# Aktive Gegenangriffe
# Reactive Components and Systems

Studienarbeit zur Erlangung eines Leistungsnachweises im 6. Semester

Maik Dobryn  Sven Ludwig  Sascha Sertel  Richard Stevens

Fachhochschule Bonn-Rhein-Sieg
Sankt Augustin, Sommersemester 2002

## Zusammenfassung

Dieser Bericht befasst sich mit automatisierten Gegenangriffen gegen Angreifer im Cyberspace. Ein erkannter Angriff auf ein Informationssystem soll nicht nur gestoppt oder seine Wirkung abgeschwächt werden, sondern es soll das System des Angreifers attakiert werden. Die Gegenangriffe können verschiedene Ziele haben. Sie sollen zum Beispiel zu einer Identifizierung des Angreifers führen, möglichst ohne, dass dieser es bemerkt. Oder das System des Angreifers soll ausser Gefecht gesetzt werden. Außerdem können Gegenangriffe eine abschreckende Wirkung auf weitere Angreifer haben. Wir betrachten Geganangriffe als aggressive Gegenmaßnahmen. Um die Zusammenhänge deutlich zu machen, beschäftigen wir uns auch mit defensiven Gegenmaßnahmen und mit besonderen Mitteln zur Informationsbeschaffung. Zudem gehen wir auf die derzeitige Entwicklung von verteilten Sicherheitssystemem ein, die durch eine Zusammenarbeit verschiederner Komponenten im Netzwerk eine effektivere Bekämpfung von Angriffen ermöglichen sollen. Eine Beispielkonfiguration eines Systems, dass auf einen Angriff direkt ohne weitere Prüfung mir einem Gegenangriff reagiert, schließt den Bericht ab.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# 1 Einführung

Maik Dobryn, Sven Ludwig, Sascha Sertel, Richard Stevens

Dieses Kapitel bietet eine kurze Einführung in die Thematik dieses Berichts. Es wird eine Einteilung der Begriffe 'Angriff' und 'Maßnahme' in Bezug auf die Informationssicherheit vorgenommen. Anschließend wird auf das Thema Intrusion Detection eingegangen, da die Erkennung eines Angriffs eine notwendige Voraussetzung für einen Gegenangriff ist.

## 1.1 Historischer Abriss

Betrachtet man die Entwicklung der Computertechnik seit der Mitte des 20. Jahrhunderts, so stellt man fest, dass gleichzeitig auch eine Entwicklung der Angreifer und Angriffsmethoden stattgefunden hat. Bevor es weitreichende Netze wie das Internet gab, waren primär Telefonleitungen und die Firmen, die Telefondienste anboten, Ziel von Angriffen. Mit dem Übergang zu Rechnernetzen zogen auch die Angreifer mit und nutzten die geringen Sicherheitsvorkehrungen in den neuen Systemen zu ihrem Vorteil aus. Trotz der bereits dreißigjährigen Geschichte von Angriffen auf Rechnersysteme ist das Sicherheitsbewusstsein in den Köpfen der meisten Leute viel zu niedrig, die Gefahr für das eigene Rechnersystem wird als zu gering eingestuft, und mögliche Angreifer können ungehindert mit ihren Aktivitäten fortfahren.

Hier ein Auszug einiger wichtiger Meilensteine in der Entwicklung von Technik und Angriffen:

- **1876** Alexander Graham Bell erfindet das Telefon.
- **1971** John Draper findet heraus, dass eine Spielzeugpfeife aus einer Müsli-Schachtel genau den Ton reproduziert, der eine freie Telefonleitung öffnet. Er wird bekannt unter dem Namen „Captain Crunch“.
- **1977** Steve Jobs und Steve Wozniak, die sich bisher mit der Produktion von sogenannten „blue boxes“, Geräten zum Hacken der Computer von Telefongesellschaften, beschäftigt haben, gründen Apple Computer.
- **1981** IBM führt seine erste Version des Personal Computer auf dem Markt ein.
- **1983** Der Film „War Games“ mit Matthew Broderick läuft in den Kinos an.[1]

[1] Die Auflistung dieses Ereignisses mag in der Liste auf den ersten Blick deplaziert erscheinen. Bedenkt man jedoch, dass dieser Film erstmalig Angriffe von Rechnersystemen, in diesem Fall Militärische Computersyteme, thematisierte und realistisch die Gefahren aufzeigte, wird das entstehende Bewusstsein für Systemsicherheit deutlich.

- **1983-1984** AT& T wird in sieben Firmen aufgeteilt, die neuen Boden für Angriffe liefern.
- **1984** In den USA wird der Comprehensive Crime Control Act verabschiedet, ein Gesetz, dass dem Geheimdienst mehr Möglichkeiten gibt, Kreidtkartenbetrügern und Hackern das Handwerk zu legen.
- **1984** Gründung von 2600: The Hacker Quarterly, einem Magazin für den Hackeruntergrund.[2]
- **1986** In den USA werden zwei weitere Gesetze, die sich mit Angriffen auf Computersystemen beschäftigen, verabschiedet: Der Computer Fraud and Abuse Act und der Electronic Communications Privacy Act.
- **1988** Robert Morris bringt 6,000 Computer im Internet mit einem Virus zum Absturz und wird zu einer Strafe von $10.000 verurteilt.[3]
- **1990, 7.-9. Mai** Der Geheimdienst der USA startet in vielen Städten der Vereinigten Staaten Razzien unter dem Namen „Operation Sundevil“.
- **1991, 25.-28. März** Die erste Konferenz zum Thema Systemsicherheit mit dem Namen „Computers, Freedom and Privacy“ findet in San Francisco statt.
- **1993** Der amerikanische Geheimdienst verhaftet Mitglieder der Hackergruppe Masters of Deception. Alle werden für Computer Verbrechen schuldig gesprochen.
- **1994, Sommer** Vladimir Levin, Absolvent der St. Petersburg Tekhnologichesky Universität, stiehlt mit einer russischen Hackergruppe 10 Millionen US-Dollar von der Citibank. Er wird 1995 in London festgenommen.
- **1995, 15. Februar** Kevin Mitnik wird verhaftet und angeklagt, 20.000 gültige Kreditkartennummern gestohlen zu haben. Er bekennt sich 1996 schuldig.
- **1998, Frühling** Pentagon Computer werden wiederholt angegriffen. Der israelische Jugendliche Ehud Tenebaum, auch bekannt als „The Analyzer“, behauptet, zwei kalifornischen Jugendlichen erklärt zu haben, wie sie das Pentagon angreifen können.
- **1998, 19. Mai** Mitglieder der Hackergruppe L0pht sagen vor dem amerikanischen Senat aus und warnen vor ernsthaften Sicherheitslücken. Sie behaupten, das Internet innerhalb einer halben Stunde lahmlegen zu können.

Das Spektrum von Angreifern und Angegriffenen ist heutzutage sehr breit gefächert, von einzelnen Jugendlichen bis hin zu organisierten Hackergruppen finden Angriffe sowohl auf Einzelrechner von Privatpersonen, als auch auf namhafte (und daher als Angriffsziel beliebte) Institutionen weltweit statt. Genauso verhält es sich mit den Motiven für diese Angriffe, man findet starke politische Ziele ebenso vor wie einfache Neugierde.

[2] Der Name 2600 entstand in Anlehnung an die Entdeckung, die John Draper 1971 gemacht hatte. Die Frequenz zum Öffnen einer Telefonleitung betrug damals 2600 Hz.

[3] Zu dieser Zeit bestand das Internet nur aus wenig mehr als 6,000 Hosts, daher ist das Verhältnis des angerichteten Schadens weitaus größer, als auf den ersten Blick zu vermuten ist.

## 1.2 Begriffsbildung

Im folgenden werden Begriffe definiert, um deren Bedeutung innerhalb dieses Berichts festzulegen. Die hier aufgeführten Definitionen erheben keinen Anspruch auf Allgemeingültigkeit. Bei den beiden Definitionen zu passiven und aktiven Angriffen handelt es sich jedoch um gängige Interpretaionen [Service n.d.]. Bei der Begriffsbildung wird die Verwendung von medienwirksamen Begriffen wie „Hacker“ oder „Cracker“ und damit verbundenen Verben vermieden, da die Betrachtung möglichst neutral erfolgen soll.

### 1.2.1 Angreifer

Allgemein versteht man unter einem Angreifer jemanden, der unbefugt versucht auf Daten zuzugreifen, die sich auf einem Rechnersystem befinden. Dabei werden oftmals Sicherheitsmechanismen umgangen oder außer Kraft gesetzt, um dieses Ziel zu erreichen. Der Zugriffsversuch des Unberechtigten kann dabei sowohl intern, also direkt auf dem gleichen Rechner, als auch verteilt, also über eine Netzwerkstruktur hinweg, erfolgen. Eine spezielle Form des Angreifers ist das sogenannte Tigerteam. So werden Angreifer genannt, die im Auftrag von Sicherheitssystembetreibern ein System auf Schwachstellen überprüfen sollen.

### 1.2.2 Passiver Angriff

Der passive Angriff bezeichnet einen Angriff, bei dem der Angegriffene im Idealfall gar nicht merkt, dass er angegriffen wird. Dazu zählen Angriffe wie Abhören von Leitungen, Mitlesen von Datenpaketen und Passwörtern oder auch die Installation Trojanischer Pferde, die Daten auf dem System sammeln und unbemerkt zum Angreifer übermitteln. Auch John Draper, der 1971 eine Spielzeugpfeife zum Freischalten einer Telefonleitung benutzte (siehe 1.1 auf Seite 6), übte damit einen passiven Angriff auf die Telefongesellschaft aus. Der Angegriffene kann sich vor solchen Angriffen schützen, indem er beispielsweise seine Daten sowie seine Kommunikation verschlüsselt und somit dem Angreifer den Aufwand erschwert, Informationen aus den Daten zu gewinnen.

### 1.2.3 Aktiver Angriff

Im Gegensatz zum passiven Angriff ist der aktive Angriff mit einer direkten, spürbaren Wirkung beim Angegriffenen verbunden. Der bekannteste aktive Angriff ist die *Denial of Service Attack*, die durch eine sehr hohe Anzahl dauerhafter Anfragen an ein Rechnersystem dessen Netzwerkverbindung deaktiviert. Zu aktiven Angriffen zählen aber auch die Verwendung von Sicherheitslücken in Systemdiensten, mit deren Hilfe sich ein Unberechtigter Zugriff auf ein System verschafft. Gegen aktive Angriffe müssen sich die Kommunikationspartner durch Authentifizierungsverfahren, digitale Unterschriften und weitere Sicherheitsmaßnahmen schützen.

### 1.2.4 Schutzmaßnahmen

Wie bereits bei den Definitionen der Angriffsarten beschrieben, müssen Angegriffene den Widerstandswert ihrer Systeme durch geeignete Sicherheitsmechanismen erhöhen. Die möglichen Schutzvorkehrungen bewegen sich dabei auf unterschiedlichen Ebenen:

- **Zugriffsebene:** Durch Identifizierungs- und Authentifizierungsmechanismen wird vermieden, dass Unberechtigte ohne Weiteres Zugriff auf das System erlangen.
- **Administrationsebene:** Jeder Benutzer sollte immer nur so viele Rechte auf einem System erhalten, wie er unbedingt benötigt. Die Aktionen der Benutzer sollten außerdem in Logdateien abgespeichert werden.
- **Netzwerkebene:** Netzwerkverbindungen sollten über eine (oder mehrere) Firewall ablaufen, die den Verkehr mit Kommunikationspartnern kontrolliert und gegebenenfalls Unberechtigten den Netzwerkzugriff verweigert.

Es gibt außerdem noch weitere Schutzmaßnahmen, für diese Definition soll ein grober Überblick aber genügen.

### 1.2.5 Gegenmaßnahmen und Gegenangriffe

Gegenmaßnahmen sind Maßnahmen, die sich gegen einen zuvor entdeckten Angriff richten und diesen verhindern oder dessen Wirkung abschwächen sollen. Wir unterteilen Gegenmaßnahmen in defensive und offensive bzw. aggressive Gegenmaßnahmen. Defensive Gegenmaßnahmen können oft als eine temporäre Änderung der bestehenden Konfiguration von Sicherheitskomponenten angesehen werden. Dies betrifft zum Beispiel die vorübergehende Sperrung eines Ports auf einer Firewall. Defensive Gegenmaßnahmen können zusätzlich zur Abwendung eines Angriffs auch der weiteren Informationsbeschaffung dienen. Damit sind Gegenmaßnahmen gemeint, die den Angreifer in die Irre führen und ihn Weile binden, so dass Informationen über ihn gesammelt werden können. Gegenmaßnahmen, die einen Angriff auf das System des Angreifers darstellen, fassen wir als offensive bzw. aggressive Gegenmaßnahmen auf. Es sind Gegenangriffe.

## 1.3 Intrusion Detection

Nachdem definiert wurde, was ein Angriff ist, geht es in diesem Abschnitt um die Frage, wie sich ein Angriff erkennen lässt und welche Informationen über den Angreifer sich daraus gewinnen lassen. Die Betrachtung bewegt sich dabei auf einer Abstraktionsebene, das heißt es werden keine bestimmten auf dem Markt befindlichen Produkte und deren Funtionalität betrachtet, sondern generell erläutert, welche Möglichkeiten (praktisch oder theoretisch) existieren, um überhaupt einen Angriff zu erkennen.

### 1.3.1 Erkennen eines Angriffs

Wie bereits beschrieben hat sich die Geschichte der Angriffe ebenso wie die Technik über einen langen Zeitraum entwickelt, daher sind die verschiednene existierenden Arten von Angriffen

sehr vielfältig, ebenso bieten heutige Systeme eine Vielzahl von Angriffsmöglichkeiten, die das Erkennen eines Angriffes erschweren. In der Theorie wie in der Praxis ist dabei die Methode weitverbreitet, Aktionen rund um das System, das heißt Zugriffe auf das System sowie Netzwerkkommunikation, nach bestimmten Angriffsmustern zu durchsuchen und zu bewerten. Wurde ein solches Muster gefunden, kann nicht mit Sicherheit gesagt werden, ob es sich tatsächlich um einen Angriff handelt oder nur zufällig eine Abfolge von erlaubten Zugriffen erfolgte, die dem Angriffsmuster gleicht. Auf den ersten Blick möchte man meinen, dass es besser sei, jeden möglichen Angriff zu melden, um sicher zu sein, dass echte Angriffe auf jeden Fall entdeckt werden. Allerdings gibt es hier zwei Punkte, an denen diese Einstellung scheitern kann:

- Durch eine hohe Zahl an Fehlalarmen durch falsch erkannte Angriffsmuster wird das Verhalten der Sicherheitsbeauftragten eines Rechnersystems beeinträchtigt. Nach einiger Zeit nimmt die Sensitivität für einen Alarm ab, da man wieder nur mit einem Fehlalarm rechnet, und kann sogar zur Abschaltung von Sicherheitsmechanismen führen, um nicht mehr weiter Fehlalarme zu erhalten.
- Auf der anderen Seite wird durch zu viele Alarme ein falsches Sicherheitsbewusstsein geschaffen, da sich der Benutzer in Sicherheit wägt, dabei aber vergisst, dass es auch Angriffe gibt, deren Muster nicht im System gespeichert ist und daher auch nicht erkannt wird.

### 1.3.2 Welche Informationen liefert ein IDS

Intrusion Detection Systeme sind primär dazu ausgelegt, über Anomalien im Netzverkehr und gezielte Angriffe zu informieren. Sie arbeiten dabei meist Filterbasiert. Der Netzverkehr wird von einem Sensor überwacht und wenn ein bekanntes Muster vorliegt, wird eine definierte Reaktion ausgelöst, meist sind dies Logeinträge oder Benachrichtigungen per Email, Pager oder SMS. Zusätzlich können aber auch aktive Maßnahmen durchgeführt werden.

Netbasierte Intrusion Detection basiert auf der Analyse der Pakete, die im Netzwerk transportiert werden. Daher kann ein solches System alle Informationen liefern, die aus einem Protokollpaket ersichtlich sind. Dazu gehören Quell- und Zieladresse des Pakets, Protokollart, mitgeführte Daten und Protokollparameter. Neben diesen Protokollspezifischen Informationen erkennt man auch noch die Art der Anomalie oder des Angriffs.

Neben der Analyse einzelner unabhängiger Pakete bieten Intrusion Detection Systeme auch die Möglichkeit, Protokollsitzungen und deren Inhalte zu analysieren. Dadurch ist es z.B. möglich, Versuche zu erkennen, ein Passwort durch manuelles oder automatisiertes Ausprobieren zu knacken.

Damit die Zahl der Fehlalarme gering gehalten werden kann, können Grenzen definiert werden bis zu denen registrierte Anomalien als normal eingestuft werden. Ein gelegentliches verdächtiges Paket oder ein gelegentlich falsch eingegebenes Passwort muss nicht gleich auf einen Angriff hindeuten.

# 2 Informationsbeschaffung

Richard Stevens

Lernen ist wie Rudern gegen den Strom. Sobald man aufhört, treibt man zurück.
*Benjamin Britten (1913-76), brit. Komponist*

Die Informationsbeschaffung ist eine der Hauptaufgaben, wenn es um den Schutz von Rechnersystemen vor Angreifern geht. Dabei ist es absolut notwendig, immer über aktuelle Lücken und Fehler in der eingesetzten Software informiert zu sein und diese umgehend zu schließen. Diese reaktive Vorgehensweise ist jedoch bei weitem nicht ausreichend, um mit hoher Wahrscheinlichket erfolgreich Gegenmaßnahmen gegen einen Angriff durchführen zu können. Das Wissen über die potentiellen Angreifer und deren Methoden ermöglicht erst Aktion und nicht nur Reaktion.

In diesem Kapitel sollen daher Werkzeuge und Methoden vorgestellt werden, die es ermöglichen, mehr über die Gefahren für vernetzte Rechnersysteme und deren Auslöser zu erfahren. Abschließend sollen noch Anregungen für eine Aus- und Verwertung der gesammelten Daten geliefert werden. Dabei ist es wichtig, die Informationsbeschaffung analog zu obenstehendem Zitat von Benjamin Britten als Prozeß und nicht als einmaliges Ereignis zu sehen. Wer nicht bereit oder in der Lage ist, permanent sein Wissen über Angriffsmethoden und Angreifer zu erweitern und zu validieren, sowie die gewonnenen Erkenntnisse in Schutzmaßnahmen einfließen zu lassen, der wird sehr viel leichter nicht nur Ziel, sondern Opfer eines Angriffs.

## 2.1 Klassifizierung der Methoden zur Informationsbeschaffung

Es gibt verschiedene Wege, über die sich sicherheitsrelevante Informationen beschaffen lassen. Ganz offensichtlich ist die Möglichkeit, sich über einschlägige Webseiten und Literatur zu Informieren. Als Beispiele könnten hier die Webseiten der Hersteller und Entwickler, die Webseiten und Warnungen des CERT[1] des Software Engineering Institutes der Carnegie Mellon Universität oder Security Focus[2] genannt werden. Quellen dieser Art bieten zeitnah Informationen zu bekannten Sicherheitslücken. Häufig werden auch Grundsatzartikel zu Sicherheitsthemen veröffentlicht.

Ein zweiter Weg sind aktive Verfahren. Dazu zählen die Überwachung und Analyse des Netzverkehrs, sowie automatisierte Scans eines potentiellen Angreifers mit geeigneten Werkzeugen. Die Überwachung des Netzverkehrs findet dabei lokal im eigenen Netz statt, ohne das ein potentieller Angreifer davon Kenntnis erlangen muss. Die Analyse des Systems eines potentiellen

[1] Computer Emergency Response Team (http://www.cert.org)
[2] http://www.securityfocus.com

Angreifers verursacht Netzwerkverbindungen zum entfernten System, die dort erkannt werden können.

## 2.2 Honeypots und Honeynets

Angriffe können durch Analyse des ein und ausgehenden Netzwerkverkehrs ekannt werden. Voraussetzung für eine solche Analyse ist umfangreiches Logging des Netzverkehrs. Hieraus ergeben sich allerdings schwerwiegende Probleme. Firmennetze sind heute meist über Leitungen mit hoher Bandbreite mit dem Internet verbunden. In Zukunft kann damit gerechnet werden, dass die Kapazität dieser Anbindungen weiter steigt. Wie können die zwischen den harmlosen Paketen versteckten verdächtigen Pakete für eine Analyse herausgefiltert werden. Wie stellt man sicher, dass alle relevanten Pakete gefiltert und analysiert werden? Soll der über die Leitung gehende Datenstrom vollständig archiviert werden, oder beschränkt man sich auf speziell herausgefilterte Pakete. In Kapitel 1 wurden Verfahren zur Erkennung von Angriffen vorgestellt. Dabei überwachen Intrusion Detection Systeme den durch das eigene Netz fließenden Datenstrom und Vergleichen die Paktete mit Signaturen bekannter Angriffe. Wird ein verdächtiges Paket erkannt, kann es durch einen Logmechanismus vollständig archiviert werden. Dabei werden aber ausschließlich Pakete mit schon bekannten Signaturen gefiltert. Es gibt keine Möglichkeit das automatisierte Filtern von bisher unbekannten gefährlichen Paketen zu automatisieren. Hinzu kommt, dass Informationen über die von Angreifern verwendeten Maschinen zwar aufschlußreich sind, aber nur wenig Informationen über die Vorgehensweise und kaum Informationen über die Motive liefern können. Aber gerade Kenntnis über die Vorgehensweisen der Angreifer ermöglicht die Realisierung gezielter Schutzmechanismen. Darüber hinaus könenn die so gelernten Vorgehensweisen in die Verfahren für Gegenangriffe eingebunden werden.

