Wi...	22:52:18.825.979
1303	129.187.19.106	129.187.10.31	TFTP: Read Request, File: msblast.exe, Transfer type: octet	22:52:19.019.745
1310	129.187.19.106	129.187.10.31	TFTP: Read Request, File: msblast.exe, Transfer type: octet	22:52:27.021.054
1311	129.187.19.106	129.187.10.31	TFTP: Read Request, File: msblast.exe, Transfer type: octet	22:52:35.022.854
1312	129.187.19.106	129.187.10.31	TFTP: Error Code, Code: Not defined, Message: timeout on receive	22:52:43.039.466

Abbildung 23: Angriff des Blaster-Wurms am 11. August um 22:51 Uhr (Analyse mit *Packetyzer*)

5.4.2.4.4 W32.HLLW.Gaobot.AA

Dieser Wurm nutzt dieselbe Schwachstelle wie der Blaster-Wurm aus, nur mit der Erweiterung, dass er zusätzlich eine Schwachstelle des Locator Services [msb01 03] verwendet. Dadurch kann der Rechner nicht nur über den Port 135 sondern auch über den Port 139 bzw. Port 445 angegriffen werden. Das erste Auftreten wurde bei Symantec [sym 03] am 21. August 2003 bekannt, auf dem Windows™-Honeypot war dieser Wurm bereits am 20. August um 2:10 Uhr zu beobachten. Da der Port 135 seit 12. August gesperrt wurde, starteten die Angriffe auf Port 139 bzw. Port 445.

Die einzelnen Angriffe müssen in zwei Gruppen unterschieden werden:

- Speichern der Datei `winhlpp32.exe` im Verzeichnis `%WinDir%\system32`
  Diese Datei enthält den Wurm-Code, sie wurde aber nie ausgeführt, da dazu der Port 135 geöffnet sein müsste. Die Dateigröße betrug 32 KByte. Es wurden keine weiteren Aktionen durchgeführt.

- Beendigung des RPC-Dienstes
  Bevor die Datei übertragen werden konnte, wurde der RPC-Dienst beendet und die bereits bekannten Meldungen in der *Ereignisanzeige* festgehalten. Durch einen Neustart wurde der RPC-Dienst wieder zurückgesetzt.

Diagramm 3 zeigt den zeitlichen Ablauf der Angriffe dieses Wurms.

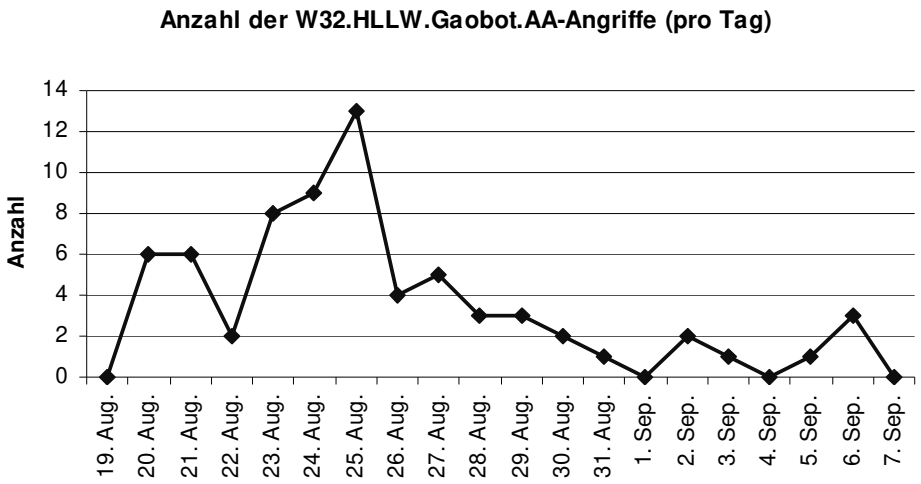

Diagramm 3: Anzahl der W32.HLLW.Gaobot.AA-Angriffe (pro Tag)

5.4.2.5 Sonstige Beobachtungen

Neben den bisherigen Angriffen sind noch folgende Ereignisse zu beobachten:

MySQL-Datenbank des Linux-Honeypot
Es fanden Login-Versuche von 3 verschiedenen Rechnern statt. Eine Übersicht ist in Tabelle 7 dargestellt. Der erste Rechner versuchte insgesamt 20-mal einen Zugriff auf die Datenbank zu bekommen, jeweils 10-mal als Benutzer `root` und `mysql`. Dabei verwendete der Angreifer 3-mal kein Passwort, die anderen 17 Versuche wahrscheinlich einfache Passwörter. Da diese verschlüsselt übertragen werden, kann nicht festgestellt werden, welche Passwörter probiert wurden.
Es war zu beobachten, dass alle aufgeführten Rechner auch Anfragen an den Windows™-Honeypot stellten, dort war aber keine MySQL-Datenbank installiert.

IP-Adresse	DNS-Name	Betriebssystem (ermittelt mit *p0f*)	Land	Login-Versuche
80.228.76.7	dynadsl-080-228-76-007.ewetel.net	Windows™ 2000/XP (DSL)	DE	`root` (2 mal ohne Passwort, 8 mal mit Passwort) `mysql` (1 mal ohne Passwort, 9 mal mit Passwort)
217.228.97.92	pD9E4615C.dip.t-dialin.net	Windows™ 2000/XP mit NAT (DSL)	DE	`mysql` (1 mal ohne Passwort)
147.229.132.100	autnt.fme.vutbr.cz	Windows™ NT	CZ	`root` (1 mal ohne Passwort, 10 mal mit Passwort)

Tabelle 7: Login-Versuche für die MySQL-Datenbank auf dem Linux-Honeypot

SSH-Verbindungen

Eingehende SSH-Verbindungen sind von 14 verschiedenen IP-Adressen registriert worden. Ein Login-Versuch fand dabei nie statt. In drei Fällen erfolgte eine Abfrage der eingesetzten SSH-Version auf dem Linux-Honeypot (z.B. mit `telnet 129.187.19.105 22` und anschließendem Abbruch der Verbindung).

Eingehender Verkehr mit Ziel-Port 1740

Zwischen dem 26. August und dem 12. September war bei 456 Verbindungen von 45 unterschiedlichen Rechnern der Ziel-Port auf den Wert 1740 gesetzt. Auf diesem Port ist der Dienst „encore“ registriert.

Das seltsame an diesen Verbindungen war, dass bei diesen Paketen entweder das SYN- und das ACK-Flag bzw. das RST- und das ACK-Flag gesetzt waren, d.h. es fehlte der erste Schritt bzw. die ersten beiden Schritte für den Handshake, da dieser Port auf beiden Honeypots geschlossen ist. Außerdem waren die Quell-Ports oft auf „well-known“ Ports, z.B. 53, 80 gesetzt. Bei genauerer Betrachtung der Quell-IP-Adressen stellte sich heraus, dass 11 dieser Rechner DNS-Server sind. Es ist nicht auszuschließen, dass diese Rechner Ziel eines DoS-Angriffes waren und zufällig die IP-Adressen des Honeynets als gespoofte IP-Adressen der Angreifer eingesetzt wurden.

Meldungen an den Nachrichtendienst von Windows™

Über den Nachrichtendienst von Windows™ können Nachrichten in Form von Popup-Fenstern auf einem anderen Rechner übermittelt werden. Dies erfolgt über UDP auf Port 135. Jeweils eine Meldung hat die Honeypots erreicht, beim Windows™-Honeypot wurde sie angezeigt (Abbildung 24).

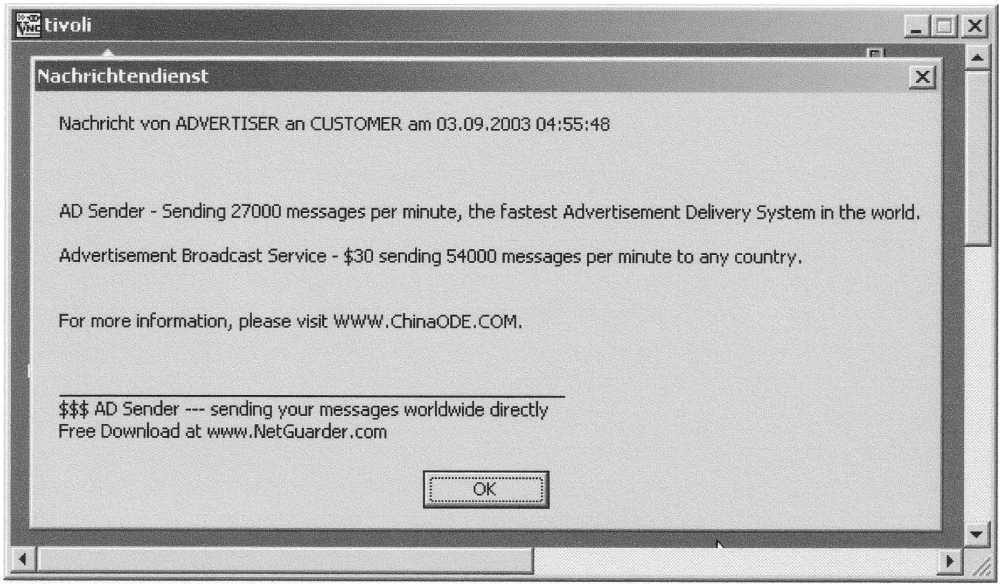

Abbildung 24: Popup-Nachricht auf dem Windows™-Honeypot

Port 139 und Port 445

Neben den Wurm-Angriffen steigerte sich zwischen Ende August und Anfang September das Verkehrsaufkommen auf den Ports 139 und 445 beträchtlich. Am 1. September wurden zwischen 14:50 Uhr und 17 Uhr 1895 Verbindungen auf Port 139 und 656 Verbindungen auf Port 445 registriert.

Der Rechner mit der IP-Adresse *207.236.37.170* versuchte ohne Erfolg von 14:50 Uhr bis 15:40 permanent sich mit der `$IPC`-Freigabe (als „Administrator“ der Domäne „FIREWALL“) des Windows™-Honeypot zu verbinden.

Konsequenzen für den Betrieb hatten diese „Angriffe“ nicht.

Anfragen mit Quell-Port 0

Bei 46 Verbindungen war der Quell-Port auf den Wert „0“ gesetzt, dabei handelte es sich bis auf eine Verbindung um DSL Accounts, die alle in den USA lokalisiert sind (Analyse mit *Visualroute*). Durch die Verwendung des Quell-Ports 0 versucht der Angreifer eine evtl. vorhandene Firewall auszuweichen, da diese unter Umständen nur Verbindungen mit den Quell-Ports 1 bis 65535 überwacht [mcc 02]. Als Ziel-Port wurden nahezu nur Proxy-Ports und als Ziel-Adresse nahezu nur die des Windows™-Honeypot verwendet. Dies legt die Vermutung nahe, dass In Tabelle 8 ist eine Übersicht über die Verbindungen mit Quell-Port 0 aufgelistet.

Domain	Ziel-Ports	Häufigkeit
Dial1.SaintLouis1.Level3.net	25, 80, 1080, 1180, 1182, 3128, 4480, 6588, 8080	24
dialup.stlsmo.swbell.net	80, 1080, 1182, 6588, 8080	8
se.client2.attbi.com	80, 1080, 3128, 1080	8
dsl.rcsntx.swbell.net	25	3
us.da.qwest.net	3128, 8080	2
unknown	57669	1

Tabelle 8: Registrierte Verbindungen mit Quell-Port 0

5.4.3 Erfolgreiche Kompromittierungen

Während des Betriebes fanden, abgesehen von Wurm- und Web-Server-Angriffen, keine erfolgreichen Kompromittierungen statt.
Dies heißt jedoch nicht, dass keine Versuche unternommen wurden, Zugriff auf einen der Honeypots zu bekommen.
Es wurde vielmehr deutlich, dass das Einspielen von Updates in einer Rechnerumgebung unerlässlich ist. Viele der Wurm- und Web-Server-Angriffe hätten keinen Erfolg gehabt, wenn die Systeme auf dem neuesten Stand gewesen wären. Hier hat sich gezeigt, dass der Linux-Honeypot weniger in der Praxis angegriffene Schwachstellen als der Windows™-Honeypot aufwies.
Es hat sich herausgestellt, dass deutlich mehr Verbindungen zum Windows™-Honeypot als zum Linux-Honeypot registriert wurden, d.h. es wurde wesentlich öfter versucht über die NetBIOS-Ports Zugriff auf den Windows™-Honeypot zu bekommen. Da diese Ports beim Linux-Honeypot geschlossen sind, bestand hier keine Angriffsmöglichkeit. Eine ausführliche Analyse ist in Kapitel 5.5.4 beschrieben.
Die IP-Adressen bzw. die DNS-Namen der Honeypots wurden nicht veröffentlicht, dadurch wurde man nur zufällig bzw. durch groß angelegte Netzwerk-Scans auf die Systeme aufmerksam. Der einzige Hinweis auf die Existenz eines Honeynet im MWN war die Beschreibung dieser Diplomarbeit auf der Homepage des Lehrstuhls „Kommunikationssysteme und Systemprogrammierung“ der LMU München.

5.5 Statistiken

In diesem Kapitel werden die unterschiedlichen Vorkommnisse statistisch ausgewertet und dargestellt. Diese Statistiken beruhen auf den gesammelten Daten der beiden Honeypots bzw. des Gateway-Rechners (insbesondere auf die Firewall-Log Datei des Gateways). Die zur Analyse verwendeten PHP-Skripte sind im Anhang (Kapitel A.5) verzeichnet.

5.5.1 Allgemeines

Zunächst folgen einige kurze Statistiken über allgemeine Daten, z.B. über die Laufzeit, den verbrauchten Speicherplatz für Log-Dateien, sowie die Anzahl der erzeugten Einträge bei *ACID*.

Das Honeynet war zwischen dem 15. Juli 2003 (8:54 Uhr) und dem 12. September 2003 (19:16 Uhr) in Betrieb. Dies entspricht also einer Online-Zeit von 60 Tagen. Während dieser Zeit sind beim Windows™-Honeypot 30 Neustarts durchgeführt worden, beim Linux-Honeypot ein Neustart (verursacht durch einen kurzen Spannungsabfall). Die Neustarts beim Windows™-Honeypot wurden entweder durch einen CodeRed-Angriff oder auf Grund des beendeten RPC-Dienstes (verursacht durch den Blaster- bzw. Gaobot-Wurm) notwendig. Dadurch wurde der Windows™-Honeypot wieder in den ursprünglichen Zustand zurückgesetzt.

Der erste erfolgreiche Angriff erfolgte am 15. Juli um 8:55 Uhr (1 Minute nach der Freischaltung): Ein CodeRed2-Angriff auf den *IIS* Web-Server des Windows™-Honeypot.

Für die Analyse und Auswertung wurden zwei Rechner (neben dem Gateway-Rechner für die Online-Analyse) mit folgender Hardwareausstattung verwendet (Tabelle 9).

für *ACID*	für *ethereal*, *Packetyzer*
• Intel Pentium® IV mit 2,4 GHz (800 MHz Systembus) mit Hyper-Threading-Technologie • Intel i865G Chipsatz (Springdale, Dual-Channel) • 512 MB DDR Hauptspeicher • 20 GB Festplattenpartition • SuSE Linux 8.2	• AMD Athlon™ mit 1,4 GHz (Thunderbird) • SiS 735 Chipsatz • 512 MB DDR Hauptspeicher • 10 GB Festplattenpartition • Windows™ XP Professional SP1

Tabelle 9: verwendete Hardware für die Auswertung

In Diagramm 4 ist die absolute Anzahl der Alarme dargestellt, die täglich von *snort* generiert wurden. Dabei ist folgendes zu beachten: In diesen Alarmen sind natürlich auch Meldungen enthalten, die von den Honeypots selber erzeugt wurden.

Dazu zählen

- NTP-Abfragen
- DNS-Anfragen
- Kopieren der Accounting-Daten für den Linux-Honeypot
- Mail-Benachrichtigungen des Integritätscheckers
- NetBIOS-Broadcasts des Windows™-Honeypots.

Aus diesem Grund wurden auch für diese Alarme eigene Regeln für *snort* definiert (siehe Kapitel 4.5.1), um diese schnell löschen zu können. Dadurch reduziert sich die Anzahl deutlich („Alarme gefiltert“). Im späteren Verlauf des Betriebs beeinflussten diese Alarme die Gesamtanzahl weniger (ab dem 14. August).
Außerdem beinhalten die Alarme auch Verbindungen, die vom Betreuer zu Management-Zwecken initiiert wurden. Eine Erweiterungsmöglichkeit, damit in den Log-Dateien keine Management-Verbindungen gespeichert werden, wird in Kapitel 6 kurz vorgestellt.
Zwischen dem 15. Juli und dem 29. Juli versuchte der Windows™-Honeypot seinen DNS-Namen zu registrieren. Da dies im LRZ nicht unterstützt wird, versendete dieser Rechner eine Vielzahl (bis zu 3000) von Anfragen. Am 29. Juli wurde diese Funktion deaktiviert.
Am 12. August wurden die Beschränkungen am Zugangsrouter zum G-WiN für die Honeypot-Rechner aufgehoben und ein Portscan zum Überprüfen der offenen Ports dieser Rechner von einem Mitarbeiter des LRZ ausgeführt. An diesem Tag startete ebenfalls die Verbreitung des Blaster-Wurms.

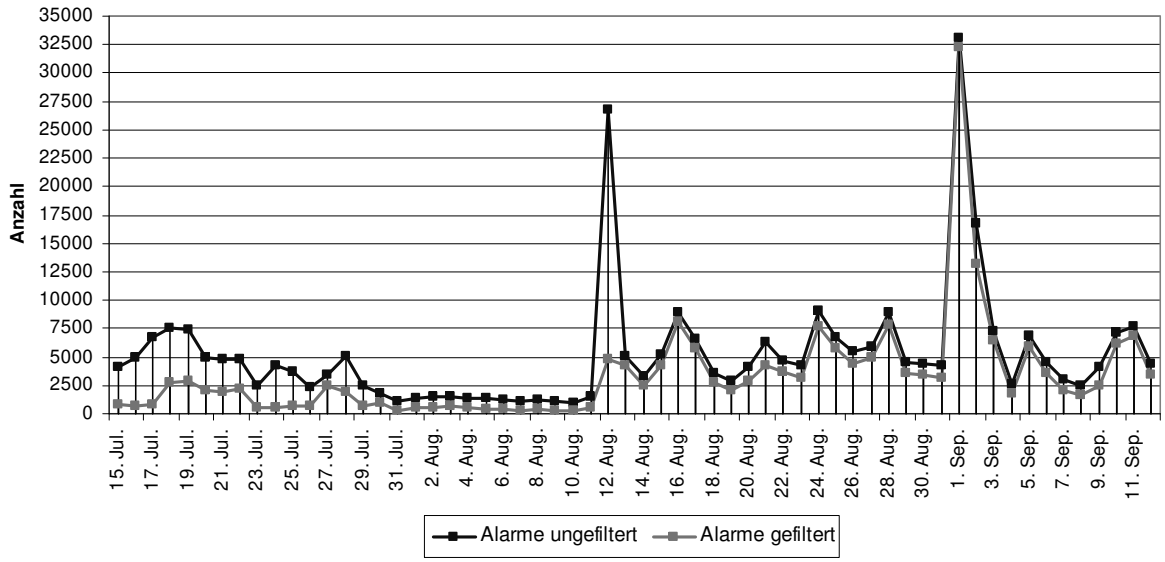

Diagramm 4: Anzahl der Alarme in *ACID* pro Tag

In den letzten Wochen des Betriebes waren deutlich höhere Anfragen auf den NetBIOS-Ports 139 und 445 zu beobachten, insbesondere am 1. September und 2. September (ohne erkennbaren Grund, z.B. neue Viruswarnungen).
Im Unterschied zum Firewall-Log, wo nur für jede neue Verbindung ein Eintrag erzeugt wird, erscheint bei *snort* für jede Aktion ein eigener Alarm, deshalb ist die Anzahl bei *snort* deutlich höher.

Eine weitere interessante Beobachtung sind die Größen der Log-Dateien. Diagramm 5 zeigt die Dateigrößen (Y-Achse in logarithmischer Skalierung) der Netzwerkdumps, die mit *tcpdump* aufgezeichnet wurden. Durch die CodeRed-Angriffe am 15. Juli, 16. Juli, 5. August und am 10. August sind die Log-Dateien auf über 100 MB angewachsen. Durch die Deaktivierung der Filterung am 12. August ist auch der Netzverkehr gestiegen, insbesondere durch „Angriffe“ auf die Ports 139 und 445 am 1. und 2. September.
Bei wenigen „Angriffen“ bzw. bei keinem „Angriff“ lag die Dateigröße vor dem 12. August bei ca. 1 MByte, danach zwischen 3 MByte und 5 MByte pro Tag.

