Einbruchserkennung in Netzwerke

mit Intrusion Detection Systemen und Honeypots

Stefan Schumacher

<stefan@net-tex.de>, PGP: 0xB3FBAE33
http://www.net-tex.de/net/ids.html

5. März 2005

$Id: report.tex,v 1.19 2005/03/05 11:37:29 stefan Exp $

Zusammenfassung

Dieser Artikel entstand im Rahmen des Vortrags ‘‘Einbruchserkennung in Netzwerke mit Intrusion Detection Systemen und Honeypots’’ auf den Chemnitzer Linux-Tagen 2005. Er dient als Ergänzung zu meiner Präsentation, sollte aber auch für sich genommen genug Informationen enthalten.

Schlüsselwörter

Einbruchserkennung, IDS, Intrusion Detection System, Honeypot, mids, intrusion, evasion

Inhaltsverzeichnis

1 Einführung **1**
- 1.1 Wozu Einbruchserkennung? 1
- 1.2 Was ist/kann IDS nicht? 2

2 Prinzip der Einbruchserkennung **2**
- 2.1 Datensammlung 3
- 2.2 Angriffserkennung 4
  - 2.2.1 Integritätsprüfung 4
  - 2.2.2 Signaturerkennung 4
  - 2.2.3 Anomalieerkennung 5

3 Angriffe gegen IDS **5**
3.1 Integritätsprüfung 5
3.2 Signaturerkennung 6
3.2.1 Insertion 6
3.2.2 Evasion 7
3.3 Programme 7
3.3.1 AIDE 7
3.3.2 `verifiedexec(4)` 8
3.3.3 Portsentry 8
3.3.4 snort 8

4 Multilayer IDS (mIDS) **9**

5 Honeypot **9**
5.1 Analyse 10
5.2 Implementierungsprobleme bzw. Angriffsmöglichkeiten 11
5.3 Programme 13
5.3.1 LaBrea 13
5.3.2 honeyd 13

6 Fazit **13**

7 Literatur **15**

1 Einführung

1.1 Wozu Einbruchserkennung?

Das Gebiet der Einbruchserkennung ergänzt als Teil des Sicherheitskonzeptes andere technische Sicherheitsstrategien wie Zugangsschutz, Identifikation, Autorisierung, Datensicherung und soziale Maßnahmen (Vorschriften, Policies, Verträge ...). Die Einbruchserkennung ist notwendig bzw. sinnvoll, da in der Regel die o.g. Maßnahmen die Sicherheit als Prozess nicht komplett gewährleisten können bzw. nicht alle Angriffsszenarien abdecken oder gar selbst angegriffen und ausgehebelt werden können.

1.2 Was ist/kann IDS nicht?

Auch ein IDS bietet keine 100%-ige Sicherheit, da Sicherheit nun mal kein Zustand ist, der durch Installation eines Programms erreicht wird, sondern ein Prozess mit vielen veränderlichen Variablen.
Außerdem kann selbst ein installiertes IDS Angriffziel werden und den Zugang zu weiteren Systemen öffnen.

Idealerweise ist ein IDS in der Lage auch unbekannte neuartige Einbruchsszenarien zu erkennen und abzuwehren, aber es dürfte jedem klar sein das dies nur ein Wunschtraum bleiben wird und ein IDS ständige Pflege und Überwachung benötigt. Außerdem ist verständlicherweise ein tiefes Verständniss der zugrundeliegenden Protokolle und Anwendungen notwendig. Durch die steigende Komplexität der IT-Infrastrukturen und Sicherheitsmaßnahmen ist in größeren Netzen eine koordinierte Arbeit mehrerer Administratoren notwendig.

2 Prinzip der Einbruchserkennung

Ein System zur Einbruchserkennung[1] besteht im Prinzip aus drei Komponenten, die die jeweilige Funktionalität zur Verfügung stellen:

- **Datensammlung**
  Das System muss die anfallenden Daten sammeln und zur Analyse aufbereiten
- **Datenanalyse**
  Die angesammelten Daten müssen bezgl. Angriffe untersucht werden
- **Datenpräsentation**
  Bei entsprechenden Treffern ist eine Information der Administratoren notwendig.

Zusätzlich existieren noch sogenannte Intrusion Prevention Systeme[2], die anstatt der, oder zusätzlich zur, Präsentation Reaktionen starten um den Angriff abzuwehren. Diese System gleichen den IDS in den wichtigsten Teilen und sind lediglich um eine recht "simple"[3] Reaktionslogik erweitert, daher werde ich nicht gesondert darauf eingehen.

Weiterhin lassen sich IDS in weitere Klassen unterteilen, bspw. bezgl. der zeitlichen Reaktion in reaktive, also solche die nach einem Einbruch reagieren und proaktive, die während eines Einbruchs reagieren und versuchen diesen zu unterbinden. Außerdem existieren noch Postmortem Werkzeuge, die eine Analyse "toter" Systeme ermöglicht.