Eine Möglichkeit die gewünschten Informationen zu erhalten ohne mit den geschilderten Problemen der hohen Datenmengen kämpfen zu müssen bieten sogenannte Honeypots oder sogar Honeynets. Honeypots sind Systeme, die gezielt dafür eingerichtet werden, dass sie Ziel von Angreifern werden. Sie sind mit geeigneten Warn- und Logmechanismen ausgestattet und ermöglichen so, einen Angriff zu beobachten und später genau zu analysieren. Honeynets sind mehrere Honeypots in einem überwachten und stark kontrollierten Netzsegment.

Das Honeypot Konzept gibt es schon einige Jahre. Der Grundgedanke ist, einem Angreifer ein System zur Verfügung zu stellen, das aufgrund von Sicherheitslücken ein interessantes Ziel für einen Angriff darstellt. Das System untersteht einer starken Überwachung, die es ermöglicht, den Angreifer genau zu beobachten. Ursprünglich wurden Honeypots in bestehende produktive Netze eingebunden, um Angreifer auf das vermeindlich einfach Ziel zu lenken.

Dadurch ist es möglich, sich die nötige Zeit zu erkaufen, Gegenmaßnahmen zu ergreifen und den Angreifer zu stoppen. Es existieren Produkte auf dem Markt, die es ermöglichen, ein ganzes Netz von Rechnern mit sehr limitierten Funktionen zu simulieren, bis hin zu kompletten Betriebssysteminstanzen in einem besonders abgeriegelten, geschützen Bereich, der auch als Jail bezeichnet wird. Das Honeynet Projeket [*The Honeynet Project* n.d.], ein Zusammenschluß von 30 Sicherheitsexperten hat das Honeypot Konzept erweitert und auf dedizierte

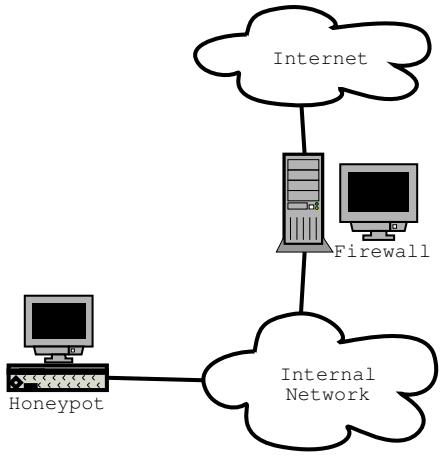


Abbildung 2.1: Honeypot hinter einer Firewall in einem Netzwerk

Netze ausgeweitet. Honeynets sind speziell abgeschottete und überwachte Netzsegmente, in denen Hardware und Systeme wie in einem produktiven Umfeld eingesetzt werden. Diese mit heterogenen Systemen bestückten Netze ermöglichen eine viel genauere Analyse der Vorgehensweisen und Motive der Angreifer, da sie aufgrund der unterschiedlichen Systeme Ziele für Angreifer mit unterschiedlichen Fähigkeiten bieten und äusserlich kaum von poduktiven Netzwerken unterscheidbar sind. Das Ziel des Honeynet Projekts ist es, mit diesen Netzwerken, Sicherheitslücken in den Verwendeten Systemen, sowie Vorgehensweisen und Motive der Angreifer zu analysieren und zu veröffentlichen.

Honeynets sind hochkontrollierte Umgebungen. Es muss sichergestellt werden, dass ein erfolgreicher Angreifer keinen oder wenn überhaupt möglichst wenige Schritte vollziehen kann, ohne dass diese aufgezeichnet werden. Weiterhin darf ein in einem Honeynet überwundenes System nicht als Sprungbrett für Attacken gegen andere Systeme verwendet werden können. Aus diesem Grund werden die Log- und Sicherheitsmechanismen mehrschichtig und redundant ausgelegt. Die vom Honeynet Projekt verwendete Konfiguration besteht aus einer Reihe von verschiedenen Systemen, die hinter einer Firewall platziert sind. Die Firewall erlaubt freien Zugriff von außen auf das Honeynet. Sämtliche ausgehenden Verbindungen sind jedoch auf eine festgelegte Anzahl begrenzt, da ausgehende Verbindungen auch für Atacken verwendet werden können. Eine gewisse Anzahl von Verbindungen nach außen wird jedoch bewusst erlaubt, da die Angreifer im Glauben gelassen werden sollen, die überwundenen Systeme seien für ihre Zwecke Nutzbar. Oft werden Hilfsmittel zur Einrichtung von Hintertüren in die überwundenen Systeme aus dem Internet heruntergeladen und installiert. Wäre dies nicht möglich, würden Angreifer möglicherweise erkennen, dass sie sich in einem Honeynet befinden oder dass die Systeme für ihre Zwecke wertlos sind. Außerdem würden Möglichkeiten zur Analyse der Vorgehensweise deutlich eingeschränkt.

Eine der Wichtigsten Eigenschaften eines Honeynets ist, dass sämtliche Aktivitäten eines Eindringlings genau analysiert werden können. Dafür ist vollständiges Logging nötig. Angreifer versuchen in der Regel, ihre Spuren auf dem Überwundenen System zu verwischen oder vollständig zu beseitigen. Daher würden Logdateien auf dem erfolgreich angegriffenen System nicht ausreichen, eine zuverlässige Überwachung zu realisieren. Die Mitglieder des Honeynet

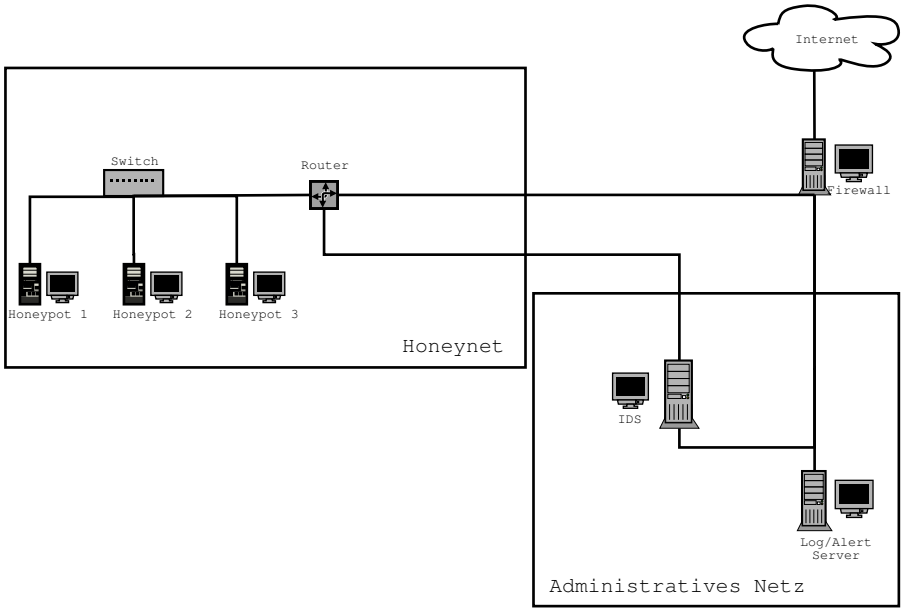


Abbildung 2.2: Beispiel eines Honeynets

Projekts haben eine für ihre Zwecke zuverlässige Methode entwickelt, vollständiges Logging zu gewährleisten, auch wenn der Angreifer die Logs auf dem System unbrauchbar machen kann. Die erste überwachende Instanz ist die Firewall. Ein Honeynet ist ein Netzwerk, das sich in einem Punkt deutlich von produktiven Systemen unterscheidet. Es verursacht bewusst keinen oder nur sehr wenig Netzverkehr nach außen. Jede ein- und ausgehende Verbindung ist daher verdächtig. Die Firewall vermerkt daher jede Verbindung in und aus dem Honeynet auf einem Besonderen System in einem zusätzlichen administrativen Netzsegment, das auf keinen Fall aus dem Honeynet erreichbar sein darf. Weiterhin befindet sich hinter der Firewall ein Intrusion Detection System, das den gesamten Datenverkehr innerhalb des Honeynets analysiert und archiviert.

Die Positionierung dieses Intrusion Detection System (IDS) hat dabei Auswirkungen auf die Art der Informationen, die gesammelt werden können. Eine Positionierung des IDS außerhalb der Firewall analysiert alle Pakete, die an dieser Position über die Leitung kommen. Ein zwischen Router und und Firewall positioniertes IDS liefert sämtlichen Netzwerkverkehr, der zwischen den Honeynetsystemen und der Firewall erfolgt. Pakete, die nicht für das Honeynet bestimmt sind, werden nicht erfasst. Dadurch enthalten die Logs zwar weniger, aber relevantere Informationen. Die letzte Möglichkeit ist eine Positionierung hinter dem Router. Dadurch ist es möglich, auch den Netzverkehr zwischen den Honeypot Rechnern zu erfassen. Diese Position ermöglicht umfangreiches Logging auch der Meldungen an den Syslog Server. Daher ist diese Position zu präferieren.

Das IDS bildet die zweite Instanz neben der Firewall, die die Administratoren des Netzwerks über festgestellte Angriffe informiert. Das Intrusion Detection System besitzt keinen IP Stack und ist daher vor Angriffen aus dem Honeynet geschützt. Innerhalb des Honeynets befindet sich noch ein besonders gesichtertes System, das über das Netzwerk Syslog Meldun-

gen aller Honeynetsysteme erhält. Angreifer, die ein System überwunden haben, stellen unter Umständen fest, dass dieses Logsystem existiert. Um unerkannt zu bleiben, versuchen versierte Angreifer mit hoher Wahrscheinlichkeit, das besonders gesicherte Logsystem zu überwinden. Dadurch bietet dieses System ein Ziel, mit dem auch besonders fähige Angreifer und Angriffsmethoden analysiert werden können. Selbst wenn es einem Angreifer gelingen sollte, das Logsystem zu überwinden und die Logs unbrauchbar zu machen, hat das Intrusion Detection System mit seinem Logmechanismus sämtliche an das Logsystem gesendeten Nachrichten aufgezeichnet, daher sind die Daten nicht verloren.

In zunehmendem Maße werden zum Verbindungsaufbau zu den angegriffenen Systemen verschlüsselte Dienste verwendet. Sämtliche Ein- und Ausgaben einer Telnetsitzung können ohne weiteres mitgelesen werden. Bei einer SSH-Sitzung ist dies nicht möglich. Daher werden auf den Honeynetsystemen modifizierte Shells und Kernel eingesetzt, die die Ein- und Ausgaben unverschlüsselt an den Syslogdienst senden.

Für die erfolgreiche Durchführung von Gegenmaßnahmen ist es notwendig, über umfangreiches Wissen über Angriffsmöglichkeiten und -methoden zu verfügen. Die vom Honeynet Projekt eingerichteten Netze bestehen in der Regel aus Standardinstallationen der Betriebssysteme. Ein Honeynet kann jedoch auch ein produktives Firmennetz nachbilden. Dabei können die gleichen Betriebssysteme und Hardwarekomponenten eingesetzt werden, die auch im Firmennetz zum Einsatz kommen. Eine dem Honeynet vorgeschaltete offener Konfigurierte Firewall ermöglicht es Angreifern, nicht produktiv eingesetzte Klone der im Firmennetz hinter der Firewall platzierten Systeme anzugreifen. Dadurch können wertvolle Informationen über die Sicherheitslücken der Systeme hinter der schützenden Firewall gewonnen werden. Diese Informationen können genutzt werden, die Produktivsysteme durch gezielte aktive Gegenmaßnahmen für die erfolgreichen Angriffe zu schützen. Im Fall von Gegenangriffen können die ermittelten Vorgehensweisen in den “Werkzeugkasten” für Gegenangriffe aufgenommen werden.

## 2.3 Analyse eines Angriffs

Die im letzten Abschnitt vorgestellten Verfahren dienen hauptsächlich dazu, sich über die Methoden der Angreifer zu informieren die, eigenen Systeme zu schützen und Methoden kennenzulernen, die man für eigene Angriffe nutzen kann. Für erfolgreiche Gegenangriffe muss jedoch möglichst viel über die Systeme des Angreifers herausgefunden werden, damit geeignete Gegenangriffe ausgewählt werden können.

Deise Informationen lassen sich schon durch die Analyse des Angriffs durchführen, auf den reagiert werden soll. Das dafür verwendete Verfahren heisst passives Fingerprinting. Es basiert darauf, die durch das Intrusion Detection System gesammelten Pakete auf Besonderheiten zu untersuchen, die durch Kenntnis über die spezielle Implementierung der Protokollstacks in verschiedenen Betriebssystemen Rückschlüsse auf das verwendete System zulassen.

Passive Fingerprintverfahren lassen sich jedoch auch im Zielnetzwerk anwenden. Dies setzt voraus, dass im Rahmen des Gegenangriffs ein System erfolgreich übernommen werden kann.

Dann kann auf diesem System ein Sensor installiert werden, der passiv Hosts und deren Betriebssysteme innerhalb des Zielnetzwerks analysiert. Weiterhin lassen sich die in Kapitel 3 vorgestellten verteilten Verfahren auch zur Einrichtung verteilter Sensoren für passives Fingerprinting benutzen.

Passive Verfahren haben gegenüber aktiven Verfahren den Vorteil, dass sie nahezu nicht erkannt werden können. Es hängt jedoch von der Platzierung des Sensors oder der Sensoren ab, wie viele Informationen gesammelt werden können. Sensoren im eigenen Netzwerk können schon bei Scanversuchen oder Angriffen Informationen über das Betriebssystem des Angreifers liefern. Sollte eine Platzierung eines Sensors im oder in der Nähe des Netzwerks des Angreifers möglich sein, gibt es drei Stufen. Die wertvollsten Informationen liefert ein Sensor tief im internen Netzwerk des Angreifers oder auf dessen Rechner. Es dürfte jedoch schwerfallen, dieses Ziel im Rahmen eins automatisierten Gegenangriffs zu erreichen. Die zweite Stufe bildet ein Sensor innerhalb der demilitarisierten Zone (DMZ) des Zielnetzwerks. Hier können alle Rechner innerhalb deser Zone erkannt und möglicherweise das verwendete Betriebssystem identifiziert werden. Weiterhin können Systeme erkannt werden, die mit Rechnern in der DMZ kommunizieren. Aber selbst ein Sensor auf der dritten Stufe, ausserhalb, aber in der Nähe des Zielnetzwerks, liefert immerhin noch Informationen über die Rechner im Zielnetzwerk, die mit dem Internet kommunizieren dürfen.

## 2.4 Aktive Informationsbeschaffung

Die Informationsbeschaffung im eigenen Netz kann erste Aufschlüsse über die zum Angriff verwendeten Betriebssysteme liefern. Deutlich mehr und genauere Informationen liefert aber die aktive Analyse des angreifenden Systems oder des Netzwerks in dem sich das angreifende System befindet. Wenn der angreifende Rechner verhältnismässig ungeschützt direkt erreichbar ist, lassen sich Portscans oder Vulnerability Scanner einsetzen. Portscans geben Aufschluss über die Dienste die auf dem angreifenden System zur Verfügung stehen und mögliche Angriffspunkte darsellen, Vulnerability Scanner suchen darüber hinaus gezielt nach bekannten Sicherheitslücken. Ist das angreifende System nicht direkt erreichbar, können durch verschiedene gezielte Methoden des ICMP Scannens trotzdem für einen Angriff wertvolle Informationen gesammelt werden.

### 2.4.1 Portscans

Portscans versuchen Verbindungen zu verschiedenen Ports des Zielrechners aufzubauen. Ist eine Verbindung erfolgreich, so wird dieser Port für einen Dienst verwendet, ist sie nicht erfolgreich, wird dieser Port entweder durch eine Firewall gesperrt, oder er ist unbenutzt.

Werkzeuge wie `nmap` können ganze Subnetze auf einmal scannen und verschiedene Verfahren anwenden [Fyodor n.d.]. Die einfachste Form von Portscans versucht TCP oder UDP Verbindungen zu den Ports eines Rechners zu öffnen. Dabei findet der Verbindungsaufbau vollständig statt. Ein vollständiger Verbindungsaufbau wird in vielen Fällen von den Anwendungen, die den Service bereitstellen mitprotokolliert, daher bieten Portscanner verstecktere Verfahren an. Etwas unauffälliger als ein offener Portscan ist ein sogenannter halboffener Porstcan. Dafür wird ein TCP SYN Paket zum Verbindungsaufbau an den Zielrechner geschickt. Bei einem

offenen Port antwortet der Zielrechner mit SYN/ACK und bestätigt die Erreichbarkeit. Bei einem geschlossenen Port wird ein RST Paket gesendet, um die Verbindung wieder abzubauen. Wenn ein SYN/ACK empfangen wurde, schickt der Portscanner unmittelbar ein RST Paket an den Zielrechner, um den versuchten Verbindungsaufbar zu beenden. Ein Normaler Verbindungsaufbau würde nach SYN/ACK ein ACK Paket verlangen. Diese Form des versuchten und dann wieder abgebrochenen Verbindungsaufbaus wird oft nicht mitprotokolliert. Weitere Portscanverfahren bedienen sich der übrigen Flags in einem TCP Paket. Bei einem sogenannten FIN Scan wird nur ein Paket verschickt, das im Normalfall den Abbau einer bestehenden TCP/IP Verbindung einleitet. Solche Pakete sollen laut RFC 793 für nichtgenutzte Ports eine RST Nachricht auslösen, offene Ports sollen sie ignorieren. Daher ist es möglich durch dieses Verhalten offene Ports auszumachen.

### 2.4.2 Vulnerabilityscanner

Vulnerabilityscanner gehen weiter als Portscanner. Sie prüfen nicht nur, ob bestimmte Ports offen sind, sondern versuchen darüberhinaus noch herauszufinden, welche Software in welchen Versionen die Dienste hinter den Ports anbietet. Die meisten Werkzeuge dieser Art sind dafür gedacht, das eigene Netz nach Software abzusuchen, die nicht die nicht mit den neuesten fehlerbehebenden Updates versehen ist oder aufgrund eines Konfigurationsfehlers Sicherheitslücken offen lässt oder mehr Informationen als nötig liefert. Je aktueller die diesen Anwendungen zugrundeliegende Bibliothek mit bekannten Sicherheitslücken ist, desto wirkungsvoller lassen sich diese Werkzeuge einsetzen. Auch wenn viele dieser Werkzeuge dafür konzipiert sind, Sicherheitslücken im eigenen Netzwerk aufzuspüren, lassen sie sich auch dazu verwenden, potentielle Ziele für einen Gegenangriff automatisiert zu analysieren.

### 2.4.3 Informationsbeschaffung mit ICMP

Das ICMP Protokoll dient dazu, IP Netzwerke zu untersuchen und Fehler aufzuspüren. Es wurde erstmals im RFC 792 beschrieben und später durch die RFCs 1122, 1256, 1349 und 1812 erweitert. Die auf den ersten Blick harmlose Funktionalität kann jedoch sehr gut zur Informationsbeschaffung dienen. Häufig ist diese ICMP basierte Informationsbeschaffung der erste Schritt eines Angriffs. Einem automatisierten Gegenangriff sollte daher auch möglichst umfangreiche Informationsbeschaffung mit ICMP vorausgehen, um möglichst genaue Informationen über das anzugreifende System zu erhalten. In den nächsten Abschnitten werden einige Methoden vorgestellt und beispielhaft durchgeführt, mit denen sich Informationen über Rechner und Netzwerkinfrastruktur sammeln lassen. Zur Durchführung kommen zwei Rechner und ein Router zum Einsatz. Der die Scans ausführende Rechner hat die IP 192.168.10.110 und läuft mit dem Betriebssystem Linux (Kernel Version 2.4.19-gentoo-r7 #4 SMP). Dieser Kernel ist mit zusätzlichen Sicherheitsfunktionen ausgestattet, die allerdings unerheblich für die Tests sind, da ihre Ausrichtung nicht Schutz vor ICMP Scans, sondern ihre Durchführung ist. Der Zielrechner mit der IP 192.168.10.151 läuft entweder unter Windows 2000 Service Pack 2 oder unter SuSE Linux 7.3 (Kernel Version 2.4.10). Als dritte Hardwarekomponente kommt mit der IP 192.168.10.1 ein NETGEAR RO318 DSL/Cable Router zum Einsatz. Die speziellen ICMP Pakete werden mit `sing`[3] und `isic`[4] erzeugt, als Sniffer kommt `ethereal`[5]

[3] http://sourceforge.net/projects/sing
[4] http://www.packetfactory.net/Projects/ISIC/
[5] http://www.ethereal.org

zum Einsatz und für die Portscans wird `nmap`[6] eingesetzt.

Es wird bewusst darauf verzichtet, die hier vorgestellten, nicht standardkonformen Verfahren an Rechnern im Internet durchzuführen, da vor allem die im späteren Teil vorgestellten Scans von Transitsystemen auf dem Weg zum Ziel mit hoher Wahrscheinlichkeit als Vorstufe eines Angriffs identifiziert würden, selbst wenn für das Zielsystem eine Erlaubnis zur Durchführung existiert.