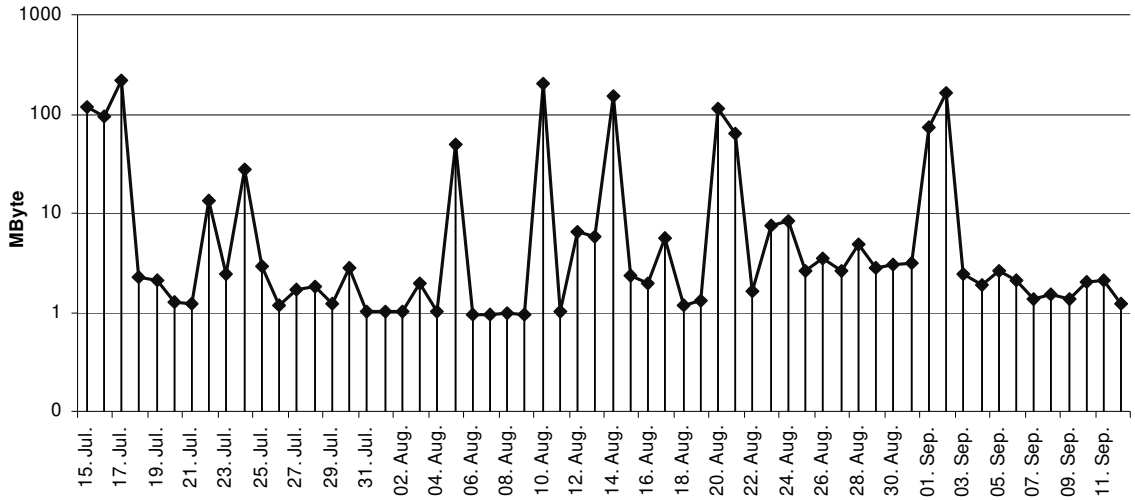

Diagramm 5: Dateigrößen der mitprotokollierten *tcpdump*-Dateien in MByte

Beim Vergleich zwischen Diagramm 4 und Diagramm 5 fällt auf, dass z.B. am 14. August der Netzwerkdump ca. 170 MB groß war, jedoch die Alarme in ACID relativ gering ausfielen. Dies ist dadurch zu erklären, dass am 13. August um 23:55 Uhr ein CodeRed2-Wurm den Windows™-Honeypot befiel und der Rechner erst am 14. August neu gestartet wurde. Die HTTP-Anfragen des Windows™-Honeypot „nach draußen“ wurden in der Netzwerkdump-Datei gespeichert, von *snort* wurden sie jedoch ignoriert, da in diesen Paketen keine relevanten Informationen stecken. Das Gleiche gilt auch für die Spitze in Diagramm 5 am 20. August.

5.5.2 Häufigkeits-Statistiken

Bei dem Betrieb eines Honeynets ist die Anfragehäufigkeit der einzelnen Ports aufschlussreich. Dadurch lässt sich z.B. analysieren, welche Ports für Angreifer interessant sind. Diagramm 6 zeigt die Statistik über die Anfragehäufigkeit der Ports (prozentual und absolut), zu denen mehr als 100 Verbindungen aufgebaut wurden. Dabei ist klar zu sehen, dass Anfragen auf Web-Server (Port 80) am häufigsten registriert wurden (36%). Auch Verbindungen zu den NetBIOS-Ports 135, 139 und 445 wurden sehr häufig erfasst. Auf Grund des Blaster-Wurms sind 98% der Verbindungen zu Port 135 von der Firewall geblockt worden. In der Ausschnittvergrößerung (es wurden die ursprünglichen 8% auf 100% hochgerechnet) kann man erkennen, dass Port-Anfragen zu bekannten Trojanern (z.B.

SubSeven) eher eine untergeordnete Rolle spielten. Interessanterweise wurden auch nur 257 Verbindungen zu den FTP-Servern aufgebaut.

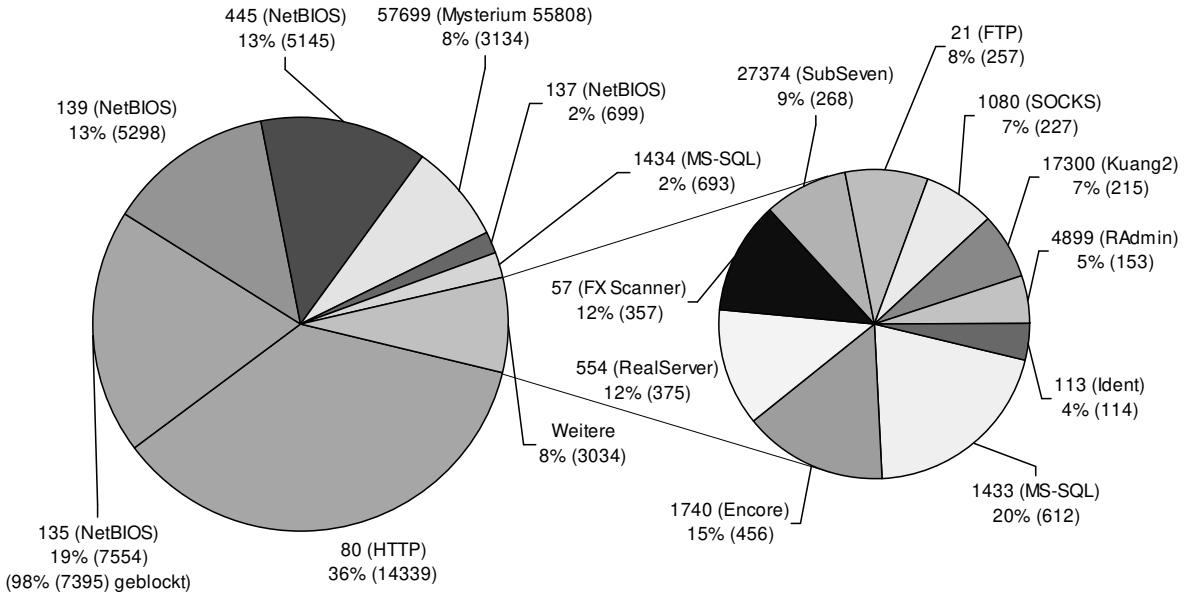

Diagramm 6: Ports zu denen mehr als 100 Verbindungen während des Betriebs aufgebaut wurden

Neben dem Angriffsziel interessiert den Betreuer eines Honeynets woher die Angriffe kamen. Dabei ist zunächst die Domain des Angreifers ein Indiz. In Diagramm 7 sind die Domainzuordnungen der eingehenden Verbindungen dargestellt. Es sind nur die 16 am häufigsten aufgetretenen Domains abgebildet: Dies entspricht mindestens 250 Verbindungen von einer Domain. Der größte Anteil der Verbindungen (40%) nehmen T-Online DSL Accounts (Domain *t-dialin.net*) ein. Bei 36% der Verbindungen war kein DNS-Name registriert. Die Domain *lrz-muenchen.de* konnte bei 3% der Verbindungen ermittelt werden. Hierbei handelt es sich um die Ausbreitungsphase des Blaster-Wurms. Im Vergleich dazu ist in Diagramm 8 die Anzahl der verschiedenen IP-Adressen der jeweiligen Domain abgebildet. 1477 IP-Adressen hatten keinen DNS-Eintrag (nicht im Diagramm dargestellt). Allerdings ist auch hier die Domain *t-dialin.net* mit 269 IP-Adressen am stärksten vertreten. Mit 169 IP-Adressen ist der amerikanische Provider *RoadRunner* (Domain *rr.com*) erfasst worden. Bei den 80 IP-Adressen der Domain *lrz-muenchen.de* handelt es sich, wie auch schon weiter oben beschrieben, fast ausschließlich um Rechner, die den Dialin-Zugang des LRZ nutzen. Bei diesen Rechnern liegt der Verdacht nahe, dass sie sehr wahrscheinlich selber Opfer des Blaster-Wurms waren und sich der Wurm weiter ausbreitet. Selbst Anfang November werden noch 86 Rechner im MWN[57] registriert, die mit dem Blaster-Wurm infiziert sind.

[57] Stand: 9. November, http://nm2.lrz-muenchen.de/infizierte-rechner/blaster.html

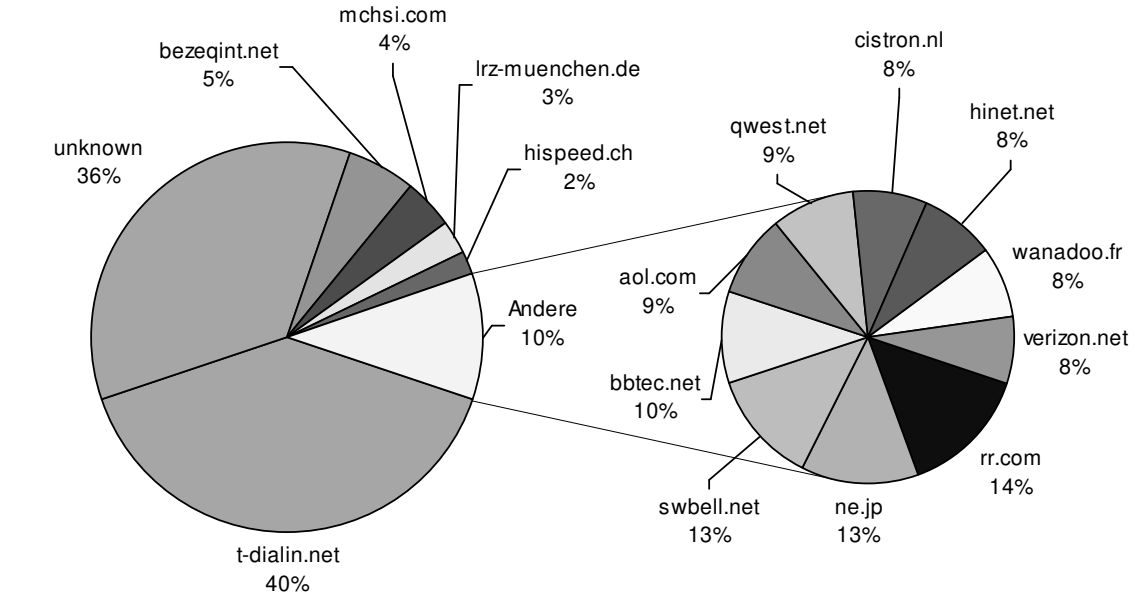

Diagramm 7: Domainzuordnungen der eingehenden Verbindungen

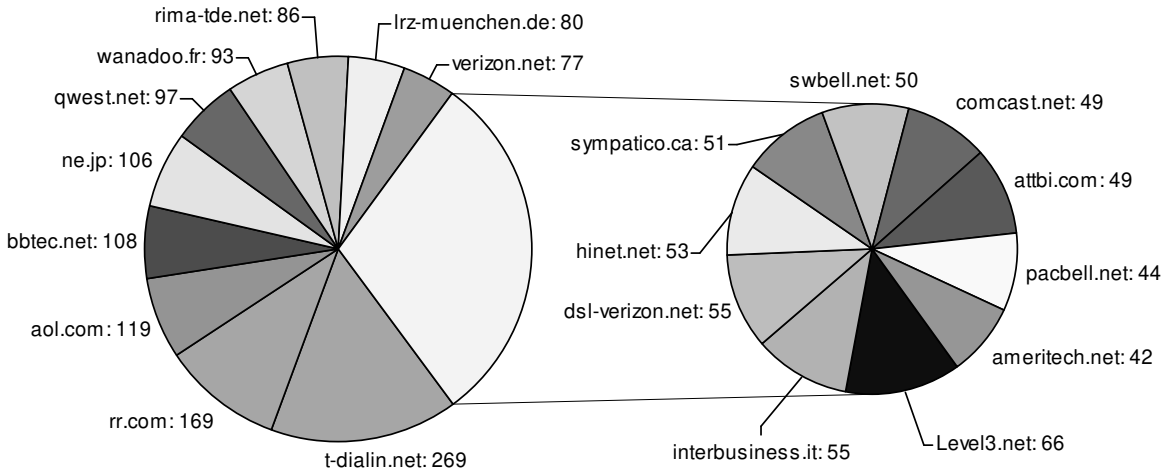

Diagramm 8: Anzahl unterschiedlicher IP-Adressen einer Domain, die Verbindungen zum Honeynet aufbauten

5.5.3 „Mysterium 55808“

Insgesamt kamen von 183 IP-Adressen 3134 TCP-Pakete, die eine *WindowSize* von 55808 aufwiesen. Außerdem war zu beobachten, dass die Anzahl der Pakete pro Tag, über die gesamte Betriebsphase abnehmend war (Diagramm 9).

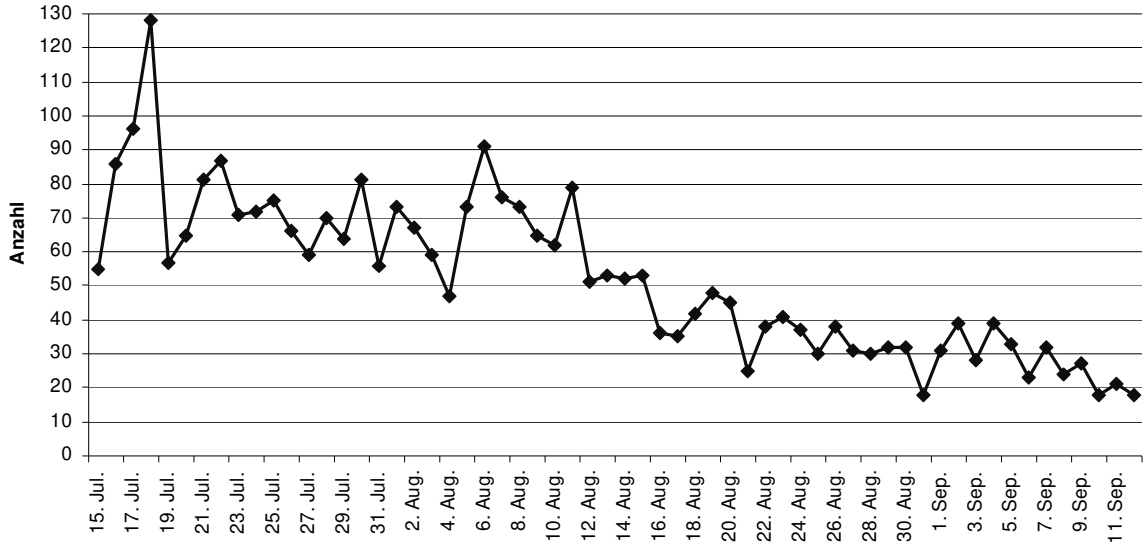

Diagramm 9: Anzahl der eingegangenen Pakete mit der *WindowSize* 55808 (*Stumbler*)

Von den meisten IP-Adressen kam nur eine einzige Anfrage. In Tabelle 10 sind die häufigsten IP-Adressen, die die *WindowSize* von 55808 hatten, aufgelistet. Bei einigen liefert die DNS-Auflösung ein Ergebnis, wobei nicht sichergestellt werden kann, dass dies auch der wirkliche Herkunftsort des Paketes ist, da die IP-Adressen von *Stumbler* gefälscht sein könnten. Würden die IP-Adressen nicht gefälscht sein, so wären einige Rechner in den USA beheimatet. Hierfür spricht die Domain-Endung „.com“ und „md“ deutet auf den US-Bundesstaat Maryland (Abkürzung: MD) hin.

IP-Adresse	DNS-Name (falls registriert)	Häufigkeit
84.185.038.091	---	2730
12.251.107.193	12-251-107-193.client.attbi.com	26
210.075.206.015	---	16
200.093.065.049	---	12
68.048.145.159	pcp01582926pcs.nrockv01.md.comcast.net	9
218.018.115.083	---	9
211.243.102.059	---	8
218.093.011.170	---	8
12.253.216.150	12-253-216-150.client.attbi.com	7
67.166.120.224	c-67-166-120-224.client.comcast.net	7
211.109.40.222	---	7
24.093.127.005	dhcp93127005.columbus.rr.com	6

Tabelle 10: *Stumbler* IP-Adressen, von denen mindestens 6 Pakete den Linux-Honeypot erreichten

5.5.4 Vergleich der Aktivitäten zwischen Windows™ und Linux

Durch die Wahl der Betriebssysteme Windows™ und Linux für die Honeypots ist interessant, ob sich daraus Erkenntnisse über das Angreiferverhalten gewinnen lassen bzw. ob es bei den Angreifern eine gewisse Präferenz für einen der beiden Honeypots gegeben hat. Es hat sich gezeigt, dass fast 28791 Verbindungen (dies entspricht knapp 70%) von 4759 verschiedenen IP-Adressen zum Windows™-Honeypot aufgebaut wurden (Tabelle 11). Beim Linux-Honeypot wurden von 4898 IP-Adressen 13066 Verbindungen registriert. Die höhere Anzahl der IP-Adressen ist zum einen durch die „Mysterium 55808“-Pakete zu erklären. Zum anderen wurden viele Verbindungen auf dem Linux-Honeypot sofort wieder geschlossen, da der angefragte Port nicht geöffnet war (z.B. Port 135, 139 oder 445). Auch bei den Anfragen auf Port 80 lässt sich erkennen, dass deutlich mehr Verbindungen zum Windows™-Honeypot aufgebaut wurden. Bei dem Port 135 ist die Verteilung auf die beiden Honeypot-Rechner nahezu identisch, da es sich hier um Wurm-Angriffe handelte.
Auch bei den Trojaner-Ports (unterer Teil der Tabelle) sind fast keine Unterschiede zwischen dem Anfrageverhalten zu sehen. Hier liegt die Vermutung nahe, dass Netzwerk-Scans auf diesen Ports durchgeführt wurden, um auf bereits installierte Trojaner zu prüfen.

	Windows™		Linux	
	IP-Adressen	Verbindungen	IP-Adressen	Verbindungen
Gesamt	4759	28971	4898	13066
Port 21	61	117	60	129
Port 80	438	13512	357	597
Port 135 (nicht geblockt)	28	69	28	90
Port 135 (geblockt)	2643	3516	2631	3879
Port 139	549	5008	90	290
Port 445	520	3818	450	1327
Port 554	68	184	67	191
Port 4000	14	44	14	48
Port 4899	24	76	24	77
Port 12345	3	9	3	9
Port 17300	38	103	41	112
Port 27374	45	138	46	130

Tabelle 11: Vergleich der Verteilung der Anfragen auf die beiden Honeypots

Betrachtet man die Anzahl der Verbindungen, die pro Tag für jeden der beiden Honeypots registriert wurden, so kann man ebenfalls Rückschlüsse auf das Angreiferverhalten ziehen. So ist in Diagramm 10 an den Wochenenden ein leichter Anstieg der Verbindungen beim Windows™-Honeypot zu erkennen. Beim Linux-Honeypot sind hingegen kaum Schwankungen festzustellen. Die Anzahl der Verbindungen stieg bei beiden Systemen am 12. August durch die Entfernung der Portsperren an.
Anfang August bis Mitte August waren sehr wenige Verbindungen zu beobachten: In dieser Zeit hatten die meisten Bundesländer Schulferien und somit Hauptreisezeit. Außerdem lagen die Tagestemperaturen über 30° C. Daraus könnte man den Schluss ziehen, dass viele „T-Online Script Kiddies“ im Urlaub oder beim Baden waren.

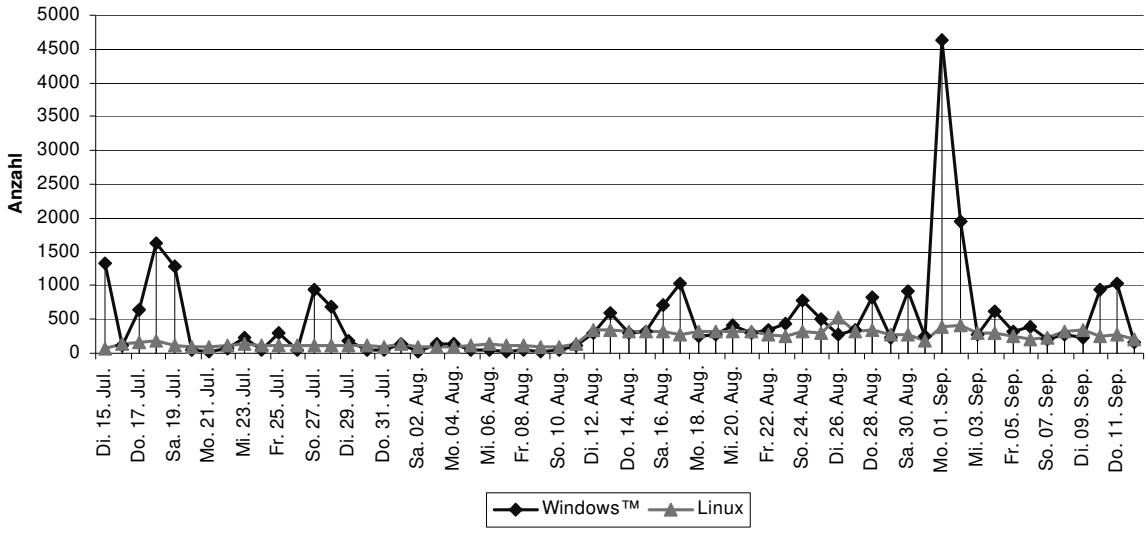

Diagramm 10: Anzahl der registrierten Verbindungen pro Tag

In Diagramm 11 ist die Anzahl der registrierten Verbindungen (Gesamt, zu dem Linux-Honeypot und zu dem Windows™-Honeypot) während der Betriebsphase abhängig von der Tageszeit dargestellt. Die Anzahl der Verbindungen nimmt dabei am Vormittag ab und steigert sich bis den Abend hinein. Auffällig ist, dass die Anzahl der Verbindungen, die als Ziel den Linux-Honeypot hatten, nahezu konstant ist. Insgesamt schwankt die Anzahl der Verbindungen zum Windows™-Honeypot relativ stark („Zick-Zack-Muster"), d.h. hier lassen sich kaum Rückschlüsse ziehen. Viele der Angriffe finden in den Nachmittagsstunden sowie in der Nacht statt.

