

# Generierung von Intrusion-Detection-Konfigurationen aus Firewall-Regeln

Michael Mai  
Technische Universität München  
maim@in.tum.de

15. März 2004

## **Zusammenfassung**

Ziel dieser Arbeit war es ein Tool zu entwickeln, welches aus einer Konfiguration einer Firewall Policy-Skripte für ein Intrusion-Detection-System (IDS) ableitet. Dieses Projekt beschränkt sich auf eine Linux-basierende `filter`-Tabelle einer `iptables`-Firewall und das ebenfalls für Linux entwickelte IDS `Bro`. Das Tool ist erweiterbar für andere Firewalls und ID-Systeme. Die erzeugten Skripte können zusammen mit anderen Policies in `Bro` geladen werden, um einen umfangreicheren Schutz durch Überprüfung der Firewall-Tätigkeit, Warnung vor Veränderung der Firewall-Konfiguration, sowie durch Konfigurationsfehler der Firewall selbst, zu gewährleisten.

# Inhaltsverzeichnis

<b>1</b>	<b>Einführung</b>	<b>3</b>
<b>2</b>	<b>Voraussetzungen</b>	<b>3</b>
2.1	Eingesetzte Software . . . . .	3
2.2	Konfiguration der <code>iptables</code> -Firewall . . . . .	3
2.3	Topologie des Netzwerks . . . . .	4
2.4	Konfiguration des Intrusion Detection Systems Bro . . . . .	5
<b>3</b>	<b>Benutzung des Skripts <code>ipt2bro</code></b>	<b>5</b>
3.1	Das Skript <code>ipt2ids.pl</code> . . . . .	6
3.2	Das Skript <code>fire2bro.pl</code> . . . . .	6
<b>4</b>	<b>Umsetzung der Firewall-Regeln</b>	<b>7</b>
4.1	Ablaufdiagramm eines Pakets . . . . .	7
4.2	TCP-Regeln . . . . .	7
4.3	UDP-Regeln . . . . .	8
4.4	ICMP-Regeln . . . . .	9
4.5	Selbstdefinierte Chains . . . . .	9
4.6	Targets . . . . .	10
<b>5</b>	<b>Vorschläge zur Weiterentwicklung</b>	<b>10</b>
5.1	<code>connection-state</code> von <code>iptables</code> . . . . .	10
5.2	Target <code>MIRROR</code> . . . . .	11
5.3	Erweiterbarkeit . . . . .	11
<b>6</b>	<b>Literaturverzeichnis</b>	<b>11</b>

# 1 Einführung

Kaum ein Netzwerk läuft heutzutage noch ohne den Schutz einer Firewall. Die wohl verbreitetste Art von Firewalls stellen Paketfilter dar. Diese überprüfen einzelne Pakete anhand von zuvor festgelegten Regeln auf Zulässigkeit. Anschliessend wird das Paket weitertransportiert oder verworfen. Da derartige Firewalls jedoch nur beschränkten Schutz gegenüber Angriffen höherer Schichten oder getunnelten Verkehrs bieten können, werden häufig zusätzlich Intrusion-Detection- bis hin zu Intrusion-Prevention-Systeme eingesetzt. Diese sind in der Lage bekannte Verfahrensmuster von Angriffen, sowie annormale Verhaltensweisen eines Hosts, zu erkennen (Detection) oder darauf zu reagieren (Prevention). Wird ein Angriff als solcher spezifiziert, kann unterschiedlich verfahren werden. Es kann ein Alarm generiert werden, der die Administratoren auf diesen Vorfall hinweist, es kann die Verbindung unterbrochen werden oder es wird ein Gegenangriff gestartet. Um die Chancen einen Angriff richtig zu erkennen, ist es sinnvoll mehrere Komponenten zum Schutz einzusetzen, die sich auch gegenseitig überwachen. Nur so kann ein ausreichender Schutz gegen infiltrierte und manipulierte Software gewährleistet werden. Aus diesen Gründen wurde in dieser Arbeit ein Tool entwickelt, welches es dem IDS Bro ermöglicht, den unter Linux eingesetzten Paketfilter `iptables` auf Zuverlässigkeit zu überprüfen. Die Schwierigkeit lag darin, die paketorientierte Firewall in Bro umzusetzen, das nicht jedes Paket, sondern ganze Verbindungen analysiert.

## 2 Voraussetzungen

Damit die erstellten Policies für Bro eine semantisch korrekte Übersetzung der Firewall-Regeln darstellen, müssen einige Voraussetzungen getroffen werden, die in den nachfolgenden Abschnitten näher erläutert werden.

### 2.1 Eingesetzte Software

Dieses Tool wurde unter `Suse Linux 8.0 Professional` mit der Bro-Version `0.8a48` entwickelt. Zusätzlich wurde es unter `RedHat-` und `Debian-Linux` getestet. Das Programm sollte aber mit allen aktuelleren Linux-Distributionen zusammenarbeiten.