Außerdem kann man die Verteilung bzw. den Aktionsort der IDS einteilen in hostbasierte IDS (HIDS), die nur auf einem System arbeiten (bspw. Virenscanner) und netzwerkbasierte (NIDS) die auf verteilten Systemen in Netzwerken arbeiten oder in Hybridsystem, die Eigenschaften der NIDS und HIDS vereinen.

[1] Intrusion Detection System, IDS
[2] IPS
[3] wer schon mal 150 heterogene Netzwerkrechner verarzten musste, weiß wie simpel "simpel" ist ;-)

2.1 Datensammlung

Die Datensammlung ist ein technisches und ein soziales Problem, da ein IDS idealerweise alle anfallenden Daten sammeln und analysieren sollte, dabei aber an juristische und moralische Grenzen bezgl. des Datenschutzes stößt.

Nur weil ich alle eMails der Nutzer sammeln *kann*, heißt dies nicht auch das ich das tun[4] *muss*. Ebenso sind potentielle Angriffe die ins Leere laufen, bspw. gegen nicht installierte CGIs auf Webservern oder bestimmte Netzdienste, die nicht angeboten werden uninteressant und können getrost ignoriert werden.

Laut [IEEE17] steigt zwar die Zahl der Angriffe gegen IT-Systeme kontinuierlich an, dabei ist allerdings die schöpferische Höhe der Angriffe in den meisten Fällen eher gering. Insbesondere Angriffe mittels Skripte sind relativ leicht aufklärbar und können daher auch sehr leicht bekämpft werden. Prinzipiell ist es aber auch hier einfacher und sinnvoller die (vorhandenen) Angriffsvektoren auszumerzen.

Prinzipiell unterscheidet sich die Datensammlung bei HIDS und NIDS erheblich voneinander, da ein HIDS nur die relevanten Daten auf einem Host analysieren muss. Um beim Virenscanner als Beispiel zu bleiben, muss dieser auf seinem Host nur "gefährliche" Dateien scannen, also solche die schadhaften Code enthalten können. Durch die Anpassung an lediglich ein bekanntes System ermöglicht ein HIDS natürlich eine bessere Analyse der Vorgänge, ist aber bei Ausfall des Hosts ebenfalls inaktiv.

Das NIDS besteht aber in der Regel aus mehreren verteilten Monitoren im Netz und einer Managementkonsole, da Teile des Netzwerkverkehrs gesammelt werden müssen. Die Monitore arbeiten dabei bspw. auf verschiedenen Rechnern oder überwachen das Netz an strategischen Punkten (Gateway, DMZ, Switch) und werden von der Managementkonsole aus gesteuert bzw. wird von dieser aus die Datenanalyse und Datenpräsentation ausgeführt. Ein NIDS ermöglicht die Überwachung eines kompletten Netzes und erfordert dazu lediglich einige Monitore. Außerdem ist ein NIDS nicht von Angriffen/Ausfällen eines Zielsystems betroffen.

Problematisch ist hierbei eine geswitchte Umgebung, wenn die Switches bzw. die Infrastruktur nicht auf Überwachung (Monitoringports o.ä.) ausgelegt sind oder die Datenströme mittels Kryptographie gesichert werden. Zwar wäre es im zweiten Fall möglich mittels Key Escrow[5] den Verkehr vom IDS entschlüsseln und verarbeiten zu lassen, dies korrumpiert aber das gesamte Kryptosystem, da dessen Sicherheit nicht mehr gewährleistet werden kann. Möglich praktikabel ist es aber noch Schnittstellen beim Benutzer einzurichten, die bei dessem Zugriff auf die Chiffrate den Datenstrom analysieren. Hierbei sollte man genau abwägen ob der Einsatz des IDS die Kompromottierung des Kryptosystems rechtfertigt.

[4] vom technischen Sinn mal ganz abgesehen, Schwanzverlängerungsspam ist zwar nervig, aber i.d.R. nicht wirklich IDS-relevant

[5] Schlüsselhinterlegung

2.2 Angriffserkennung

Ein weites, und weitaus schwierigeres Feld als die Datensammlung ist die Datenanalyse. Auch hierbei existieren wieder verschiedene Vorgehensweisen:

- **Integritätsprüfung**
  Integrität der Daten wird geprüft
- **Signaturerkennung**
  bekannte Angriffssignaturen werden erkannt
- **Anomalieerkennung**
  vom Normalbetrieb abweichende Operationen werden erkannt

2.2.1 Integritätsprüfung

Vom technischen Standpunkt aus die einfachste Variante, Daten (i.d.R. Dateien) werden mit kryptographischen Signaturen oder Prüfsummen (PGP, SHA1) signiert und bei Bedarf gegen eine initiale Datenbank verglichen. Dies kann Postmortem oder reaktiv erfolgen, es wird also aufgeklärt welche Dateien verändert wurden, aber auch ein reaktives System ist möglich, bspw. `verifiedexec(4)` unter NetBSD, das bei jedem `exec()` im Kernel eine aktuelle Prüfsumme der Binärdatei mit einer alten Datenbank vergleicht und bei Fehlern die Ausführung der Datei verweigert.