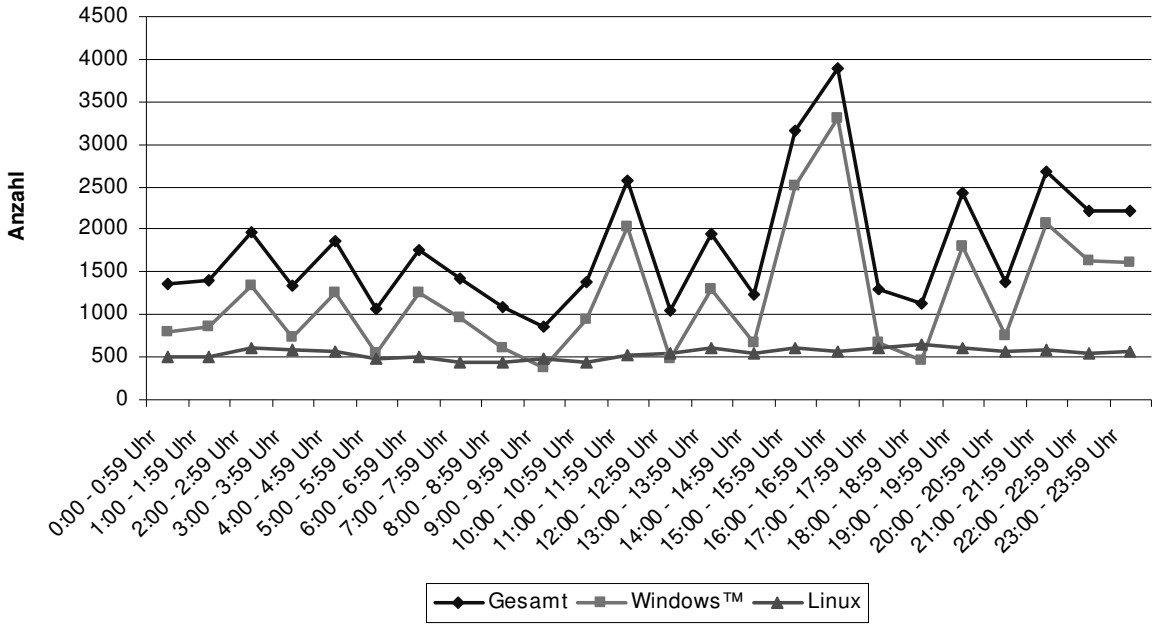

Diagramm 11: Anzahl der auf dem Gateway registrierten Verbindungen pro Stunde zwischen dem 15. Juli und dem 12. September

Bei der Analyse der unterschiedlichen IP-Adressen (unterschiedliche Rechner) sieht man die Beeinflussung der Portsperren am deutlichsten (Diagramm 12). So waren zwischen dem 15. Juli und dem 11. August zwischen 36 und 70 verschiedene IP-Adressen im Firewall-Log registriert. Im weiteren Verlauf schwankt auch hier die Anzahl der verschiedenen Rechner pro Tag stark (zwischen 236 am 1. September und 134 am 12. September). Dabei haben mehr Rechner Verbindungen zum Linux-Honeypot als zum Windows™-Honeypot aufgebaut. Der Ausbruch des Blaster-Wurms ab dem 12. August ist in diesem Diagramm gut zu erkennen. Es wird auch deutlich, dass die viele der Rechner zu beiden Honeypots Verbindungen aufbauen, da die blaue Kurve (Gesamt-Anzahl aller IP-Adressen pro Tag) meistens nicht viel über der roten Kurve (Anzahl der IP-Adressen mit Ziel-Adresse Linux-Honeypot) liegt. 4278 verschiedene Rechner stellten Anfragen an beide Honeypots, 481 unterschiedliche Rechner nur an den Windows™-Honeypot und 605 Rechner nur an den Linux-Honeypot.

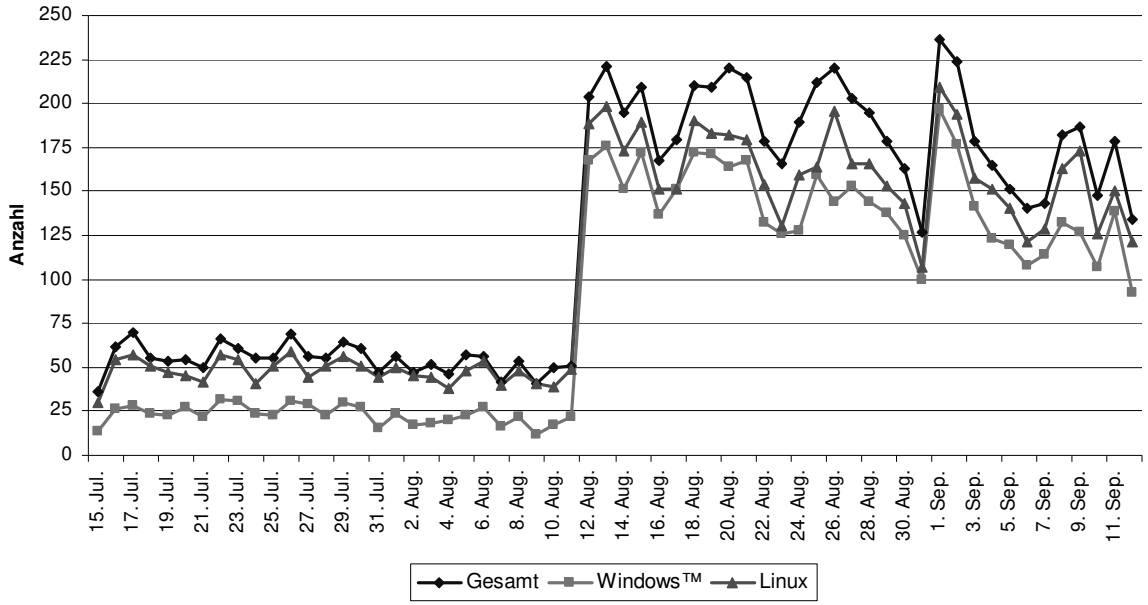

Diagramm 12: Anzahl der unterschiedlichen IP-Adressen pro Tag

Neben der Analyse der Verbindungen pro Stunde lassen sich auch die Anzahl der verschiedenen Rechner, die zu den Honeypots Verbindungen aufgebaut haben, untersuchen. In Diagramm 13 ist die Anzahl der verschiedenen IP-Adressen pro Stunde zu den Honeypots zwischen dem 15. Juli und dem 12. September dargestellt. Im Vergleich zu Diagramm 11 werden die Aktivitäten deutlicher: In den frühen Morgenstunden nehmen die Aktivitäten ab, steigen dann bis zum frühen Abend wieder an und fallen nach 21 Uhr wieder etwas ab.

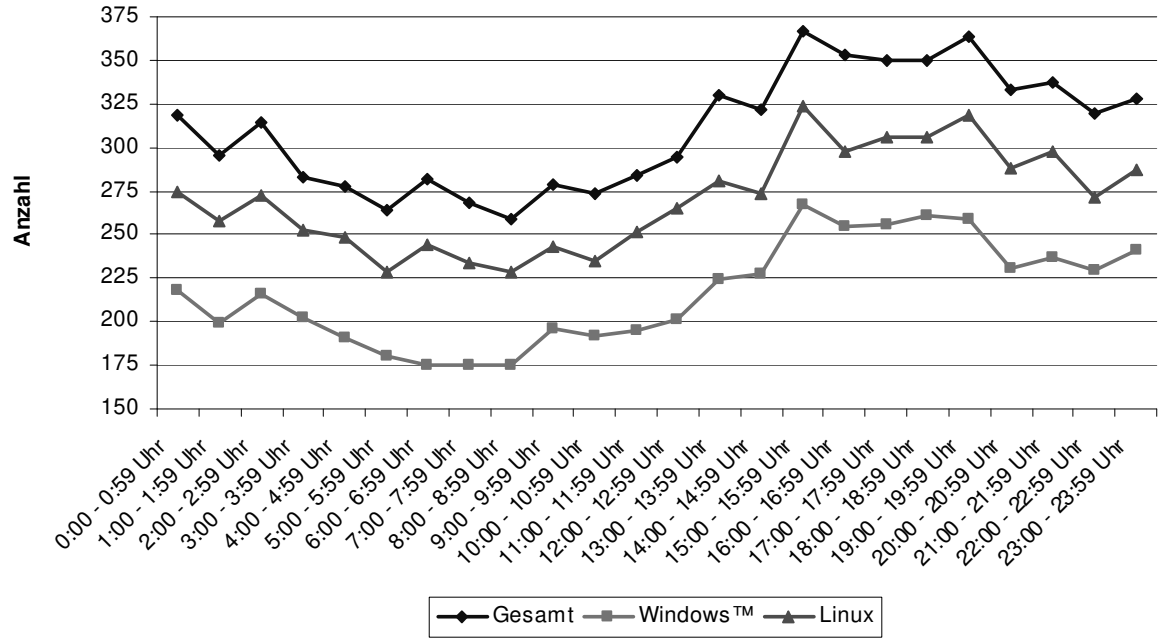

Diagramm 13: Anzahl der auf dem Gateway registrierten unterschiedlichen IP-Adressen pro Stunde zwischen dem 15. Juli und dem 12. September

Zum Abschluss dieses Kapitels erfolgt noch eine getrennte Betrachtung der Rechner, die die meisten Verbindungen zu den beiden Honeypot-Systemen aufbauten. In Tabelle 12 sind die Rechner aufgelistet, die die meisten Verbindungen zum Windows™-Honeypot, in Tabelle 13, die die meisten Verbindungen zum Linux-Honeypot aufgebaut haben.
Auffällig ist der hohe Anteil an T-Online DSL-Accounts sowie die Anzahl der Verbindungen (oft zu den Ports 80 und 139) innerhalb der meist kurzen Zeitspanne. Hier wird auch wieder deutlich, dass wesentlich mehr Verbindungen zum Windows™-Honeypot aufgebaut wurden. Das Herkunftsland wurde mit *Visualroute* bestimmt.
Auf dem Linux-Honeypot wurde von einem Rechner (Zeile 5) ein Portscan ausgeführt, jedoch ohne den Port 80 abzufragen. Bei 7 der 10 Rechner handelte es sich entweder um „Mysterium 55808"-Pakete (Kapitel 5.4.2.3) oder um Antworten eines DoS-Angriffs (Kapitel 5.4.2.5).

	IP-Adresse	**DNS-Name (Land)**	**Anzahl Verb.**	**Port (*Häufigkeit*)**	**Erstes Auftreten**	**Letztes Auftreten**
1	207.236.37.170	unknown (Kanada)	1889	ICMP (*1*) 57 (*4*) 139 (*1880*) 1433 (*4*)	01.09.2003 14:59:17	01.09.2003 15:29:27
2	80.135.118.191	p508776BF.dip.t-dialin.net	1458	ICMP (*1*) 80 (*1457*)	10.09.2003 21:58:40	11.09.2003 02:35:53
3	81.218.224.250	unknown (Israel)	1315	ICMP (*2*) 80 (*656*) 139 (*1*) 445 (*656*)	01.09.2003 16:46:39	01.09.2003 16:59:54
4	217.230.99.230	pD9E663E6.dip.t-dialin.net	1302	80 (*1302*)	15.07.2003 21:07:04	16.07.2003 14:38:09
5	12.217.27.192	12-217-27-192. client.mchsi.com (USA)	1191	ICMP (*2*) 21 (*1*) 57 (*4*) 80 (*1184*)	19.07.2003 11:20:49	19.07.2003 11:30:15
6	80.133.19.25	p50851319.dip.t-dialin.net	1000	80 (*1000*)	17.07.2003 23:18:22	18.07.2003 04:58:35
7	80.131.151.9	p50839709.dip.t-dialin.net	960	ICMP (*2*) 139 (*705*) 445 (*253*)	02.09.2003 05:34:05	02.09.2003 11:38:16
8	217.226.199.215	pD9E2C7D7.dip.t-dialin.net	893	ICMP (*2*) 57 (*3*) 80 (*888*)	27.07.2003 05:48:03	27.07.2003 06:58:13
9	217.228.35.249	pD9E423F9.dip.t-dialin.net	710	ICMP (*1*) 21 (*1*) 57 (*3*) 80 (*705*)	17.08.2003 01:56:38	17.08.2003 02:14:45
10	217.84.15.225	pD9540FE1.dip.t-dialin.net	657	ICMP (*1*) 21 (*1*) 57 (*3*) 80 (*652*)	28.07.2003 16:33:12	28.07.2003 16:53:10

Tabelle 12: Top-10 Rechner, die die meisten Verbindungen zum Windows™-Honeypot aufbauten

	IP-Adresse	DNS-Name	Anzahl Verb.	Port (*Häufigkeit*)	Erstes Auftreten	Letztes Auftreten
1	84.185.38.91	unknown	2730	57669 (*2730*)	15.07.2003 10:34:40	12.09.2003 18:15:58
2	207.71.44.121	ns2.rackspace.com (USA)	90	1504 (*2*) 1740 (*88*)	26.08.2003 00:18:23	06.09.2003 17:26:02
3	64.39.2.185	cirrus.rackspace.com (USA)	74	1740 (*74*)	26.08.2003 02:28:04	28.08.2003 04:36:27
4	81.101.117.58	pc1-nthc4-6-cust58. nrth.cable.ntl.com (GB)	58	ICMP (*17*) 135 (*41*)	02.09.2003 20:07:26	12.09.2003 10:56:34
5	62.143.1.207	ip207.1.1411J-CUD12K-02.ish.de (Deutschland)	54	ICMP (*2*) 23 (*3*) 66 (*3*) 69 (*3*) 107 (*3*) 135 (*1*) 138 (*3*) 139 (*3*) 445 (*3*) 539 (*3*) 554 (*3*) 593 (*3*) 1025 (*3*) 1080 (*3*) 1433 (*3*) 1434 (*3*) 1527 (*3*) 1529 (*3*) 4899 (*3*)	02.09.2003 03:46:31	02.09.2003 03:48:40
6	207.235.16.2	ns.rackspace.com (USA)	46	1504 (*1*) 1740 (*45*)	26.08.2003 00:55:16	28.08.2003 04:36:11
7	80.228.76.7	dynadsl-080-228-76-007. ewetel.net (Deutschland)	33	80 (*2*) 135 (*6*) 1433 (*3*) 3306 (*22*)	16.08.2003 00:57:30	16.08.2003 02:00:37
8	216.157.55.51	ns2.hostcentric.net (USA)	32	1740 (*32*)	08.09.2003 20:36:09	08.09.2003 21:06:25
9	12.251.107.193	12-251-107-193. client.attbi.com (USA)	26	57669 (*26*)	29.07.2003 10:17:59	11.09.2003 00:16:12
10	198.41.3.45	wx2.dnsmanaged.com (USA)	23	1740 (*23*)	10.09.2003 16:33:27	12.09.2003 10:51:06

Tabelle 13: Top-10 Rechner, die die meisten Verbindungen zum Linux-Honeypot aufbauten

5.5.5 Zusammenfassung der Analyse

Obwohl keine „feindliche Übernahme“ auf den Honeypots stattfand, gab es doch viele Angriffe zu analysieren.
Die Angriffe lassen sich grob in zwei Kategorien unterteilen. Bei der einen Kategorie handelte es sich, wie vermutet, überwiegend um Script Kiddies, die im Internet Werkzeuge zum Aufspüren von Schwachstellen angewendet haben (z.B. FX Scanner). Die andere Kategorie der Angriffe waren Wurm-Angriffe.
Gezielte Angriffe (z.B. auf den SSH-Server des Linux-Honeypot) waren leider nicht zu beobachten. Die einzige Ausnahme stellten hier die Login-Versuche bei der MySQL-Datenbank des Linux-Honeypot dar.
Auch die Verteilung der Angriffe war so zu erwarten gewesen: Es wurden wesentlich mehr Verbindungen zum Windows™-Honeypot registriert als zum Linux-Honeypot.
Bei der Analyse wurde davon ausgegangen, dass es sich bei den IP-Adressen um die IP-Adresse des jeweiligen Angreifers handelt. Bei einem aktiven Angreifer ist davon auszugehen, dass er ein anderes System, möglicherweise eines, dass von ihm schon erfolgreich kompromittiert wurde, für seine Angriffe verwendet. Die Herkunft der Pakete kann zwar ermittelt werden, jedoch nicht der eigentliche Ursprung der Aktivitäten.

6 Zusammenfassung und Ausblick

In der Betriebsphase des Honeynet waren zwar keine „richtigen“ Kompromittierungen zu beobachten, so dass ein Angreifer die volle Kontrolle über einen der Honeypots erhielt, jedoch konnten z.B. einige Erkenntnisse über die Ausbreitung von Würmern gewonnen werden. Besondere Beachtung verdient die gewonnene Erkenntnis über die Zusammensetzung des Datenverkehrs. Der aufgetretene Verkehr ließ sich grob in folgende Kategorien unterteilen:

- Netzverkehr zu den Web-Servern: Dieser Verkehr muss durchaus kritisch eingestuft werden, insbesondere dann, wenn der *IIS* Web-Server von Microsoft eingesetzt wird: Hier sind eine Vielzahl an Schwachstellen bekannt, für die es auch spezielle Werkzeuge gibt, um ein System auf diese Schwachstellen zu überprüfen. Das Risiko eines erfolgreichen Angriffs kann durch regelmäßige Updates sehr reduziert werden.

- Netzverkehr zu NetBIOS-Ports: Viele Computerwürmer nutzen diese Ports für ihre Ausbreitung. Hier können mit Hilfe eines Windows™-Honeynet Erkenntnisse und Rückschlüsse auf neue Würmer bzw. auf Varianten bereits bekannter Würmer gezogen werden. Dieser Netzverkehr sollte kritisch betrachtet werden. Durch das Sperren dieser Ports, wie es am G-WiN Zugang des LRZ der Fall ist, können Wurm-Angriffe dieser Art auf Produktiv-Systeme verhindert werden.

- Netzverkehr zu Trojaner-Ports: Prinzipiell handelt es sich hier um harmlosen Verkehr, solange sich auf keinem der Honeypots ein Trojaner durch einen Angreifer installiert wurde. Die betroffenen Ports sowie die einzelnen Programme sind meistens schon länger bekannt. Diese Anfragen sind Nebeneffekte von Scans in größeren Netzen.

Für den Betrieb eines Honeynet sind folgende Aspekte zu beachten:

<u>Vorteile eines Honeynet</u>

- Beobachtung und Analyse des Angreiferverhaltens: Hieraus lassen sich neue Methoden der Angreifer erkennen.

- Der Wert der gesammelten Daten ist hoch: Da jeder Netzverkehr als Angriff zu werten ist, enthalten die Daten wertvolle Information in Bezug auf sicherheitskritische Vorfälle.

- Flexibilität bei der Realisierung: Ein Honeynet kann individuell an eine Umgebung angepasst werden, z.B. eine
    - Überwachung des internen Netzverkehrs, falls ein Angreifer dort vermutet wird

  oder eine
    - Überwachung des Netzverkehrs außerhalb einer Firewall und/oder in der DMZ, falls der Verdacht besteht, dass eigene Rechner angegriffen werden.

Diese beiden Szenarien sind geeignet, um einen Angreifer „auf frischer Tat“ zu ertappen bzw. das Verhalten des Angreifers zu verstehen und zu lernen.

- Ergänzung vorhandener Sicherheitseinrichtungen: Ein Honeynet kann als Unterstützung für ein (bereits installiertes) IDS dienen oder als Frühwarnsystem eingesetzt werden (z.B. bei Wurm-Angriffen). Eine Beschreibung, wie speziell mit Honeypots Internet-Würmer bekämpft werden können, ist in [Oud 03] zu finden.

Nachteile eines Honeynet

- Beschränkte Sicht: Es können nur Angriffe beobachtet werden, wenn das Honeynet auch Ziel eines Angriffs war.

- Nicht für alle Bedrohungen geeignet: E-Mail-Viren und -Würmer können z.B. nicht mit einem Honeynet erfasst werden, da ja diese Systeme nicht produktiv genutzt werden und deshalb keine E-Mails abgerufen werden.

- Risiken für andere Systeme: Durch die Kompromittierung einer oder mehrerer Honeypots in einem Honeynet könnten Drittsysteme angegriffen werden.

- Keine zusätzliche Sicherheit: Ein Honeynet schützt verwundbare Systeme nicht. Potentielle Angreifer könnten durch ein Honeynet auf die Umgebung aufmerksam gemacht werden.

Hardware-Anforderungen

Die Leistung der eingesetzten Rechner in einem Honeynet sollte nicht zu klein dimensioniert werden. Es hat sich gezeigt, dass der Gateway-Rechner für die Online-Analyse schnell an seine Belastungsgrenzen gestoßen ist, vor allem beim Löschen von Alarmen in *ACID*. Erfolgt die Analyse auf einem anderen Rechner, so kann dieses Problem gelöst werden, jedoch erhöht sich der Netzverkehr und die übermittelten Daten könnten unter Umständen abgehört werden. Bei den Honeypots können ältere Rechner verwendet werden. Hier ist die Installation der Rechner die längere Aufgabe.

Personelle Anforderungen

Der Aufwand, den eine Honeynet-Installation erfordert, darf nicht unterschätzt werden. Dies beginnt bei der Planung, wird fortgeführt bei der Installation und Konfiguration des Gateways sowie der einzelnen Honeypots und endet in der Analyse der gewonnenen Daten. Bei der Planung ist vor allem wichtig, die Erwartungen klar zu definieren. Während der Betriebsphase ist eine kontinuierliche Überwachung unerlässlich, d.h. es müssen personelle Kapazitäten vorhanden sein. Auch muss ein gewisses Know-how über Betriebssysteme, Angriffsmöglichkeiten und Computernetze existieren.
Die Frage, ab wann bzw. ob sich der Einsatz eines Honeynet und der damit verbundene Aufwand lohnt ist schwer zu beurteilen und zu bewerten. Sind z.B. durch den Einsatz eines Honeynets Schäden verhindert worden, so hat sich der Aufwand gelohnt.