### 2.4.4 Rechnererkennung durch das ICMP Protokol

Das offensichtlichste Verfahren mögliche Ziele für einen Gegenangriff herauszufinden ist der ICMP Echo Request. Ein Echo Request wird mit dem `ping` Befehl erreicht.

```
ping www.richardstevens.de
PING www.richardstevens.de (217.172.179.61): 56 octets data
64 octets from 217.172.179.61: icmp_seq=0 ttl=246 time=152.5 ms
```

Diese Ausgabe zeigt, dass der Rechner mit der IP Adresse 217.172.179.61 erreichbar ist und die Antwort auf die Echo Anfrage 152.5 Millisekunden gebraucht hat.

```
    Source: risingsites.risingsites.homelinux.org (192.168.10.110)
    Destination: www.richardstevens.de (217.172.179.61)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x6c68 (correct)
    Identifier: 0xa179
    Sequence number: 00:00
    Data (56 bytes)

0000  c3 a1 2c 3d 0b 3c 04 00 08 09 0a 0b 0c 0d 0e 0f   ..,=.<..........
0010  10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f   ................
0020  20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f    !"#$%&'()*+,-./
0030  30 31 32 33 34 35 36 37                          01234567


    Source: www.richardstevens.de (217.172.179.61)
    Destination: risingsites.risingsites.homelinux.org (192.168.10.110)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x7468 (correct)
    Identifier: 0xa179
    Sequence number: 00:00
    Data (56 bytes)
```

[6] http://www.nmap.org

```
0000  c3 a1 2c 3d 0b 3c 04 00 08 09 0a 0b 0c 0d 0e 0f   ..,=.<..........
0010  10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f   ................
0020  20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f    !"#$%&'()*+,-./
0030  30 31 32 33 34 35 36 37                          01234567
```

Um die Hosterkennung zu Beschleunigen, gibt es auch Werkzeuge, die ganze IP Adressbereiche oder Subnetze parallel scannen können. Eine Serie von Echo Requests auf einen Adressbereich wird auch Ping Sweep genannt.

```
risingsites root # nmap -n -sP -PI 192.168.10.1-254

Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Host  (192.168.10.1) appears to be up.
Host  (192.168.10.110) appears to be up.
Host  (192.168.10.151) appears to be up.
Nmap run completed -- 254 IP addresses (3 hosts up) scanned in 9 seconds
```

Eine Analyse des lokalen Netzes des Autors zeigt 3 aktive Rechner.

Neben ICMP Echo Anfragen unterstützt das ICMP Protokoll auch weitere Pakettypen für verschiedene Aufgaben. Dazu zählen

- Zeitstempel (Timestamp Request): Erfragt die Zeit in Millisekunden, die seit Mitternacht (GMT) verstrichen ist. Zeitstempelanfragen können dazu benutzt werden, Latenzen im Netzwerk zu messen. Die Implementierung dieser Anfrage ist laut RFC 1122 optional.
- Informationsanfrage (Information Request): Diese Anfrage diente ursprünglich dazu, selbstkonfigurierende Netzwerke aufzubauen. Laut RFC 1122 soll diese Funktion nicht mehr implementiert werden.
- Adressmaskenanfrage (Adress Mask Request): Diese Anfrage dient dazu, Plattenlosen Rechnern ihre Subnetzmaske mitzuteilen. Eine Implementierung ist laut RFC 1122 optional.

Bis auf die Informationsanfrage, die nicht mehr implementiert sein sollte, lassen sich die anderen ICMP Anfragen genaus dazu verwenden, herauszufinden, ob eine IP Adresse in Benutzung ist oder nicht. Besonders interessant ist die Adressmaskenanfrage, wenn sie beantwortet wird, da sie Imformationen über die Konfiguration des Netzwerkes liefert. Adressmaskenanfragen sollen laut 1122 nur von Geräten beantwortet werden, die für das entsprechende Netzsegment zuständig sind. Meist wird diese Zuständigkeit von Routern oder Gateways übernommen. Laut RFC 1812 müssen Router diese Anfragen beantworten, sie sollen jedoch nicht in andere Netzwerke geroutet werden. Ein Test dieser Anfrage ist jedoch in der Testumgebung nicht möglich, da sich der RO318 in diesem Punkt nicht standardkonform verhält. Diese Einschränkung wird jedoch nicht immer genau befolgt, daher ist es möglich, Router auf dem Weg zum Zielnetzwerk ausfindig zu machen. Diese Information ist für Gegenangriffe wertvoll, da so speziell die Infrastruktur angegriffen werden kann.

### 2.4.5 Fortgeschrittenere ICMP Scanverfahren

Die im letzten Abschnitt vorgestellten Verfahren basieren alle auf Standardfunktionen innerhalb der Spezifikation des ICMP Protokolls. Durch bewusstes Erzeugen nicht standardkonformer Pakete, können den Zielssytemen weitere Informationen entlockt werden.

#### Modifikation der Felds für die IP Header Länge

Eine Möglichkeit ist, die zum Scannen verwendeten Pakete mit einem Fehlerhaften Wert für die IP Header Länge zu versehen. So angesprochene Netzwerkhardware reagiert mit hoher Wahrscheinlichkeit mit einer ICMP Prarameter Problem Nachricht.

```
    Source: 192.168.10.151 (192.168.10.151)
    Destination: risingsites.risingsites.homelinux.org (192.168.10.110)
Internet Control Message Protocol
    Type: 12 (Parameter problem)
    Code: 0 (IP header bad)
    Checksum: 0xf019 (correct)
    Pointer: 21
    Internet Protocol, Src Addr: risingsites.risingsites.homelinux.org
    (192.168.10.110), Dst Addr: 192.168.10.151 (192.168.10.151)
        Version: 4
        Header length: 56 bytes
        Differentiated Services Field: 0xb6
        (DSCP 0x2d: Unknown DSCP; ECN: 0x02)
            1011 01.. = Differentiated Services Codepoint: Unknown (0x2d)
            .... ..1. = ECN-Capable Transport (ECT): 1
            .... ...0 = ECN-CE: 0
        Total Length: 112
        Identification: 0x000f
        Flags: 0x00
            .0.. = Don't fragment: Not set
            ..0. = More fragments: Not set
        Fragment offset: 0
        Time to live: 105
        Protocol: st (0x05)
        Header checksum: 0xb874 (correct)
        Source: risingsites.risingsites.homelinux.org (192.168.10.110)
        Destination: 192.168.10.151 (192.168.10.151)
        Options: (36 bytes)
            Unknown (0x7e)
            (option length = 193 bytes says option goes past end of options)
    Data (8 bytes)

0000  13 f2 cb 3e 24 34 eb 80                              ...>$4..
```

Die obige Nachricht zeigt ein solches ICMP Fehlerpaket. Die relevanten Meldungen sind hervorgehoben. Pakete mit Fehlerhaften Angaben zur Länge des IP Headers werden jedoch auch

von einigen Routern verworfen. Dann schickt der Router die ICMP Fehlermeldung. Sollte der das Paket verwerfende Router schon Teil des Zielnetzwerks sein, lassen sich daraus Rückschlüsse auf den Hersteller und das Fabrikat erhalten. Wird keine Antwort auf das fehlerhafte Paket gesendet, deutet das auf eine Firewall hin, die das Paket aufgrund der eingestellten Regeln verworfen hat. Beide Fälle liefern für einen Angriff interessante Informationen, verhindern jedoch, dass das Paket den Zielrechner erreicht.

### Modifikation des Feldes für die Länge des Datagrams

Neben der Information zur Länge des IP Headers gibt es nur noch wenige Felder in einem IP Paket, die man verändern kann, ohne dass zum Erreichen des Ziels notwendige Informationen modifiziert werden. Es verbleiben noch das Feld für die IP Optionen und für die Gesamtlänge des transportierten Datagrams. Im Gegensatz zu den Feldern im IP Header ist das Feld für die Gesamtlänge des Datagrams für Router weniger interessant als die Länge des IP Headers. Das erhöht die Wahrscheinlichkeit, dass das Paket bis zum Zielnetzwerk vordringen kann, damit wäre das erste Hindernis überbrückt. Eine Firewall wird das Paket aber dennoch verwerfen, da es sich weiterhin um ein ICMP Paket handelt. Da die Paketfehler jedoch in den IP Header eingebracht werden, kann jedes auf IP aufsetzende Protokoll in das IP Paket eingebettet werden. Dadurch können beispielsweise TCP oder UDP Pakete auf wohlbekannte Ports, wie 80 (WWW) oder 21 (FTP) generiert werden. Viele Firewalls prüfen nur den TCP oder UDP Header, um das Paket einem bestimmten Dienst zuzuordnen. Wenn das Regelwerk der Firewall es zulässt, wird das Paket an den Zielrechner weitergeleitet. Der Zielrechner versucht im Anschluss, das gesamte TCP Paket aus dem IP Paket zu extrahieren und erkennt den Fehler. Laut RFC wird dann ein ICMP Parameter Problem Paket als Antwort generiert, um das Paket erneut anzufordern.

Diese Antwort liefert gleich zwei für einen Angriff relevante Informationen. Zum einen Bestätigt sie die Existenz des Rechners, zum anderen liefert sie erste Informationen über die Regeln der Firewall. Durch Verwendung verschiedenartiger Kombinationen aus Protokoll und Portnummern in Verbindung mit Scans auf einen, mehrere oder alle Rechner eines Subnets kann das Regelwerk der Firewall und die Erreichbarkeit verschiedener Rechner im geschützen Zielnetz ermittelt werden.
Dieses Verfahren stellt daher ein sehr leistungsfähiges Mittel zur Informationsbeschaffung dar. Allerdings werden spätestens durch diesen Scan, hinreichend viele Scanpakete durch die Firewall blockiert und ICMP Fehlerantworten generiert, dass sowohl Warnmechanismen der Firewall als auch vorhandene Intrusion Detection Systeme auslösen sollten. Daher eignet sich dieses Verfahren nur, wenn die Vorboten des kommenden Angriffs nicht verschleiert werden sollen. Zusätzlich könnten dynamische Firewallregeln dazu führen, dass alle Pakete von der scannenden Adresse verworfen werden. Dann ist ein umfassender Scan des Netzwerks hinter der Firewall nicht merh möglich. Einen Ausweg könnten die in Kapitel (REFERENZ) beschriebenen agentenbasierten Systeme liefern, da sie jeweils einzelne Scans von verscheidenen IP Adressen ausführen könnten. Ein derart verteilter Scan hat bedeutend höhere Chancen, keine Blockade der Firewall zu verursachen.

### Auswertung der Fehlemeldung für nichtunterstützte Protokolle

Im Rahmen eines automatisierten Gegenagriffs ist damit zu rechnen, dass der Angriff nahezu unmittelbar nach dem Scan erfolgt. Daher ist ein erkannter Scan unproblematisch. Ein weiteres sehr offensichtliches Verfahren eignet sich, Router oder Firwalls von Rechnern zu unterscheiden. Viele Betriebssysteme reagieren bei unbekannten Werten im Protokoll Feld des IP Pakets mit einer ICMP Destination unreachable Nachricht, die die Zusatzinformation Protocol unreachable enthält. Firewalls und Router senden diese Information mit hoher Wahrscheinlichkeit nicht, da im Fall von Routern alle IP basierten Protokolle unterstützt werden und Firewalls diese Pakete aufgrund des Regelwerks verwerfen. Diese Methode ist jedoch keine Sichere Bestätigung für die Existenz einer Firewall, da beispielsweise AIX, HP-UX und Digital UNIX auch keine ICMP Destination unreachable Pakete verschicken. Sollte durch aktives oder passives Fingerprinting (Siehe Seite 24 und 15) das Betriebssystem eines Rechners bekannt sein und zu der Gruppe gehören, die diese Fehlermeldungen sendet, kann dadurch eine Firewall auf dem Weg zum Ziel erkannt werden. Bei `nmap` nennt sich dieser Scan Protocol Scan. Die Untersuchung des Windows 2000 Systems liefert folgendes Ergebnis:

```
nmap -vv -sO 192.168.10.151

Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Host  (192.168.10.151) appears to be up ... good.
Initiating IPProto Scan against  (192.168.10.151)
The IPProto Scan took 3 seconds to scan 255 ports.
Adding open port 6/udp
Adding open port 1/udp
Adding open port 2/udp
Adding open port 17/udp
Interesting protocols on  (192.168.10.151):
(The 251 protocols scanned but not shown below are in state: closed)
Protocol   State       Name
1          open        icmp
2          open        igmp
6          open        tcp
17         open        udp

Nmap run completed -- 1 IP address (1 host up) scanned in 3 seconds
```

Dabei wurde für jedes nicht unterstützte Protokoll eine entsprechende ICMP Destination unreachable Nachricht generiert.

```
    Source: 192.168.10.151 (192.168.10.151)
    Destination: risingsites.risingsites.homelinux.org (192.168.10.110)
Internet Control Message Protocol
    Type: 3 (Destination unreachable)
    Code: 2 (Protocol unreachable)
    Checksum: 0xfcfd (correct)
    Internet Protocol, Src Addr:
    risingsites.risingsites.homelinux.org (192.168.10.110),
```

```
    Dst Addr: 192.168.10.151 (192.168.10.151)
        Version: 4
        Header length: 20 bytes
        Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
            0000 00.. = Differentiated Services Codepoint: Default (0x00)
            .... ..0. = ECN-Capable Transport (ECT): 0
            .... ...0 = ECN-CE: 0
        Total Length: 20
        Identification: 0x9d81
        Flags: 0x00
            .0.. = Don't fragment: Not set
            ..0. = More fragments: Not set
        Fragment offset: 0
        Time to live: 57
        Protocol: Unknown (0x41)
        Header checksum: 0x4dd2 (correct)
        Source: risingsites.risingsites.homelinux.org (192.168.10.110)
        Destination: 192.168.10.151 (192.168.10.151)
```

Der Scan des Routers ergibt hingegen, dass alle 256 möglichen Protokollwerte unterstützt werden.

```
nmap -vv -sO 192.168.10.1

Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Host  (192.168.10.1) appears to be up ... good.
Initiating IPProto Scan against  (192.168.10.1)
The IPProto Scan took 16 seconds to scan 255 ports.
Adding open port 244/udp
Adding open port 162/udp
(...)
Adding open port 247/udp
Adding open port 26/udp
Interesting protocols on  (192.168.10.1):
Protocol   State       Name
1          open        icmp
2          open        igmp
3          open        ggp
4          open        ip
5          open        st
6          open        tcp
(...)
253        open        unknown
254        open        unknown

Nmap run completed -- 1 IP address (1 host up) scanned in 16 seconds
```

Die Darstellung ist gekürzt, alle 256 möglichen Werte sind mit dem Status open gekennzeichnet. Dabei treten nur einige ICMP Malformed Packet Meldungen für die wirklich unterstützen Protokolle auf. Alle weiteren Pakete werden von der integrierten Firewall verworfen. Ein reiner Router würde sich völlig transparent verhalten und keine ICMP Fehlermeldungen generieren.

### Weitere Verfahren

Arkin [S. 34-77] gibt in seinem Papier noch weitere Möglichkeiten an, durch ICMP Pakete und Fehlermeldungen Informationen über das Zielnetzwerk zu erhalten. Dabei werden verschiedene Eigenschaften augenutzt, die durch die Heterogenität des Internets zustande kommen. Aufgrund der Zusammenschaltung unterschiedlicher Netzwerktypen und Kapselung verschiedenartigster Protokolle ineinander ergeben sich beispielsweise unterschiedliche Maximalgrößen für Pakete. Ein für ein bestimmtes Netzsegment zu großes Paket wird am Netzübergang zu diesem Netzsegment verworfen und der entsprechende Router generiert eine Meldung, die das Paket in fragmentierter Form, also aufgeteilt auf mehrere Pakete anfordert. Diese Fragmentierten Pakete werden am Zielhost wieder zusammengesetzt, damit der Inhalt entnommen werden kann. Erhält ein Rechner nicht alle Fragmente eines aufgeteilten Pakets, sendet er nach einem definierten Zeitintervall eine ICMP Fragment Reassembly Time Exceeded Fehlermeldung. Mit dieser Meldung fordert er den erneuten Versand des Pakets beim Sender an. Dieses Verhalten ist völlig normal und sorgt unter normalen Umständen dafür, dass alle Pakete beim Empfänger ankommen. Bewusst unvollständig verschickte fragmentierte Pakete erzwingen dieses Verhalten. Die ICMP Antwort liefert wieder Informationen über das Zielsystem. Das Ausbleiben dieser Nachrichten lässt wieder Rückschlüsse auf eine mögliche Firewall zu.

In fehlerhaft konfigurierten Zielnetzen können durch Fragmentierung Rückschlüsse auf die innere Topologie gezogen werden. Wenn ein zu großes Paket einen Router innerhalb des Zielnetzes erreicht, weil der Router zum Internet diese für das interne Netz zu große Paket durchlässt, generiert der interne Router eine ICMP Fragmentation needed Fehlermeldung. In einem fehlerfrei konfigurierten Netz wäre diese Meldung durch den Router zum Internet generiert worden. Durch diesen Fehler lassen sich Router innerhalb des Zielnetzwerks ausfindig machen.

Eine Andere möglichkeit sind UDP basierte Scans. UDP Pakete, die an nicht unterstützten Ports des Zielsystems ankommen, generieren eine ICMP Port Unreachable Fehlermeldung. Das Ausbleiben dieser Pakete deutet wieder auf eine Firewall hin.

In bestimmten Fällen sperrt die Firewall ICMP Destination Unreachable Fehlermeldungen aus dem internen Netz nicht. Gezielte Anfragen an Router im internen Netz können so genutzt werden um Infomrationen über benutzte und unbenutzte IP Adressen zu erhalten.

### Aktive Fingerprints mit ICMP

Die bisher beschriebenen aktiven Verfahren dienen dazu, mögliche Ziele für den Gegenangriff im Netzwerk des Angreifers ausfindig zu machen. Damit die Gegenangriffe erfolgreich durchgeführt werden können, müssen Schwächen und Lücken in der Software der Zielrechner

ausfindig gemacht werden. Mit aktiven Fingerprint verfahren können Informationen zum verwendeten Betriebssystem gesammelt werden, auch wenn der Zielrechner sonst gut geschützt ist. Aktives Fingerprinting macht sich wie passives Fingerprinting (siehe Abschnitt 2.3) Unterschiede in der Implementierung der Netzwerkprotokolle zu Nutze. Im Gegensatz zu passiven Fingerprintverfahren werden für aktive Fingerprints gezielt Pakete an die zu untersuchenden Rechner geschickt, um aufgrund der erhaltenen Antworten Rückschlüsse ziehen zu können. Aktives Fingerprinting lässt sich nicht nur über ICMP durchführen. ICMP hat jedoch den Vorteil, dass die Menge der benötigten Pakete im Vergleich zu anderen Protokollen sehr gering ist.

Die einzelnen Testferfahren lassen oft keine genaue Indentifikation des Betriebssystems zu. Meist liefern sie nur die Information, dass das Zielsystem zu einer Gruppe von Betriebssystemen gehört, da verschiedene Betriebssysteme auf den test gleich reagieren. Die Gruppierungen sind jedoch von Test zu Test unterschiedlich, sodass nach einer Reihe von Tests oft ein oder einige wenige Betriebssystemkandidaten übrig bleiben. Eine genaue Analyse der Eigenschaften einzelner Betriebssysteme und ihre Klassifizierung würde den Rahmen dieses Papiers deutlich sprengen, daher werden hier nur einige mögliche Methoden vorgestellt.

Schon durch die Verwendung regulärer ICMP Pakete, lassen sich Betriebssysteme kategorisieren. Besonders aufschlussreich ist das Verhalten bei ICMP Nachrichten an die Broadcastadresse des Netzwerks. Mit diesem Verfahren lassen sich Linux Systeme mit Kernel 2.2.x, Sun Solaris und HPUX 10.20 erkennen. Alle genannten Antworten auf eine ICMP Timestamp Anfrage auf die Broadcastadresse. Nur HPUX antwortet auf eine ICMP Information Anfrage. Von den beiden auf diese Anfrage nicht antwortenden Betriebsystemen antwortet Solaris auf die ICMP Adress Mask Anfrage, Linux nicht. Alle anderen heute gängigen Betriebssysteme antworten auf diese Art von ICMP Paketen nicht, daher lassen sich auch keine weiteren Rückschlüsse ziehen.

IP Pakete besitzen eine Reihe von Feldern für Zähler und Statusinformationen. Das IP-ID Feld dient dazu, einzelne IP Pakete zu identifizieren. Die meisten Betriebssysteme zählen die IP-ID für jedes ausgehende Paket der Reihe nach durch. Ältere Windowsbetriebssysteme bis ausschließlich Windows 2000 erhöhen den Wert um 256 je Paket. Der Linux Kernel der Version 2.4.X verhält sich bei ICMP Echo Anfragen sehr auffällig. Diese Art von Anfragen wird immer mit einer IP-ID von 0 gestellt und beantwortet.

Eine weitere Möglichkeit zur Gruppierung bietet das Verhalten im Bezug auf die Fragmentierung von IP-Paketen. Ein IP Paket darf entweder nicht fragmentiert werden, dann ist das Don't Fragment Bit gesetzt, oder eine Fragmentierung ist erlaubt. Ist eine Fragmentierung erlaubt, kennzeichnet ein weiteres Bit, ob es sich um das letzte Fragment handelt, oder ob weitere Fragmente erwartet werden. Linux Kernel der Version 2.4.x, HPUX 10.30 und 11.0x, sowie AIX 4.3.x setzen bei den verschiedenen ICMP Anfragen das Don't Fragment bit. In Verbindung mit dem Verhalten bei der IP-ID lässt sich Linux 2.4.x von den anderen Betriebssystemen unterscheiden.