Angriffe sind in der Regel nur gegen das kryptographische System (Passwort/Schlüssel) möglich.

2.2.2 Signaturerkennung

Ein System wie bspw. Snort analysiert den gesamten, oder zumindest relevante Teile des, Netzwerkverkehrs und überprüft den Verkehr auf Signaturen verschiedener Angriffe. Die Signatur enthält dabei bestimmte Auffälligkeiten oder Regelmäßigkeiten und Muster die einen Angriff kennzeichnen, z. B. bestimmte Pakettypen an bestimmte Ports oder markante Textstellen in Viren.

Angewendet werden in der Regel Pattern Matching Verfahren die den Verkehr analysieren, zusätzlich sind meist noch andere Verfahren nötig um die Daten vor der eigentlichen Analyse aufzubereiten, bspw. das Zusammensetzen fragmentierten Verkehrs.

Angriffe gegen diese Verfahren sind über Insertion und Evasion möglich.

2.2.3 Anomalieerkennung

Ein System der Anomalieerkennung überwacht das Verhalten der Benutzer, welches in einem Profil analysiert wird. Entsprechende Abweichungen vom

Profil können als Angriff gewertet werden, verwendet also bspw. eine Sekretärin die sonst nur Webbrowser, Emailprogramm und Textverarbeitung von 08:00 bis 16:00 Uhr benutzt auf einmal um 23:00 Uhr `nmap` und `gcc`, deutet dies auf eine Kompromittierung des Benutzerkontos hin, höchstwahrscheinlich wurde also in das System eingebrochen.

Ebenfalls möglich ist die manchmal als Bottleneck bezeichnete Methode der Alarmierung bei privilegierten Operationen, die normalerweise nicht von Benutzer ausgeführt werden dürfen. Gelangt ein Benutzer also an Rechte die ihm nicht zustehen (bspw. `wheel`) und kann so Programme aufrufen die er sonst nicht benutzen durfte (bspw. `su(1)`) alarmiert das System den Administrator.

Problematisch ist hierbei vor allem die faschistoide Überwachung der Benutzer, man sollte also nur Server oder wirklich wichtige Spezialsysteme damit sichern. Außerdem ist es fraglich ob sich ein relativ eindeutiges Profil für jeden Benutzer erstellen lässt. Weiterhin ist eine Implementierung des IDS nicht trivial, da es auch gegen Angriffe durch `root` geschützt sein muss.

3 Angriffe gegen IDS

Generell kann jedes IDS angegriffen werden, in dem seine Module attackiert werden und die "üblichen Verdächtigen" (Buffer Overflow, Format String, Privilege Escalation ...) ausgenutzt werden. Diese technischen Attacken gelten aber prinzipiell für jedes Programm und können mit bekannten Methoden (Privilege Separation, Kontrolle der Quellen) bekämpft werden.

Trotzdem gibt es noch Angriffe die sich gezielt gegen Methoden der IDS richten.

3.1 Integritätsprüfung

Die Sicherheit der Integritätsprüfung liegt im allgemeinen in der Sicherheit des verwendeten Kryptosystems, d.h. die verwendeten mathematischen Verfahren und deren Implementierung gelten als sicher.

Weiterhin ist ein ggf. verwendetes Passwort und/oder ein privater Schlüssel explizit zu schützen, d.h. man sollte den privaten Schlüssel nicht auf dem Produktivssystem belassen.

Bei Systemen die keine kryptographische Signatur sondern einfache Prüfsummen einsetzen liegt die Integrität des Systems in der Integrität der initialen Datenbank. Diese sollte hinreichend gesichert werden, in dem man sie bspw. ausdruckt, auf ein WORM Medium bringt oder mit einer kryptographischen Signatur explizit sichert.

3.2 Signaturerkennung

Aufgrund ihrer Natur und dem Einsatz in verteilten Systemen kann sich ein NIDS schenll zu einem Engpass in Systemen entwickeln und so bspw. auch über DDoS-Attacken ausgeschaltet oder umgangen werden.

Insertion und Evasion sind direkte Angriffe gegen die Signaturerkennungsmechanismen des IDS und versuchen hierbei die logische Reihenfolge des IDS auszuhebeln, in dem manipulierte Pakete in den Verkehr eingeschleust werden. Im Prinzip geht es also nur darum eine Angriffsignatur zu verschleiern, um zu verhindern daß das IDS sie erkennt.