Während der Betriebsphase des Honeynets konnte die Ausbreitung des Blaster-Wurms im LRZ genau untersucht werden. Durch das verwendete Sicherheitskonzept wurden weitere Infizierungen durch die Honeypots erfolgreich verhindert, jedoch ist der Blaster-Wurm immer noch im MWN präsent. Mit Hilfe des Honeynets konnten bereits am Abend des 11. August die ersten Blaster-Angriffe beobachtet werden, während die eigentliche „Blaster-Welle“ im LRZ erst am Vormittag des 12. August begann. Durch sofortige Maßnahmen am 11. August hätte man die Ausbreitung zwar nicht stoppen, aber zumindest reduzieren können. Dazu wären allerdings eine Benachrichtigung rund um die Uhr notwendig, sowie ein Bereitschaftsdienst, der über die notwendigen Kenntnisse verfügt.

Rechtliche Aspekte

Rechtliche Aspekte wurden im Rahmen dieser Arbeit nicht betrachtet. Beispielhaft sind die folgenden drei Fragestellungen, die vor einem Betrieb zweifelsfrei geklärt werden müssen:

- **Datenschutz**: Ist es erlaubt, den übertragenen Datenverkehr zu speichern und zu analysieren?

- **Verleitung, Anstiftung**: Ist die Bereitstellung von Honeypots/Honeynets eine Verleitung bzw. Anstiftung zum Angriff dieser Systeme?

- **Haftung**: Wer muss für den entstandenen Schaden haften, falls von einem Honeypot ein Drittsystem angegriffen wird?

Lance Spitzner hat bereits Überlegungen zu diesem Thema in [Spi 03b] angestellt.

Meines Erachtens hat sich in dieser Diplomarbeit herausgestellt, dass der Einsatz eines Honeynet für Forschungseinrichtungen zwar interessant, aber für einen Dauerbetrieb, insbesondere in Unternehmen, zu betreuungsintensiv ist. Vor allem für IT-Sicherheitsunternehmen und Antiviren-Software Hersteller kann ein Honeynet aufschlussreiche Ergebnisse liefern.

Ausblick und Erweiterungen

Neben dem in dieser Arbeit vorgestellten Honeynet lassen sich natürlich eine Vielzahl von Einsatzszenarien für Honeynets vorstellen.
Erweitert man das vorgestellte Honeynet mit einem Management-Netz (Abbildung 25) so können die Management-Zugriffe über dieses Netz erfolgen, ohne auf dem Gateway-Rechner unnötige Alarme zu erzeugen. Das Problem liegt dabei im „Verstecken“ der zusätzlichen Netzwerkkarten der Honeypots, damit diese vom Angreifer nicht entdeckt werden.

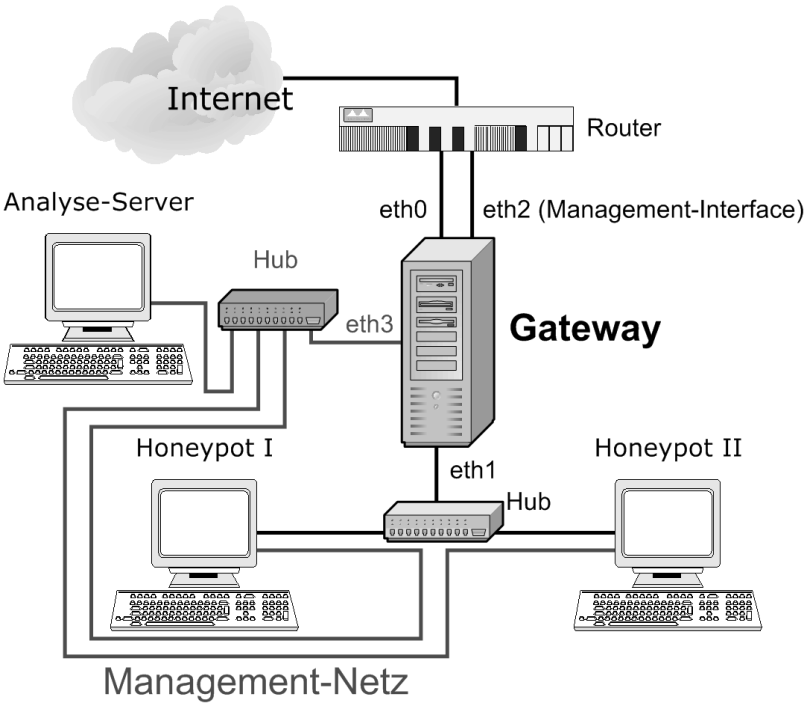

Abbildung 25: Honeynet mit Management-Netz (Erweiterungsmöglichkeit)

Um z.B. zu untersuchen, wie „verwundbar“ Windows™ 2000 Systeme mit unterschiedlichen Servicepacks sind, wäre es vorstellbar ein Honeynet aus Windows™ 2000 Systemen aufzubauen. Auf jedem der Honeypots ist ein unterschiedliches Servicepack installiert.

Bei einem Bait and Switch Honeypot[58] erfolgt die Umleitung der Angriffe von dem eigentlichen Netz auf einen Honeypot. Dazu entscheidet ein vorgeschalteter Gateway-Rechner (mit Hilfe eines IDS), ab wann es sich um einen Angriff handelt und leitet den Angreifer unbemerkt auf den Honeypot um. Für das LRZ wäre diese Technik bei Systemen interessant, bei denen bereits der Verdacht besteht, dass Angriffe ausgeübt werden.

Lance Spitzner hatte Mitte September die Idee der Dynamic Honeypots [Spi 03c] entwickelt: Die einsetzten Honeypots sollen sich dabei dynamisch ihrer Umgebung anpassen und damit die eigentliche Umgebung nachbilden. Für die Erkennung der Systeme im eigenen Netz kann z.B. die Methode des Passive Fingerprinting verwendet werden. Mit Dynamic Honeypots soll der Betreuungsaufwand verringert werden.
Diese Idee ist bisher noch nicht umgesetzt worden.

[58] http://baitnswitch.sourceforge.net/

7 Glossar

ACID	Analysis Console for Intrusion Databases
ADOdb	Active Data Objects Data Base
ARP	Address Resolution Protocol
ASCII	American Standard Code for Information Interchange
CERT	Computer Emergency Response Team
CGI	Common Gateway Interface
CPU	Central Processing Unit
DBI	Database Interface
DCOM	Distributed Component Object Model
DFN	Deutsches Forschungsnetz
DMZ	Demilitarisierte Zone bzw. demilitarized zone
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
DSN	Data Source Name
FTP	File Transfer Protocol
GUI	Graphical User Interface
G-WiN	Gigabit-Wissenschaftsnetz
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDS	Intrusion Detection System
ICMP	Internet Control Message Protocol
IIS	Internet Information Services
IP	Internet Protocol
IPC	Interprocess Communication
ISO	International Organization of Standardization
MAC	Media Access Control
LAMP	Linux, Apache, MySQL und PHP
LAN	Local Area Network
LMU	Ludwig-Maximilians-Universität München
LRZ	Leibniz-Rechenzentrum
MTU	Maximum Transmission Unit
MWN	Münchner Wissenschaftsnetz
NAT	Network Address Translation

NetBIOS	Network Basic Input/Output System
NFS	Network File System
NTFS	NT File System
NTP	Network Time Protocol
OSI	Open Systems Interconnection
PHP	PHP: Hypertext Preprocessor
RFC	Request For Comments
RPC	Remote Procedure Call
RSA	Asymmetrisches Verschlüsselungsverfahren, benannt nach Rivest, Shamir, Adleman
SMB	Server Message Block
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Socket Layer
STP	Spanning Tree Protocol
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
TLS	Transport Layer Security
TTL	Time-to-Live
TUM	Technische Universität München
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VPN	Virtual Private Network
VNC	Virtual Network Computing
YaST	Yet Another Setup Tool: Installationshilfe bei SuSE Linux

8 Literaturverzeichnis

Literatur

[Ano 03] Anonymous, *Hacker's Guide*, Markt + Technik, 2003, ISBN: 3-8272-6522-3

[Bal 03] Balas, E., *Know Your Enemy: Sebek2*, 13. September 2003 *http://www.honeynet.org/papers/sebek.pdf*

[Bka 02] Bundeskriminalamt, *Polizeiliche Kriminalstatistik 2001 – Computerkriminalität http://www.bundeskriminalamt.de/pks/pks2001/p_3_21.pdf*

[Bka 03] Bundeskriminalamt, *Polizeiliche Kriminalstatistik 2002 – Computerkriminalität http://www.bundeskriminalamt.de/pks/pks2002/p_3_21.pdf*

[BP 02] Baumann, R., Plattner, A., *Diploma Thesis: Honeypots*, Institut für Technische Informatik und Kommunikationsnetze der Eidgenössischen Technischen Hochschule Zürich, 2002 *http://security.rbaumann.net/download/diplomathesis.pdf*

[DFN 01] DFN-Betriebstagung 2001, Code Red2 – Analyse und Gegenmaßnahmen, 2001, *http://www.dfn-cert.de/dfn/bt/2001/bt-2001-codered.pdf*

[Eck 01] Eckert, C., IT-Sicherheit – Konzepte, Verfahren, Protokolle, Oldenburg Verlag, 2001, ISBN: 3-486-25298-4

[EW 01] Estermann, C., M. Waldburger, *Computerviren (Seminar IT-Sicherheit)*, Institut für Informatik - Forschungsgruppe Informations- und Kommunikations-management der Universität Zürich, Dezember 2001 *http://www.ifi.unizh.ch/ikm/Vorlesungen/Sem_Sich01/Estermann.pdf*

[Fun 03] Funk, C., *Diplomarbeit: Konzeption und Realisierung einer IDS-Prototypinstallation für das LRZ*, Institut für Informatik der Ludwig-Maximilians-Universität München, 2003

[HP 00] The Honeynet Project, *Know Your Enemy– The Tools and Methodologies of the Skript Kiddie*, 21. Juli 2000, *http://www.honeynet.org/papers/enemy/*

[HP 01] The Honeynet Project, *Know Your Enemy: II – Tracking the blackhat's moves*, 18. Juni 2003, *http://www.honeynet.org/papers/enemy2/*

[HP 02] The Honeynet Project, *Know Your Enemy: Passive Fingerprinting – Identifying remote hosts, without them knowing*, 4. März 2002, *http://www.honeynet.org/papers/finger/*

[HP 03] The Honeynet Project, *Honeynet – Definitions, Requirements and Standards*, 12. April 2003, *http://project.honeynet.org/alliance/requirements.html*

[HR 02] Hegering, H.-G., H. Reiser, *Vorlesungsunterlagen IT-Sicherheit (Sicherheit vernetzter Systeme)*, Wintersemester 2002/03, Ludwig-Maximilians-Universität München, *http://www.nm.informatik.uni-muenchen.de/Vorlesungen/ws0203/itsec/skript/k3-v0.5.pdf*

[HROG 03] Hegering, H.-G., H. Reiser, B. Oberhaitzinger, H. Gerloni, *Praktikumsunterlagen IT-Sicherheit (Sommersemester 2003)*, Ludwig-Maximilians-Universität München *http://www.nm.informatik.uni.-muenchen.de/Praktika/ss03/secp/Unterlagen/main_secp_student.pdf*

[Kro 03] Krooß, M., *Honeypots – Einsatzmöglichkeiten beim Schutz von IT-Systemen*, Seminararbeit, Fachbereich Informatik der Universität Hamburg, 15. Mai 2003, *http://www.informatik.uni-hamburg.de/RZ/lehre/18.415/seminararbeit/11_Honeypots.pdf*

[Oud 03] Oudot, L., *Fighting Internet Worms With Honeypots*, WWW-Seite, *http://www.securityfocus.com/infocus/1740*

[RFC 1244] Holbrook P., J. Reynolds, *RFC 1244: Site Security Handbook*, Juli 1991, *ftp://ftp.isi.edu/in-notes/rfc1244.txt*

[RFC 2246] Dierks, T., C. Allen, *RFC 2246: The TLS Protocol Version 1.0. RFC*, IETF, Januar 1999, *ftp://ftp.isi.edu/in-notes/rfc2246.txt*

[Spi 02] Spitzner, L., *Honeypots – Tracking Hackers*, Addison-Wesley, 2002, ISBN: 3-211-0895-7

[Spi 03a] Spitzner, L., *Honeypots – Definitions and Value of Honeypots*, 29. Mai 2003, *http://www.tracking-hackers.com/papers/honeypots.html*

[Spi 03b] Spitzner, L., *Honeypots: Are They Illegal?*, 12. Juni 2003, WWW-Seite, *http://www.securityfocus.com/infocus/1703*

[Spi 03c] Spitzner, L., *Dynamic Honeypots*, 15. September 2003, WWW-Seite, *http://www.securityfocus.com/infocus/1731*

[Sto 03] Stone, V., *W32 Deloder Worm: The Building of an Army*, 2003, *http://www.giac.org/practical/GCIH/Vance_Stone_GCIH.pdf*

[Zan 03] Zander, R., *Honeypots (Seminar Ausgewählte Aspekte zum Thema IT-Sicherheit)*, Fachbereich Informatik (Fachgebiet Sicherheit in der Informationstechnik, Prof. Dr. C. Eckert) der Technischen Universität Darmstadt, Januar 2003, *http://www.sec.informatik.tu-darmstadt.de/de/lehre/WS02-03/seminar/ausarbeitungen/Honeypots.pdf*

Internet-Quellen

[aci 03] *Analysis Console for Intrusion Databases*, WWW-Seite, *http://www.andrew.cmu.edu/~rdanyliw/snort/snortacid.html*

[ado 03] *ADOdb Database Library for PHP: Create Portable DB Apps*, WWW-Seite, *http://php.weblogs.com/ADODB*

[ant 03] Anti-Trojan, *Netbus 1.70*, WWW-Seite, *http://www.anti-trojan.net/de/trojaninfo.aspx?id=71*

[bof 03] *BackOfficerFriendly*, WWW-Seite, *http://www.nfr.com/resource/backOfficer.php*

[bsi 03] *Bundesamt für Sicherheit in der Informationstechnik*, WWW-Seite, *http://bsi.bund.de*

[cert 01] *RUS CERT® - [MS/IIS] zahlreiche Schwachstellen im Microsoft IIS 4.0/5.0*, WWW-Seite, *http://cert.uni-stuttgart.de/ticker/article.php?mid=352*

[cert 02] *CERT® Advisory CA-2002-27 Apache/mod_ssl Worm*, WWW-Seite, *http://www.cert.org/advisories/CA-2002-27.html*

[cert 03a] *CERT® Coordination Center*, WWW-Seite, *http://www.cert.org*

[cert 03b] *CERT®/CC Statistics 1988-2003*, WWW-Seite, *http://www.cert.org/stats/cert_stats.html*

[certa20 03] CERT®, *CERT® Advisory CA-2003-20 W32/Blaster worm*, 11. August 2003, WWW-Seite, *http://www.cert.org/advisories/CA-2003-20.html*

[ciac 89] The Computer Incident Advisory Capability, *Information about the PC CYBORG (AIDS) trojan horse*, 19. Dezember 1989, *http://www.ciac.org/ciac/bulletins/a-10.shtml*

[dsh 03] Dshield, *Port 4899 RADMIN?*, Mailling-Liste, WWW-Seite, *http://www.dshield.org/pipermail/list/2003-April/007742.php*

[eth 03] *Ethereal*, WWW-Seite, *http://www.ethereal.com*

[hei 03a] Heise Newsticker, *SQLSlammer -- winziger Wurm erzeugt riesigen Internet-Traffic*, 25. Januar 2003, *http://www.heise.de/newsticker/data/pab-25.01.03-000/*

[hei 03b] Heise Newsticker, *Mysterium 55808*, 20. Juni 2003, *http://www.heise.de/newsticker/data/pab-20.06.03-000/*

[heis 03] *Heise Security*, WWW-Seite, *http://www.heisec.de*

[ian 03] Internet Assigned Numbers Authority, *Port Assignments*, WWW-Seite, *http://www.iana.org/assignments/port-numbers*

[ipt 02] Garcia, G., *IPTables log analyzer*, *http://www.gege.org/iptables/*

[isc 03] *Internet Storm Center*, WWW-Seite, *http://isc.incidents.org/*

[iss 03] Infoworld, *ISS reports snort vulnerability*, 4. März 2003 *http://www.infoworld.com/article/03/03/04/HNsnort_1.html*

[jpg 03] *JpGraph - OO Graph Library for PHP*, WWW-Seite, *http://www.aditus.nu/jpgraph/index.php*

[lrzntp 03] Shabani, S., W. Beyer, *NTP-Zeitserver im MWN*, LRZ, 22. August 2003, WWW-Seite, *http://www.lrz-muenchen.de/services/netzdienste/ntp/*

[lrzport 03] Leibniz-Rechenzentrum, *Einschränkungen und Regeln im Netzbetrieb*, 13. August 2003, WWW-Seite, *http://www.lrz-muenchen.de/services/netz/einschraenkung/*

[mcc 02] McCain J., *Re: [Snort-users] Proxy Scanner?*, 20. Dezember 2002, Neohapsis Mail Archive, WWW-Seite, *http://archives.neohapsis.com/archives/snort/2002-12/0521.html*

[mkb 03] Microsoft, *Description of the Windows File Protection Feature*, 14. Oktober 2003, WWW-Seite, *http://support.microsoft.com/support/kb/articles/Q222/1/93.ASP*

[msb52 00] Microsoft, *Microsoft Security Bulletin (MS00-052) – Patch Available for „Relative Shell Path" Vulnerability*, 28. Juli 2000, WWW-Seite, *http://www.microsoft.com/technet/security/bulletin/MS00-052.asp*

[msb01 03] Microsoft, *Microsoft Security Bulletin MS03-001 – Unchecked Buffer in Locator Service Could Lead to Code Execution (810833)*, WWW-Seite, *http://www.microsoft.com/technet/security/bulletin/MS03-001.asp*

[msb26 03] Microsoft, *Microsoft Security Bulletin (MS03-026) – Buffer Overrun In RPC Interface Could Allow Code Execution (823980)*, 16. Juli 2003, WWW-Seite, *http://www.microsoft.com/technet/security/bulletin/MS03-026.asp*

[na 03] Network Associates, *W32/Randbot.worm*, WWW-Seite, *http://vil.nai.com/vil/content/v_100401.htm*

[net 03] *Netlexikon von akademie.de*, WWW-Seite, *http://netlexikon.akademie.de/*

[ngr 03] *Ngrep*, WWW-Seite, *http://ngrep.sourceforge.net/*

[p0f 03] Zalewski, M., *The new p0f*, WWW-Seite, *http://lcamtuf.coredump.cx/p0f.shtml*

[pac 03] *Packetyzer*, WWW-Seite, *http://www.packetyzer.com*

[par 03] *Partimage for Linux Homepage*, WWW-Seite, *http://www.partimage.org*

[sans 03] *SANS*, WWW-Seite, *http://www.sans.org*

[sf 03] Security Focus, *Real Networks Helix Universal Server Remote Buffer Overflow Vulnerability*, 22. August 2003, WWW-Seite, *http://www.securityfocus.com/bid/8476*

[sky 00] G-Lock Software, *Skydance*, WWW-Seite, *http://www.glocksoft.com/trojan_list/SkyDance.htm*

[smi 03] Smith, C., *Eventlog to Syslog Utility*, 12. Juni 2003, *https://engineering.purdue.edu/ECN/Resources/Documents/UNIX/evtsys*

[sym 03] Symantec, *W32.HLLW.Gaobot.AA*, 21. August 2003, WWW-Seite, *http://www.symantec.com/avcenter/venc/data/w32.hllw.gaobot.aa.html*

[tcp 03] *Tcpdump*, WWW-Seite, *http://www.tcpdump.org/*

[tra 99] *Traceroute*, WWW-Seite, *http://freshmeat.net/projects/traceroute/?topic_id=150*

[tri 03a] *Tripwire*, WWW-Seite, *http://www.tripwire.org*

[tri 03b] *Tripwire 30 Tage Testversion für Windows™*, *http://www.tripwire.com/downloads/index.cfm*

[vis 03] *Visualroute*, WWW-Seite, *http://www.visualware.com/personal/products/visualroute/index.html*

Software

[bfw 02] *Kernel-Patch für Linux-Kernel 2.4.19 (oder neuer) für Bridge-Firewalling*, *http://bridge.sourceforge.net/devel/bridge-nf/bridge-nf-0.0.7-against-2.4.19.diff*

[bri 03] *Bridge-Utilities für das Management von Bridges unter Linux*, *http://bridge.sourceforge.net/bridge-utils/bridge-utils-0.9.6.tar.gz*

[com 03] SecurIT Informatique Inc, *Comlog 1.05*, WWW-Seite, *ftp://ftp.digitalvoodoo.org/pub/mirrors/securit/comlog105free.zip*

[con 03] Spitzner L., *The Honeynet Project, Shell-Skript zur Konvertierung von snort-Regeln für snort_inline*, 27. November 2002, *http://www.honeynet.org/papers/honeynet/tools/convert.sh*

[hsn 03] The Honeynet Project, *Start-Skript für snort*, *http://www.honeynet.org/papers/honeynet/tools/snort.sh*

[hsi 03] The Honeynet Project, *Start-Skript für snort_inline*, *http://www.honeynet.org/papers/honeynet/tools/snort_inline.sh*

[lkp 03] *Kernel-Patch für Linux-Kernel 2.4.18 für syslog*, *http://www.axehind.com/honeynet/exec_syslog_linux_2.4.18.patch*

[mil 03] McMillen R., *Firewall-Skript für Honeynets*, *http://www.honeynet.org/tools/rc.firewall*

[seb 03a] Balas, E., *The Honeynet Project – Sebek*, *http://www.honeynet.org/papers/honeynet/tools/sebek-linux-2.0.1.tar.gz*

[seb 03b] Balas, E., *The Honeynet Project – Sebeksniff, http://www.honeynet.org/papers/honeynet/tools/sebeksniff-2.0.1.tar.gz*

[si 03] The Honeynet Project, *snort_inline Toolkit*, *http://www.honeynet.org/papers/honeynet/tools/snort_inline.tgz*

A Anhang

A.1 Skripte für Honeynet-Rechner

Im Folgenden werden die wichtigsten Skripte und Programme, die für den Betrieb des Honeynets verwendet wurden, angegeben.