Arkin [S. 78-102] beschreibt in seinem Papier noch weitere Möglichkeiten, aus bestimmten Reaktionen auf gültige ICMP Anfragen weitere Informationen zu erlangen. Neben gültigen ICMP Paketen können wie auch bei der Hosterkennung (siehe Seite 20) ungültige oder gezielt

hergestellte ICMP Anfragen und ihre Reaktionen zur Betriebssystemerkennung verwendet werden. Dazu zählen besondere Wert im TOS (Type of Service) Feld, der IP Pakete oder ICMP Fehlermeldungen.

## 2.5 Fazit

Die in diesem Kapitel vorgestellten Verfahren zur Vorbereitung eines Gegenangriffs können ein detailliertes Bild des Zielneztwerks liefern. Sie sind jedoch bekannt und publiziert. Daher gibt es auch Wege, sich vor Scans dieser Art zu schützen. Viele der zur Host- und Betriebssystemerkennung verwendeten ICMP Pakete können an der Firewall gefiltert werden, ohne dass gravierende Probleme in der Netzkommunikation auftreten. Wenn diese Filterung konsequent betrieben wird, lassen die aktiven Verfahren nicht viel mehr, als den Schluss zu, dass das Zielnetzwerk durch eine Firewall geschützt ist. Es ist empfehlenswert, das eigene Netzwerk durch strikte Filterung dieser Pakete zu schützen und das Intrusion System mit Filtern für diese Art von Scans auszustatten, um ein Frühwarnsystem für mögliche Angriffe zu realisieren. Auf diese Weise geschütze Zielnetzwerke erschweren jedoch auch einen automtisch erzeugten Gegenangriff, da zu einer Auswahl der geeigneten Angriffe notwendige Informationen fehlen. Es ist sehr wahrscheinlich, dass sich fähigere Angreifer, die fortgeschrittenere Scanmethoden verwenden auch vor diesen Schützen. Daher verbleiben in vielen Fällen nur die Informationen, die sich aus passiven Verfahren sammeln lassen. Für einen erfolgreichen Gegenangriff ist daher möglichst vollständiges Logging des Netzverkehrs nötig.

# 3 Verteilte Sicherheitssysteme

Maik Dobryn, Sven Ludwig

In diesem Kapitel gehen wir auf aktuelle Entwicklungen von verteilten Sicherheitssystemen ein, bei denen Angriffe in Zusammenarbeit von verschiedenen Netzwerkkomponenten bekämpft werden sollen. Zur Zeit ist leider noch keines dieser Systeme für uns verfügbar. Im Internet gibt es jedoch bereits Informationen dazu, ausreichend, um hier zwei Ansätze mit ihren Lösungen und offenen Problemen darstellen zu können.

## 3.1 Allgemeine Aspekte

Maik Dobryn, Sven Ludwig

An dieser Stelle werden einige allgemeine Aspekte aufgeführt werden, um die Thematik des gesamten Kapitels etwas zu umreißen.

### 3.1.1 Grundlegende Eigenschaften des Internets

Das Internet ist in seiner Entstehungsphase als ein offenes Netzwerk mit möglichst hoher Ausfallsicherheit und Flexibilität entwickelt worden. Es wurde zunächst im militärischen Bereich eingesetzt, wo das Hauptaugenmerk auf der Verfügbarkeit der logischen Verbindungen zwischen Computern gelegt wurde. Etwas später wurde es in der Wissenschaft für den offenen Austausch von Informationen genutzt. In den 90er Jahren wurden mit der Ausdehnung des Internets verschiedene Sicherheitsaspekte relevant. Aus den grundlegenden Eigenschaften des Internets resultieren heutzutage Probleme in Bezug auf die Sicherheit. Einige Beispiele sollen im folgenden kurz genannt werden. Im Internet kann der Inhalt von Paketen mitgelesen und verändert werden. Um Integrität und Vertraulichkeit zu gewährleisten, müssen Maßnahmen auf der Applikationsschicht getroffen werden. Die beiden Protokolle IP und TCP, die den mittleren Schichten des OSI Modells zuzuordnen sind, bieten keinerlei Schutz gegen Man-In-The-Middle Angriffe. Außerdem kann jeder Rechner zunächst beliebige Pakete an beliebige Empfänger senden, wodurch die heutzutage sehr gefürchteten Distributed Denial of Service (DDoS) Angriffe ermöglicht werden [Costa-Requena 2001]. Deren Abwehr wird wiederum durch das flexible Routing der IP-Pakete erschwert.

### 3.1.2 Rückverfolgung

Das optimale Ergebnis der Rückverfolgung eines Angriffs wäre die genaue Bestimmung des Rechners, von dem aus der Angriff gestartet wurde. Meist verlaufen Angriffe jedoch über mehrere so genannte Hops. Dies sind Stationen auf einer Verbindungsstrecke zwischen zwei

Hosts, auf denen eine logische Verbindung endet und eine neue logische Verbindung beginnt. Es besteht also keine durchgehende logische Verbindung zwischen den beiden Hosts. Auf einer Station besteht die Möglichkeit, dass Pakete der Applikationsschicht einfach weitergeleitet werden, oder dass die Station selbst dazu genutzt wird, um Daten zu senden. Jedenfalls ist dem Empfänger der Daten die Adresse des Senders nicht bekannt, sofern sie nicht auf der Applikationsschicht mit übertragen wird. Um das System nun die Adresse des Systems eines Angreifers auszumachen, müssen Hops erkannt und von den Stationen selbst aus muss die Strecke nachvollzogen werden. Dies ist jedoch nur möglich, falls man auf allen Stationen hin zu dem System des Angreifers genügend Berechtigungen hat, um an die benötigten Informationen zu gelangen. Fehlen Berechtigungen, so kann man nur mit Hilfe des Besitzers des betreffenden Rechners fortschreiten oder sich die fehlenden Berechtigungen durch einen Angriff auf diesen Rechner aneignen. Aus diesem Grunde ist eine Rückverfolgung zunächst an den äußeren Grenzen des eigenen Netzwerks beendet.

### 3.1.3 Auswirkungen von Gegenmaßnahmen

Vor dem Ergreifen einer Gegenmaßnahme muss zunächst abgewogen werden, ob die Auswirkungen auf das eigene System mehr Schaden bedeuten würden als die Auswirkungen des Angriffs, gegen den sich die Gegenmaßnahme richten würde. Wenn dies der Fall ist, sollte die Gegenmaßnahme nicht ergriffen werden. Es könnte dann nach einer alternativen Gegenmaßnahme gesucht werden oder für den Moment ganz auf Gegenmaßnahmen verzichtet werden. Ein weiterer Aspekt ist, dass für den Einsatz von netzwerkbasierten Gegenmaßnahmen in verteiten Sicherheitssystemen eine optimale Position im Netzwerk bestimmt werden, an der oder von der aus die Gegenmaßnahme eingesetzt wird. An dieser Position sollte die Gegenmaßnahme am besten gegen den Angriff wirken und gleichzeitig am wenigsten negative Auswirkungen auf den normalen Verkehr im Netzwerk haben.

### 3.1.4 Risiken der Automatisierung

Die Automatisierung von Sicherheitssystemen in der Informationstechnik bietet zunächst einige Vorteile. Der personelle Aufwand, der für die Überwachung der Systeme notwendig ist, kann verringert werden. Zudem können automatische Reaktionen schneller und zuverlässiger sein als Menschliche. Für Aufgaben der Angriffserkennung und Alarmierung ist eine Automatisierung zunächst unproblematisch, mal abgesehen von Fehlalarmen. Es wird jedoch kritisch, sobald Entscheidungen über Gegenmaßnahmen getroffen werde sollen. Zum einen müssen die Auswirkungen der Gegenmaßnahmen mit denen der Angriffe verglichen werden. Zum anderen kann ein Angreifer automatische Gegenmaßnahme ausnutzen, um sein Ziel zu erreichen. Dies bezieht sich insbesondere auf Denial of Service Angriffe. Ein Angreifer kann zum Beispiel versuchen, mit erzeugten Angriffen eine Gegenmaßnahme auszulösen, die einen Port auf dem Zielsystem sperrt. Der Dienst, der unter diesem Port angeboten wird, ist dann zumindest vorübergehend nicht mehr verfügbar.

## 3.2 Cooperative Intrusion Traceback and Response Architecture

Sven Ludwig

Die Cooperative Intrusion Traceback and Response Architecture wird unter der Schirmherrschaft der Defense Advanced Research Projects Agency (DARPA) von Mitgliedern der Boeing's Phantom Works, der Network Associates' NAI Labs [NetworkAssociates n.d.*b*] und des University of California Davis' Computer Security Lab entwickelt [Schnackenberg, Holliday, Smith, Djahandari & Sterne 2001]. Die Architektur setzt auf dem zuvor von den selben Institutionen entwickelten Intruder Detection and Isolation Protocol (IDIP) auf [Schnackenberg, Djahandari & Sterne 2000]. Das Projekt verfolgt Ziele, die man bisher nur in Teilen oder nur durch stark herstellerabhängige Lösungen erreichen konnte. Angriffe sollen im Zusammenwirken von verschiedenen Netzwerkkomponenten und Hosts verfolgt werden. An zentraler Stelle fließen dabei gewonnene Informationen zusammen (information pooling). Aufgrund ihrer Auswertung können Gegenmaßnahmen gezielt koordiniert werden. Die Integration weiterer Komponenten in ein CITRA System soll möglichst einfach und kostengünstig gehalten werden. Zudem möchte man einen hohen Grad an Automatisierung ermöglichen, um den personellen Aufwand für die Überwachung eines Netzwerks zu verringern. Die Rolle des Administrators soll sich ändern. Er soll nicht mehr manuell das Netzwerk überwachen und Gegenmaßnahmen ergreifen, falls diese von Nöten sind, sondern er soll das verteilte Sicherheitssystem konfigurieren und dessen Funktion überwachen. Das Grundprinzip der CITRA ist, existierende Komponenten wie Intrusion Detection Systeme, Firewalls oder Router über eine gemeinsame Infrastruktur zu einem verteilten Sicherheitssystem zusammenzuschließen.

### 3.2.1 Struktur

In einem CITRA Netzwerk gibt es CITRA Komponenten und normale Komponenten. Letztere werden von den CITRA Komponenten verteidigt und nehmen selbst nicht an der organisierten Verteidigung teil. Das gesamte Netzwerk in „Communities“ unterteilt, die wiederum in „Neighborhoods“ unterteilt werden. Ein Neighborhood besteht aus mehreren CITRA Komponenten, zwischen denen keine weiteren CITRA Komponenten stehen. Das bedeutet, dass bei einer Bus Topologie alle an den Bus angeschlossenen Komponenten zu einem Neighborhood gehören. An den Grenzen des Busses befindliche Router oder andere weiter vermittelnde Komponenten, so genannte „Boundary Controller“, sind vorzugsweise immer CITRA Komponenten. Diese können dabei gleichzeitig zu mehreren Neighborhoods gehören. Eine Community ist eine administrative Domäne, in der es eine zentrale Verwaltungsstelle gibt. Zu dieser Stelle wird ein so genannter „Discovery Coordinator“ (kurz DC) hinzugefügt. Abbildung 3.1 zeigt die hier beschriebene Struktur.

### 3.2.2 Intruder Detection and Isolation Protocol (IDIP)

Alle CITRA Komponenten kommunizieren untereinander über das Intruder Detection und Isolation Protocol (IDIP), das ein fester Bestandteil der CITRA ist. Es definiert, welche Informationen wie ausgetauscht werden können. Das Protokoll arbeitet auf der Applikationsschicht des OSI-Modells. Es ist neben dem HTTP Protokoll angesiedelt und verwendet auf der darunter liegenden Transportschicht das User Datagramm Protocol (UDP). IDIP ist selbst nochmal ist in zwei Schichten unterteilt:

- Message Layer
- Application Layer

Der Message Layer liegt unter dem Application Layer. Er wird auch als IDIP Backplane bezeichnet. Er bildet eine Infrastruktur zum Austausch von Nachrichten nach dem Publisher/Subscriber Modell (Herausgeber/Abonnent), das für die Datenverarbeitung im Bereich des Message Queuing neben dem Point-to-Point (P2P) Modell bekannt geworden ist. Über die API des Message Layers *veröffentlichen* die CITRA Komponenten ihre Nachrichten, *abonnieren* bestimmte Typen von Nachrichten und *erhalten* Nachrichten der abonnierten Typen. Wie letztendlich die Verteilung der Nachrichten implementiert ist, läßt die API des Message Layers offen. Die Implementierung bleibt also austauschbar. In der aktuellen Version sieht man die Verwendung von IP-Multicast innerhalb von CITRA Neighborhoods vor. Dadurch kann eine an mehrere Empfänger adressierte Nachricht mit einem einzigen Sendevorgang verschickt werden.

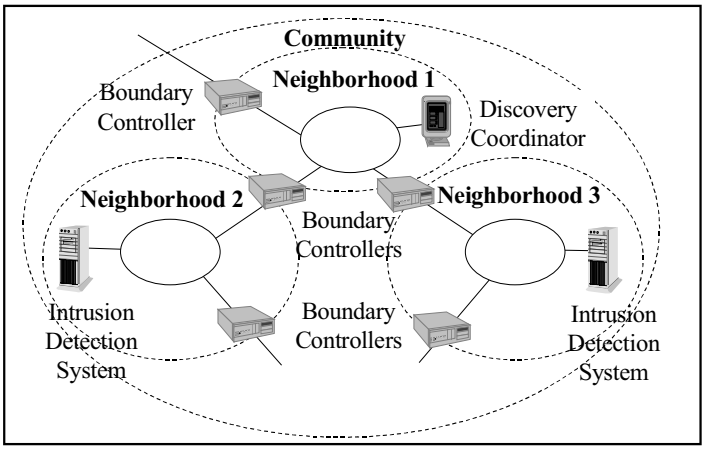


Abbildung 3.1: CITRA Community
(Quelle: [Schnackenberg et al. 2000])

An den Message Layer werden weiterhin einige Anforderungen gestellt. Die Auslieferung von Nachrichten muss zuverlässig sein. Da das verwendete Transport Protokoll UDP nicht prüft, ob die versendeten Daten beim Empfänger angekommen sind, ist jeweils eine Empfangsbestätigung pro Nachricht und Empfänger auf der Applikationsschicht notwendig. Falls der Sender nicht von allen vorgesehenen Empfängern eine Bestätigung erhält, muss er die Nachricht erneut verschicken oder akzeptieren, dass die betroffenen Empfänger sie wahrscheinlich nicht erhalten haben. Die genaue Vorgehensweise in einem solchen Fall, der Fallback, ist für das IDIP nicht beschrieben. Eine weitere Anforderung an den Message Layer ist, dass Doppelt empfange Nachrichten erkannt und verworfen werden. Dazu bekommt jede Nachricht eine eindeutige Identifikationsnummer, so dass doppelt empfangene Nachrichten durch einen einfachen Vergleich der Nummern erkannt werden können.
Der Application Layer wird durch einen CITRA Agenten realisiert, der Schnittstellen für den Datenaustausch mit der eigentlichen Komponente (Firewall, IDS, etc.) bietet. Auf dem

Discovery Coordinator kommt dazu noch die Discovery Coordinator Applikation, die aus einer Implementierung der Discovery Coordinator API und darüber integrierten Programmen zur Auswertung der gesammelten Informationen und Koordinierung von Gegenmaßnahmen besteht. Die Applikationen auf dem Application Layer abonnieren beim Message Layer die benötigten Typen von Nachrichten. Sie erhalten dann alle ankommenden Nachrichten von diesen Typen, inklusive der lokal auf der Komponente erzeugten Nachrichten. Dadurch kann die Kommunikation zwischen den auf dem Discovery Coordinator angesiedelten Programmen auch über den Message Layer abgewickelt werden. Bisher stellt IDIP die folgenden Typen von Nachrichten zur Verfügung:

- trace - eine Anweisung zur Verfolgung mit bisher ermittelten Informationen
- report - ein Bericht über Ereignisse
- directive - eine Direktive vom Discovery Coordinator
  - do - eine Maßnahme ergreifen
  - undo - eine Maßnahme rückgängig machen

Welche Informationen die verschiedenen Typen von Nachrichten enthalten können und wann sie zum Einsatz kommen wird in späteren Abschnitten verdeutlicht. Zunächst werden weitere Anforderungen an das Protokoll und deren Umsetzung beschrieben.
Das Protokoll muss ein Zeitmanagement zur Verfügung stellen, dass eine zeitliche Synchronisation zwischen den CITRA Komponenten erlaubt. Dazu werden die Zeitunterschiede zwischen den internen Uhren der CITRA Komponenten innerhalb eines Neighborhoods regelmäßig gemessen und für die zeitliche Einordnung von Informationen aus Nachrichten verwendet. Über die Messung der Zeitunterschiede ist nur bekannt, dass sie aufgrund von Verzögerungen bei der Verteilung von Nachrichten berechnet werden. Im einfachen Sinne kann der Zeitunterschied zwischen zwei Komponenten als der aktuelle Zeitpunkt des Empfangs einer Nachricht minus dem Zeitstempel der Nachricht minus dem Intervall zwischen Versendung und Empfang der Nachricht angesehen werden. Allerdings ergeben sich dabei einige Schwierigkeiten, da das Intervall zwischen Versendung und Empfang kein konstanter Wert ist.
Eine weitere Anforderung an das Protokoll ist die Bekanntmachung der Komponenten untereinander und die Verwaltung ihrer Statusinformationen. Für CITRA Komponenten ist eine automatische Erkennung ihrer Nachbarn geplant. Erste Implementierungen verwenden jedoch zunächst Listen, die vom Discovery Coordinator ausgegeben wurden. Der Status einer Komponente sagt neben der Anwesenheit etwas über ihre Vertrauenswürdigkeit aus. Wurden auf einer Komponente Anomalien entdeckt, so kann ihr Status innerhalb der CITRA als weniger bis nicht vertrauenswürdig eingestuft werden. Die anderen Komponenten verhalten sich in der Kommunikation mit einer eventuell kompromittierten Komponente zurückhaltend oder schließen sie ganz aus dem System aus. Diese Vorgehensweise soll verhindern, dass ein Angreifer mit der Übernahme einer CITRA Komponente das System täuschen oder missbrauchen kann.
Weiterhin bestehen Gefahren, dass ein Angreifer versucht, die ausgetauschten Informationen zu seinem eigenen Vorteil zu nutzen oder sie zu verändern, um die CITRA Komponenten zu beeinflussen. Darüber hinaus könnte ein Angreifer versuchen, eine normale Komponente als CITRA Komponente zu tarnen und von dort aus sein Unwesen zu treiben. Um diesen

Gefahren entgegen zukommen, muss die Kommunikation zwischen den CITRA Komponenten vertrauenswürdig sein. Allerdings muss sie schnell abgewickelt werden, damit Reaktionen auf Angriffe rechtzeitig zum Tragen kommen können. Daraus folgt, dass die Verschlüsselung möglichst schnell ablaufen sollen und die Auswirkung der Verschlüsselung auf die Größe der IDIP Nachrichten möglichst gering sein soll. Für das IDIP Protokoll wurde ein eigenes portables Verfahren zum Schutz der IDIP Nachrichten entwickelt, da bereits existierende Verfahren den Anforderungen nicht gerecht werden konnten. Als Basis für die Implementierung des Verfahrens diente OpenSSL [*OpenSSL Project* n.d.]OpenSSL. OpenSSL stellt eine portable Open Source Implementierung der Protokolle SSL (Version 2 und 3) und TSL sowie eine Software Bibliothek für kryptographische Zwecke zur Verfügung. Die für das IDIP Protokoll verwendeten Module, importierte sowie selbst entwickelte Module, wurde insgesamt möglichst portabel gehalten, um die Verwendung der CITRA in heterogenen Netzwerken zu ermöglichen.

### 3.2.3 Konfiguration und INFOCON

Beim Design der Architektur legt man sehr großen Wert auf die Konfiguration. Es wird eine hohe Flexibilität geboten, um eine ausgewogene, auf die jeweiligen Anforderungen zugeschnittene Wirkungsweise des Systems zu ermöglichen. Kompromisse bezüglich der Abwehr eines Angriffs und den damit verbundenen Auswirkungen auf die eigenen Systeme können umgesetzt werden. Jede CITRA Komponente erhält ihre eigene CITRA-spezifische Konfiguration mit Parametern und Regeln (engl. Policies), die ihre Verhaltensweise bestimmen.
Die CITRA unterstützt das Prinzip eines übergeordneten Sicherheitslevels in Anlehnung an den vom Verteidigungsministerium der Vereinigten Staaten eingeführten Begriff „Information Operation Condition“ (INFOCON). Es ist ein Wert, der die aktuelle Situation im elektronischen Netzwerk und die Wahrscheinlichkeit eines Angriffs repräsentiert. Nutzt man das Feature in CITRA, so muss auf jeder CITRA Komponente jedem möglichen INFOCON Level eine Konfiguration zugeordnet werden. Die dem aktuellen INFOCON Level zugeordnete Konfiguration wird verwendet. Ändert sich der Level, so schickt der Discovery Coordinator der Community eine Nachricht an alle CITRA Komponenten, die daraufhin ihre entsprechende Konfiguration laden. Genaueres zur Konfiguration der Komponenten findet sich in den folgenden Abschnitten.