3.2.1 Insertion

In der Insertion werden Pakete propagiert, die vom IDS, nicht aber vom Zielsystem akzeptiert werden. Das IDS akzeptiert also das manipulierte Paket, welches die Angriffssignatur tarnt und erkennt den Angriff somit nicht. Das Zielsystem allerdings dropt das manipulierte Paket und wird vom Angriff getötet.

Erläuterung der Insertionattacke an einem Beispiel:
Annahme:

- jedes Kästchen sei ein Paket
- das IDS akzeptiert Pakete mit einem *, das Zielsystem nicht
- die Pakete werden in der indizierten Reihenfolge vom IDS untersucht
- erhält das Zielsystem die Pakete H, A, L, T in genau dieser Reihenfolge stürzt es ab
- das IDS kann diesen Angriff verhindern

1.) Der Angreifer schleust folgende Pakete ins Netz ein:

$$\boxed{H_1}\ \boxed{X_2^*}\ \boxed{A_3}\ \boxed{L_4}\ \boxed{T_5}$$

2.) Das IDS untersucht und akzeptiert alle Pakete, liest also `H X A L T`
3.) es existiert keine Angriffssignatur zu `H X A L T`
4.) alle Pakete werden an das Zielsystem weitergeleitet
5.) das Zielsystem erhält alle Pakete, dropt aber X_2^*
6.) das Zielsystem erhält $\boxed{H_1}\ \boxed{A_3}\ \boxed{L_4}\ \boxed{T_5}$ und stirbt qualvoll

3.2.2 Evasion

Das umgekehrte Prinzip zur Insertion, das IDS dropt ein Paket, welches das Zielsystem akzeptiert, wird Evasion genannt.
Erläuterung der Evasionattacke an einem Beispiel:
Annahme:

- jedes Kästchen sei ein Paket
- das IDS akzeptiert Pakete mit einem * nicht, das Zielsystem schon
- die Pakete werden in der indizierten Reihenfolge vom IDS untersucht
- erhält das Zielsystem die Pakete H, A, L, T in genau dieser Reihenfolge stürzt es ab
- das IDS kann diesen Angriff verhindern

1.) Der Angreifer schleust folgende Pakete ins Netz ein:

$$\boxed{H_1}\ \boxed{A_2^*}\ \boxed{L_3}\ \boxed{T_4}$$

2.) Das IDS untersucht alle Pakete, dropt aber A_2^*, liest also `H L T`
3.) es existiert keine Angriffssignatur zu `H L T`
4.) alle Pakete werden an das Zielsystem weitergeleitet
5.) das Zielsystem erhält alle Pakete, akzeptiert aber A_2^*
6.) das Zielsystem erhält $\boxed{H_1}\ \boxed{A_2^*}\ \boxed{L_3}\ \boxed{T_4}$ und stirbt qualvoll

3.3 Programme

3.3.1 AIDE

AIDE ist ein reaktives, HIDS bzw. genauer gesagt ein Dateisystemintegritätsprüfer. AIDE erzeugt eine einfache Datenbank, in der Dateiname, Dateistatus und Prüfsumme über der Datei abgelegt werden. Sofern man diese initiale Datenbank entsprechend schützt (z. B. auf einem schreibgeschützten Medium oder verschlüsselt), kann man diese Datenbank erneut über das aktuelle System erzeugen lassen und mit der initialen Datenbank vergleichen. Weicht die Prüfsumme einer Datei ab wurde diese offensichtlich manipuliert.

Ziel des Systems ist es, insbesondere Rootkits zu enttarnen, diese Kits werden in der Regel nach einem erfolgreichen Einbruch installiert und dienen dazu die Anwesenheit eines Einbrechers zu verschleiern. Meist werden hierzu manipulierte Binaries wichtiger Systemprogramme installiert, um z. B. zu verhindern daß `ps -aux` die laufenden Prozesse des Einbrechers anzeigt. Nachteilig ist hierbei allerdings das AIDE reaktiv funktioniert, d. h. es gelingt dem Einbrecher das Rootkit zu installieren und er wird erst beim nächsten Durchlauf von AIDE enttarnt.