A.1.1 Gateway

Der Gateway-Rechner übernimmt die zentrale Rolle des Honeynets. Neben dem Firewall-Skript sind auch Skripte zur Benachrichtigung (z.B. per SMS notwendig).

A.1.1.1 Firewall-Skript

Das zentrale Kontrollskript auf dem Gateway-Rechner ist das Firewall-Skript. Als Grundlage diente das vom Honeynet Project entwickelte Skript `rc.firewall` [mil 03].

```
#!/bin/bash
#
# **********************
# Set mode to bridging
# **********************
#
MODE="bridge"
#
# **********************
# Set the IP-Addresses of the honeypots
# **********************
#
PUBLIC_IP="129.187.19.105 129.187.19.106"
#
# **********************
# Set the external interface
# **********************
#
INET_IFACE="eth0"                       # D-Link-NIC (external)
#
# **********************
# Set the internal interface
# **********************
#
LAN_IFACE="eth1"                        # Intel-NIC (internal)
LAN_BCAST_ADDRESS="129.187.19.111"      # IP Broadcast range for internal network
#
# **********************
# Enable queue support for snort_inline
# **********************
#
QUEUE="yes"
#
# **********************
# Set the connection limits
# **********************
#
SCALE="day"                             # second, minute, hour, etc.
TCPRATE="15"                            # Number of TCP connections per $SCALE
UDPRATE="20"                            # Number of UDP connections per $SCALE
ICMPRATE="10"                           # Number of ICMP connections per $SCALE
OTHERRATE="15"                          # Number of other IP connections per $SCALE
#
```

```
# **********************
# Enable sebek support
# **********************
#
SEBEK="yes"
#
SEBEK_FATE="ACCEPT"
#
SEBEK_DST_IP="129.187.19.108"           # set to unused IP-Address
SEBEK_DST_PORT="3141"                   # different from standard port (1101)!
#
SEBEK_LOG="no"
#
# **********************
# Set the DNS-Servers
# **********************
#
DNS_SVRS="129.187.10.25 129.187.16.1"
#
# **********************
# Set the management interface
# **********************
#
MANAGE_IFACE="eth2"                     # Compaq-NIC
MANAGE_IP="129.187.18.202"
MANAGE_NETMASK="255.255.255.0"
#
ALLOWED_TCP_IN="9004 36987"             # WWW + SSH on non standard port
MANAGER="141.84.218.33 141.84.218.1 131.159.4.197 129.187.15.176"
#        pcheger3,     sunheger1,   athalle5,     LRZ-Rechner
#
# **********************
# Set the logging interface
# **********************
#
LOG_IFACE="eth3"                        # 3com-NIC
LOG_IP="129.187.19.107"                 # Non-mapped IP

# **********************
# Incoming server without logging
# **********************
#
WWWSRC_IP="10.155.10.35"                # Source of incoming Web-Data
WWWDST_IP="129.187.19.105"              # Destination of incoming Web-Data

# **********************
# Enable outbound traffic generated by the firewall
# **********************
#
RESTRICT="yes"
#
ALLOWED_UDP_OUT="53 123"
ALLOWED_TCP_OUT="22 25 43 80 443 36999"
#
# **********************
# Set PATH
# **********************
#
PATH="/sbin:/usr/sbin:/usr/local/sbin:/bin"

##########################################
##########################################
##########################################

# ***********
# Flush rules
# ***********
iptables -F
iptables -F -t nat
iptables -F -t mangle
iptables -X

echo ""
# ***********
# Starting bridge mode
# ***********
if [ $MODE = "bridge" ]
then
```

```
   echo "Starting up Bridging mode ... "

   # ***********
   # Cleaning up the bridge.
   # ***********
   brctl delif br0 ${INET_IFACE} 2> /dev/null
   brctl delif br0 ${LAN_IFACE} 2> /dev/null
   brctl delif br0 ${LOG_IFACE} 2> /dev/null
   ifconfig br0 down 2> /dev/null
   brctl delbr br0 2> /dev/null

   # ***********
   # Disabling arp information of the interfaces
   # ***********
   ifconfig $INET_IFACE 0.0.0.0 up -arp
   ifconfig $LAN_IFACE 0.0.0.0 up -arp
   ifconfig $LOG_IFACE 0.0.0.0 up -arp


   # ***********
   # Let's start the bridge
   # ***********
   brctl addbr br0
   brctl addif br0 ${LAN_IFACE}
   brctl addif br0 ${INET_IFACE}
   brctl addif br0 ${LOG_IFACE}

   # ***********
   # Disable BPDUs
   # ***********
   brctl stp br0 off

   # ***********
   # Enable Management-Interface
   # ***********
   ifconfig $MANAGE_IFACE $MANAGE_IP netmask $MANAGE_NETMASK up
   ifconfig br0 0.0.0.0 up -arp
fi
echo "done."

# ***********
# Let's figure out dns
# ***********
if [ $DNS_HOST -z ]
then
   if [ $MODE = "bridge" ]
   then
      DNS_HOST=$PUBLIC_IP
   else
      DNS_HOST=$HPOT_IP
   fi
fi

# ***********
# Load all required IPTables modules
# ***********

# ***********
# Needed to initially load modules
# ***********
/sbin/depmod -a

# ***********
# Add iptables target LOG.
# ***********
modprobe ipt_LOG

# ***********
# Enable iptables QUEUE support
# ***********
if test $QUEUE = "yes"
then
  # ***********
  # Insert kernel mod
  # ***********
  modprobe ip_queue
```

```
  # ***********
  # check to see if it worked, if not exit with error
  # ***********
  lsmod | grep ip_queue
  IPQUEUE=$?

  if [ "$IPQUEUE" = 1 ]; then
    echo ""
    echo "It appears you do not have the ip_queue kernel module compiled"
    echo "for your kernel.  This module is required for Snort-Inline and"
    echo "QUEUE capabilities.  You either have to disable QUEUE, or compile"
    echo "the ip_queue kernel module for your kernel.  This module is part"
    echo "of the kernel source."
    exit
  fi
   echo "Enabling Snort-Inline capabilities, make sure Snort-Inline is"
   echo "running in -Q mode, or all outbound traffic will be blocked"
fi

# ***********
# Support for connection tracking of FTP and IRC.
# ***********
modprobe ip_conntrack_ftp
modprobe ip_conntrack_irc

# ***********
# Enable ip_forward
# ***********
echo "1" > /proc/sys/net/ipv4/ip_forward

# ***********
# Create protocol handling chains
# ***********
if [ -z $STOP_OUT ] || [ "$STOP_OUT" != "yes" ]
then
   iptables -N tcpHandler
   iptables -N udpHandler
   iptables -N icmpHandler
   iptables -N otherHandler
fi

# Since this is a bridge, we want to allow broadcast.  By default, we allow all
# inbound traffic (including broadcast).  We also want to allow outbound broadcast
# (such as NetBIOS) but we do not want to count it as an outbound session.  So
# we allow it here *before* we begin counting outbound connections
iptables -A FORWARD -d ${LAN_BCAST_ADDRESS} -j ACCEPT
iptables -A FORWARD -d 255.255.255.255 -j ACCEPT

### Inbound SSH to logserver
iptables -A FORWARD -p tcp -s $MANAGE_IP -d $LOG_IP --dport 36999 -m state↵
--state NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp -s 129.187.254.32 -d 129.187.18.202 -m state↵
--state NEW,ESTABLISHED,RELATED -j ACCEPT

iptables -A FORWARD -p tcp -s 129.187.51.0/24 -d 129.187.19.106 --dport 5978 -m state↵
--state NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -p tcp -d 129.187.51.0/24 -s 129.187.19.106 --sport 5978 -m state↵
--state ESTABLISHED,RELATED -j ACCEPT

### Allow NTP at syslog server
iptables -A FORWARD -i $LOG_IFACE -p udp -d 129.187.254.32 --dport 123 -m state↵
--state NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i $INET_IFACE -p udp -s 129.187.254.32 --sport 123 -m state↵
--state NEW,ESTABLISHED,RELATED -j ACCEPT

### Allow NTP from the Honeypots
for host in ${PUBLIC_IP}; do
    iptables -A FORWARD -p udp -i $LAN_IFACE -s ${host} -d 129.187.254.32 --dport 123↵
    -j ACCEPT
done

### Drop traffic from the Internet to the logserver
iptables -A FORWARD -i $INET_IFACE -d $LOG_IP -m state --state NEW,ESTABLISHED,RELATED -j DROP

### Inbound SSH to internal (webpages)
iptables -A FORWARD -p tcp -s $WWWSRC_IP -d $WWWDST_IP --dport 22 -m state↵
```

```
--state NEW,ESTABLISHED,RELATED -j ACCEPT

### Inbound TCP_DROP
iptables -A FORWARD -i $INET_IFACE -p tcp --dport 135 -m state --state NEW -j LOG↵
--log-prefix "HW IN_135DROP "
iptables -A FORWARD -i $INET_IFACE -p tcp --dport 135 -m state --state NEW -j DROP

### Inbound TCP
iptables -A FORWARD -i $INET_IFACE -p tcp -m state --state NEW -j LOG↵
--log-prefix "HW IN_TCP "
iptables -A FORWARD -i $INET_IFACE -p tcp -m state --state NEW -j ACCEPT

### Inbound UDP
iptables -A FORWARD -i $INET_IFACE -p udp -m state --state NEW -j LOG↵
--log-prefix "HW IN_UDP "
iptables -A FORWARD -i $INET_IFACE -p udp -m state --state NEW -j ACCEPT

### Inbound ICMP
iptables -A FORWARD -i $INET_IFACE -p icmp -m state --state NEW -j LOG↵
--log-prefix "HW IN_ICMP "
iptables -A FORWARD -i $INET_IFACE -p icmp -m state --state NEW -j ACCEPT

### Inbound anything else
iptables -A FORWARD -i $INET_IFACE -m state --state NEW -j LOG --log-prefix "HW IN_OTHER "
iptables -A FORWARD -i $INET_IFACE -m state --state NEW -j ACCEPT

# The remainder of established connections will be ACCEPTED.  The rules above
#   are required in order to log new inbound connections.
iptables -A FORWARD -i $INET_IFACE -j ACCEPT


#  Okay, this is where the magic all happens.  All outbound traffic is counted,
#  logged, and limited here.   Targets (called Handlers) are what actually limit
#  the connections.  All 'Handlers' are defined at the bottom of the script.


# This rule is for use with sebek.  If sebek is used, and we don't want
#   the logs filled by SPOOFED SOURCE entries because sebek uses spoofed
#   ips, we should drop all traffic in the sebek ip range.
if [ "$SEBEK" = "yes" ]
then
        if [ "$SEBEK_LOG" = "yes" ]
        then
                iptables -A FORWARD -i $LAN_IFACE -p udp -d $SEBEK_DST_IP -dport↵
                $SEBEK_DST_PORT -j LOG --log-prefix "SEBEK"
        fi
        iptables -A FORWARD -i $LAN_IFACE -p udp -d $SEBEK_DST_IP --dport $SEBEK_DST_PORT↵
        -j $SEBEK_FATE
fi

# Allow syslog-packets to server
iptables -A FORWARD -i $LAN_IFACE -p udp --sport 514 --dport 514 -d $LOG_IP -j ACCEPT
iptables -A FORWARD -i $LAN_IFACE -o $LOG_IFACE -p udp --dport 514 -d $LOG_IP -j ACCEPT

# Allow NTP at syslog server
iptables -A FORWARD -i $LOG_IFACE -p udp --dport 123 -m state↵
--state NEW,ESTABLISHED,RELATED-j ACCEPT

# Allow SSH from syslogserver to Mgmt-Interface
iptables -A FORWARD -p tcp -s $LOG_IP --sport 36999 -d $MANAGE_IP -m state↵
--state NEW,ESTABLISHED,RELATED -j ACCEPT

# Block everything else to the syslogserver
iptables -A FORWARD -o $LOG_IFACE -j DROP

### DNS / NTP Perhaps one of your honeypots needs consistent
### outbound access to provide internal service.
for srvr in ${DNS_SVRS}; do
    for host in ${DNS_HOST}; do
        iptables -A FORWARD -p udp -i $LAN_IFACE -s ${host} -d ${srvr} --dport 53 -j ACCEPT
        iptables -A FORWARD -p tcp -i $LAN_IFACE -s ${host} -d ${srvr} --dport 53 -j ACCEPT
    done
done

if [ $MODE = "nat" ]
then
```

```
    LIMIT_IP=$HPOT_IP
elif [ $MODE = "bridge" ]
then
    LIMIT_IP=$PUBLIC_IP
fi

### Count and limit all other outbound connections
if [ -z $STOP_OUT ] || [ "$STOP_OUT" != "yes" ]
then
   for host in ${LIMIT_IP}; do

# This will ensure we don't restrict Honeypots talking to eachother, and
#   we don't log them as outbound connections (in bridge mode, the
#   firewall sees all packets; therefore, we have to make sure it doesn't
#   log packets incorrectly and give false positives).
#      If you do not want to see this log, comment out the logging rule.
#   You will still need the ACCEPT rule to ensure they honeypots can talk
#   to eachother freely.
      iptables -A FORWARD -i $LAN_IFACE -o $LAN_IFACE -j LOG --log-prefix↵
      "HW Honeypot->Honeypot"
      iptables -A FORWARD -i $LAN_IFACE -o $LAN_IFACE -j ACCEPT

#  TCP
#        This next rule is the connection limiter.  If it has not exceeded
#        the limit, the packet will be sent to the tcpHandler.  The
#        tcpHandler will log and either QUEUE or ACCEPT depending on
#        the Architecture selected.
#
#     NOTE: The purpose of the drop rule is to ensure we can catch 'other'
#        protocols that enter our network.  If this statement is not here
#        we will get false log entries stating Drop other after xxx
#        connections.
      iptables -A FORWARD -p tcp -i $LAN_IFACE -m state --state NEW -m limit ↵
      --limit ${TCPRATE}/${SCALE} --limit-burst ${TCPRATE} -s ${host} -j tcpHandler
      iptables -A FORWARD -p tcp -i $LAN_IFACE -m state --state NEW -m limit↵
      --limit 1/${SCALE} --limit-burst 1  -s ${host} -j LOG↵
      --log-prefix "HW OUT_DROP_TCP_${TCPRATE} "
      iptables -A FORWARD -p tcp -i $LAN_IFACE -m state --state NEW -s ${host} -j DROP

   # This rule is for Mike Clark in order to give him RELATED information.  For
   #   example, this will tell him the data channel related to an ftp command
   #   channel of a connection.
      iptables -A FORWARD -p tcp -i $LAN_IFACE -m state --state RELATED↵
      -s ${host} -j tcpHandler

#
#  UDP - see TCP comments above.
#
      iptables -A FORWARD -p udp -i $LAN_IFACE -m state --state NEW -m limit↵
      --limit ${UDPRATE}/${SCALE} --limit-burst ${UDPRATE} -s ${host} -j udpHandler
      iptables -A FORWARD -p udp -i $LAN_IFACE -m state --state NEW -m limit↵
      --limit 1/${SCALE} --limit-burst 1 -s ${host} -j LOG↵
      --log-prefix "HW OUT_DROP_UDP_${UDPRATE} "
      iptables -A FORWARD -p udp -i $LAN_IFACE -m state --state NEW -s ${host} -j DROP

#
#  ICMP - see TCP comments above.
#
      iptables -A FORWARD -p icmp -i $LAN_IFACE -m state --state NEW -m limit↵
      --limit ${ICMPRATE}/${SCALE} --limit-burst ${ICMPRATE} -s ${host} -j icmpHandler
      iptables -A FORWARD -p icmp -i $LAN_IFACE -m state --state NEW -m limit↵
      --limit 1/${SCALE} --limit-burst 1 -s ${host} -j LOG↵
      --log-prefix "HW OUT_DROP_ICMP_${ICMPRATE} "
      iptables -A FORWARD -p icmp -i $LAN_IFACE -m state --state NEW -s ${host} -j DROP

#
#  EVERYTHING ELSE - see TCP comments above.
#
      iptables -A FORWARD -i $LAN_IFACE -m state --state NEW -m limit↵
      --limit ${OTHERRATE}/${SCALE} --limit-burst ${OTHERRATE} -s ${host} -j otherHandler
      iptables -A FORWARD -i $LAN_IFACE -m state --state NEW -m limit↵
      --limit 1/${SCALE} --limit-burst 1 -s ${host} -j LOG↵
      --log-prefix "HW OUT_DROP_OTHER_${OTHERRATE} "
   done
```

```
# This portion of the script will ensure that established or related
#    connections that were allowed, continue to work.  If these lines
#    are not here, only the first packet of each connection that hasn't
#    reached the limit will be allowed in because we are dropping
#    all outbound connections by default.
   if test $QUEUE = "yes"
      then
      iptables -A FORWARD -i $LAN_IFACE -m state --state RELATED,ESTABLISHED -j QUEUE
   fi
   iptables -A FORWARD -i $LAN_IFACE -m state --state RELATED,ESTABLISHED -j ACCEPT

###  These define the handlers that actually limit outbound connection.
#
# tcpHandler - The only packets that should make it into these chains are new
#                connections, as long as the host has not exceeded their limit.
#
   iptables -A tcpHandler -j LOG --log-prefix "HW OUT_TCP "
   if test $QUEUE = "yes"
      then
      iptables -A tcpHandler -j QUEUE
   fi
   iptables -A tcpHandler -j ACCEPT

#
# udpHandler - see tcpHandler comments above.
#
   iptables -A udpHandler -j LOG --log-prefix "HW OUT_UDP "
   if test $QUEUE = "yes"
      then
      iptables -A udpHandler -j QUEUE
   fi
   iptables -A udpHandler -j ACCEPT

#
# icmpHandler - see tcpHandler comments above.
#
   iptables -A icmpHandler -j LOG --log-prefix "HW OUT_ICMP "
   if test $QUEUE = "yes"
      then
      iptables -A icmpHandler -j QUEUE
   fi
   iptables -A icmpHandler -j ACCEPT

#
# otherHandler - see tcpHandler comments above.
#
   iptables -A otherHandler -j LOG --log-prefix "HW OUT_OTHER "
   if test $QUEUE = "yes"
      then
      iptables -A otherHandler -j QUEUE
   fi
   iptables -A otherHandler -j ACCEPT
fi # STOP_OUT

iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

### Lets make sure our firewall can talk to itself
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT


##############################
# MANAGEMENT INTERFACE RULES #
##############################
if [ $MANAGE_IFACE != "none" ]
then
   for ports in $ALLOWED_TCP_IN; do

      if [ "$MANAGER" = "any" ]
      then
         iptables -A INPUT -i $MANAGE_IFACE -p tcp --dport $ports -m state --state NEW↵
         -j ACCEPT
      else
         for ips in $MANAGER; do
            iptables -A INPUT -i $MANAGE_IFACE -p tcp -s $ips --dport $ports↵
            -m state --state NEW -j ACCEPT
         done
      fi
```

```
   done
        iptables -A OUTPUT -o $MANAGE_IFACE -p tcp -m state↵
        --state RELATED,ESTABLISHED -j ACCEPT
fi

### Set default policies for the INPUT, FORWARD and OUTPUT chains
# By default, drop all connections sent to firewall
iptables -P INPUT DROP

# If we selected to restrict the firewall, lets implement it here.
if [ $RESTRICT = "yes" ]
then

   for port in $ALLOWED_TCP_OUT; do
      iptables -A OUTPUT -p tcp --dport $port -m state↵
      --state NEW,ESTABLISHED,RELATED -j ACCEPT
   done

   for port in $ALLOWED_UDP_OUT; do
      iptables -A OUTPUT -p udp --dport $port -m state↵
      --state NEW,ESTABLISHED,RELATED -j ACCEPT
   done

   # By default, drop firewall outbound connection
   iptables -P OUTPUT DROP

else

   # By default, accept firewall outbound connection
   iptables -P OUTPUT ACCEPT

fi

iptables -P FORWARD DROP

### set new default gateway
route add default gw 129.187.18.254
```