### 3.2.4 Integration von Komponenten

Jeder Komponente, die am CITRA System teilnehmen soll, wird ein CITRA Agent zur Seite gestellt, der für die Anbindung an das System zuständig ist. Der Agent, ein portables Stück Software, übernimmt für die Komponente den kompletten Teil der IDIP Applikationsschicht, d.h. er verarbeitet alle ankommenden Nachrichten, generiert und versendet ausgehende Nachrichten und hält die CITRA spezifische Konfiguration. Je nach Einsatzzweck erhält der Agent Informationen von der Komponente oder er ist in der Lage, ihr Anweisungen zu geben. Es kann auch beides zutreffen. Mit Agent ist hier nicht die Art von Agent gemeint, die später in Kapitel 3.3 beschrieben wird. In dem hiesigen Fall ist der Agent stationär auf der CITRA Komponente angesiedelt und sorgt für ihre Integration in das CITRA System.
Für den Datenaustausch mit der Komponente selbst verwendet der Agent die Common Intrusion Specification Language (CISL) aus dem Common Intrusion Detection Framework (CIDF) [*Common Intrusion Detection Framework* n.d.]. Diese Sprache ermöglicht eine Beschreibung von Verbindungen, Anomalien, Angriffen und Gegenmaßnahmen. Arbeitet die Komponente

selbst nicht mit der CISL, muss für deren Integration eine Brückensoftware geschrieben werden, um das Format der Komponente in die CISL zu übersetzen. Sofern die Komponente eine Veränderung ihrer Konfiguration zur Laufzeit erlaubt und diese Funktionalität genutzt werden soll, sprich der Agent ihr Anweisungen erteilen soll, ist eine Übersetzung auch in der anderen Richtung notwendig. Die CISL wird in Zukunft wahrscheinlich von dem Intrusion Detection Message Exchange Format (IDMEF) abgelöst, dass von der Internet Engineering Task Force in Hinblick auf einen neuen Internet Standard entwickelt wird [*Intrusion Detection Message Exchange Format Working Group* n.d.]. Das Format wird mit XML realisiert, wodurch für die Verarbeitung von IDMEF Dokumenten bereits existierende Software herangezogen werden kann (XML-Parser, etc.). Die Struktur des Formats ist objektorientiert und erlaubt im Gegensatz zur CISL keine Mehrdeutigkeiten bei der Interpretation der Inhalte. Es ist denkbar, dass bei einer Weiterentwicklung der CITRA das IDMEF für die Integration der Komponenten genutzt wird.

### 3.2.5 Integration von Sensoren

Als Sensoren können alle Systeme verwendet werden, die Anomalien oder Angriffe entdecken können, wie z.B. Intrusion Detection Systeme, Werkzeuge zur Überwachung der Systemintegrität oder Viren Scanner. Auch Firewalls können dafür in Frage kommen, wenn sie auftretende Verletzungen ihrer Regeln aufzeigen. Wenn auf einer CITRA Komponente mit einem Sensor ein Alarm ausgelöst wird, entscheidet der CITRA Agent der Komponente anhand seiner Konfiguration, welche Reaktion in Frage kommt. Dazu gibt es die folgenden Möglichkeiten, wobei die letzte Möglichkeit über die Funktionalität eines reinen Sensors hinausgeht und nur dann zur Verfügung steht, falls die Komponente Gegenmaßnahmen ergreifen kann.

1. keine Reaktion
2. einen Report an den Discovery Coordinator schicken
3. den Angriff zurückverfolgen lassen
4. den Log- bzw. Prüfungslevel erhöhen
5. den Angriff mit einer defensiven Gegenmaßnahme stoppen oder abschwächen

Eine stärkere Reaktion schließt alle vorhergehenden Reaktionen mit ein. Gibt es eine Reaktion, wird also immer ein Report an den Discovery Coordinator geschickt. Für die Entscheidung über die Reaktion führt der Agent eine Berechnung durch, für die er drei Parameter benötigt:

- die Schwere des Angriffs
- die Bestimmtheit, hier: die Wahrscheinlichkeit, das der Angriff wirklich ein Angriff ist (certainty)
- Grenzwerte für die fünf verschiedenen Maßnahmen

Die Schwere eines Angriffs ergibt sich aus dem Wert der betroffenen Ressource zur aktuellen Tageszeit und ihrer der Verletzlichkeit gegenüber dem Angriffstyp. Die Wertangaben über die Ressource und ihre Verletzlichkeiten gegenüber den Angriffstypen werden in der CITRA-spezifischen Konfiguration der Komponente festgelegt. Die Bestimmtheit eines Angriffs liefern

einige Sensoren schon von sich aus. Wenn dies nicht der Fall ist oder die Angaben des Sensors nicht verwendet werden sollen, muss der CITRA Agent die Bestimmtheit hinzufügen. Dazu verwendet er eine Tabelle aus der Konfiguration, in der für jeden Angrifftyp die Wahrscheinlichkeit eines Fehlalarms abgelegt ist. Die Grenzwerte für die fünf möglichen Maßnahmen beziehen sich auf die beiden ersten Parameter Schwere und Wahrscheinlichkeit. Sie legen fest, bei welcher Kombination von Schwere und Bestimmtheit welches Vorgehen in Frage kommt. So kann zum Beispiel bei Angriffen mit einer Bestimmtheit von 0 Prozent auf Maßnahmen verzichtet werden, bei mittlerer Bestimmtheit und mittlerer Schwere kann die Rückverfolgung des Angriffs eingeleitet werden, und bei schwerwiegenden Angriffen ab mittlerer Bestimmtheit können abwehrende Maßnahmen getroffen werden (falls die Komponente es erlaubt). Nachdem der Agent berechnet hat, welche Reaktion in Frage kommt, führt er die entsprechenden Aktionen aus. Allerdings muss er für das Ergreifen einer Gegenmaßnahme noch eine weitere Konfiguration auswerten, in der die Zeitliche Dauer einer von ihm angeordneten Gegenmaßnahme begrenzt wird. Mehr dazu später in Abschnitt 3.2.8. Eine CITRA Komponente mit einem Sensor bildet das erste Glied in der Kette einer Angriffsbekämpfung durch die CITRA Komponenten und kann eventuell schon selbst eine Gegenmaßnahme ergreifen.

### 3.2.6 Integration von Vulnerability Assesment Tools

Die CITRA unterstützt die Integration von Vulnerability Assesment Tools. Es sind Programme, häufig Scanner, die im eigenen System bzw. Netzwerk nach Schwachstellen und Verletzlichen Stellen suchen. Sie führen beispielsweise Portscans an den eigenen Systemen durch, um alle offenen Ports zu ermitteln. Ein Beispiel für derartige Tools ist der CyberCop Scanner der Firma Network Associates [NetworkAssociates n.d.*c*]. Mit ihm lassen sich Systeme auf diverse bekannte Schwachstellen überprüfen. Zusätzlich lassen sich Intrusion Detection Systeme auf die Erkennung von Angriffen testen. Die Bibliotheken von bekannten Angriffen und Schwachstellen werden von Network Associates ständig aktualisiert.
Damit Vulnerability Scanner keine Alarme in der CITRA auslösen, müssen ihre Handlungen den CITRA Komponenten bekannt gemacht werden, die den Scan entdecken könnten. Dazu lässt sich in der Konfiguration einer CITRA Komponente zunächst festlegen, wie sie auf friedliche Scans reagieren soll. Beispielsweise soll sie über friedliche bzw. angemeldete Scans lediglich berichten, während feindliche bzw. unangemeldete Scans verfolgt und verhindert werden sollen. Die Anmeldung und Durchführung von friedlichen Scans wird in der folgenden Ablaufbeschreibung verdeutlicht.
Am Discovery Coordinator wird ein Plan über den nächsten Vulnerability Scan erstellt. Diesen sendet der DC in Form einer scan-Message an den CITRA Scan Agent. Eine scan-Message beschreibt, welche Schwachstellen an welchen Komponenten aus dem gesamten Netzwerk zu überprüfen sind. Der Scan Agent berechnet willkürlich einen in einem bestimmten Intervall liegenden Zeitpunkt, an dem der Scan starten soll. Kurz vor dem Beginn des Scans sendet er eine Anmeldung des Scans an alle CITRA Komponenten, die den Scan entdecken könnten. Anschließend lässt der Scan Agent den Scan von einem Scanner durchführen. Die integrierten Sensoren an den umliegenden CITRA Komponenten entdecken evtl. den Scan. Sie erkennen ihn jedoch als friedlichen, rechtmäßig angemeldeten Scan an und Handeln nach ihrer Konfiguration entsprechend anders als bei einem unangemeldeten Scan. Sie senden zum Beispiel einen Bericht über den Scan an den Discovery Coordinator. Nach der Beendigung des Scans sendet der Scan Agent erneut eine Nachricht an die entsprechenden CITRA Komponenten, um den Scan wieder abzumelden. Die Scan-Ergebnisse sendet er an den Discovery Coordinator.

Bei dem Verfahren ist es sehr wichtig, dass die Kommunikation zwischen den CITRA Komponenten vertrauenswürdig ist. Ansonsten könnte ein Angreifer die Funktionalität ausnutzen und zum Beispiel seinen eigenen Scan vorher anmelden. Oder er könnte Ergebnisse von Scans mitlesen und diese Informationen zu seinem Vorteil nutzen. Damit blieben ihm eigene Scans erspart. Um das CITRA System zu täuschen, könnte er versuchen, die Ergebnisse auf ihrem Weg zum DC zu verfälschen. Diese Überlegungen zeigen, dass Vertraulichkeit, Integrität und Authentizität der ausgetauschten CITRA Nachrichten gewährleistet bzw. überprüft werden müssen. Im Abschnitt 3.2.2 über das IDIP Protokoll wurden diese Anforderungen, die für die gesamte Kommunikation der CITRA Komponenten gelten, bereits benannt.
Ein weiterer sicherheitsrelevanter Aspekt ist, dass die Überprüfungen immer in unregelmäßigen Abständen durchgeführt werden. Da der Scan Agent den Zeitpunkt eher zufällig bestimmt, kann auch bei einer regelmäßigen Konfiguration der Scans am Discovery Coordinator keine genaue Zeitangabe gemacht werden. Dadurch verhindert man, dass ein Angreifer genau ausrechnen kann, wann der nächste Scan durchgeführt wird. Mit diesem Wissen könnte er versuchen, während der friedlichen Scans selbst einige kleinere Scans hinzuzufügen, die im dann nicht unbedingt auffallen würden. Durch die Unregelmäßigkeit müsste der Angreifer, auf seinen eigenen Scan vorbereitet, auf den nächsten friedlichen Scan warten.
Die Ergebnisse einer Überprüfung mit einem Vurnerability Assesment Tools werden vom Scan Agent an den Discovery Coordinator gesendet. Dieser beginnt anschließend mit der Auswertung. Sind neue Schwachstellen aufgedeckt worden oder alte Schwachstellen verschwunden, berechnet der DC neue Regeln für einige CITRA Komponenten. Diese sendet er jeweils in Form einer Direktive (Message Typ directive) an die betreffenden Komponenten, die den Anweisungen folgen müssen und die neuen Regeln anwenden.

### 3.2.7 Die Rückverfolgung eines Angriffs: Trace Back

Die automatische Rückverfolgung eines Angriffs ist eine der Hauptaufgaben der CITRA. Entdeckt ein Sensor einen Angriff, so kontrolliert der ihm zur Seite stehende CITRA Agent anhand der Konfiguration, welche Reaktion auf den Alarm folgen soll. Wie diese Entscheidung getroffen wird, ist im obigen Abschnitt 3.2.5 beschrieben. Fällt sie auf Rückverfolgung des Angriffs oder eine höherwertige Reaktion, so verschickt der CITRA Agent eine report-Message an den Discovery Coordinator und eine trace-Message mit Informationen über den Angriff an die umliegenden Boundary Controller, die zwischen zwei oder mehreren Neighborhoods liegen. Diese suchen dann in ihren Log-Einträgen nach Verbindungen oder Paketen, die dem Angriff zuzuordnen sind. Ist das Ergebnis positiv, so senden sie wiederum eine trace-Message an die nächstliegenden Boundary Contoller, ausgenommen der Komponente, von der sie zuvor benachrichtigt wurden. Durch dieses Vorgehen kann ein Angriff entlang des Pfades, auf dem die Pakete angekommen sind, bis an die Grenze des eigenen Netzwerks zurückverfolgt werden. Abbildung 3.2 verdeutlicht den Vorgang einer Rückverfolgung.
Wenn der Agent eines Boundary Controllers eine trace-Massage erhält, vollzieht er einen ähnlichen Entscheidungsprozess wie ein Agent auf einem Sensor im Falle eines Alarms (siehe Seite 33). Der Unterschied ist nur, dass er lediglich die Wahl zwischen den Punkten 3 bis 5 hat. Eine Rückverfolgung und ein Berichterstattung an den Discovery Coordinator muss also in jedem Fall erfolgen.

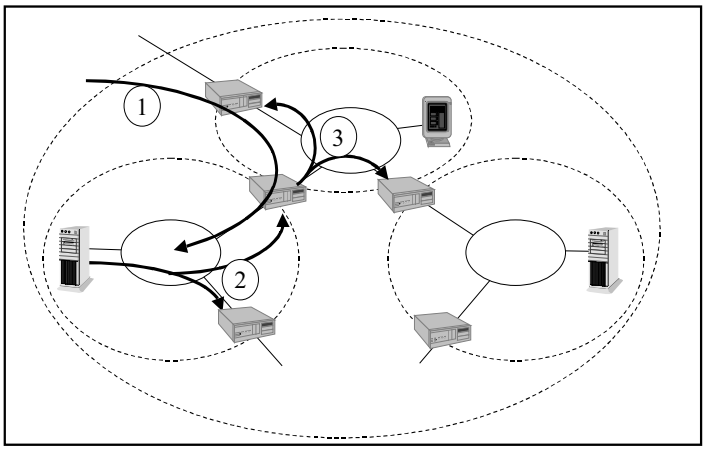


Abbildung 3.2: Rückverfolgung eines Angriffs
(Quelle: [Schnackenberg et al. 2000])

### 3.2.8 Gegenmaßnahmen und der Discovery Coordinator

Bei der CITRA kommen die Maßnahmen automatisch und direkt zum tragen, wodurch die Abwehr von Angriffen effektiver wird. Durch eine schnelle Reaktion wird ein Angriff eher unterbunden als beispielsweise durch den manuellen Eingriff eines Administrators, und der resultierende Schaden des Angriffs fällt geringer aus. Allerdings müssen hier aufgrund der Problematik bei der Automatisierung von Entscheidungsprozessen Kompromisse geschlossen werden (siehe 3.1.4). Die Kompromissbildung realisiert die CITRA in einem mehrstufigen Prozess. Zunächst werden auf den Boundary Controllern, die auf dem Pfad in Richtung Quelle des Angriffs liegen, aufgrund des in Abschnitt 3.2.5 beschriebenen Entscheidungsprozesses eventuell Gegenmaßnahmen ergriffen. Bei dieser Entscheidung wird der erste Kompromiss gemacht. Anschließend ermittelt der CITRA Agent anhand seiner Konfiguration, welche zeitliche Begrenzung für den Einsatz der Gegenmaßnahme vorgesehen ist. Diese kann je nach Tageszeit unterschiedlich sein. Damit hat ein Administrator zum Beispiel die Möglichkeit, die Dauer einer Sperrung eines Ports zur Tageszeit auf eine kurze Zeit zu begrenzen, während der Port lange Zeit gesperrt wird. Die nächste Stufe der Kompromissbildung ist die Entscheidung des Discovery Coordinators über die Gegenmaßnahme. In einer report-Message des Boundary Controllers erhält alle notwendigen Informationen über den Angriff und die Gegenmaßnahme. Er hat zuvor schon von jeder CITRA Komponente auf dem Pfad der Rückverfolgung einen Bericht erhalten. Mit den gewonnenen Informationen und seinen Informationen über den Aufbau des Netzwerks kann er nun weitere Berechnungen anstellen. Das Ziel ist, einen optimale Einsatz der Gegenmaßnahmen zu koordinieren. Kommt der DC zu dem Schluss, dass die aktuell ergriffenen Gegenmaßnahmen nicht optimal auf die Situation abgestimmt sind, sendet er Direktiven (directive-Messages) an die CITRA Komponenten, die ihr aktuelles Verhalten ändern sollen. Er kann mit einer so genannten do-message neue Gegenmaßnahmen ergreifen lassen oder mit einer undo-Message Gegenmaßnahmen wieder deaktivieren. In einer do-Message kann die zeitliche Begrenzung der Gegenmaßnahmen vorab festgelegt sein, kann aber auch offen gelassen werden. Im letzteren Fall kann die Gegenmaßnahme nur durch eine undo-Message vom DC aufgehoben werden. Die CITRA Komponenten müssen den Direktiven des DC immer folgen. Der DC stellt einen Single Point of Failure (SPOF) dar und muss aus

diesem Grunde besonders geschützt und Überwacht werden.
Bei der Konfiguration der CITRA Komponenten in Bezug auf die Gegenmaßnahmen läßt man dem Administrator weitgehend freie Hand. Ein Ziel ist, ihm die Möglichkeit zu geben, das System nur Gegenmaßnahmen ergreifen zu lassen, die er selbst in dem jeweiligen Kontext einsetzen würde. Zu Beginn der Entwicklung einer Konfiguration empfiehlt es sich, die CITRA Komponenten noch keine Gegenmaßnahmen ausführen zu lassen, sondern das System auf das Sammeln und Auswerten von Informationen zu beschränken. Von diesem Stand ausgehend kann man gezielt die Gegenmaßnahmen aktivieren und testen, und behält einen besseren Überblick über das Geschehen.
In den Versuchen, die von den Entwicklern der CITRA durchgeführt wurden, gab es bisher nur eine Integration von defensiven Gegenmaßnahmen. Gegenangriffe blieben außen vor. In den folgenden beiden Abschnitten werden die bisher in der CITRA verwirklichten Gegenmaßnahmen beschrieben.

#### Netzwerk-basierte Gegenmaßnahmen

In die CITRA wurden bisher ein paar von Netzwerk-basierten Gegenmaßnahmen testweise integriert. Eine dieser Maßnahmen ist die Verwendung eines Routers zur Begrenzung der Bandbreite und damit der Datenrate für bestimmte Verbindungen. Dies ist eine Maßnahme, die sich speziell gegen Denial of Service Angriffe richtet. Eine weitere Maßnahme ist eine dynamische Rekonfiguration eines Routers, um Pakete auf bestimmten Routen nicht mehr weiterzuvermitteln. Diese Vorgehensweise bezeichnet man allgemein als blockende Operation oder nur *block*. Eine blockende Operation kann auch mit einer Firewall oder mit einem Gateway umgesetzt werden. Mehr als diese defensiven Netzwerk-basierten Gegenmaßnahmen wurden in der CITRA nicht integriert. Eine Integration weiterer Maßnahmen ist jedoch aufgrund der Flexibilität der Architektur problemlos möglich.

#### Host-basierte Gegenmaßnahmen

Unter host-basierten Gegenmaßnahmen versteht man in Bezug auf die CITRA Maßnahmen, die sich lokal auf das Geschehen auf dem Host selbst beziehen. Die ausführenden Instanzen können zum Beispiel Programme bzw. Dienste sein, die mit root-Rechten auf einem Host laufen und vom CITRA Agenten Anweisungen erhalten können. Auf diese Art und Weise können Prozesse beendet werden, die als *angreifende* Prozesse identifiziert wurden. Ein weiterer Schritt wäre die Deaktivierung des Kontos des Benutzers auf dem Host, unter dessen UserID die Prozesse laufen. In besonders hartnäckigen Fällen kann auch ein reboot durchgeführt werden, um eine den Angriff zu unterbinden.
Eine weitere Art von ausführenden Instanzen von Host-basierten Gegenmaßnahmen stellen Betriebssystem- bzw. OS-Wrapper dar. Sie bieten Möglichkeiten, auf einen Angriff auf der Systemcall-Ebene des Betriebssystems zu reagieren. Die erfolgreiche Ausführung von Prozessen des Angreifers kann verhindert werden, indem bei Aufrufen von systeminternen Funktionen nur noch Fehler produziert werden. Eine andere, etwas fortgeschrittene Methode ist, für einen böswilligen Prozess eine falsche Umgebung zu emulieren. Auf diese Art und Weise kann der Prozess auf wertlosen, nicht der Wirklichkeit entsprechenden Fülldaten weiterarbeiten. Ein Angreifer kann so in dem Glauben gelassen werden, dass er tatsächlich auf dem realen System voranschreitet. Er wird seinen Angriff zunächst nicht abbrechen und kann beobachtet

werden.
Für den Einsatz von Host-basierten Gegenmaßnahmen sind ausreichende Rechte auf dem Host notwendig. Es sind also zunächst defensive Gegenmaßnahmen auf den eigenen Hosts. Darauf beschränkt sich auch ihr bisheriger Einsatz innerhalb der CITRA. Allerdings kann ein Bezug zu Gegenangriffen hergestellt werden. Gelingt es im Zuge eines Gegenangriffs, genügend Rechte auf dem System des Angreifers zu erlangen, so sind Host-basierte Gegenmaßnahmen, mal abgesehen von den OS-Wrappern, auch als offensive Gegenmaßnahmen einsetzbar.