Ein Anleitung zur Installation und Konfiguration findet sich unter:
http://www.net-tex.de/unix/aide.html

3.3.2 `verifiedexec(4)`

Ähnlich wie AIDE, allerdings proaktiv, funktioniert `verifiedexec`. `verifiedexec` implementiert ab NetBSD 2.0 einen Prüfsummenvergleich in den `exec()`-Pfad des Kernels. Dazu wird ebenfalls eine initiale Datenbank (bspw. mit

`/usr/share/examples/veriexecctl/gen_sha1`) erzeugt und im Bootprozess via `veriexecctl(8)` in den Kernelspeicher geladen. Ruft man nun ein Binary auf, wird die Prüfsumme erneut gebildet und mit der Datenbank verglichen. Im Securitylevel 1 warnt der Kernel bei abweichender Prüfsumme vor Ausführung, im Level 2 wird die Ausführung des Binaries verweigert.
Eine englischsprachige Anleitung zu `verifiedexec(4)` findet sich in http://www.net-tex.de/unix/veriexec.html

3.3.3 Portsentry

Portsentry ist ein vergleichbar einfaches IDS, das dazu eingesetzt wird auf einem Server die Verbindungsversuche auf bestimmte Ports zu überwachen und so Portscans aufzuklären. Wurde ein (vermeintlicher) Portscan detektiert, kann Portsentry die `/etc/hosts.deny` anpassen und so die Quelle des Angriffs blockieren.
Portsentry lässt sich bequem aus pkgsrc heraus installieren und anschließend auch recht einfach konfigurieren. Es müssen lediglich `/usr/pkg/etc/portsentry.conf` und `/usr/pkg/etc/portsentry.ignore` angepast werden. Erstere definiert die zu überwachenden Hosts und die entsprechende Reaktion bei Angriffen, zweitere zu ignorierende IP Adressen/Subnetze. Insbesondere die `portsentry.conf` ist ausreichend kommentiert und bereits mit Beispielen gefüllt.

3.3.4 snort

Snort ist *das* NIDS schlechthin. Es ist unter der GPL als Open Source erhältlich und erfreut sich weltweit großer Beliebtheit. Es funktioniert als NIDS in dem es den Netzwerkverkehr an einem Netzwerkinterface mitschneidet und bezgl. Angriffsignaturen analysiert. Trifft eine Angriffssignatur, kann es über die Rules bestimmte Aktionen starten.

Die Aktualisierung der Signaturdatenbank ist unter Snort sehr wichtig, da eben nur bereits bekannte Angriffe erkannt werden können.

Der Einsatz und die Konfiguration eines Snort-IDS ist für größere Netze relativ komplex und würde den Rahmen des Artikels sprengen. Außerdem habe ich keine Lust die FAQ abzuschreiben ;-), daher verweise ich hier auf [sr1], [snof], [sno23], [reh03] und [groups,www].google.de

4 Multilayer IDS (mIDS)

Ein Problem der Systemüberwachung ist die Analyse verschiedenster Logdateien, die jeweils eine eigene Syntax und Semantik haben und nur bestimmte Teile der Infrastruktur überwachen. Die manuelle Analyse ist zeitaufwändig und fehleranfällig und ermöglicht nur sehr eingeschränkt das Erkennen von Verbindungen in den einzelnen Logs. Ein mIDS soll nun dazu dienen die verschiedenen Logdateiein in ein einziges Log zu aggregieren und ein Analyse

aller Logs auf Zusammenhänge zu ermöglichen. Dazu ist es notwendig die Syntax und Semantik der Rohlogs zu kennen und diese ohne Informationsverlust in ein wohldefiniertes Einheitsformat des mIDS zu überführen.
Vorteile sind die Automatisierung algorithmisierbarer Vorgänge, so wird der Administrator entlastet, indem er nicht jedes Log manuell analysieren muss. Weiterhin wird die Analyse erleichtert indem logische Zusammenhänge zusammenfassend dargestellt werden.
Notwendige Design- und Implementierungsschritte:

1. **Datensammlung:**
   Zusammführen aller notwendigen Daten aus verteilten Logs im mIDS
2. **Datenaufbereitung:**
   verlustfreie Konvertierung der Rohdaten in das mIDS-Datenformat,
3. **Analyse der aufbereiteten Daten:**
   Aggregation der Informationen um Beziehungen aufzudecken
4. **Alarmierung / Reaktion:**
   regelbasierte Analyse der Daten bezgl. Angriffe und Alarmierung bzw. auch Gegenmaßnahmen
5. **Präsentation der Ergebnisse:**
   Darstellung der Analyse, von der blinken LED bis zum atomaren Erstschlag

Prinzipiell lassen sich nach den ersten beiden Schritten (Sammlung & Aufbereitung) beliebige Technologien ansetzen, man kann die Daten bspw. statistisch oder topologisch analysieren.

5 Honeypot

Ein Honeypot (oder Honeynet, wenn ein komplettes Netz betrieben wird), ist ein System, das als Lockmittel für Angreifer dient. Es läuft auf einem extra dafür bereitgestelltem Server und bietet keine sinnvollen Dienste an. Seine Existenz ist den normalen Netzbenutzern nicht bekannt, so das im Prinzip schon jeder einfache Verbindungsversuch als Angriff gewertet werden kann. Zusätzlich simuliert der Honeypot Dienste oder Betriebsyssteme und versucht so den Angreifer aus sich zu lenken. Im Hintergrund werden die Aktionen des Angreifers protokolliert und ggf. analysiert.