A.1.1.2 Regeln für *snort*

Zusätzlich installierte Regeln, um die Analyse und Auswertung einfacher durchführen zu können.

```
pass udp any 520 <> 255.255.255.255/32 520 (msg:"Routing Information"; session:printable;)
pass udp $HOME_NET 514 <> 129.187.19.107/32 514 (msg:"syslog"; session:printable;)
pass tcp 129.187.51.0/24 any <> 129.187.19.106/32 5978 (msg:"VNC"; session:printable;)
alert tcp 129.187.10.25/32 53 <> 129.187.19.106/32 any (msg:"DNS out (TCP)";
       session:printable; classtype:honeynet;)
alert udp 129.187.10.25/32 53 <> $HOME_NET  any (msg:"DNS out (UDP)"; session:printable;
       classtype:honeynet;)
alert tcp 129.187.254.111/32 25 <> 129.187.19.106/32 any (msg:"Mailserver 1";
       session:printable; classtype:honeynet;)
alert tcp 129.187.254.112/32 25 <> 129.187.19.106/32 any (msg:"Mailserver 2";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.106/32 21 (msg:"FTP (21) -> Windows";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.105/32 21 (msg:"FTP (21) -> Linux";
       session:printable; classtype:honeynet;)
alert tcp 10.155.10.35 any <> 129.187.19.105/32 22 (msg:"accounting <-> internal";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.106/32 22 (msg:"SSH (22) -> Windows";
       session:printable; classtype:honeynet;)
alert tcp !10.155.10.35 any <> 129.187.19.105/32 22 (msg:"SSH (22) -> Linux";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.106/32 23 (msg:"Telnet (23) -> Windows";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.105/32 23 (msg:"Telnet (23) -> Linux";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> $HOME_NET 57669 (msg:"55808 Window Size"; session:printable;
       classtype:honeynet;)
alert udp 129.187.19.106/32 137 <> 129.187.19.111/32 137 (msg:"Windows Port 137";
```

```
       session:printable; classtype:honeynet;)
alert udp 129.187.19.106/32 138 <> 129.187.19.111/32 138 (msg:"Windows Port 138";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.105/32 135 (msg:"NetBIOS (135) -> Linux";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.106/32 135 (msg:"NetBIOS (135) -> Windows";
       session:printable; classtype:honeynet;)
alert udp $EXTERNAL_NET any <> 129.187.19.105/32 137 (msg:"NetBIOS (137) UDP -> Linux";
       session:printable; classtype:honeynet;)
alert udp $EXTERNAL_NET any <> 129.187.19.106/32 137 (msg:"NetBIOS (137) UDP -> Windows";
       session:printable; classtype:honeynet;)
alert udp $EXTERNAL_NET any <> 129.187.19.105/32 139 (msg:"NetBIOS (139) UDP -> Linux";
       session:printable; classtype:honeynet;)
alert udp $EXTERNAL_NET any <> 129.187.19.106/32 139 (msg:"NetBIOS (139) UDP -> Windows";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.105/32 139 (msg:"NetBIOS (139) TCP -> Linux";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.106/32 139 (msg:"NetBIOS (139) TCP -> Windows";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.105/32 445 (msg:"NetBIOS (445) -> Linux";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.106/32 445 (msg:"NetBIOS (445) -> Windows";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.105/32 57 (msg:"FX-Scanner (57) -> Linux";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.106/32 57 (msg:"FX-Scanner (57) -> Windows";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET !80 <> 129.187.19.105/32 1433 (msg:"SQL Server (1433) TCP -> Linux";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET !80 <> 129.187.19.106/32 1433 (msg:"SQL Server (1433) TCP -> Windows";
       session:printable; classtype:honeynet;)
alert udp $EXTERNAL_NET !80 <> 129.187.19.105/32 1433 (msg:"SQL Server (1433) UDP -> Linux";
       session:printable; classtype:honeynet;)
alert udp $EXTERNAL_NET !80 <> 129.187.19.106/32 1433 (msg:"SQL Server (1433) UDP -> Windows";
       session:printable; classtype:honeynet;)
alert udp $EXTERNAL_NET !80 <> 129.187.19.105/32 1434 (msg:"SQL Server (1434) UDP -> Linux";
       session:printable; classtype:honeynet;)
alert udp $EXTERNAL_NET !80 <> 129.187.19.106/32 1434 (msg:"SQL Server (1434) UDP -> Windows";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET !80 <> 129.187.19.105/32 1434 (msg:"SQL Server (1434) TCP -> Linux";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET !80 <> 129.187.19.106/32 1434 (msg:"SQL Server (1434) TCP -> Windows";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET !80 <> 129.187.19.105/32 1740 (msg:"Port 1740 -> Linux";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET !80 <> 129.187.19.106/32 1740 (msg:"Port 1740 -> Windows";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET !80 <> 129.187.19.105/32 3306 (msg:"MySQL (3306) -> Linux";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET !80 <> 129.187.19.106/32 3306 (msg:"MySQL (3306) -> Windows";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET !80 <> 129.187.19.105/32 3389 (msg:"Microsoft Terminal Services (3389)
       -> Linux"; session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET !80 <> 129.187.19.106/32 3389 (msg:"Microsoft Terminal Services (3389)
       -> Windows"; session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET !80 <> 129.187.19.105/32 4000 (msg:"SkyDance (4000) -> Linux";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET !80 <> 129.187.19.106/32 4000 (msg:"SkyDance (4000) -> Windows";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.105/32 20168 (msg:"Lovgate.M (20168) -> Linux";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.106/32 20168 (msg:"Lovgate.M (20168) -> Windows";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET 22226 <> 129.187.19.105/32 any (msg:"W32/HLLW.Gaobot-AA (22226) ->
       Linux"; session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET 22226 <> 129.187.19.106/32 any (msg:"W32/HLLW.Gaobot-AA (22226) ->
       Windows"; session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET 22227 <> 129.187.19.105/32 any (msg:"W32/HLLW.Gaobot-AA (22227) ->
       Linux"; session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET 22227 <> 129.187.19.106/32 any (msg:"W32/HLLW.Gaobot-AA (22227) ->
       Windows"; session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET !80 <> 129.187.19.105/32 4899 (msg:"RAdmin (4899) -> Linux";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET !80 <> 129.187.19.106/32 4899 (msg:"RAdmin (4899) -> Windows";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.105/32 12345 (msg:"NetBus (12345) -> Linux";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.106/32 12345 (msg:"NetBus (12345) -> Windows";
```

```
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.105/32 17300 (msg:"Kuang2TheVirus (17300) -> Linux";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.106/32 17300 (msg:"Kuang2TheVirus (17300) ->
       Windows"; session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.105/32 27374 (msg:"SubSeven (27374) -> Linux";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.106/32 27374 (msg:"SubSeven (27374) -> Windows";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.105/32 554 (msg:"Real Server (554) <-> Linux";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET any <> 129.187.19.106/32 554 (msg:"Real Server (554) <-> Windows";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET !80 <> 129.187.19.105/32 593 (msg:"HTTP RPC Ep Map (593) <-> Linux";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET !80 <> 129.187.19.106/32 593 (msg:"HTTP RPC Ep Map (593) <-> Windows";
       session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET !80 -> 129.187.19.105/32 4444 (msg:"Port 4444 (RPC-Exploit Shell) ->
       Linux"; session:printable; classtype:honeynet;)
alert tcp $EXTERNAL_NET !80 -> 129.187.19.106/32 4444 (msg:"Port 4444 (RPC-Exploit Shell) ->
       Windows"; session:printable; classtype:honeynet;)
alert udp $HOME_NET any <> 129.187.254.32/32 123 (msg:"NTP"; session:printable;
       classtype:honeynet;)
alert udp $HOME_NET any <> 129.187.10.32/32 123 (msg:"NTP old"; session:printable;
       classtype:honeynet;)
alert udp 129.187.19.105/32 1101 -> 129.187.19.108/32 3141 (msg:"Sebek"; session:printable;
       classtype:honeynet;)
alert udp 129.187.19.106/32 any -> 129.187.19.107/32 514 (msg:"syslog Windows";
       session:printable; classtype:honeynet;)
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"ICMP Destination Unreachable (Destination
       Host Unknown)"; itype: 3; icode: 7; sid:394;  classtype:misc-activity; rev:4;)
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"ICMP Destination Unreachable (Destination
       Network Unknown)"; itype: 3; icode: 6; sid:395;  classtype:misc-activity; rev:4;)
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"ICMP Destination Unreachable
       (Fragmentation Needed and DF bit was set)"; itype: 3; icode:4; sid:396;  classtype:misc
       activity; rev:4;)
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"ICMP Destination Unreachable (Host
       Precedence Violation)"; itype: 3; icode: 14; sid:397;  classtype:misc-activity; rev:4;)
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"ICMP Destination Unreachable (Host
       Unreachable for Type of Service)"; itype: 3; icode: 12; sid:398;  classtype:misc-
       activity; rev:4;)
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"ICMP Destination Unreachable (Host
       Unreachable)"; itype: 3; icode: 1; sid:399;  classtype:misc-activity; rev:4;)
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"ICMP Destination Unreachable (Network
       Unreachable for Type of Service)"; itype: 3; icode:11; sid:400;   classtype:misc-
       activity; rev:4;)
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"ICMP Destination Unreachable (Network
       Unreachable)"; itype: 3; icode: 0; sid:401;  classtype:misc-activity; rev:4;)
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"ICMP Destination Unreachable (Port
       Unreachable)"; itype: 3; icode: 3; sid:402;  classtype:misc-activity; rev:4;)
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"ICMP Destination Unreachable (Port
       Unreachable) 2"; itype: 3; icode: 3;)
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"ICMP Destination Unreachable (Precedence
       Cutoff in effect)"; itype: 3; icode: 15; sid:403;  classtype:misc-activity; rev:4;)
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"ICMP Destination Unreachable (Protocol
       Unreachable)"; itype: 3; icode: 2; sid:404;  classtype:misc-activity; rev:4;)
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"ICMP Destination Unreachable (Source Host
       Isolated)"; itype: 3; icode: 8; sid:405;  classtype:misc-activity; rev:4;)
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"ICMP Destination Unreachable (Source Route
       Failed)"; itype: 3; icode: 5; sid:406;  classtype:misc-activity; rev:4;)
alert icmp $HOME_NET any -> $EXTERNAL_NET any (msg:"ICMP Destination Unreachable (Undefined
       Code!)"; itype: 3; sid:407;  classtype:misc-activity; rev:4;)
alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"ICMP Echo Reply"; itype: 0; icode: 0;
       sid:408;  classtype:misc-activity; rev:4;)
pass tcp $EXTERNAL_NET any <> $HOME_NET $HTTP_PORTS (flags: S;)
pass tcp $HOME_NET $HTTP_PORTS <> $EXTERNAL_NET any (flags: S;)
pass tcp $EXTERNAL_NET any <> $HOME_NET $HTTP_PORTS (flags: A;)
pass tcp $HOME_NET $HTTP_PORTS <> $EXTERNAL_NET any (flags: A;)
pass tcp $EXTERNAL_NET any <> $HOME_NET $HTTP_PORTS (flags: R;)
pass tcp $HOME_NET $HTTP_PORTS <> $EXTERNAL_NET any (flags: R;)
pass tcp $EXTERNAL_NET any <> $HOME_NET $HTTP_PORTS (flags: SA;)
pass tcp $HOME_NET $HTTP_PORTS <> $EXTERNAL_NET any (flags: SA;)
pass tcp $EXTERNAL_NET any <> $HOME_NET $HTTP_PORTS (flags: AF;)
pass tcp $HOME_NET $HTTP_PORTS <> $EXTERNAL_NET any (flags: AF;)
pass tcp $EXTERNAL_NET any <> $HOME_NET $HTTP_PORTS (flags: AR;)
pass tcp $HOME_NET $HTTP_PORTS <> $EXTERNAL_NET any (flags: AR;)
alert tcp $EXTERNAL_NET any <> $HOME_NET $HTTP_PORTS (msg:"HTTP Traffic"; flags: AP+;
       session:printable;)
```

```
pass tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (flags: S; msg:"Potential Worm http
       Traffic"; session: printable;)
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:"Incoming HTTP Traffic on Source
       Port 80"; session:printable;)
alert tcp $EXTERNAL_NET 0 -> $HOME_NET any (msg:"Source Port 0 Traffic"; session:printable;)
log ip any any <> any any (msg:"Snort Unmatched"; session:printable;)
```

A.1.1.3 Skript für *tcpdump*

Skript für die tägliche Protokollierung des Netzverkehrs. Skript wurde in `crontab` eingetragen und jeweils um 0:01 Uhr ausgeführt.

```
#! /bin/sh
. /etc/rc.status
DATE=`date +%b_%d`
TIME=`date +_%T`
TCPDUMP_BIN="/usr/sbin/tcpdump"
TCPDUMP_OPTIONS="-i eth1 -w /var/log/tcpdump/$DATE.log"
if [ -e /var/log/tcpdump/$DATE.log ]; then
       mv /var/log/tcpdump/$DATE.log /var/log/tcpdump/$DATE$TIME.log
fi
. /etc/rc.status

rc_reset
case "$1" in
    start)
        echo -n "Starting "
        echo $TCPDUMP_BIN $TCPDUMP_OPTIONS
        /sbin/startproc $TCPDUMP_BIN $TCPDUMP_OPTIONS
        rc_status -v
        ;;
    stop)
        echo -n "Shutting down tcpdump"
        /sbin/killproc -TERM $TCPDUMP_BIN
        rc_status -v
        ;;
    restart)
        $0 stop
        $0 start
        rc_status
        ;;
    status)
        echo -n "Checking for tcpdump: "
        /sbin/checkproc $TCPDUMP_BIN
        rc_status -v
        ;;
    *)
        echo "Usage: $0 {start|stop|status|restart}"
        exit 1
        ;;
esac
rc_exit
```

A.1.1.4 Konfiguration von *swatch*

Mit *swatch* wird `/var/log/messages` überwacht:

- Enthält die aktuelle Zeile einen der Schlüsselbegriffe *IN_TCP*, *IN_UDP*, *IN_ICMP* oder *IN_OTHER* wird das Skript `/root/alarmin` mit den Parametern Quell-Adresse (`$12`), Ziel-Adresse (`$13`), Ziel-Port (`$21`) und *WindowSize* (`$22`) ausgeführt.

- Enthält die aktuelle Zeile den Schlüsselbegriff *PHYSOUT=eth0 SRC=129.187.19.105* wird das Skript `/root/alarmoutlinux` mit den Parametern Quell-Adresse (`$12`), Ziel-Adresse (`$13`), Quell-Port (`$21`) und Ziel-Port (`$22`) ausgeführt.

- Enthält die aktuelle Zeile den Schlüsselbegriff *PHYSOUT=eth0 SRC=129.187.19.106* wird das Skript `/root/alarmoutwin` mit den Parametern Quell-Adresse (`$12`), Ziel-Adresse (`$13`), Quell-Port (`$21`) und Ziel-Port (`$22`) ausgeführt.

```
# Personal Swatch configuration file
#
watchfor /IN_TCP|IN_UDP|IN_ICMP|IN_OTHER/
        exec /root/alarmin $12 $13 $21 $22
watchfor /PHYSOUT=eth0 SRC=129.187.19.105/
        exec /root/alarmoutlinux $12 $13 $21 $22
watchfor /PHYSOUT=eth0 SRC=129.187.19.106/
        exec /root/alarmoutwin $12 $13 $21 $22
```

A.1.1.5 Skripte für den SMS-Versand

alarmin (Eingehende Verbindungen)

Zunächst wird der Wert der Datei `/root/mailin` überprüft. Ist dieser kleiner als 10, so wird eine E-Mail mit den übergebenen Parametern gesendet, die Zählvariable inkrementiert und gespeichert. Zur vollen Stunde wird der Wert wieder auf „0“ gesetzt.

```
#!/bin/bash
TESTMAIL=`cat /root/mailin`
if [ $TESTMAIL -lt 10 ]; then
        echo "$1 $2 $3 $4" | mail -s"---- Inbound Connection ($2) ----"↵
          gereon.volker@stud.tu-muenchen.de
fi
let TESTMAIL=$TESTMAIL+1
echo $TESTMAIL > /root/mailin
exit 0
```

alarmoutlinux (Ausgehende Verbindungen vom Linux-Honeypot)

Zunächst wird der Wert der Datei `/root/mailoutlinux` überprüft. Ist dieser kleiner als 10, so wird eine E-Mail mit den übergebenen Parametern gesendet, die Zählvariable inkrementiert und gespeichert. Zur vollen Stunde wird der Wert wieder auf „0“ gesetzt.
Zwischen 0 Uhr und 6 Uhr wird keine SMS versendet. Analog zur E-Mail Benachrichtigung wird die Anzahl der bereits gesendeten SMS überprüft (`/root/sentsms`). Falls dieser Wert kleiner als 2 ist, erfolgt der Versand der SMS über den Rechner *nm1.lrz-muenchen.de*. Die Zählvariable wird ebenfalls zur vollen Stunde wieder auf „0“ gesetzt.

```
#!/bin/bash
TESTMAIL=`cat /root/mailoutlinux`
if [ $TESTMAIL -lt 10 ]; then
        let TESTMAIL=$TESTMAIL+1
        echo "$1 $2 $3 $4" | mail -s"---- Outbound Connection Linux ($2)↵
          ----" gereon.volker@stud.tu-muenchen.de
        echo $TESTMAIL > /root/mailoutlinux
fi
HOUR=`date +%k`
if [ $HOUR -gt 6 ]; then
        NUMBER=`cat /root/sentsms`
        if [ $NUMBER -lt 2 ]; then
                let NUMBER=$NUMBER+1
                echo $NUMBER > /root/sentsms
                echo "ALARM OUTBOUND LINUX: $1 $2 $3 $4" > /root/message
```

```
                scp /root/message honeynet@nm1:/home/honeynet
                ssh honeynet@nm1 /opt/management/sendxms/sendxms↵
                   01794924752 -f/ home/honeynet/message
        fi
fi
exit 0
```

alarmoutwin

Dieses Skript arbeitet analog zum vorherigen Skript für den Windows™-Honeypot.

```
#!/bin/bash
TESTMAIL=`cat /root/mailoutwin`
if [ $TESTMAIL -lt 10 ]; then
        let TESTMAIL=$TESTMAIL+1
        echo "$1 $2 $3 $4" | mail -s"---- Outbound Connection Windows ($2) ----"↵
         gereon.volker@stud.tu-muenchen.de
        echo $TESTMAIL > /root/mailoutwin
fi
HOUR=`date +%k`
if [ $HOUR -gt 5 ]; then
        NUMBER=`cat /root/sentsms`
        if [ $NUMBER -lt 2 ]; then
                let NUMBER=$NUMBER+1
                echo $NUMBER > /root/sentsms
                echo "ALARM OUTBOUND WINDOWS: $1 $2 $3 $4" > /root/message
                scp /root/message honeynet@nm1:/home/honeynet
                ssh honeynet@nm1 /opt/management/sendxms/sendxms↵
                   01794924752 -f/home/honeynet/message
        fi
fi
exit 0
```

sendsmsin

Dieses Skript dient zur Benachrichtigung per SMS falls in innerhalb einer Stunde mehr als 25 eingehende Verbindungen registriert werden und wird in der `crontab` eingetragen. Die Ausführung des Skriptes erfolgt immer 5 Minuten vor der vollen Stunde.

```
#!/bin/bash
NUMBER=`cat /root/mailin`
HOUR=`date +%k`
if [ $NUMBER -gt 25 -a $HOUR -gt 5 ]; then
        TIME1=`date +%H`
        TIME2=`date +%H:%M`
        echo "$NUMBER INBOUND Connections zwischen $TIME1:00 und $TIME2" >↵
          /root/message2
        scp /root/message2 honeynet@nm1:/home/honeynet
        ssh honeynet@nm1 /opt/management/sendxms/sendxms 01794924752↵
          -f/home/honeynet/message2
fi
exit 0
```

A.1.2 Linux-Honeypot

A.1.2.1 „fake-syslog-Daemon“

Dieses Programm soll dem Angreifer den syslog-Daemon vortäuschen. Es handelt sich dabei um ein einfaches C-Programm, das alle 1000 Sekunden kurz „aufwacht“.

```
#include <stdio.h>
#include <unistd.h>
main()
{
        while (1)
        {
                sleep(1000);
        }
}
```

A.1.2.2 Perl-Skripte für den Webauftritt

Zur Modifikation der HTML-Dateien werden zwei Skripte ausgeführt: Das Skript „update“ überprüft periodisch, ob jeweils eine neue Quell-Datei vorhanden ist. Ist dies der Falll, so wird das Skript „gendata“ gestartet und im Anschluss die Original-Datei gelöscht.

update

```
#!/bin/bash
INFILE="/tmp/www/top-out-gwin-hosts-day.html"
if [ -e $INFILE ]; then
        perl /usr/local/bin/gendata $INFILE
        rm $INFILE
fi
INFILE="/tmp/www/top-out-gwin-hosts-night.html"
if [ -e $INFILE ]; then
        perl /usr/local/bin/gendata $INFILE
        rm $INFILE
fi
exit 0
```

gendata

```
#!/usr/bin/perl

use CGI;
use DBI;

my $datum = time();
my $webserver = "129.187.19.105";
my $dbserver = "129.187.19.105";
my $database = "trafficdata";
my $login = "updater";
my $passwd = "traffic";

my $outfile = $datum . ".html";
if (@ARGV[0])
{
        my $infile = @ARGV[0];
        open (INDAT,"< $infile");
        $i = 0;
        while ($zeile=<INDAT>){
                if ($zeile =~ />\d{3}\.\d{3}\.\d{3}\.\d{3}/)
                {
                        $zeile =~ />(\d{3}\.\d{3}\.\d{3}\.\d{3})/;
                        $ip[$i] = $1;
                        #print ".";
                        $i++;
```

```
                }
        }
        for ($j = 0; $j < $i; $j++)
        {
                @srcipdata = split(/\./,$ip[$j]);
                $pos0 = int(rand 255) + 1;
                if ($pos0 < 10) { $pos0 = "00$pos0"; }
                if ($pos0 < 100 && $pos0 > 9) { $pos0 = "0$pos0"; }
                $pos1 = int(rand 255) + 1;
                if ($pos1 < 10) { $pos1 = "00$pos1"; }
                if ($pos1 < 100 && $pos1 > 9) { $pos1 = "0$pos1"; }
                $repip[$j] = "$srcipdata[0].$srcipdata[1].$pos0.$pos1";
                #print ".";
        }
        seek(INDAT, 0, SEEK_SET);
        open (OUTDAT,"> $outfile");
        while ( <INDAT> )
        {
                s/<p><b>Statistik/<h1>Accounting-Daten<\/h1><p><b>Statistik/g;
                for ($j = 0; $j < $i; $j++)
                {
                        s/$ip[$j]/$repip[$j]/g;
                }
                s/<i>Out bedeutet: MWN-Hosts --> GWiN-Host \(der G-Win Host↵
                ist Datensenke\)<BR>//g;
                s/In bedeutet: GWiN-Host --> MWN-Hosts \(der G-WiN Host↵
                ist Datenquelle\)<\/i>//g;
                s/Accounting MWN\/GWiN/Accounting Daten/g;
                s/Top Out GWiN Hosts/Top Out Hosts/g;
                s/<th>G-WiN Hostname<\/th>//g;
                s/, erstellt um \d{2}:\d{2}:\d{2} \d{2}\.\d{2}\.\d{4}//g;
                s/<td  >([\w-]+\.)*[\w-]+\.[\w-]+<\/td>//g;
                print OUTDAT;
        }
        close (INDAT);
        close (OUTDAT);
        my $cgi_obj = new CGI;
        my $dbh = DBI->connect('dbi:mysql:'. $database . ':' . $dbserver, ↵
        $login, $passwd) || die "Cannot connect to MySQL server: ↵
        $DBI::errstr\n";

        my $sql = "UPDATE list SET valid = 0";
        my $ins = $dbh->prepare( $sql ) || die "Error while ↵
        preparing statement: $DBI::errstr\n";
        $ins->execute || die "Error while executing query:  $DBI::errstr\n";

        $sql = "INSERT INTO list VALUES(0," . $datum . ",1)";
        $ins = $dbh->prepare( $sql ) || die "Error while preparing ↵
        statement: $DBI::errstr\n";
        $ins->execute || die "Error while executing query:  $DBI::errstr\n";

        $ins = $dbh->prepare('SELECT * FROM list') || die "Error↵
        while preparing statement: $DBI::errstr\n";
        $ins->execute || die "Error while executing query:  $DBI::errstr\n";

        $ins->finish;
        $dbh->disconnect;
        `mv $outfile /usr/local/httpd/htdocs/data`
}
```