### 3.2.9 Weitere Entwicklung

Die CITRA stellt eine der ersten Realisierungen eines verteilten Sicherheitssystems dar. Sie soll als Beispiel und als Grundlage für weitere Entwicklungen dienen. Es bleiben einige offene Fragen und ungelöste Probleme. Im Bereich der Auswertung von zentral gesammelten Informationen über den Zustand eines Netzwerks sucht man nach neuen Strategien und Wegen zur Verknüpfung dieser Strategien. Im Umfeld der Beschreibungssprachen von Ereignissen wie Angriffen, Anomalien und Maßnahmen ist man auf der Suche nach einer möglichst optimalen Sprache, die bisher noch nicht gefunden wurde. Weiterhin sind die Anforderungen an die Sicherheit der Kommunikation zwischen den CITRA Komponenten bisher nicht zufriedenstellend erfüllt. Und nicht zuletzt steht die Problematik der Automatisierung. Die Automatisierung kann bietet große Vorteile, ist jedoch sehr mit Vorsicht zu genießen.
Unter dem Namen „Active Network Intrusion Detection and Response“ (kurz: AN-IDR) hat eine neue Entwicklung im Bereich der verteilten Sicherheitssysteme begonnen [NetworkAssociates n.d.*a*]. Die Ergebnisse in der Entwicklung der der CITRA sollen dabei in die Entwicklung einfließen. Das Projekt zielt auf den Bereich der so genannten Active Networks. Unter dem Slogan „From Internet to ActiveNet“ arbeiten Wissenschaftler für die DARPA an der Umsetzung einer neuen Philosophie für das Netzwerkdesign [Tennenhouse, Garland, Shrira & Kaashoek 1996]. Dabei sollen Netzwerkkomponenten wie Router oder sogar Switches portable Programme ausführen können. Die Programme sollen die über die Komponente übertragene Daten in gewisser Weise verarbeiten können. Damit soll die Verhaltensweise einer Netzwerkkomponente sehr variabel gehalten werden, so dass Reaktionen auf veränderte Anforderungen an das Netzwerk schneller umgesetzt werden können als bisher. Netzwerkdienste sollen „on demand“ eingespielt werden können. Zudem möchte man eine vollständige Trennung zwischen den Netzwerkdiensten und der zugrunde liegenden Hardware erreichen. Das AN-IDR Projekt setzt auf dieser Technologie auf, um von sich aus anpassungsfähige und sich selbst einrichtende Mechanismen zur Erkennung, Rückverfolgung und Abwehr von Angriffen sowie zur Reperatur von Softwareschäden zu entwickeln.

## 3.3 Das Java Aglet Framework

Maik Dobryn

### 3.3.1 Einführung — Agentenbasierte Software

In der Vergangenheit sind verteilte Anwendungen oft in herkömmlichen Programmiersprachen, wie C oder C++ geschrieben worden. Das Ergebnis waren maschinenabhängige Binärprogramme, mit denen sich eine Reihe von Anforderungen nur unter großen Aufwendungen umsetzen ließen.

- Übertragbarkeit der Applikationskomponenten auf andere Plattformen
- Interoperabilität von Anwendungsmodulen, die mit verschiedenen Programmiersprachen entwickelt wurden
- Datenaustausch zwischen unterschiedlichen Rechnerarchitekturen
- Errichtung von Zugriffsbeschränkungen für sicherheitsrelevante Programmteile

Die inzwischen sehr populäre Programmiersprache Java löst durch ihren Aufbau und Funktionsumfang u.a. die vorgenannten Schwierigkeiten und eignet sich in besonderer Weise als Ausgangsbasis für die Entwicklung von verteilten Anwendungen. Java unterstützt Low-Level-Netzwerkschnittstellen und gängige Kommunikationsprotokolle wie FTP, HTTP, Telnet usw. Somit ist es sehr leicht, Software-Module zu entwickeln, die in der Lage sind über ein Netzwerk Daten miteinander austauschen. Darüberhinaus ist es möglich, selbstdefinierte Inhalts- und Protokoll-Handler zu implementieren und damit neue Kommunikationsmodelle oder anwendungsspezifische Protokolle zu entwerfen. Hierbei kann festgehalten werden, dass diese Flexibilität nicht zwangsläufig an hoch strukturierte Kommunikationsprotokolle wie Remote Method Invocation (RMI) gebunden ist, welche bei traditionellen Client-Server-Architekturen zum Einsatz kommen.
Wenn Anwendungsteile von unterschiedlichen Netzwerkrechnern aus interoperieren, ist die Frage nach einer effizienten Aufgaben- und Kommunikationsverteilung von Bedeutung. So zeigt sich häufig der Fall, dass Applikationsmodule über das Netzwerk permanent in Kontakt stehen, was je nach Anwendung für umfangreichen Netzverkehr sorgt. Manchmal ist es allerdings sinnvoller eine Anwendungskomponente zu einem entfernten Hostrechner zu transferieren, sie dort ihre Aufgabe erledigen zu lassen und sie dann entweder zurückzubeordern oder sich selbst vernichten zu lassen. Solche mobilen Komponenten arbeiten weitestgehend selbständig und bedürfen meist keiner Aufsicht durch eine Koordinationskomponente, was für viele Anwendungsfälle Vorteile gegenüber statischen Modulen bringt.
Derzeit existieren mehrere Application Programming Interfaces (API), die mobile Komponenten unterstützen und durch die Bereitstellung von Frameworks und Protokollen eine Entwicklung von mobilen Agenten möglich machen. Die Agententechnik liefert also Lösungsansätze für die Umsetzung von kooperativen, verteilten Anwendungen. Weiterhin bieten solche Frameworks wichtige Unterstützung für den Entwicklungsprozeß verteilter Software, denn mithilfe der zur Verfügung gestellten Funktionen erleichtern sie die fortlaufende Verbesserung von bestehenden Applikationen.

Das Aglets Software Development Kit (ASDK) von IBMs Tokyo Research Laboratory ist ein Framework für leichtgewichtige mobile Agenten. In dem vorliegenden Abschnitt wird der Aufbau und die Funktionsweise des ASDK-Framworks vorgestellt und gezeigt, dass es mithilfe von Aglets verhältnismäßig einfach ist, eigenständige verteilte Anwendungen zu programmieren. Diese können unabhängig von zentralen Applikationsservern (mit gängigen Middleware Services) agieren und bilden eine hervorragende Ausgangsbasis für die Entwicklung von netzübergreiffenden Sicherheitsanwendungen.
Das ASDK Framework von IBM beinhaltet die Java Klassendateien der Aglet-API, Dokumentation, verschiedene Beispiele, den Quell-Code und Tahiti, ein Aglet Server/Viewer. Durch zwei auf verschiedenen Rechnern gestarteten Tahiti-Server, ist es z.B. leicht möglich Aglets zu entwerfen, die sich zwischen den beiden Hosts bewegen und mit anderen Aglets über das Netzwerk kommunizieren. Weiterhin hält die API für Java-Applikationen Klassen zur Implementierung der Aglet-Server-Funktionalität bereit.

### 3.3.2 Installation und Einrichtung

#### Installation der ASDK-Distribution

An dieser Stelle soll eine Schritt-für-Schritt-Installation des Aglet Frameworks gegeben werden. Hierbei sind alle nötigen Voraussetzungen genannt und es wurde auf eine zusammenhängende, nachvollziehbare Darstellung geachtet. Die Originaldokumentation ist in den diesen Punkten leider nicht immer konsistent. Die nachfolgend benutzen Shell-Kommandos sind exemplarisch für das Vorgehen auf einem unix-kompatiblen Betriebssystem.

Zunächst kann der Download der Binärdistribution (aktuell und vom Autor getestet in der Version 2.0.2) von der Internet-Adresse

`http://www.trl.ibm.com/aglets/index_e.htm`

bzw. direkt von

`http://sourceforge.net/projects/aglets/`

vorgenommen werden. Im Anschluß daran folgt das Erstellen eines Zielverzeichnisses, in das die Installationsdatei kopiert wird.

```
$ mkdir /opt/AgletAPI
$ cp aglets-2.0.2.jar /opt/AgletAPI/
```

In diesem Verzeichnis muß die Distribution entpackt werden.

```
$ cd /opt/AgletAPI
$ jar -xvf aglets-2.0.2.jar
```

Für den folgenden Schritt ist das Installationstool ANT Voraussetzung. Falls nicht vorhanden, kann es von der Site

`http://jakarta.apache.org/ant/`

bezogen werden. Insbesondere muss die Umgebungsvariable `ANT_HOME` gesetzt sein. Ist dies der Fall, kann das Installationsskript gestartet werden.

```
$ /opt/AgletAPI/bin/ant
```

Mit Beendigung des Skripts hat sich das ASDK Framework in dem Zielverzeichnis eingerichtet und einige Konfigurationsdateien angelegt.

#### Einrichtung des Tahiti Aglet-Servers

Für die Benutzung des Tahiti-Startskripts ist es nötig, die Umgebungsvariable `AGLET_HOME` zu setzen.

```
$ set AGLET_HOME=/opt/AgletAPI
```

Auch `JDK_HOME` sollte vorhanden sein. Als nächstes muß die Beispiel-Keystore-Datei in das Home-Verzeichnis des aktuellen Benutzers kopiert werden.

```
$ cp /opt/AgletAPI/bin/.keystore ~/
```

Alternativ hierzu kann mit dem Java-Kommando `keytool` eine eigene Keystore-Datei erstellt werden. Nun kann eine Tahiti-Session mithilfe des Startskripts eröffnet werden.

```
$ /opt/AgletAPI/bin/agletsd
```

Bei Verwendung der vorgegebenen Keystore-Datei lautet der Benutzername: `aglet_key`, das zugehörige Passwort ist: `aglets`.

### 3.3.3 Aufbau und Betrieb von Aglets

#### Grundlagen

Ein Aglet ist im einfachsten Fall eine Java-Klasse, die von `com.ibm.aglet.Aglet` abgeleitet wurde.

```
import com.ibm.aglet.*;
import com.ibm.aglet.event.*;

public class HalloAglet extends Aglet {
   public void onCreation(Object init) {
      setText("Hallo Welt!");
   }
}
```

In diesem Beispiel implementiert `HalloAglet` lediglich die `onCreation()`-Methode, die immer dann zur Ausführung kommt, wenn eine Aglet-Instanz neu erzeugt wurde. Alle Aglets erben eine statische Variable für einfachen Text, die von der Aglet-Umgebung, dem sogenannten Aglet-Kontext, in eigener Weise dargestellt werden kann. Aglets können nur in einer Umgebung existieren, die eine Ausführung von mobilen Agenten unterstützen. Im ASDK Framework werden Aglets grundsätzlich in einem Aglet-Kontext erzeugt. Die API bietet Funktionen zur Generierung und Verwaltung von solchen Umgebungen. So zum Beispiel liefert die Methode `createAgletContext()` einer Instanz von `com.ibm.aglet.system.AgletRuntime`


Abbildung 3.3: Der Tahiti Aglet-Server/Viewer

einen Aglet-Kontext an das aufrufende Programm. Dieser Kontex kann entweder von einer eigenständigen Anwendung implementiert oder von speziellen, vorgegebenen Aglet-Servern bereitgestellt werden. Der dem ASDK beiliegende Aglet-Viewer Tahiti ist ein solcher allgemeiner Server. Tahiti hat eine ganze Reihe von Funktionen, wobei das Laden bzw. Erzeugen von Aglets, die Weiterleitung an Remote-Hosts sowie das Zurückrufen von Aglets zu den wichtigsten gehören. Mithilfe des Tahiti-Servers können diese Kommandos sehr leicht umgesetzt werden.
Wenn der `HalloAglet`-Agent innerhalb der Tahiti-Umgebung erzeugt wird, erscheint ein Log-Eintrag zusammen mit dem Begrüßungstext in einer auswählbaren Liste (siehe Abbildung 3.3).

Tahiti bietet Kommandoschaltflächen zur Steuerung des markierten Aglets. Der `HalloAglet`-Agent zum Beispiel kann angewählt und zu einem entfernten Host-Rechner delegiert und kurz darauf wieder zurückgeholt werden. Tahiti benutzt Dialogfenster um zusätzliche Informationen zu den jeweiligen Befehlen einzuholen.

### Aglet-Operationen und Lebenszyklus

Innerhalb der ASDK-API gibt es drei Operationen, die sich unmittelbar auf den Aglet-Lebenszyklus auswirken.

- **Create.** Sobald ein Aglet in einem Kontex bzw. in Bezug auf einen Kontex erzeugt wurde, erhält es einen einmaligen Identifier. Weiterhin wird es initialisiert und sein Ausführungsthread beginnt.
- **Clone.** Diese Operation erzeugt ein eigenständiges Dublikat von dem aufgerufenen Aglet mit eigenem Identifier und Thread.
- **Dispose.** Wenn ein Aglet seine Aufgaben ausgeführt hat und nicht mehr benötigt wird, kann es gelöscht werden. Dieser Prozeß beendet die Ausführung des Aglets und entfernt es aus dem Kontext.

Während der Lebensdauer eines Aglets kann es an weiteren Grundoperationen beteiligt sein.

- **Dispatch.** Ein Aglet wird aus seiner Umgebung gelöst und zu einer anderen Umgebung weitergeleitet. Dabei befindet sich der Zielkontext meist auf einem entfernten Netzwerk-Host. Dort angekommen wird die `run()`-Methode des Aglets ausgeführt.
- **Retract.** Das weggeschickte Aglet wird vom Remote-Host zurückgeholt, wieder in die Aglet-Umgebung eingegliedert und erneut gestartet.
- **Deactivate.** Die Ausführung des Aglets unterbricht ohne dass es den Bezug zu seinem Kontext verliert. Es wird gespeichert und verbleibt in diesem Zustand bis zu seiner Aktivierung.
- **Activate.** Das Aglet wird erneut gestartet, sobald die inaktive Phase beendet ist.

Diese fundamentalen Operationen bilden im ASDK Framework, zusammen mit den aglet-spezifischen Methoden `onCreation()`, `onDisposing()` sowie `run()` der Klasse `Aglet`, die Grundlage für das delegationsbasierte Ereignismodell.

#### Aglet-Management und -Kommunikation

Da Aglets nichts anderes als Java-Klassen sind, können diese vom Grundsatz her alles ausführen, was in der Java-API implementiert ist. Dadurch ist es möglich, Aglets auch herkömmliche Aufgaben wie Datenbankabfragen oder Dateisystemoperationen durchführen zu lassen. Die Herausforderung liegt in der sinnvollen Verwaltung von verteilten Arbeitsaufträgen. Nach [Smith 1999] lassen sich hierbei zwei Vorgehensweisen unterscheiden.

(1) Ein Manager-Aglet instruiert und koordiniert die Arbeit von mehreren Worker-Aglets.

(2) Eine Applikation, ausgestattet mit einem eigenen Kontext, bietet eingehenden Aglets einen Service-Dienst.

Was den Kommunikationsaspekt anbetrifft, so unterstützt das ASDK Framework verschiedene Arten, Nachrichten zu einem Aglet zu bringen. Die Klasse `Aglet` beinhaltet Messaging-Methoden wie `handleMessage()`, `waitMessage()`, `notifyMessage()` und weitere. Darüberhinaus bietet das Framework Interfaces und Klassen für unterschiedliche Messaging-Operationen und Verfahren (synchrone und asynchrone). Neben `Aglet`, bildet auch die Klasse `Message` einen wichtigen Ausgangspunkt, denn normalerweise erzeugen Aglets Message-Instanzen und senden diese zu bekannten Aglet-Kollegen. Ein nachrichtenempfangendes Aglet hingegen hält die Methode `handleMessage()` bereit. Abbildung 3.4 zeigt exemplarisch den Nachrichtenaustausch.

### 3.3.4 Ausblick — Angiffsgegenwehr mit Aglets

#### Einordnung in den Bereich der Informationssicherheit

Passive Schutzmaßnahmen

Der passive Bereich umfasst Maßnahmen zur dynamischen Verstärkung von Sicherheitsbarrieren. Dieser Bereich ist für den Einsatz von Aglets bestens geeignet, denn hier können ihre Vorteile gegenüber statischen Programmen voll in Erscheinnung treten.

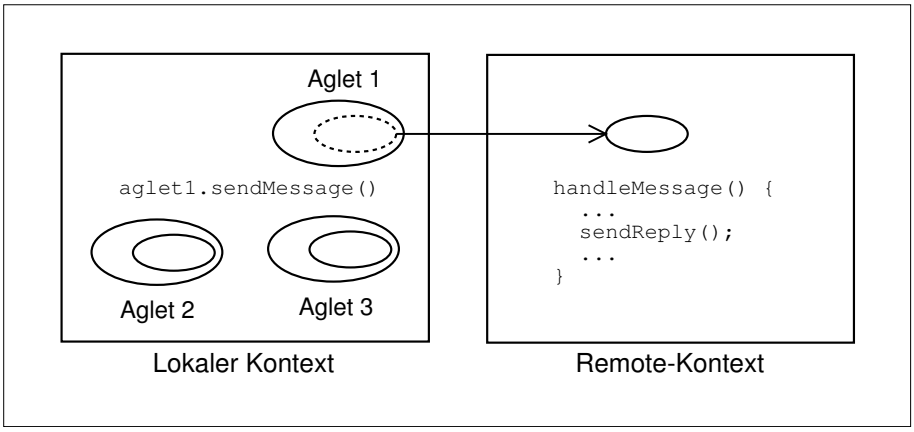


Abbildung 3.4: Nachrichtenaustausch zwischen Aglets
(Quelle: [Smith 1999])

Aktive Gegenmaßnahmen
Der aktive Bereich beschreibt Methoden zum Gegenangriff mit dem Ziel der Beendigung des Einbruchs bzw. der Vereitellung von wiederholten Angriffen des Tätersystems. In diesen Bereich können Aglets wahrscheinlich nur mit Hilfe des Java Native Interfaces (JNI) vorstoßen, da der Hauptnachteil von Aglets in der Bedingung eines Java-Aglet-Kontextes liegt.

### Abwehrszenarien mit Aglets

Allgemein sind folgende Angriffsreaktionen denkbar

- Abschottung des betroffenen Systems (oder Teile davon) vom Netzwerk oder dessen Abschaltung,
- Alarmierung des Incident Response Teams (IRT) (siehe [van Wyk & Forno 2001]) oder einer öffentlichen Behörde,
- Aktivierung dynamischer Zugriffssperren zur Einschränkung der Handlungsmöglichkeiten des Eindinglings,
- Aufrechterhaltung des Systembetriebs zur Informationsgewinnung, und
- Wiederherstellung des beschädigten Systems.

In [Frincke & Wilhite 2001] beschreiben die Autoren das Konzept eines hierachischen Netzwerks von leichtgewichtigen, mobilen, adaptiven Tools kombiniert mit einer verteilten und verbundenten Intrusion-Detection-Umgebung. Hierbei arbeiten kleine Software-Agenten mit einem integrierten Intrusion Detection System zusammen, was beiden eine größere Informationsbandbreite und somit effektive Verteidigungsstrategien verschafft. Verteilte Software-Agenten werden in drei Kategorien eingeteilt.

- **Hilfsagenten** (tool agents)
- **Aufklärungsagenten** (investigation agents)
- **Abwehragenten** (defensive agents)

### 3.3.5 Zusammenfassung

Mit der Technik der mobilen Agenten sind Entwickler nicht mehr an traditionelle Verteilungsarchitekturen wie z.B. das Zwei-Tier Client-Server-Modell oder das Drei-Tier Middlewareorientierte Modell gebunden. Das ASDK Framework ermöglicht eine flexible Code-Verteilung, womit in erster Linie modulbildende Merkmale und die Unterstützung von intelligenten Kommunikationsstrukturen gemeint sind. Damit können Applikationsteile, denen ein solches Entwurfsmuster zugrunde liegt, auf Abruf an beliebige Rechner im eigenen Netzwerk versendet und dort zur Ausführung gebracht werden.
Das ASDK Framework bietet ein Ereignismodell, das delegationsorientierte und mobilitätsgesteuerte Operationen unterstützt. Aglets können hiermit nicht nur an entfernte Aglet-Umgebungen versendet, sondern auch wieder zurückgeholt werden. Darüberhinaus können sie nach der Ankunft auf dem Ziel-Host jegliche Java-Operation ausführen, die von den Sicherheitsprivilegien zugelassen werden. Da Aglets nichts anderes als Java-Klassen sind, ist ihre Ausführung in jedem Aglet-Kontext innerhalb einer Java-Umgebung möglich, die über das Netzwerk erreichbar sind. Diese Kommunikation findet in den meisten Fällen mithilfe des Agenten-Transfer-Protokolls (ATP) statt.
Entwickler werden dazu angehalten, ihre Java-Klassen in der Weise zu entwerfen, sodass diese die Möglichkeit besitzen, sich von Host zu Host zu bewegen und jeweils ihre Aufgaben durchzuführen. Hierbei gibt es nach [Smith 1999] mindestens drei Arten die Aglet-Technik anzuwenden.