Der Honeypot dient also in der Regel folgenden Zielen:

- Anlocken und Binden des Angreifers
- Erkennung und Analyse der Angriffe
- Schutz der Produktivsysteme als "Roter Hering"[6]
- juristische Beweissicherung

[6] bewusst ausgelegte falsche Fährte

Man möchte im Allgmeinen herausfinden wie ein Einbrecher in ein System eindringt und ob es neue Sicherheitslücken gibt. Hierzu ist es verständlicherweise notwendig das Hostsystem gut zu sichern und die Aktivitäten des Honeypots adäquat zu loggen und auszuwerten. Man kann die so gewonnen Erkenntnisse verarbeiten und Sicherheitsrisiken ausmerzen.

Weiterhin kann man einen Honeypot natürlich dazu benutzen von den Produktivsystemen abzulenken und diese so aus dem Blickfeld des Angreifers bringen. Einfache Honeypots gibt es als sogenannte Tarpit, das sind Programme die Portscanner (LaBrea) oder Mailspammer aufhalten sollen. Dazu werden bspw. offene Mailrelays emuliert, die bei Verbindungsaufnahme und versuchter Maileinlieferung die IP-Adresse auf den echten Mailservern blockt. Mehr dazu in [grun05]

Da eigentlich ja schon eine einfache Kontaktaufnahme (telnet oder scan) zum Honeypot Angriffsabsichten verrät, enttarnt sich der Angreifer somit selbst und man kann geeignete Gegenmaßnahmen einleiten, während der Angreifer seine Zeit damit zubringt in ein, für ihn, nutzloses System einzudringen und dabei analysiert wird.

Wichtig ist, daß es dem Angreifer nicht gestattet wird den Honeypot für weitere Angriffe zu missbrauchen, daher sollte er im Netz entsprechend abgegrenzt werden und idealerweise mit einem Not-Aus ausgestattet werden, das den Hopeypot deaktiviert und so weiteren Schaden abwendet.

5.1 Analyse

Die Daten, die ein Honeypot abwirft, können beachtliche Ausmaße annehmen und sollten daher nach bestimmten Kriterien untersucht werden. Interessante Analysepunkte sind bspw.:

- Angriffsquelle(n)
  woher stammt der Angriff? (IP-Adresse, Region (geoiplookup))
  wieviele Angreifer?
- Ausrüstung des Angreifers
  Betriebssystem, -version und Architektur mittels passivem Fingerprinting (p0f, disco)
- well known attacker?
  ist die Angriffsquelle für Angriffe bekannt? (Blacklists)
- sequentielle/parallele Angriffe?
  wurden die Angriffe gleichzeitig oder nacheinander ausgeführt?

Wichtige Orte der Analyse sind:

- Netzwerkverkehr
  Mitschnitte des Honeypots, Gateways, Switch/Bridge
- Logdateien
  Logdateien relevanter Systeme wie Gateways, Firewalls, NIDS, physischen Systemen

- Systemplatten des physischen Honeypots
  Dateisystemintegrität prüfen und ggf. Manipulationen untersuchen

Honeypots lassen sich in verschiedene grobe Kategorien einteilen:

- **Tarpits/Traps**
  simulieren Netzwerkdienste (bspw. offene Ports oder Mailserver) und behindern den Angreifer indem sie die Verbindung unnötig lange offenhalten. Bsp: LaBrea, Portsentry
- **physisches Sytem**
  ein einfacher Rechner fungiert als äußerlich normales System, wird aber überwacht
- **logisches System**
  ein physisches System emuliert ein (1:1) oder mehrere (1:n) Systeme, bspw. mit honeyd oder NetBSD/xen

Ist ein beliebiger Honeypot installiert, ist es ratsam ihn vor dem Produktiveinsatz zu testen, dazu kann man beliebige Angriffsszenarien durchspielen, bspw. in dem man ein Angriffsteam (Samurai) anmietet oder bekannte Schwachstellenscanner (SATAN, Nesus, Saint, Sara, ADMsmb) einsetzt.

Überwachen lässt sich ein logischer Honeypot am einfachsten, indem der physische Host die logischen System überwacht, bspw. in dem XEN unter NetBSD oder Linux oder UserModeLinux eingesetzt wird. Weiterhin ist es ratsam den gesamten anfallenden Netzverkehr zum System mitzuschneiden, bspw. mit dem guten alten tcpdump(8) (siehe [sch03]) oder einem Gateway-IDS wie hogwash. Ebenfalls nützlich ist es den Netzwerkvehr zum passiven Fingerprinting zu verwenden, in dem Programme wie p0f den Datenverkehr analysieren und anhand von Signaturen das OS des Angreifers passiv identifizieren. Passiv heißt in diesem Fall, das kein Portscan oder ähnliches Verfahren gegen den Angreifer eingesetzt wird, da ihn dies verschrecken könnte, sondern einfach der sowieso von ihm erzeugte Netzwerkverkehr analysiert wird.