A.1.2.3 Tripwire-Policy

Die verwendete Policy für den Integritätschecker *tripwire*.

```
 ##############################################################################
 #                                                                            ##
############################################################################## #
#                                                                            # #
#                     Policy file for RedHat Linux 7.0                       # #
#                               V1.0.0                                       # #
#                           June 24, 2003                                    # #
```

```
#                                                                              ##
################################################################################

  ###############################################################################
 #                                                                               ##
################################################################################ #
#                                                                               # #
# Global Variable Definitions                                                   # #
#                                                                               # #
# These are defined at install time by the installation script.  You may        # #
# Manually edit these if you are using this file directly and not from the      # #
# installation script itself.                                                   # #
#                                                                               ##
################################################################################

@@section GLOBAL
TWDOCS="/.. /usr/doc/tripwire";
TWBIN="/.. /usr/sbin";
TWPOL="/.. /etc/tripwire";
TWDB="/var/lib/.. /tripwire";
TWSKEY="/.. /etc/tripwire";
TWLKEY="/.. /etc/tripwire";
TWREPORT="/var/lib/.. /tripwire/report";
HOSTNAME=linux;

@@section FS
SEC_CRIT      = $(IgnoreNone)-SHa ;  # Critical files that cannot change
SEC_SUID      = $(IgnoreNone)-SHa ;  # Binaries with the SUID or SGID flags set
SEC_BIN       = $(ReadOnly) ;        # Binaries that should not change
SEC_CONFIG    = $(Dynamic) ;         # Config files that are changed infrequently but accessed
                                     # often
SEC_LOG       = $(Growing) ;         # Files that grow, but that should never change ownership
SEC_INVARIANT = +tpug ;              # Directories that should never change permission or
                                     # ownership
SIG_LOW       = 33 ;                 # Non-critical files that are of minimal security impact
SIG_MED       = 66 ;                 # Non-critical files that are of significant security
                                     # impact
SIG_HI        = 100 ;                # Critical files that are significant points of
                                     # vulnerability


# Tripwire Binaries
(
  rulename = "Tripwire Binaries",
  severity = $(SIG_HI)
)
{
  $(TWBIN)/siggen                     -> $(SEC_BIN) ;
  $(TWBIN)/tripwire                   -> $(SEC_BIN) ;
  $(TWBIN)/twadmin                    -> $(SEC_BIN) ;
  $(TWBIN)/twprint                    -> $(SEC_BIN) ;
}

# Tripwire Data Files - Configuration Files, Policy Files, Keys, Reports, Databases
(
  rulename = "Tripwire Data Files",
  severity = $(SIG_HI)
)
{
  # NOTE: We remove the inode attribute because when Tripwire creates a backup,
  # it does so by renaming the old file and creating a new one (which will
  # have a new inode number).  Inode is left turned on for keys, which shouldn't
  # ever change.

  # NOTE: The first integrity check triggers this rule and each integrity check
  # afterward triggers this rule until a database update is run, since the
  # database file does not exist before that point.

  $(TWDB)                             -> $(SEC_CONFIG) -i ;
  $(TWPOL)/tw.pol                     -> $(SEC_BIN) -i ;
  $(TWPOL)/tw.cfg                     -> $(SEC_BIN) -i ;
  $(TWLKEY)/$(HOSTNAME)-local.key     -> $(SEC_BIN) ;
  $(TWSKEY)/site.key                  -> $(SEC_BIN) ;

  #don't scan the individual reports
  $(TWREPORT)                         -> $(SEC_CONFIG) (recurse=0) ;
}
```

```
# Commonly accessed directories that should remain static with regards to owner and group
(
  rulename = "Invariant Directories",
  severity = $(SIG_MED)
)
{
  /                                   -> $(SEC_INVARIANT) (recurse = 0) ;
  /home                               -> $(SEC_INVARIANT) (recurse = 0) ;
  /etc                                -> $(SEC_INVARIANT) (recurse = 0) ;
}
  ################################################
 #                                               ##
################################################ #
#                                               # #
# File System and Disk Administration Programs # #
#                                               ##
################################################

(
  rulename = "File System and Disk Administraton Programs",
  severity = $(SIG_HI)
)
{
  /sbin/badblocks                     -> $(SEC_CRIT) ;
  /sbin/dosfsck                       -> $(SEC_CRIT) ;
  /sbin/e2fsck                        -> $(SEC_CRIT) ;
  /sbin/debugfs                       -> $(SEC_CRIT) ;
  /sbin/dumpe2fs                      -> $(SEC_CRIT) ;
  /sbin/e2label                       -> $(SEC_CRIT) ;
  /sbin/fdisk                         -> $(SEC_CRIT) ;
  /sbin/fsck                          -> $(SEC_CRIT) ;
  /sbin/fsck.ext2                     -> $(SEC_CRIT) ;
  /sbin/fsck.minix                    -> $(SEC_CRIT) ;
  /sbin/fsck.msdos                    -> $(SEC_CRIT) ;
  /sbin/hdparm                        -> $(SEC_CRIT) ;
  /sbin/mkdosfs                       -> $(SEC_CRIT) ;
  /sbin/mke2fs                        -> $(SEC_CRIT) ;
  /sbin/mkfs                          -> $(SEC_CRIT) ;
  /sbin/mkfs.ext2                     -> $(SEC_CRIT) ;
  /sbin/mkfs.minix                    -> $(SEC_CRIT) ;
  /sbin/mkfs.msdos                    -> $(SEC_CRIT) ;
  /sbin/mkinitrd                      -> $(SEC_CRIT) ;
  /sbin/mkpv                          -> $(SEC_CRIT) ;
  /sbin/mkraid                        -> $(SEC_CRIT) ;
  /sbin/mkswap                        -> $(SEC_CRIT) ;
  /sbin/quotacheck                    -> $(SEC_CRIT) ;
  /sbin/quotaon                       -> $(SEC_CRIT) ;
  /sbin/raidstart                     -> $(SEC_CRIT) ;
  /sbin/resize2fs                     -> $(SEC_CRIT) ;
  /sbin/sfdisk                        -> $(SEC_CRIT) ;
  /sbin/tune2fs                       -> $(SEC_CRIT) ;
  /bin/mount                          -> $(SEC_CRIT) ;
  /bin/umount                         -> $(SEC_CRIT) ;
  /bin/touch                          -> $(SEC_CRIT) ;
  /bin/mkdir                          -> $(SEC_CRIT) ;
  /bin/mknod                          -> $(SEC_CRIT) ;
  /bin/mktemp                         -> $(SEC_CRIT) ;
  /bin/rm                             -> $(SEC_CRIT) ;
  /bin/rmdir                          -> $(SEC_CRIT) ;
  /bin/chgrp                          -> $(SEC_CRIT) ;
  /bin/chmod                          -> $(SEC_CRIT) ;
  /bin/chown                          -> $(SEC_CRIT) ;
  /bin/cp                             -> $(SEC_CRIT) ;
  /bin/cpio                           -> $(SEC_CRIT) ;
}

  ###################################
 #                                  ##
################################### #
#                                  # #
# Kernel Administration Programs # #
#                                  ##
###################################

(
  rulename = "Kernel Administration Programs",
  severity = $(SIG_HI)
```

```
)
{
  /sbin/depmod                          -> $(SEC_CRIT) ;
  /sbin/ctrlaltdel                      -> $(SEC_CRIT) ;
  /sbin/insmod                          -> $(SEC_CRIT) ;
  /sbin/insmod.static                   -> $(SEC_CRIT) ;
  /sbin/insmod_ksymoops_clean           -> $(SEC_CRIT) ;
  /sbin/klogd                           -> $(SEC_CRIT) ;
  /sbin/ldconfig                        -> $(SEC_CRIT) ;
  /sbin/modinfo                         -> $(SEC_CRIT) ;
  /sbin/sysctl                          -> $(SEC_CRIT) ;
}

  ########################
 #                       ##
###################### #
#                     # #
# Networking Programs # #
#                     ##
########################

(
  rulename = "Networking Programs",
  severity = $(SIG_HI)
)
{
  /sbin/arp                             -> $(SEC_CRIT) ;
  /sbin/dhcpcd                          -> $(SEC_CRIT) ;
  /sbin/ifconfig                        -> $(SEC_CRIT) ;
  /sbin/ifdown                          -> $(SEC_CRIT) ;
  /sbin/ifup                            -> $(SEC_CRIT) ;
  /sbin/ifuser                          -> $(SEC_CRIT) ;
  /sbin/ip                              -> $(SEC_CRIT) ;
  /sbin/ipmaddr                         -> $(SEC_CRIT) ;
  /sbin/iptunnel                        -> $(SEC_CRIT) ;
  /sbin/plipconfig                      -> $(SEC_CRIT) ;
  /sbin/portmap                         -> $(SEC_CRIT) ;
  /sbin/rarp                            -> $(SEC_CRIT) ;
  /sbin/route                           -> $(SEC_CRIT) ;
  /sbin/slattach                        -> $(SEC_CRIT) ;
  /bin/ping                             -> $(SEC_CRIT) ;
}

  ##################################
 #                                 ##
################################## #
#                                # #
# System Administration Programs # #
#                                ##
##################################

(
  rulename = "System Administration Programs",
  severity = $(SIG_HI)
)
{
  /sbin/chkconfig                       -> $(SEC_CRIT) ;
  /sbin/halt                            -> $(SEC_CRIT) ;
  /sbin/init                            -> $(SEC_CRIT) ;
  /sbin/killall5                        -> $(SEC_CRIT) ;
  /sbin/rpc.lockd                       -> $(SEC_CRIT) ;
  /sbin/rpc.statd                       -> $(SEC_CRIT) ;
  /sbin/shutdown                        -> $(SEC_CRIT) ;
  /sbin/sulogin                         -> $(SEC_CRIT) ;
  /sbin/swapon                          -> $(SEC_CRIT) ;
  /sbin/syslogd                         -> $(SEC_CRIT) ;
  /sbin/unix_chkpwd                     -> $(SEC_CRIT) ;
  /bin/pwd                              -> $(SEC_CRIT) ;
  /bin/uname                            -> $(SEC_CRIT) ;
}

  ######################################
 #                                   ##
###################################### #
#                                    # #
# Hardware and Device Control Programs # #
#                                    ##
######################################
```

```
(
  rulename = "Hardware and Device Control Programs",
  severity = $(SIG_HI)
)
{
  /sbin/hwclock                         -> $(SEC_CRIT) ;
  /sbin/isapnp                          -> $(SEC_CRIT) ;
  /sbin/kbdrate                         -> $(SEC_CRIT) ;
  /sbin/losetup                         -> $(SEC_CRIT) ;
  /sbin/lspci                           -> $(SEC_CRIT) ;
  /sbin/pnpdump                         -> $(SEC_CRIT) ;
  /sbin/setpci                          -> $(SEC_CRIT) ;
}

  ##############################
 #                            ##
############################## #
#                            # #
# System Information Programs # #
#                            ##
##############################
(
  rulename = "System Information Programs",
  severity = $(SIG_HI)
)
{
  /sbin/kernelversion                   -> $(SEC_CRIT) ;
  /sbin/runlevel                        -> $(SEC_CRIT) ;
}

  ###################################
 #                                 ##
################################### #
#                                 # #
# Application Information Programs # #
#                                 ##
###################################

(
  rulename = "Application Information Programs",
  severity = $(SIG_HI)
)
{
  /sbin/genksyms                        -> $(SEC_CRIT) ;
  /sbin/sln                             -> $(SEC_CRIT) ;
}

  ################
 #              ##
################ #
#              # #
# OS Utilities # #
#              ##
################
(
  rulename = "Operating System Utilities",
  severity = $(SIG_HI)
)
{
  /bin/cat                              -> $(SEC_CRIT) ;
  /bin/date                             -> $(SEC_CRIT) ;
  /bin/dd                               -> $(SEC_CRIT) ;
  /bin/df                               -> $(SEC_CRIT) ;
  /bin/echo                             -> $(SEC_CRIT) ;
  /bin/egrep                            -> $(SEC_CRIT) ;
  /bin/false                            -> $(SEC_CRIT) ;
  /bin/fgrep                            -> $(SEC_CRIT) ;
  /bin/gawk                             -> $(SEC_CRIT) ;
  /bin/grep                             -> $(SEC_CRIT) ;
  /bin/true                             -> $(SEC_CRIT) ;
  /bin/arch                             -> $(SEC_CRIT) ;
  /bin/ash                              -> $(SEC_CRIT) ;
  /bin/ash.static                       -> $(SEC_CRIT) ;
  /bin/basename                         -> $(SEC_CRIT) ;
  /bin/dmesg                            -> $(SEC_CRIT) ;
  /bin/gunzip                           -> $(SEC_CRIT) ;
  /bin/gzip                             -> $(SEC_CRIT) ;
  /bin/hostname                         -> $(SEC_CRIT) ;
```

```
  /bin/kill                             -> $(SEC_CRIT) ;
  /bin/ln                               -> $(SEC_CRIT) ;
  /bin/loadkeys                         -> $(SEC_CRIT) ;
  /bin/login                            -> $(SEC_CRIT) ;
  /bin/ls                               -> $(SEC_CRIT) ;
  /bin/mail                             -> $(SEC_CRIT) ;
  /bin/more                             -> $(SEC_CRIT) ;
  /bin/mv                               -> $(SEC_CRIT) ;
  /bin/netstat                          -> $(SEC_CRIT) ;
  /bin/ps                               -> $(SEC_CRIT) ;
  /bin/rpm                              -> $(SEC_CRIT) ;
  /bin/sed                              -> $(SEC_CRIT) ;
  /bin/setserial                        -> $(SEC_CRIT) ;
  /bin/sleep                            -> $(SEC_CRIT) ;
  /bin/sort                             -> $(SEC_CRIT) ;
  /bin/stty                             -> $(SEC_CRIT) ;
  /bin/su                               -> $(SEC_CRIT) ;
  /bin/sync                             -> $(SEC_CRIT) ;
  /bin/tar                              -> $(SEC_CRIT) ;
  /bin/usleep                           -> $(SEC_CRIT) ;
  /bin/vi                               -> $(SEC_CRIT) ;
  /bin/zcat                             -> $(SEC_CRIT) ;
}

  ##############################
 #                             ##
############################## #
#                             # #
# Critical Utility Sym-Links # #
#                             ##
##############################
(
  rulename = "Critical Utility Sym-Links",
  severity = $(SIG_HI)
)
{
  /sbin/clock                           -> $(SEC_CRIT) ;
  /sbin/kallsyms                        -> $(SEC_CRIT) ;
  /sbin/ksyms                           -> $(SEC_CRIT) ;
  /sbin/lsmod                           -> $(SEC_CRIT) ;
  /sbin/modprobe                        -> $(SEC_CRIT) ;
  /sbin/mount.smbfs                     -> $(SEC_CRIT) ;
  /sbin/pidof                           -> $(SEC_CRIT) ;
  /sbin/poweroff                        -> $(SEC_CRIT) ;
  /sbin/quotaoff                        -> $(SEC_CRIT) ;
  /sbin/raid0run                        -> $(SEC_CRIT) ;
  /sbin/raidhotadd                      -> $(SEC_CRIT) ;
  /sbin/raidhotremove                   -> $(SEC_CRIT) ;
  /sbin/raidstop                        -> $(SEC_CRIT) ;
  /sbin/swapoff                         -> $(SEC_CRIT) ;
  /sbin/reboot                          -> $(SEC_CRIT) ;
  /sbin/rmmod                           -> $(SEC_CRIT) ;
  /sbin/telinit                         -> $(SEC_CRIT) ;
  /bin/awk                              -> $(SEC_CRIT) ;
  /bin/dnsdomainname                    -> $(SEC_CRIT) ;
  /bin/domainname                       -> $(SEC_CRIT) ;
  /bin/nisdomainname                    -> $(SEC_CRIT) ;
  /bin/ypdomainname                     -> $(SEC_CRIT) ;
}


  ##########################
 #                         ##
########################## #
#                         # #
# Temporary directories # #
#                         ##
##########################
(
  rulename = "Temporary directories",
  recurse = false,
  severity = $(SIG_LOW)
)
{
  /usr/tmp                              -> $(SEC_INVARIANT) ;
  /var/tmp                              -> $(SEC_INVARIANT) ;
  /tmp                                  -> $(SEC_INVARIANT) ;
}
```

```
  ###############
 #             ##
############### #
#             # #
# Local files # #
#             ##
###############
(
  rulename = "User binaries",
  severity = $(SIG_MED)
)
{
  /sbin                                 -> $(SEC_BIN) (recurse = 1) ;
  /usr/local/bin                        -> $(SEC_BIN) (recurse = 1) ;
  /usr/sbin                             -> $(SEC_BIN) (recurse = 1) ;
  /usr/bin                              -> $(SEC_BIN) (recurse = 1) ;
}

(
  rulename = "Shell Binaries",
  severity = $(SIG_HI)
)
{
  /bin/csh                              -> $(SEC_BIN) ;
  /bin/sh                               -> $(SEC_BIN) ;
  /bin/bash                             -> $(SEC_BIN) ;
  /bin/tcsh                             -> $(SEC_BIN) ;
}

(
  rulename = "Security Control",
  severity = $(SIG_HI)
)
{
  /etc/group                            -> $(SEC_CRIT) ;
  /etc/security/                        -> $(SEC_CRIT) ;
}


(
  rulename = "Login Scripts",
  severity = $(SIG_HI)
)
{
  /etc/csh.cshrc                        -> $(SEC_CONFIG) ;
  /etc/csh.login                        -> $(SEC_CONFIG) ;
  /etc/profile                          -> $(SEC_CONFIG) ;
}

# Libraries
(
  rulename = "Libraries",
  severity = $(SIG_MED)
)
{
  /usr/lib                              -> $(SEC_BIN) ;
  /usr/local/lib                        -> $(SEC_BIN) ;
}


  ######################################################
 #                                                    ##
###################################################### #
#                                                    # #
# Critical System Boot Files                         # #
# These files are critical to a correct system boot. # #
#                                                    ##
######################################################

(
  rulename = "Critical system boot files",
  severity = $(SIG_HI)
)
{
     /boot                              -> $(SEC_CRIT) ;
     /sbin/lilo                         -> $(SEC_CRIT) ;
     !/boot/System.map ;
```

```
     !/boot/module-info ;
}
   ################################################
  ################################################
  # These files change every time the system boots ##
  ################################################
(
  rulename = "System boot changes",
  severity = $(SIG_HI)
)
{
     !/var/run/ftp.pids-all ; # Comes and goes on reboot.
     !/root/.enlightenment ;
     /dev/log                          -> $(SEC_CONFIG) ;
     /dev/cua0                         -> $(SEC_CONFIG) ;
     /dev/console                      -> $(SEC_CONFIG) -u ;
     /dev/tty3                         -> $(SEC_CONFIG) ; # are extremely
     /dev/tty4                         -> $(SEC_CONFIG) ; # variable
     /dev/tty5                         -> $(SEC_CONFIG) ;
     /dev/tty6                         -> $(SEC_CONFIG) ;
     /dev/urandom                      -> $(SEC_CONFIG) ;
     /dev/initctl                      -> $(SEC_CONFIG) ;
     /var/run                          -> $(SEC_CONFIG) ; # daemon PIDs
     /var/log                          -> $(SEC_CONFIG) ;
     /etc/issue.net                    -> $(SEC_CONFIG) -i ; # Inode number changes
     /etc/ioctl.save                   -> $(SEC_CONFIG) ;
     /etc/issue                        -> $(SEC_CONFIG) ;
     /etc/.pwd.lock                    -> $(SEC_CONFIG) ;
     /etc/mtab                         -> $(SEC_CONFIG) -i ;
     /lib/modules                      -> $(SEC_CONFIG) ;
}

# These files change the behavior of the root account
(
  rulename = "Root config files",
  severity = 100
)
{
     /root                             -> $(SEC_CRIT) ; # Catch all additions to /root
     /root/mbox                        -> $(SEC_CONFIG) ;
     /root/.xsession-errors            -> $(SEC_CONFIG) ;
     /root/.bash_history               -> $(SEC_CONFIG) ;
}

  ################################
 #                               ##
################################ #
#                               # #
# Critical configuration files # #
#                               ##
################################
(
  rulename = "Critical configuration files",
  severity = $(SIG_HI)
)
{
     /etc/crontab                      -> $(SEC_BIN) ;
     /etc/cron.hourly                  -> $(SEC_BIN) ;
     /etc/cron.daily                   -> $(SEC_BIN) ;
     /etc/cron.weekly                  -> $(SEC_BIN) ;
     /etc/cron.monthly                 -> $(SEC_BIN) ;
     /etc/default                      -> $(SEC_BIN) ;
     /etc/fstab                        -> $(SEC_BIN) ;
     /etc/exports                      -> $(SEC_BIN) ;
     /etc/group-                       -> $(SEC_BIN) ;  # changes should be infrequent
     /etc/host.conf                    -> $(SEC_BIN) ;
     /etc/hosts.allow                  -> $(SEC_BIN) ;
     /etc/hosts.deny                   -> $(SEC_BIN) ;
     /etc/httpd/httpd.conf             -> $(SEC_BIN) ;  # changes should be infrequent
     /etc/protocols                    -> $(SEC_BIN) ;
     /etc/services                     -> $(SEC_BIN) ;
     /etc/init.d                       -> $(SEC_BIN) ;
     /etc/rc.d                         -> $(SEC_BIN) ;
     /etc/mail.rc                      -> $(SEC_BIN) ;
     /etc/motd                         -> $(SEC_BIN) ;
     /etc/passwd                       -> $(SEC_CONFIG) ;
     /etc/passwd-                      -> $(SEC_CONFIG) ;
     /etc/profile.d                    -> $(SEC_BIN) ;
```

```
     /var/lib/nfs/rmtab                -> $(SEC_BIN) ;
     /etc/rpc                          -> $(SEC_BIN) ;
     /etc/sysconfig                    -> $(SEC_BIN) ;
     /etc/nsswitch.conf                -> $(SEC_BIN) ;
     /etc/hosts                        -> $(SEC_CONFIG) ;
     /etc/inetd.conf                   -> $(SEC_CONFIG) ;
     /etc/inittab                      -> $(SEC_CONFIG) ;
     /etc/resolv.conf                  -> $(SEC_CONFIG) ;
     /etc/syslog.conf                  -> $(SEC_CONFIG) ;

}

  ####################
 #                   ##
################### #
#                   # #
# Critical devices # #
#                   ##
####################
(
  rulename = "Critical devices",
  severity = $(SIG_HI),
  recurse = false
)
{
     /dev/kmem                         -> $(Device) ;
     /dev/mem                          -> $(Device) ;
     /dev/null                         -> $(Device) ;
     /dev/zero                         -> $(Device) ;
     /proc/devices                     -> $(Device) ;
     /proc/net                         -> $(Device) ;
     /proc/sys                         -> $(Device) ;
     /proc/cpuinfo                     -> $(Device) ;
     /proc/modules                     -> $(Device) ;
     /proc/mounts                      -> $(Device) ;
     /proc/dma                         -> $(Device) ;
     /proc/filesystems                 -> $(Device) ;
     /proc/pci                         -> $(Device) ;
     /proc/interrupts                  -> $(Device) ;
     /proc/ioports                     -> $(Device) ;
     /proc/scsi                        -> $(Device) ;
     /proc/kcore                       -> $(Device) ;
     /proc/self                        -> $(Device) ;
     /proc/kmsg                        -> $(Device) ;
     /proc/stat                        -> $(Device) ;
     /proc/ksyms                       -> $(Device) ;
     /proc/loadavg                     -> $(Device) ;
     /proc/uptime                      -> $(Device) ;
     /proc/locks                       -> $(Device) ;
     /proc/version                     -> $(Device) ;
     /proc/mdstat                      -> $(Device) ;
     /proc/meminfo                     -> $(Device) ;
     /proc/cmdline                     -> $(Device) ;
     /proc/misc                        -> $(Device) ;
}

# Rest of critical system binaries
(
  rulename = "OS executables and libraries",
  severity = $(SIG_HI)
)
{
     /bin                              -> $(SEC_BIN) ;
     /lib                              -> $(SEC_BIN) ;
}

#=============================================================================
#
# Copyright 2000 Tripwire, Inc. Tripwire is a registered trademark of Tripwire,
# Inc. in the United States and other countries. All rights reserved.
#
# Linux is a registered trademark of Linus Torvalds.
#
# UNIX is a registered trademark of The Open Group.
#
#=============================================================================
#
# Permission is granted to make and distribute verbatim copies of this document
```

```
# provided the copyright notice and this permission notice are preserved on all
# copies.
#
# Permission is granted to copy and distribute modified versions of this
# document under the conditions for verbatim copying, provided that the entire
# resulting derived work is distributed under the terms of a permission notice
# identical to this one.
#
# Permission is granted to copy and distribute translations of this document
# into another language, under the above conditions for modified versions,
# except that this permission notice may be stated in a translation approved by
# Tripwire, Inc.
#
# DCM
```