- Als Kommunikationsmechanismus. Kommunikation kann auf zwei Ebenen betrachtet werden. Zum Einen werden Aglets zwischen Hostrechnern hin- und hergesendet; dies geschied über das URL-basierte Kommunikationsprotokoll ATP. Zum Anderen kommt es vor, dass zwei Anwendungsteile miteinander kommunizieren müssen, was über den Austausch von Nachrichten realisiert wird.

- Als Datentransportmittel zwischen verschieden Netzwerklokationen. Für die Versendung von ganzen Datenpaketen bzw. Objekten jeden Inhalts stehen Aglets mit ihren Instanzvariablen zur Verfügung.

- Als Vorgabe zur Aufteilung von Anwendungsfunktionen. Der wohl wichtigste Vorteil beim Einsatz der Aglet-Technologie ergibt sich durch neue verteilte Applikationsstrukturen, die sich am Prinzip der Aufgabendelegation orientieren.

Abschließend kann festgehalten werden, dass das Paradigma der mobilen Agenten eine hervorragende Ausgangsbasis für eine übergreifende Anwendungsarchitektur bietet, was für den Bereich der aktiven Informationssicherheit ein großes Potential darstellt.

# 4 Realisierung eines Systems für automatisierte Gegenangriffe

Sascha Sertel

Das vorliegende Kapitel erweitert die bekannten Verfahren der Intrusion Detection um die Reaktionsmöglichkeiten der aktiven Gegenwehr. Dazu müssen bestimmte Voraussetzungen gegeben sein, die in einem einleitenden Teil dargestellt werden. Darauf aufbauend soll anhand ausgewählter Beispielkonfigurationen gezeigt werden, wie - wenn auch mit Einschränkungen - das Intrusion Detection Tool Snort [Caswell n.d.] zur automatisierten Gegenwehr benutzt werden kann. Dazu wurden zwei Angriffsmuster ausgewählt, auf die das System reagieren wird. Nach der Vorstellung der Beispiele werden die Einschränkungen aufgezeigt, denen diese Verfahrensweise unterliegt und aus denen sich die genannten Voraussetzungen ergeben. Zum Schluss wird ein Ausblick gegeben auf die mögliche Entwicklung von zukünftigen Systemen zur Gegenwehr.

## 4.1 Einleitung

Um aktive Gegenwehr durchführen zu können, gehen wir von folgender Situation aus:

1. **Opfer:**
   Es gibt einen Rechner in einem Netzwerk, auf dem ein ID&RS installiert ist und ausgeführt wird.

2. **Angriff:**
   Dieser Rechner wird von einem Angreifer mit einem bestimmten Angriffsmuster attackiert.

3. **Intrusion Detection:**
   Das ID&RS erkennt das Angriffsmuster und stellt einen Angriff fest.

4. **Vorbereitung der Gegenwehr:**
   Das ID&RS kann aus den Daten des Angreifers dessen Quellrechner ermitteln.

Die Situationsbeschreibung wurde bewusst auf einem abstrakten Niveau erstellt, da die Art des Angriffs und die Netzwerkkonfiguration, mit der der Angriff stattfand, sehr verschieden sein können. Aus Komplexitätsgründen sehen wir deshalb von einer differenzierteren Unterscheidung des Angriffs ab. Stattdessen interessieren wir uns für den Schritt, der dem Erkennen des Angreifers folgt, nämlich der eigentlichen Gegenwehr.

In Kapitel 2 auf Seite 11 wurde bereits auf die Probleme eingegangen, die das Auffinden des tatsächlichen Hosts des Angreifers bereiten. In diesem Kapitel gehen wir davon aus, dass dieses Problem gelöst wurde und der anzugreifende Zielrechner bekannt ist. Demzufolge wurden die in Kapitel 4.2 durchgeführten Beispiele so aufgebaut, dass eine Direktverbindung zwischen Angreifer und Opfer besteht und keine Zwischensysteme für den Angriff verwendet werden.

Der Angegriffene besitzt also jetzt die Möglichkeit, den Angriff zu erwidern. Dies erfolgt durch eine vorprogrammierte Reaktion, die mit der Rechneradresse des Angreifers ausgeführt wird. Auf die dabei existierenden Einschränkungen wird in Kapitel 4.3 auf Seite 55 eingegangen.

## 4.2 Anwendungsbeispiel

Die hier beschriebenen Beispiele stellen keine Anleitung zum Entwurf eines Gegenangriffs-Werkzeugs dar, sondern sollen demonstrieren, wie sich mit den zur Zeit der Erstellung dieses Berichtes verfügbaren Mitteln eine automatisierte aktive Gegenwehr realisieren lässt. Dafür wird zunächst der Aufbau der Testsysteme, deren Netzwerkstruktur, sowie die verwendete Systemkonfiguration beschrieben.

### 4.2.1 Angriffsmuster

Für die Beispielkonfigurationen wurden folgende Angriffsmuster ausgewählt:

- Portscan:
  Ein Portscan selbst ist zwar kein direktes Angriffsverfahren, allerdings stellt er meist die erste Stufe eines Gegners dar, mögliche Sicherheitslöcher in einem System zu finden. Dabei werden auf dem Zielsystem systematisch alle offenen Ports ermittelt und die auf dem System installierten Dienste dabei untersucht. Viele Dienste haben Standardports, unter denen sie typischerweise zu erreichen sind, beispielsweise benutzen Web-Server meist den Port 80, die sichere Remote-Shell SSH besitzt den Standardport 22.
  Heutige Intrusion Detection Systeme sind in der Lage, solche Portscans festzustellen. In der Unix/Linux Welt ist der Portscanner Nmap [Fyodor n.d.] sehr weit verbreitet, deshalb wird er auch für den hier dargestellten Beispielangriff mit einem Portscanner verwendet.

- Das Systemsicherheitstool Nessus [Deraison n.d.]:
  Nessus selbst ist kein Angriffstool, sondern im Gegenteil ein Programm zum Auffinden von Sicherheitslöchern in einem System. Dazu verwendet es allerdings Techniken, die sich von Intrusion Detection Systemen feststellen und als Angriffsmuster erkennen lassen. Nessus wurde als zweites Beispiel ausgewählt, um zu zeigen, dass aktive Gegenwehr für jedes erkennbares Angriffsmuster eingesetzt werden kann.

Die Angriffsmuster wurden aufgrund ihrer Verbreitung und Nachvollziehbarkeit ausgewählt, es ist aber darüber hinaus möglich, jeden Angriff, der von einem ID&RS erkannt werden kann, mit einem Gegenangriff zu erwidern. An der gezeigten Beispielkonfiguration wird außerdem deutlich, dass sich für verschiedene Angriffsmuster unterschiedliche Reaktionen definieren lassen, die eine subtilere Reaktion auf den jeweiligen Angriff ermöglichen.

### 4.2.2 Aufbau

Wie zu Beginn des Kapitels erklärt, werden für die Beispiele vereinfachte Annahmen getroffen. Weitere Erläuterungen dazu sind in Kapitel 4.3 auf Seite 55 zu finden:

- Es gibt keine Zwischenrechner, die zum Angriff benutzt wurden, das heißt die vom Intrusion Detection System erkannte IP-Adresse ist die tatsächliche IP-Adresse des Angreifers.
- Es findet nur ein Angriff gleichzeitig von einem einzigen Host statt. So lässt sich der Ablauf Angriff zu Gegenangriff besser in der Testumgebung nachvollziehen.
- Der Angriff findet nur kurz statt. Das Erkennen des Anfangs und Endes eines länger anhaltenden Angriffs ist eine weitere Problematik, die damit vermieden werden soll.

Demzufolge besteht das Testnetzwerk aus zwei Rechnersystemen, die über einen 100 MBit Switch miteinander verbunden sind. Auf dem System, das angegriffen werden soll, ist die Linux Distribution Mandrake 8.2 installiert, auf dem Angreifer-System befinden sich eine Microsoft Windows XP Installation sowie eine Mandrake Linux 8.2 Installation[1].

Die Angriffserkennung sowie die Reaktion in Form einer Gegenwehr findet also unter Linux statt, während Angriffe sowohl von Linux als auch von Windows XP aus ausgeführt werden. Die genauen Hardwarespezifikationen sind für die weiteren Tests unerheblich, es sei lediglich noch gesagt, dass beide Systeme über annähernd gleiche Prozessorstärke verfügen und sich daher aus der Hardwarekonfiguration keine Vorteile oder Nachteile für Angreifer oder Opfer ergeben.

Als Intrusion Detection System wird unter Linux das Tool Snort [Caswell n.d.] verwendet, für die Gegenwehr werden gleich zwei Programme vorgestellt, Swatch [Atkins n.d.] und FWLogWatch [Wesslowski n.d.]. Zum Angriff werden unter Linux und Windows XP die Programme Nmap [Fyodor n.d.] und Nessus [Deraison n.d.] eingesetzt. Keines dieser beiden Programme ist ein reines Angriffstool, jedoch besitzen sie Funktionalitäten, die sich zu Angriffszwecken verwenden lassen[2].

Für den Aufbau eines zur Gegenwehr fähigen Systems sind vor allem auf dem Linux System einige Softwarepakete notwendig, deren Installation und Konfiguration im folgenden kurz erläutert wird.

### 4.2.3 Systemkonfiguration

#### Mandrake Linux 8.2

Um die folgenden Angriffsbeispiele möglichst allgemeingültig zu halten, wurde außer den Anpassungen am Intrusion Detection System eine Standard Mandrake Linux Installation durchgeführt. Sollen die Beispiele auf einer anderen Linux Distribution wie beispielsweise

[1] Mandrake Linux ist ein eigetragenes Warenzeichen der Firma Mandrake, Windows XP ist eingetragenes Warenzeichen der Firma Microsoft.

[2] Daran erkennt man, wie nah auf technischer Seite die Begriffe Sicherheitsüberprüfung und Angriff zusammenliegen.

SuSE, Redhat oder Debian nachvollzogen werden, gilt zu beachten, dass jede Distribution mit unterschiedlichen Standardeinstellungen vor allem in Bezug auf die Konfiguration von Sicherheitsmaßnahmen wie Firewall oder Intrusion Detection Systemen ausgeliefert wird. Es kann daher nicht garantiert oder davon ausgegangen werden, dass die hier beschriebenen Anpassungen auf einer anderen Distribution funktionieren oder die gleichen Ergebnisse liefern werden.

Im Folgenden gilt für alle genannten Versionsnummern, dass, falls nicht anders angegeben, es sich um die zum Zeitpunkt der Erstellung dieses Berichts aktuellste Version handelt. Außerdem werden für die Installation der Anwendungsprogramme nur die wichtigsten Punkte beschrieben, es wird als bekannt vorrausgesetzt, wie eine gepackte Quellcodedatei unter Linux heruntergeladen und entpackt wird.

### Windows XP

Bei der Windows Installation wurde eine Standard-Installation durchgeführt. Es ist keine weitere Vorbereitung des Systems nötig abgesehen von der Installation der Angriffstools. Die Beschreibung dazu befindet sich in den jeweiligen Abschnitten zu Nessus und Nmap.

### Snort 1.8.3

Das Programm Snort wird in der Version 1.8.3 verwendet, die bei Mandrake Linux enthalten und bereits vorkonfiguriert ist. Zum Zeitpunkt der Erstellung dieses Berichts gab es zwar auf der Homepage von Snort bereits eine neuere Version, allerdings beinhalten die neueren Versionen meist nur Updates für neue Angriffsmuster sowie behobene Fehler oder Sicherheitslücken. Da aber der Nachweis der Sicherheit von Snort nicht Gegenstand dieses Berichtes ist, sondern die Verwendung der Reaktionsfunktionalität von Snort, die sich in den letzten Versionen nicht geändert hat, ist die vorliegende Version ausreichend. Zur Nachvollziehbarkeit auf einem anderen System wird allerdings mindestens eine Snort Version neuer als 1.80 vorausgesetzt, da mit dieser Version eine neue Syntax für die Konfigurationsdateien eingeführt wurde. Wie bei den Linux Distributionen gilt auch hier, dass Syntax und Verhalten von Snort in zukünftigen Versionen möglicherweise erneut geändert werden und dadurch die beschrieben Beispiele nicht mehr funktionieren könnten.

Snort läuft standardmäßig als Dienst im Hintergrund und schreibt die detektierten Attacken in Logfiles im Verzeichnis `/var/log/snort`. Zusätzlich gibt es das Kommando `snort`, mit dem man auf der Standardausgabe die Daten sehen kann, die Snort produziert.

### Swatch 3.0.2

Snort ist selbst nicht in der Lage, als Reaktion auf einen Angriff ein Kommando auszuführen. In der Dokumentation werden dazu auch Gründe formuliert:

- Snort soll möglichst echtzeitnah ablaufen. Durch den Aufruf externer Anwendungen und möglicherweise sogar Warten auf ein Ergebnis dieser Anwendung verzögert die Verarbeitung der eintreffenden Pakete, so dass dadurch Pakete unverarbeitet bleiben oder verloren gehen können. Dieses Nachteil möchte man nicht in Kauf nehmen.

- Selbst wenn Snort nicht die Anwendung starten, sondern nur eine im Hintergrund schlafende Anwendung aufwecken würde, entstünden Verzögerungszeiten, die die Paketverarbeitung beeinträchtigen.
- Stattdessen wird vorgeschlagen, eine aktive Anwendung zusätzlich zu Snort zu haben, die die Logeinträge von Snort permanent überwacht und dann in einer Angriffssituation sofort reagiert, ohne den Betrieb von Snort zu stören.

Swatch ist ein solches Programm, im nächsten Abschnitt wird noch eine zweite Alternative vorgestellt. Swatch ist aber kein Programm speziell zur Ausführung von Gegenangriffen, sondern ein Programm, das in der Lage ist, Logdateien zu überwachen und auf bestimmte Muster eine definierte Reaktion auszuführen.

Für die Installation wurde Swatch 3.0.2 von den Mandrake Linux CDs installiert, da es bei der Standard-Installation nicht mitinstalliert wurde. Außerdem sind für die Installation noch folgende Pakete notwendig, die sich ebenfalls auf den CDs befinden:

- `perl-File-Tail`
- `perl-Date-Calc`
- `perl-Time-HiRes`
- `perl-TimeDate`

Nach der Installation steht der Befehl `swatch` zur Verfügung. Über eine Konfigurationsdatei können die Muster in Form regulärer Ausdrücke definiert werden, auf die reagiert werden soll. Für die Beispielkonfiguration wurde eine Datei namens `/etc/swatchrc.is` angelegt mit folgendem Inhalt:

```
#
# Swatch configuration file for constant monitoring
#

# Fin Portscan Angriff
watchfor   /FIN/
        echo bold
        exec echo Fin-Angriff!!!! >> /root/gegenangriff
        exec nmap $4 >> /root/gegenangriff
        exec smbclient -M $4 < /etc/swatch.FINresponse

# Syn Portscan Angriff
watchfor  /SYN/
        echo bold
        exec echo Syn-Angriff!!!! >> /root/gegenangriff
        exec nmap $4 >> /root/gegenangriff
        exec smbclient -M $4 < /etc/swatch.SYNresponse
```

Damit werden zwei Regeln beschrieben:

- Die erste Anweisung sucht nach Logeinträgen, die die Zeichenkette `FIN` enthalten. Eine solche Zeile wird erzeugt, wenn eine bestimmte Art des Portscans ausgeführt wurde. Als Reaktion darauf wird Nmap gestartet, um ebenfalls einen Portscan auf dem angreifenden System auszuführen. Außerdem wird das Programm `smbclient` ausgeführt, das eine Nachricht an das Angreifersystem schickt. Der Text der Nachricht wird aus der Datei `/etc/swatch.FINresponse` gelesen wird. Der Text darin lautet: "Gegenangriff auf einen FIN Portscan". Die IP Adresse des angreifenden Systems wird mit dem Parameter `$4` eingefügt, damit wird die Stelle bezeichnet, in der sich die IP des Angreifers in der gefundenen Zeichenkette befindet. Die gesendete Nachricht erscheint beim Angreifer in Form eines Popup-Fensters auf dem Desktop.

- Die zweite Anweisung sucht ebenfalls nach einem Portscan Angriff, diesmal mit der Zeichenkette SYN. Es wird eine gleichartige Reaktion wie beim ersten Angriffsmuster ausgeführt, lediglich mit einer anderen Nachricht.

Hier wird deutlich, dass es möglich ist, zwei unterschiedliche Angriffsmuster mit unterschiedlichen Reaktion zu versehen. Zwar stammen beide Muster aus der Angriffsklasse der Portscans, es handelt sich aber dennoch um unterschiedliche Angriffe.

Statt der Reaktion mit Nmap ist es ebenso möglich, ein Angriffstool einzusetzen, das einen anderen Angriff auf das angreifende System ausübt, z.B. eine aggressivere DoS Attacke. Die Nachricht, die in diesem Beispiel an das System versendet wird, dient lediglich der Visualisierung des Angriffs beim Angreifer. Andernfalls ließe sich ein erfolgreicher Gegenagriff nur nachvollziehen, wenn auf dem angreifenden Windows System ebenfalls ein Intrusion Detection System installiert werden würde, das den ausgeführten Portscan erkennen und melden kann.

### FWLogWatch 0.8.1

Bei FWLogWatch handelt es sich um ein weiteres Programm zum Überwachen von Logdateien. In der Dokumentation heißt es sogar ausdrücklich, dass es besonders geeignet ist, um den Output von Intrusion Detection Systemen wie Snort nach Angriffen zu durchsuchen und darauf zu reagieren:

> *Finally, it can also run as daemon (with web interface) doing realtime log monitoring and reporting anomalies or starting attack countermeasures.*[3]

Dieser vielversprechende Ansatz ließ sich auf unserem Testsystem nicht nachvollziehen. Zwar ließ sich das Programm einwandfrei installieren, nach der Erstellung und Anpassung der entsprechenden Konfigurationsdateien fand aber trotzdem keine Reaktion auf ausgeführte Angriffe statt. Eine Ursache für dieses Fehlverhalten konnte nicht gefunden werden. Zur Zeit der Fertigstellung dieses Berichtes gab es keine Stellungnahme vom Autor zu diesem Problem.

### Nmap 2.54

Nmap ist ein Sicherheitstool, das in der Lage ist, offene Ports und Services auf einem System festzustellen, sowie das installierte Betriebssystem. Da einige der in Nessus enthaltenen Tests auf Nmap aufbauen, muss es vorher installiert werden. Es gibt Nmap sowohl für Linux als

[3] Quelle: [Wesslowski n.d.]

auch für Windows.

Die Installation unter Linux funktioniert folgendermaßen: Zuerst wurde Version 2.54 Beta 36 von [Fyodor n.d.] heruntergeladen, in ein temporäres Verzeichnis entpackt und durch Ausführen von

```
configure && make && make install
```

im entpackten Verzeichnis kompiliert und installiert. Danach ist der Befehl `nmap` verfügbar.

Für Windows gibt es auf der Homepage ein Zip-Archiv zum Download, das eine bereits kompilierte ausführbare Datei enthält. In der Windows Eingabeaufforderung lässt sich dann der Befehl `nmap` genauso verwenden wie unter Linux.

### Nessus 1.2.3

Nach der Installation von Nmap kann die Nessus installiert werden. Von [Deraison n.d.] wurde Nessus Version 1.2.3 heruntergeladen. Dabei wurde von den verschiedenen Download-Möglichkeiten die Shell-Skript Variante gewählt. Dazu wird nur die Installations-Datei mit dem Namen `nessus-installer.sh` benötigt. Diese wird mit dem Kommandozeilen-Befehl

```
sh nessus-installer.sh
```

ausgeführt, was den Installationsvorgang startet. Die Fragen zum Installationspfad wurden mit den vorgeschlagenen Standardeinstellungen beantwortet. Nach der Installation müssen noch weitere Schritte durchgeführt werden:

- Nessus Zertifikat Erstellen durch Aufruf von `nessus-mkcert`
  Bei den Fragen zur Lebensdauer des Zertifikates wurden die Vorgaben übernommen, als Land wurde `DE` für Deutschland eingetragen, als Organisation `FH Bonn-Rhein-Sieg` angegeben. Danach wird das Zertifikat erstellt und automatisch die entsprechenden Konfigurationsdateien aktualisiert.
- Nessus Benutzer Hinzufügen mit dem Befehl `nessus-adduser`
  Es wird ein Benutzer mit dem Namen `nessus` hinzugefügt, als Authentifizierungsmethode wird Passwort gewählt, das ebenfalls auf `nessus` gesetzt wird. Da es sich hier nur um eine Testumgebung handelt, muss auf die Sicherheit des Passwortes nicht geachtet werden. Die Frage nach den Regeln für den Benutzer wird unbeantwortet gelassen, d.h. ohne Regeln, danach ist der Benutzer hinzugefügt.
- Den Nessus Dienst starten durch das Kommando `nessusd -D`

Jetzt kann der Nessus-Client gestartet werden mit dem Befehl `nessus`. In dem Fenser trägt man zum Benutzer das Passwort ein, beim ersten Login wird man dann aufgefordert, das Zertifikat vom Server zu akzeptieren, danach erhält man das Nessus Hauptfenster mit den verschiedenen Test-Methoden und Optionen.

#### NessusWX Installation

Um den Nessus Dienst von Windows aus nutzen zu können, benötigt man einen Nessus Client für Windows. Es gibt verschiedene Clients für Windows, auf der Nessus Homepage wurde aber NessusWX als besonders empfehlenswert beschrieben, weshalb die Entscheidung darauf fiel.