5.2 Implementierungsprobleme bzw. Angriffsmöglichkeiten

Ein Honeypot lebt wie ein Scharfschütze von der Tarnung. Findet der Angreifer heraus, das er an einen Honeypot geraten ist, wird er den Einbruchsversuch abbrechen und der Honeypot ist nutzlos geworden, daher ist es das höchste Ziel einen Honeypot zu erschaffen, der sich wie ein reales System verhält und somit nicht mehr enttarnbar ist. Leider steigt mit der Realitätsnähe des Honeypots auch die Implementierungshöhe und die Wartbarkeit des Systems.

Ein sehr guter Honeypot wäre ein echtes System, das als Roter Hering ausgelegt wird, da dies dem Angreifer nicht auffallen würde. Nützlich ist dieser Honeypot für den Administrator aber nur, wenn er in der Lage ist ihn idealerweise komplett zu überwachen. Dies ist bei einem normalen System recht problematisch, man kann zwar den Netzwerkverkehr mitschneiden, verwendet

der Einbrecher aber selbst Krypotgraphie wie SSH nutzen die Aufzeichnungen nichts. Daher wäre es eine weitere Möglichkeit den Kernel dahingehend zu manipulieren bestimmte Ereignisse (`read()`, `execute()`) mitzuloggen. Nun entstehen aber zwei weitere Probleme:

- Wie bekomme ich die geloggten Daten sicher und unauffällig auf einen anderen Host?
- Wie verhindere ich die Enttarnung der Loggingmechanismen?

Beide Probleme sind nichttrivial, da man die Logdaten zwingend benötigt, sind sie doch der Hauptgrund für eine Installation des Honeypots. Eine Übertragung der Daten über Netzwerk, selbst verschlüsselt, könnte das Misstrauen des Einbrechers erregen und die Tarnung auffliegen lassen, einen Nadeldrucker mitlaufen zu lassen wäre zwar möglich, dazu muss aber der Druckprozess sehr gut versteckt werden.
Das zweite Problem ist auch nichttrivial zu lösen, da ein solcher Logmechanismus im Kernelspace die Systemlast hochtreibt. Logt der Kernel alle `read()`-Aufrufe mit, entsteht eine sehr hohe Systemlast, die sofort auffällt. Dies kann sehr trivial mit `dd` und `ping` geschehen:

```
dd if=/dev/zero of=/dev/null bs=1
```

erzeugt sehr viele Leseaufrufe, die bei einem anschließenden

```
ping 127.0.0.1
```

die RTT signifikant ansteigen lassen.

Die Uberwachungsprobleme lassen sich durch den Einsatz virtueller Maschinen umgehen, also ein physisches System emuliert $1, ..., n$ logische Systeme. Das physische System übernimmt die Loggingfunktionen und ist solange vertrauenswürdig, bis es selbst kompromittiert wurde. Die Tarnung der logische Systeme hängt hier von der Realitätsnähe der Implementierung ab:

- gibt die Emulation real existierende Fehler des emulierten Systems wieder?
- Differenzen zwischem echten Datenverkehr und emuliertem? (Router, Traffic)
- MAC-Adressen der NICs?

Tarpits erzeugen oftmals keinen Traffic, so simuliert FakeAP zwar WLAN-Access Points, kann für diese aber keinen Traffic generieren, so daß diese gefälschten APs leicht erkennbar sind. Ebenso öffnet LaBrea als Tarpit zwar eine TCP/IP-Verbindung zum Angreifer, setzt danach aber die Windowsize auf 0, so das der Angreifer keine Daten mehr übertragen kann. Dieses Verhalten ist deterministisch und lässt daher eine solche Tarpit leicht enttarnen.

5.3 Programme

5.3.1 LaBrea

Im Prinzip so etwas wie Portsentry im Netz, LaBrea kann dazu virtuelle Hosts mit echten IP Adressen erstellen und Verbindungen an diese durch manipulierte IP-Pakete offen halten und verzögern. Dadurch werden Angreifer (z.B. Würmer, die das Netz durchsuchen oder Portscans) verlangsamt und im Idealfall sogar aufgehalten.

5.3.2 honeyd

honeyd von Niels Provos emuliert logische Hosts oder komplette Netze auf einem physischen System und ist in der Lage zu jedem emulierten Port eine bestimmte Reaktion auszuführen. Weiterhin ermöglicht es bestimmte Reaktionen auf Verbindungen, z.B. ein RESET oder ein Skript, das einen Webserver simuliert.