A.2 FX Scanner-Anfragen

Nachfolgend die Anfragen, die vom Programm *FX Scanner* ausgeführt werden.

```
/MSADC/root.exe
/PBServer/..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe
/PBServer/..%%35c..%%35c..%%35cwinnt/system32/cmd.exe
/PBServer/..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe
/PBServer/..%255c..%255c..%255cwinnt/system32/cmd.exe
/Rpc/..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe
/Rpc/..%%35c..%%35c..%%35cwinnt/system32/cmd.exe
/Rpc/..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe
/Rpc/..%255c..%255c..%255cwinnt/system32/cmd.exe
/_mem_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe
/_vti_bin/..%%35%63..%%35%63..%%35%63..%%35%63..%%35%63../winnt/system32/cmd.exe
/_vti_bin/..%%35c..%%35c..%%35c..%%35c..%%35c../winnt/system32/cmd.exe
/_vti_bin/..%25%35%63..%25%35%63..%25%35%63..%25%35%63..%25%35%63../winnt/system32/cmd.exe
/_vti_bin/..%255c..%255c..%255c..%255c..%255c../winnt/system32/cmd.exe
/_vti_bin/..%255c../..%255c../..%255c../winnt/system32/cmd.exe
/_vti_bin/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe
/_vti_bin/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe
/_vti_cnf/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe
/_vti_cnf/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe
/adsamples/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe
/adsamples/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe
/c/winnt/system32/cmd.exe
/cgi-bin/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe
/cgi-bin/..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe
/d/winnt/system32/cmd.exe
/iisadmpwd/..%252f..%252f..%252f..%252f..%252f..%252fwinnt/system32/cmd.exe
/iisadmpwd/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe
/msaDC/..%%35%63..%%35%63..%%35%63..%%35%63winnt/system32/cmd.exe
/msaDC/..%%35c..%%35c..%%35c..%%35cwinnt/system32/cmd.exe
/msaDC/..%25%35%63..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe
/msaDC/..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe
/msadc/..%%35%63../..%%35%63../..%%35%63../winnt/system32/cmd.exe
/msadc/..%%35c../..%%35c../..%%35c../winnt/system32/cmd.exe
/msadc/..%25%35%63..%25%35%63..%25%35%63..%25%35%63winnt/system32/cmd.exe
/msadc/..%25%35%63../..%25%35%63../..%25%35%63../winnt/system32/cmd.exe
/msadc/..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe
/msadc/..%255c../..%255c../..%255c../winnt/system32/cmd.exe
/msadc/..%255c../..%255c../..%255c/..%c1%1c../..%c1%1c../..%c1%1c../winnt/system32/cmd.exe
/msadc/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe
/msadc/..%c1%af../winnt/system32/cmd.exe
/msadc/..%c1%pc../..%c1%pc../..%c1%pc../winnt/system32/cmd.exe
/msadc/..%c1%pc../winnt/system32/cmd.exe
/msadc/..%e0%80%af../..%e0%80%af../..%e0%80%af../winnt/system32/cmd.exe
/msadc/..%e0%80%af../winnt/system32/cmd.exe
/msadc/..%f0%80%80%af../..%f0%80%80%af../..%f0%80%80%af../winnt/system32/cmd.exe
/msadc/..%f0%80%80%af../winnt/system32/cmd.exe
/msadc/..%f8%80%80%80%af../..%f8%80%80%80%af../..%f8%80%80%80%af../winnt/system32/cmd.exe
/msadc/..%f8%80%80%80%af../winnt/system32/cmd.exe
/samples/..%255c..%255c..%255c..%255c..%255c..%255cwinnt/system32/cmd.exe
/samples/..%c0%af..%c0%af..%c0%af..%c0%af..%c0%af../winnt/system32/cmd.exe
/scripts..%c1%9c../winnt/system32/cmd.exe
/scripts/.%252e/.%252e/winnt/system32/cmd.exe
/scripts/..%%35%63../winnt/system32/cmd.exe
/scripts/..%%35c../winnt/system32/cmd.exe
```

```
/scripts/..%25%35%63../winnt/system32/cmd.exe
/scripts/..%252f..%252f..%252f..%252fwinnt/system32/cmd.exe
/scripts/..%252f../winnt/system32/cmd.exe
/scripts/..%255c%255c../winnt/system32/cmd.exe
/scripts/..%255c..%255cwinnt/system32/cmd.exe
/scripts/..%255c../winnt/system32/cmd.exe
/scripts/..%C0%AF..%C0%AF..%C0%AF..%C0%AFwinnt/system32/cmd.exe
/scripts/..%C1%1C..%C1%1C..%C1%1C..%C1%1Cwinnt/system32/cmd.exe
/scripts/..%C1%9C..%C1%9C..%C1%9C..%C1%9Cwinnt/system32/cmd.exe
/scripts/..%c0%9v../winnt/system32/cmd.exe
/scripts/..%c0%af../winnt/system32/cmd.exe
/scripts/..%c0%qf../winnt/system32/cmd.exe
/scripts/..%c1%1c../winnt/system32/cmd.exe
/scripts/..%c1%8s../winnt/system32/cmd.exe
/scripts/..%c1%9c../winnt/system32/cmd.exe
/scripts/..%c1%af../winnt/system32/cmd.exe
/scripts/..%c1%pc../winnt/system32/cmd.exe
/scripts/..%e0%80%af../winnt/system32/cmd.exe
/scripts/..%f0%80%80%af../winnt/system32/cmd.exe
/scripts/..%f8%80%80%80%af../winnt/system32/cmd.exe
/scripts/..%fc%80%80%80%80%af../winnt/system32/cmd.exe
/scripts/root.exe
/msadc/..%fc%80%80%80%80%af../..%fc%80%80%80%80%af../..%fc%80%80%80%80%af../winnt/↵
system32/cmd.exe
```

A.3 Passwörter des Deloder-Wurms

Folgende Passwörter werden für den Administrator-Account ausprobiert:

<no password>	123asd	foobar	pw
0	123qwe	god	pw123
000000	2002	godblessyou	pwd
00000000	2003	home	qwer
007	2600	ihavenopass	root
1	54321	Internet	secret
110	654321	Login	server
111	88888888	login	sex
111111	a	love	super
11111111	aaa	mypass	sybase
12	abc	mypass123	temp
121212	abc123	mypc	temp123
123	abcd	mypc123	test
123123	Admin	oracle	test123
1234	admin	owner	win
12345	admin123	pass	xp
123456	administrator	passwd	xxx
1234567	alpha	Password	yxcv
12345678	asdf	password	zxcv
123456789	computer	pat	
1234qwer	database	patrick	
123abc	enable	pc	

A.4 Benutzer und Passwörter des Gaobot-Wurms

Folgende Benutzer und Passwörter werden ausprobiert:

Benutzer:

Administrator	User	a	pc
admin	Administrador	aaa	test
administrator	Owner	abc	temp
Administrateur	Test	x	win
Default	Guest	xyz	asdf
mgmt	Gast	Dell	qwer
Standard	Inviter	home	login

Passwörter:

<no password>	abcd	pwd
0	Admin	qwer
000000	admin	root
00000000	admin123	secret
007	administrator	server
1	alpha	sex
110	asdf	super
111	computer	sybase
111111	database	temp
11111111	enable	temp123
12	foobar	test
121212	god	test123
123	godblessyou	win
123123	home	xp
1234	ihavenopass	xxx
12345	Internet	yxcv
123456	Login	zxcv
1234567	login	
12345678	love	
123456789	mypass	
1234qwer	mypass123	
123abc	mypc	
123asd	mypc123	
123qwe	oracle	
2002	owner	
2003	pass	
2600	passwd	
54321	Password	
654321	password	
88888888	pat	
a	patrick	
aaa	pc	
abc	pw	
abc123	pw123	

A.5 PHP-Skripte für die Auswertung

Bei den Skripten handelt es sich um prototypische Entwicklungen. Die Resultate der Datenbankabfragen wurden in Microsoft Excel übertragen und die Diagramme erstellt.

A.5.3 Bestimmung der „Mysterium-Pakete“ pro Tag

```
<?
  $server = "localhost";
  $database = "iptablesnew";
  $databaseuser = "root";
  $databasepwd = "";
  $handle = @mysql_connect($server,$databaseuser,$databasepwd);
?>
<html>
<head>
<title>Auswertung f&uuml; 55808-Pakete</title>
</head>
<?
if (isset($HTTP_POST_VARS["sent"]))
    $sent = $HTTP_POST_VARS["sent"];
  else
    $sent = 0;
if (isset($HTTP_POST_VARS["datum"]))
    $datum = $HTTP_POST_VARS["datum"];
  else
    $datum = 0;
?>
<body>
<form method="POST" action="<? echo $PHP_SELF ?>">
<select size="1" name="datum">
<?
$secondsperday = 86400;
$day = mktime(0,0,0,7,15,2003);
for ($i=0;$i<60;$i++)
{
       echo "<option value='" . ($day+$i*$secondsperday) . "'>" .↵
       strftime("%a %d.%b",($day+$i*$secondsperday)) . "</option>";
}
?>
</select>
<input type="hidden" name="sent" value="1">
<input type="submit" value="Anfrage abschicken" name="detail" class="button">
<br>
<?
if($handle && $sent)
{
  echo "<br>" . strftime("%d.%m.%y",($datum)) . "<br><br>";
  $select = mysql_select_db($database,$handle);
  $sql = "SELECT ip_src, count(ip_src), name_src FROM logs WHERE (date >= '" . ↵
  strftime("%Y-%m-%d %H:%M:%S",($datum)) . "' AND date < '" .
  strftime("%Y-%m-%d %H:%M:%S",($datum+$secondsperday)) . "') AND port_dest = '57669'↵
  GROUP BY ip_src";
  echo "<table width='50%' border='1'>";
  echo "<tr>";
  echo "<td width='30%'>IP-Adresse</td>";
  echo "<td width='10%'>Anzahl</td>";
  echo "<td width='60%'>DNS-Name</td>";
  echo "</tr>";
  $summe = 0;
  $result = mysql_query($sql,$handle);
  if ($result)
  {
    while ($row = mysql_fetch_array($result))
    {
      echo "<tr>";
      echo "<td>" . $row["ip_src"] . "</td>";
      echo "<td>" . $row[1] . "</td>";
      $summe += $row[1];
```

```
      echo "<td>" . $row["name_src"] . "</td>";
      echo "</tr>";
    }
  }
}
mysql_close();
?>
</table>
<br>
Summe: <? echo $summe ?>
</form>
</body>
</html>
```

A.5.4 Bestimmung der Anzahl der Verbindungen pro Stunde

```
<?
  $server = "localhost";
  $database = "iptablesnew";
  $databaseuser = "root";
  $databasepwd = "";
  $handle = @mysql_connect($server,$databaseuser,$databasepwd);
?>
<html>
<head>
<title>Verbindungen pro Stunde</title>
</head>
<body>
<table width='100%' border='1'>
  <tr>
  <td> </td>
  <td>0:00 - 0:59 Uhr</td>
  <td>1:00 - 1:59 Uhr</td>
  <td>2:00 - 2:59 Uhr</td>
  <td>3:00 - 3:59 Uhr</td>
  <td>4:00 - 4:59 Uhr</td>
  <td>5:00 - 5:59 Uhr</td>
  <td>6:00 - 6:59 Uhr</td>
  <td>7:00 - 7:59 Uhr</td>
  <td>8:00 - 8:59 Uhr</td>
  <td>9:00 - 9:59 Uhr</td>
  <td>10:00 - 10:59 Uhr</td>
  <td>11:00 - 11:59 Uhr</td>
  <td>12:00 - 12:59 Uhr</td>
  <td>13:00 - 13:59 Uhr</td>
  <td>14:00 - 14:59 Uhr</td>
  <td>15:00 - 15:59 Uhr</td>
  <td>16:00 - 16:59 Uhr</td>
  <td>17:00 - 17:59 Uhr</td>
  <td>18:00 - 18:59 Uhr</td>
  <td>19:00 - 19:59 Uhr</td>
  <td>20:00 - 20:59 Uhr</td>
  <td>21:00 - 21:59 Uhr</td>
  <td>22:00 - 22:59 Uhr</td>
  <td>23:00 - 23:59 Uhr</td>
  </tr>
<?
$HP = "129.187.19.105";
//$HP = "129.187.19.106";
$secondsperday = 86400;
$day = mktime(0,0,0,7,15,2003);
if($handle)
{
  $select = mysql_select_db($database,$handle);
  for ($i=0;$i<60;$i++)
  {
    echo "<tr>";
    echo "<td>" . strftime("%d.%m.%y",($day+$i*$secondsperday)) . "</td>";
    for ($j=0;$j<24;$j++)
    {
      $sql = "SELECT COUNT(*) FROM logs WHERE (date >= '" . ↵
      strftime("%Y-%m-%d %H:%M:%S",($day+$i*$secondsperday+$j*3600)) . "' AND↵
      date < '" . strftime("%Y-%m-%d %H:%M:%S",($day+$i*$secondsperday+($j+1)*3600)) . "'↵
      AND ip_dest='" . $HP . "')";
```

```
      $result = mysql_query($sql,$handle);
      if ($result)
      {
        while ($row = mysql_fetch_array($result))
       {
         echo "<td>" . $row[0] . "</td>";
       }
      }
    }
    echo "</tr>";
  }
}
mysql_close();
?>
</table>
</body>
</html>
```

A.5.5 Bestimmung der Top Hosts

```
<?
  $server = "localhost";
  $database = "iptablesnew";
  $databaseuser = "root";
  $databasepwd = "";
  $handle = @mysql_connect($server,$databaseuser,$databasepwd);
?>
<html>
<head>
<title>Top Hosts</title>
</head>
<body>
<table>
  <tr>
    <td>IP-Adresse</td>
    <td>DNS-Name</td>
    <td>Anzahl<br> Verbindungen</td>
    <td>Port (H&auml;ufigkeiten)</td>
    <td>Erstes Auftreten</td>
    <td>Letztes Auftreten</td>
</tr>
<?
$HP = "129.187.19.105";
//$HP = "129.187.19.106";
$limit = 10;
if($handle)
{
  $select = mysql_select_db($database,$handle);
  $sql = "SELECT ip_src, name_src, COUNT(ip_src) AS anzahl FROM logs WHERE ip_dest='$HP'↵
  GROUP BY ip_src ORDER BY anzahl DESC LIMIT $limit";
  $result = mysql_query($sql,$handle);
  if ($result)
  {
    while ($row = mysql_fetch_array($result))
    {
      echo "<tr>";
      echo "<td>" . $row["ip_src"] . "</td>";
      echo "<td>" . $row["name_src"] . "</td>";
      echo "<td>" . $row["anzahl"] . "</td>";
      echo "<td>";
      $sql1 = "SELECT COUNT(*) FROM logs WHERE ip_src = '" . $row["ip_src"] . "' AND↵
      chain = 'IN_ICMP' AND ip_dest='$HP'";
      $result1 = mysql_query($sql1,$handle);
      if ($result1)
      {
        while ($row1 = mysql_fetch_array($result1))
        {
          if ($row1[0] != 0)
            echo "ICMP (". $row1[0] . ")<br>";
        }
      }
      $sql2 = "SELECT port_dest, COUNT(port_dest) AS anzahl FROM logs WHERE ip_src = '" . ↵
      $row["ip_src"] . "' AND chain != 'IN_ICMP' AND ip_dest='$HP' GROUP BY port_dest ORDER↵
      BY port_dest";
```

```
      $result2 = mysql_query($sql2,$handle);
      while ($row2 = mysql_fetch_array($result2))
      {
        echo $row2["port_dest"] . "  (". $row2["anzahl"] . ")<br>";
      }
      echo "</td><td>";
      $sql3 = "SELECT date FROM logs WHERE ip_src = '" . $row["ip_src"] . "' ORDER BY date↵
      ASC LIMIT 1";
      $result3 = mysql_query($sql3,$handle);
      while ($row3 = mysql_fetch_array($result3))
      {
        echo substr($row3["date"],8,2) . "." . substr($row3["date"],5,2) .".". ↵
        substr($row3["date"],0,4). "<br>". substr($row3["date"],11,8);
      }
      $sql4 = " SELECT date FROM logs WHERE ip_src = '" . $row["ip_src"] . "' ORDER BY↵
      date DESC LIMIT 1";
      $result4 = mysql_query($sql4,$handle);
      echo "</td><td>";
      while ($row4 = mysql_fetch_array($result4))
      {
        echo substr($row4["date"],8,2) . "." . substr($row4["date"],5,2) .".". ↵
        substr($row4["date"],0,4) . "<br>" . substr($row4["date"],11,8);
      }
      echo "</td></tr>";
    }
  }
}
mysql_close();
?>
</body>
</html>
```