Von der Homepage [Kirhenshtein n.d.] gab es Version 1.3.3 als Windows-Installationsdatei zum Download, das Programm installierte sich ohne Fehler durch Starten der Installationsdatei. Danach steht das Programm `NessusWX` im Startmenü zur Verfügung. Dort ist genau wie bei `nessus` unter Linux die Anmeldung am Nessus Server möglich unter Verwendung des zuvor festgelegten Passworts.

### 4.2.4 Angriff mit Nmap

Als erster Angriff wird ein Portscan mittels Nmap gewählt. Das Windows-Sytem wird das Linux-System angreifen, letzteres wird den Angriff bemerken und mit der vorher definierten Gegenwehr antworten.

#### Systemzustand vor dem Angriff

Die beiden Testsysteme sind jetzt vorbereitet, um eine konstruierte Angriffssituation durchzuführen:

- Das Opfer-System wurde mit dem installierten Mandrake Linux gebootet, der Snort Dienst wurde dabei automatisch gestartet. Um das Gegenangriffssystem zu aktivieren, muss noch Swatch gestartet werden. Ein automatisches Starten von Swatch ist mit einem entsprechenden Konfigurationseintrag möglich, hier wurde jedoch der manuelle Weg gewählt:

  ```
  swatch  --config-file=/etc/swatchrc.is
          --tail-file=/var/log/snort/portscan.log
  ```

  Damit wird Swatch mit unserem Konfigurationsfile gestartet und lauscht auf das Snort-Logfile für die Portscans.

- Das Angreifer-System wurde mit dem installierten Windows XP gebootet, für den Angriff wird die Windows Eingabeaufforderung gestartet, in der wir im nächsten Schritt das Kommando für den Nmap Aufruf eingeben werden.

#### Ausführen des Angriffs

Auf dem Angreifer-System wird jetzt in der Eingabeaufforderung folgendes Kommando ausgeführt:

```
nmap -T insane 192.168.19.251 -P0 -p 1080 -sF -v
```

Kurze Erläuterung der verwendeten Parameter:

- Mit `-T insane` wird der Scanmodus eingestellt. Die Modi unterscheiden sich vor allen Dingen in der Geschwindigkeit, mit der einzelne Ports gescannt werden. Der Modus `insane` führt den Scan mit der höchsten Geschwindigkeit aus. Da Snort den Portscan auch im langsamsten Modus erkennt, verringert die Geschwindigkeit nur die Ausführungszeit von Nmap.

- Der anzugreifende Rechner besitzt in der vorliegenden Netzwerkkonfiguration die IP-Adresse `192.168.19.251`.

- Der Parameter `-P0` ist notwendig, da auf dem Linux System normale ICMP ECHO Pakete direkt geblockt und gar nicht erst beantwortet werden. Dies ist eine Sicherheitseinstellung, die direkt im Systemkernel verankert ist und ausgeführt wird, noch bevor Snort oder ein anderes Programm Zugriff auf das Paket erhält. Der Parameter veranlasst Nmap, eine alternative Methode zum Schicken der Pakete zu verwenden.

- In der Beschreibung der Ausgangssituation wurde gesagt, dass wir von einem möglichst kurzen Angriff ausgehen. Deshalb wird kein vollständiger Portscan durchgeführt, sondern mittels der Angabe `-p 1080` nur ein einziger Port, nämlich 1080, gescannt. Dieser Port wird meist von HTTP-Proxy Diensten verwendet und wurde hier als Beispiel ausgewählt. Es spielt für den Angriff dabei keine Rolle, ob der Dienst tatsächlich auf dem angegriffenen System vorhanden ist oder nicht. In unserem Fall ist der Dienst nicht vorhanden.

- Die Art des Portscans wird über den Parameter `-sF` geregelt. Das `F` steht für FIN und löst den Portscan mit gesetztem FIN-Flag in den Paketen aus. Im Abschnitt zur Swatch Konfiguration wurde speziell für diesen Angriff eine Reaktion definiert.

- Die letzte Kommandozeilenangabe `-v` steht für "verbose" und veranlasst Nmap dazu, bei der Ausführung zusätzliche Statusinformationen auszugeben. So kann man am angreifenden System nachvollziehen, ob der Portscan erfolgreich durchgeführt wurde.

#### Automatischer Gegenangriff

Wie erwartet wird der Portscan von Snort in die Logdatei geschrieben und von Swatch gefunden. Die von uns definierte Reaktion wird ausgeführt und auf dem Bildschirm des angreifenden Windows-Systems erscheint die festgelegte Nachricht für den FIN Portscan.

Zur Kontrolle wurde der zur Gegenwehr ausgeführte Nmap in eine Datei geloggt. Sie hat nach dem Gegenangriff folgenden Inhalt:

```
Fin-Angriff!!!!

Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Interesting ports on port-192-168-19-219.dhcp.stw-bonn.de (192.168.19.219):
(The 1551 ports scanned but not shown below are in state: closed)
Port       State       Service
135/tcp    open        loc-srv
139/tcp    open        netbios-ssn
445/tcp    open        microsoft-ds
```

```
1025/tcp   open        listen
1723/tcp   open        pptp
3389/tcp   open        msrdp
5000/tcp   open        fics

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

Der Gegenangriff wurde erfolgreich durchgeführt, das Ergebnis des Portscans in die Logdatei geschrieben. So kann der Systemadministrator später nachvollziehen, welche Gegenangriffe durchgeführt wurden und welches Ergebnis sie brachten.

### 4.2.5 Snort mit Nessus

Um mit Nessus eine Angriffssituation für Snort zu erzeugen, starten wir den Nessus Client NessusWX auf dem Windows System. Wir melden uns am Nessus Dienst des Linux Systems an und tragen als Ziel unter "Target" die IP-Adresse 192.168.19.219 des Opfer-Rechners ein. Dann starten wir den Nessus Scan.

Sobald Nessus die verschiedenen Portscans ausführt, schlagen die definierten Swatch-Regeln an, diesmal beide, weil Nessus alle möglichen Portscans durchführt auf der Suche nach Schwachstellen im System. Die Gegenreaktion findet also zweimal statt, für jeden gefundenen Angriff einmal. Darin zeigt sich auch eine der Schwächen dieser Konstruktion, auf die im folgenden Kapitel näher eingegangen wird.

## 4.3 Einschränkungen

Zwar konnte mit den Beispielkonfigurationen gezeigt werden, wie sich eine automatisierte Gegenwehr realisieren lässt, allerdings wird die Verwendbarkeit dieser Lösung maßgeblich von ihren Einschränkungen bestimmt. Das Hauptproblem besteht dabei in der genauen Auswertung von Angriffen und Angriffsmustern. Um wiederholte Angriffe des selben Angreifers oder bestimmte Verhaltensmuster zu erkennen und gezielt darauf reagieren zu können, ist eine Verarbeitungslogik notwendig, für die es zur Zeit noch keine verfügbaren Programme gibt.

### 4.3.1 Gegenangriff ohne Zwischenrechner

Das Auffinden des Gegners ist einer der schwierigsten Prozesse der Gegenwehr. Lösungsvorschläge dazu wurden in den Kapiteln 2 und 3 vorgestellt. Doch befinden sich zusammgefasst die meisten Projekte dazu noch im Entwicklungsstadium, einen verbreiteten Lösungsansatz gibt es noch nicht.

Um den Angreifer automatisiert ausfindig zu machen, ist der direkt Weg naheliegend. Mit direkt ist gemeint, dass das Opfer den ersten Rechner angreift, von dem der Angriff festgestellt wurde. Mit dem Angriff wird versucht, in das System genauso einzudringen wie der Angreifer es zuvor getan hat, um von diesem System seinen Angriff auszuführen. Danach werden die

Logdateien dieses Systems untersucht. Stell sich anhand der Logdateien heraus, dass es sich nur um ein Zwischensystem handelt, das zum Angriff benutzt wurde, wird nach Informationen auf den Angreifer dieses Systems gesucht. Danach wird der Schritt mit dem dort gefundenen Angreifer wiederholt. Die Wiederholung läuft solange, bis das tatsächliche Angreifersystem ausfindig gemacht wurde.

Der Weg scheint auf den ersten Blick ein leicht zu automatisierender Weg zu sein, da es ein Vorgehen gibt, bei dem sich ein Schritt beliebig oft wiederholt. Dabei wäre allerdings sehr viel Intelligenz beim Durchsuchen der Logdateien und Auswerten der Daten nötig, um überhaupt Informationen über weitere angreifende Systeme erhalten zu können. Darüber hinaus ist der erste Schritt, nämlich sich den Zugang zum System und dessen Logdateien zu verschaffen, sehr komplex und mit heutigen Mitteln nur schwer automatisierbar. Man müsste eine Angriffsbibliothek systematisch durchgehen, bis die Schwachstelle des Systems gefunden ist, und dann die Analyse der Logdateien starten.

Außer Acht gelassen wir hier auch, dass auf einem der Zwischensysteme der Angriff in der Zwischenzeit bemerkt worden sein kann und sich der umgekehrte Weg des Angreifers möglicherweise gar nicht mehr zurückverfolgen lässt, weil auf einem der Zwischensysteme die benutzte Schwachstelle bereits geschlossen wurde.

### 4.3.2 Ping-Pong Angriffe

Bei steigender Verbreitung von Gegenangriffstools könnte der Fall auftreten, dass der Angriff von einem Rechner erfolgt, auf dem ebenfalls ein ID&RS mit Funktionalität zur Gegenwehr installiert ist. Führt nun das Opfersystem seinen Gegenangriff aus, würde das Angreifersystem ebenfalls darauf reagieren und einen erneuten Angriff starten. Durch den erneuten Angriff würde aber auch wieder das Gegenangriffssystem des ersten Opfers aktiv werden, und daraus würde eine Endlosschleife aus Angriffen und Angegriffenwerden entsehen, die sich nur programmtechnisch lösen lässt, indem Angriffe und Angriffsmuster über einen Zeitraum in einen Kontext gebracht werden können. Diese Endlosschleife ist vergleichbar mit einem Ping Pong Spiel, bei dem es immer abwechseln ein Agieren und Reagieren der Beteiligten gibt.

### 4.3.3 Gegenangriff ohne Intelligenz

Wie am Ping-Pong Problem zu sehen, fehlt den hier gezeigten Gegenangriffen jegliche Intelligenz, d.h. die Möglichkeit, in den Angriffen eine Struktur zu erkennen oder logisch Angriffe vom gleichen Angreifer miteinander zu verknüpfen. Es wäre aber wünschenswert, bestimmte logische Zusammenhänge bilden zu können:

- Es ist zwar möglich, auf verschiedene Angriffsmuster mit verschiedenen Reaktionen zu antworten, dies sollte aber erweitert werden durch die Möglichkeit, auch das gleiche Angriffsmuster mit verschiedenen Gegenangriffen zu beantworten, wenn sich ein Gegenangriff als zwecklos erweisen sollte. Zwecklos heißt hier, dass durch den Gegenangriff die Angriffsversuche des Angreifers nicht gestoppt und dem Angreifer auch kein Schaden zugefügt werden konnte.

- Um verschiedene Reaktionen auf den gleichen Angreifer und das gleiche Angriffsmuster zu ermöglichen, muss in einer Datenbank oder ähnlichen Struktur eine History über erfolgte Angriffe und Angriffsmuster mit den Daten des Angreifers gespeichert werden. Nur so lassen sich zusammengehörige Angriffe erkennen und eine gezieltere Reaktion programmieren.
- Durch eine weitere Auswertung dieser Datenbank lassen sich zu den Angriffsmustern auch Muster der Angreifer erstellen, d.h. man kann beobachten, ob bestimmte Angreifer nur zu bestimmten Zeiten angreifen oder nur bestimmte Angriffsmuster verwenden, und somit eine Art Angriffsprofil des Angreifers erstellen.

- Es müssen ebenfalls die verwendeten Gegenreaktionen in dieser Datenbank gespeichert werden, damit sich erkennen lässt, ob und wie oft ein Angreifer bereits mit einem Gegenangriff abzuwehren versucht wurde. So kann man verschiedene Gegenangriffe durchführen, bis der gewünschte Erfolg eingetreten ist. Außerdem lassen sich so Ping-Pong Angriffe feststellen und vermeiden.

### 4.3.4 Erkennen von zusammenhängenden Angriffen

Bei Snort werden bestimmte Angriffsmuster erst erkannt, wenn das dazugehörige Angriffsmusters oft genug vorgekommen ist, um eine bestimmte Schwelle zu überschreiten. Danach wird Snort jedes Paket dieses Angriffsmusters in seine Logdateien schreiben, damit der Angriff später besser nachvollzogen werden kann. Daraus ergeben sich aber Probleme bei der Art, wie der Gegenangriff in unserer Beispielkonfiguration durchgeführt wurde:

- Eine Gegenwehr wird ausgelöst, sobald ein bestimmtes Muster in der Logdatei gefunden wird. Diese Gegenwehr wird aber nicht nur beim ersten Auffinden des Musters, sondern bei jedem Vorkommen ausgelöst. Das bedeutet, dass ab dem Punkt, an dem Snort beginnt, den Angriff aufzuzeichnen, für jeden Logeintrag eine Gegenreaktion gestartet wird.
- Es muss also beispielsweise in einer Datenbank festgehalten werden, ob bereits ein Gegenangriff auf einen Gegner läuft, bevor ein neuer gestartet wird. Erst wenn der Gegenangriff beendet ist, darf ein neuer Gegenangriff gestartet werden.
- Dies funktioniert aber auch nur solange gut, wie Pakete desselben Angriffsmusters eintreffen. Verwendet der Angreifer beispielsweise verschiedene Angriffsmuster gleichzeitig, muss entschieden werden, ob man die Angriffsmuster, auf die wegen eines bereits laufenden Gegenangriffs nicht reagiert werden kann, einfach wegwirft, oder in einer Art Schlange sammelt und danach abarbeitet.
- Wenn ein Angriff sehr schnell, d.h. mit sehr vielen Angriffspaketen erfolgt, wird das zu verarbeitende Datevolumen schnell sehr groß. Wenn dann für jedes Paket erst geprüft werden muss, ob bereits ein Gegenangriff läuf, ergeben sich daraus Verzögerungszeiten, die die Systemperformance beeinträchtigen werden.

Offensichtlich ist der Aufwand, der für einen intelligenten und zielgerichteten Gegenangriff getrieben werden muss, ungleich höher als der Aufwand des Angreifers. Diese Tatsache kann wiederum zu Angriffszwecken ausgenutzt werden, indem der Angreifer gezielt so viele Angriffe

startet, dass das Gegenangriffssystem nicht mehr schnell genug darauf reagieren kann, weil es zu sehr mit dem Verarbeiten und Interpretieren der Pakete beschäftigt ist. Umso mehr Intelligenz wir dem System hinzufügen wollen, umso mehr wird sich durch die entstehende Komplexität diese Schwäche ausbreiten.

## 4.4 Ausblick

In den Beispielkonfigurationen wurden Testfilter erstellt, die bestimmte Angriffsmuster erkennen und als Reaktion darauf ein Programm ausführen konnten. Dabei ging es vor allem auch darum, zu zeigen, dass - wenn auch dem Programm wesentliche Eigenschaften fehlen, um einen Angriff intelligent zu beantworten - sich immerhin für verschiedene Angriffsmuster auch unterschiedliche Antworten konfigurieren lassen. Statt wie in unseren Beispielen für jedes Angriffsmuster eine eigene Reaktion zu definieren sollte stattdessen die gesamte Informationen, d.h. wer greift mit welchem Angriffsmuster an, an ein weiteres Programm übergeben werden, das dann eine Verarbeitung dieser Information sowie die Reaktion darauf vornimmt. Dieses Programm kann dann mit verschiedenen intelligenten Routinen ausgestattet sein, die einige der im vorigen Kapitel genannten Einschränkungen aufheben.

Es ist sehr wahrscheinlich, dass zukünftige Programme einen solchen Weg gehen werden, d.h. das Intrusion Detection System und das Gegenangriffs-System in zwei getrennten Komponenten zu belassen. Vor allem gibt es dafür folgende Gründe:

- Erstens ließe sich so sehr leicht im Laufe der Entwicklung die “Intelligenz” des Gegenangriffs-Systems austauschen, d.h. der Teil, der die Entscheidung über die Art des Gegenangriffs trifft und bereits erfolgte Angriffe auswertet.
- Zweitens wäre es zu komplex, alle Funktionalität in einem System zu vereinen. Wenn es Fehler oder Schwachstellen in diesem System gäbe, dann wäre bei einem erfolgreichen Angriff nicht nur das Intrusion Detection System, sondern auch das Gegenangriffs-System außer Kraft gesetzt.

Kapitel 2 und 3 haben ebenfalls in der Entwicklung befindliche Wege aufgezeigt, wie sich durch geeignete Verfahren und Protokolle der Verarbeitungsaufwand für ein angegriffenes System vermindern lässt, indem auf verteilte Informationen zugegriffen werden kann, die zugleich ein schnelleres Auffinden des tatsächlichen Angreifers ermöglichen.

# Literaturverzeichnis


Arkin, O. [2001], *ICMP Usage in Scanning, The Complete Know-How*, 3.0 edn, The Sys-Security Group.

Atkins, T. [n.d.], 'Swatch: The simple watcher'. http://www.oit.ucsb.edu/~eta/swatch/.

Caswell, B. [n.d.], 'Snort - the open source network ids'. http://www.snort.org.

*Common Intrusion Detection Framework* [n.d.]. http://www.isi.edu/gost/cidf/.

*Connecting to the Internet What Connecting Institutions Should Anticipate* [n.d.]. http://www.ietf.org/rfc/rfc1359.txt.

Costa-Requena, J. [2001], Recent development in ddos research. Seminar on Network Security.

Denning, D. E. [1998], *Information Warfare and Security*, Addison-Wesley, Bonn.

Deraison, R. [n.d.], 'Nessus - open source system security scanner'. http://www.nessus.org.

Frincke, D. & Wilhite, E. [2001], Distributed network defense, *in* 'Workshop on Information Assurance and Security — United States Military Academy', pp. 236–238.

Fyodor [n.d.], 'Nmap - free stealth port scanner for network exploration and security audits'. http://www.insecure.org/nmap/index.html.

*Internet Control Message Protocol* [n.d.]. http://www.ietf.org/rfc/rfc0792.txt.

*Intrusion Detection Message Exchange Format Working Group* [n.d.]. http://www.silicondefense.com/idwg/.

Kirhenshtein, V. [n.d.], 'Nessuswx home page'. http://nessuswx.nessus.org/.

Kossakowski, K.-P. [2000], *Information Technology Incident Response Capabilities*, Libri Books on Demand. ISBN 3-8311-0059-4.

Lange, D. B. & Oshima, M. [1998], *Programming and Deploying Java Mobile Agents with Aglets*, Addison Wesley.

*Management Information Base for Network Management of TCP/IP-based internets* [n.d.]. http://www.ietf.org/rfc/rfc1156.txt.

Muuss, M. J. [n.d.], 'The story of the ping'. http://ftp.arl.army.mil/~mike/.

NetworkAssociates [n.d.*a*], 'Active network intrusion detection and response'. http://www.nai.com/research/nailabs/adaptive-network/active-networks.asp.

NetworkAssociates [n.d.*b*], 'Automatic intrusion tracing and response'. http://www.nai.com/research/nailabs/adaptive-network/aitr.asp.

NetworkAssociates [n.d.*c*], 'Cyber cop scanner technical support'. http://www.sniffer.com/services/support/technical-support/supp-home.asp?pCode=CYS.

*OpenSSL Project* [n.d.]. http://www.openssl.org.

Project, H. [2002], *Know Your Enemy*, 1 edn, Addison Wesley.

*Requirements for Internet Hosts – Communication Layers* [n.d.]. http://www.ietf.org/rfc/rfc1122.txt.

*Requirements for IP Version 4 Routers* [n.d.]. http://www.ietf.org/rfc/rfc1812.txt.

Schnackenberg, D., Djahandari, K. & Sterne, D. [2000], Infrastructure for intrusion detection and response. DARPA Information Survivability Conference and Exposition (DISCEX).

Schnackenberg, D., Holliday, H., Smith, R., Djahandari, K. & Sterne, D. [2001], Cooperative intrusion traceback and response architecture (citra). DARPA Information Survivability Conference and Exposition II (DISCEX'01).

Service, B. W. S. [n.d.], 'Glossary of security and internet terms'. http://wssg.berkeley.edu/public/projects/SecurityInfrastructure/glossary.html.

Smith, J. [1999], *Distributed Computing with Aglets*. http://www.vistabonita.com/papers/DCAglets/.

Sobirey, M. [1999], *Datenschutzorientiertes Intrusion Detection*, Vieweg.

Tennenhouse, D., Garland, S., Shrira, L. & Kaashoek, M. [1996], 'From internet to activenet'. http://www.tns.lcs.mit.edu/publications/rfc96/.

*The Honeynet Project* [n.d.]. http://www.project.honeynet.org.

*Transmission Control Protocol* [n.d.]. http://www.ietf.org/rfc/rfc0793.txt.

van Wyk, K. R. & Forno, R. [2001], *Incident Response*, O'Reilly, Cambridge.

Wesslowski, B. [n.d.], 'Boris wesslowski's homepage - my software & projects'. http://www.kybs.de/boris/software.shtml.