Dokumentation und Konfigurationshinweise zu honeyd finden sich auf http://www.honeyd.org/.

Ein Konfigurationsbeispiel:

```
create pit
set pit personality "Microsoft Windows 98SE"
set pit default tcp action tarpit open
set pit default udp action block
bind 192.168.1.1 pit
```

Es wird eine Konfiguration namens “pit” erstellt, die ein Windows 98SE System emuliert und alle TCP-Ports öffnet, aber als tarpit den Verkehr darauf verzögert. UDP Ports werden blockiert und abschließend wird diese Konfiguration an der Adresse 192.168.1.1 gebunden.

6 Fazit

IDS sind ein sehr effizientes Mittel um die Sicherheit eines Netzwerkes zu erhöhen. Prinzipiell gilt aber auch hier, das es einfacher und besser ist die Angriffsfläche zu minimieren und alle weiteren Maßnahmen zur Absicherung des Netzes zu ergreifen.

Ebenso sind alle Arten von Honeypots ein weiteres effizientes Mittel um Angreifer zu binden und zu analysieren, außerdem ist diese Methodik sehr gut geeignet um Angreifer aus dem Inneren abzulenken und zu enttarnen.

Soll ein IDS und/oder Honeypot eingesetzt werden, sind entsprechende vorbereitende Maßnahmen notwendig, da ein solches Projekt alle Methoden der Informatik erfordert und sehr schnell eine außerordentliche Komplexität erreichen kann.

7 Literatur

[IEEE17] ALLEN, Julia; CHRISTIE. Alan; MCHUGH, John
CERT Coordination Center
Defending Yourself: The Role of Intrusion Detection Systems
"IEEE Software" #17 Sept/Oct 2000

[ein04] EINWECHTER, Nathan
Multi-Layer Intrusion Detection Systems
http://www.securityfocus.com/infocus/1788

[oh1-1] OUDOT, Laurent; HOLZ, Thorsten
Defeating Honeypots: Network Issues, Part 1 & 2
http://www.securityfocus.com/infocus/1803
http://www.securityfocus.com/infocus/1805

[sch04] SCHUMACHER, Stefan
Einführung in kryptographische Methoden
http://cryptomancer.ath.cx/~stefan/21c3/21c3-kryptograhie-paper.pdf

[sch03] SCHUMACHER, Stefan
Spass im Netzwerk mit tcpdump & Co.
http://www-e.uni-magdeburg.de/steschum/tcpdump.pdf

[grun05] GRUNWALD, Lukas
Aufbau und Betrieb von Honeypot-Systemen
http://www.linuxdays.lu/agenda/lxd2005_security_tutorial/Attachment00089954/lxd2005_honeypot.pdf

[niels04] PROVOS, Niels
A Virtual Honeypot Framework
USENIX Security 2004
http://www.citi.umich.edu/u/provos/papers/honeyd.pdf

[pic03] PICKETT, Mark
A Guide to the Honeypot Concept
GSEC Practical v1.4b
http://www.giac.org/practical/GSEC/Mark_Pickett_GSEC.pdf

[PoDa03] POUGET, F.;DACIER, M.
Honeypot-based Forensics
http://www.eurecom.fr/ pouget/papiers/AusCERT_fullpaper_BIS.pdf
Institut Eurecom

[reh03] REHMAN, Rafeeq Ur
Intrusion Detection Systems with Snort
http://phptr.com/content/images/0131407333/downloads/0131407333.pdf
Prentice Hall PTR, 2003

[HJS03] HESS, A.; JUNG, M.; SCHÄFER, G.
Combining Multiple Intrusion Detection and Response Technologies

in an Active Networking Based Architecture
http://www.tkn.tu-berlin.de/publications/papers/dfn03.pdf
TU Berlin

[Car03] CARLO, Corbin Del
Intrusion detection evasion: How Attackers get past burglar alarms
http://cnscenter.future.co.kr/resource/security/ids/1284.pdf
SANS Institute 2003

[sno23] ROESCH, Martin; GREEN, Chris; Sourcefire Inc.
SnortUsers Manual 2.3.0
http://www.snort.org/docs/snort_manual/

[snof] The Snort Core Team
The Snort FAQ
http://www.snort.org/docs/faq.pdf

[Sct02] SCOTT, Steven
Threat Management: The State of Intrusion Detection
http://www.snort.org/docs/threatmanagement.pdf

[pn98] PTACEK, Thomas; NEWSHAM, Timothy
Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection
http://www.snort.org/docs/idspaper/

[sr1] SPENNEBERG, Ralf
Sensoren im Netzwerk
Linux-Magazin Security Edition
http://www.spenneberg.com/Snort(SecurityEdition)/2134.html