### 2.2 Konfiguration der `iptables`-Firewall

Es konnten nicht alle Fähigkeiten der `iptables`-Firewall nachgebildet werden. Gründe hierfür waren, die fehlende Möglichkeit in Bro effizient an diese Informationen zu gelangen, keine Synchronisation zwischen Firewall und IDS oder der Aufwand diese zu implementieren hätte den Umfang des Projekts übertraffen. Es folgt eine Liste der Features, die nicht oder nur teilweise unterstützt werden:

- Fragmentierte Pakete
- Flags (teilweise, nur SYN und ACK)

- MAC-Adresse
- Type of service (TOS)
- Time to live (TTL)
- Markierte Pakete (mark)
- TCP-option
- unclean
- limit
- owner
- state INVALID
- Target: MIRROR

Für die korrekte Erkennung von FTP-Verbindungen auf der Firewall durch den `state RELATED`, muss das `ip_conntrack_ftp`-Modul vorher geladen werden:

```
fw:~ # modprobe ip_conntrack_ftp
```

Wird dieser `state` benutzt ohne das Modul zu laden, kann es zur fehlerhaften Erkennung von FTP-Datenverbindungen kommen. Sind weitere Optionen aus dieser Liste gesetzt, so werden sie ignoriert, aber eine entsprechende Meldung oberhalb der jeweiligen Regel in das Policy-Skript geschrieben. Man sollte deshalb unbedingt die Ausgabe einer solchen Regel kontrollieren. Die Folge kann eine falsche Kategorisierung des Pakets sein und somit ein falscher oder gar fehlender Alarm.

## 2.3 Topologie des Netzwerks

Zur Vermeidung von falsch klassifizierten Verbindungen der `FORWARD-Table`, darf die Firewall nur einen Link zu den Zielnetzen und zum Internet besitzen. Eine gültige Topologie ist in Abbildung 1 zu sehen.

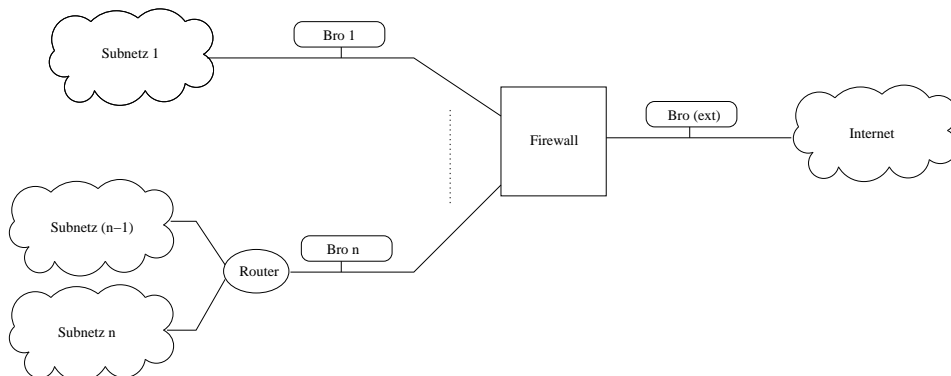


Abbildung 1: Topologie für Bro

## 2.4 Konfiguration des Intrusion Detection Systems Bro

Eine Eigenheit des Policy-Skripts `ftp.bro` verhindert, dass andere Skripte auf die Anfrage, ob eine Verbindung eine FTP-Datenverbindung ist, die korrekte Antwort erhält. Um dies zu beheben müssen die nachstehenden Zeilen 233 und 234 der Datei `ftp.bro`

```
delete ftp_data_expected[id$resp_h, id$resp_p];
delete ftp_data_expected_session[id$resp_h, id$resp_p];
```

entfernt oder auskommentiert werden.

Ausserdem müssen lokale Netze in Bro definiert werden. Diese Konfiguration ist abhängig von der Position von Bro. Es dürfen nur die lokalen Netze als solche definiert werden, von denen Bro den Netzwerkverkehr abhört. Für die `n`-te Instanz von Bro aus Abbildung 1 würde man ein neues Policy-Skript nach dem folgenden Beispiel anlegen und zusammen mit den anderen Skripten für das jeweilige Interface laden:

```
@load conn
# Subnetz (n-1): 128.1.0.0/16
# Subnetz (n-2): 128.2.0.0/16
redef local_nets=[128.1.0.0/16,128.2.0.0/16];
```

Für `Bro(ext)` müssen alle Subnetze (`1..n`) als lokal definiert werden. Diese Topologieinformation ist nötig, um einerseits die Position von Bro und andererseits die Aktion eines Pakets zu bestimmen.

## 3 Benutzung des Skripts `ipt2bro`

Die Ausführung, sowie die Parameterübergabe erledigt das Bashskript `ipt2bro`. Läuft das IDS Bro auf dem Firewall-Rechner selbst, müssen keine Parameter verwendet werden. Soll Bro an einem Netzwerksegment mithören, das an einem Interface der Firewall hängt, so muss die Konfiguration der Firewall und die Konfiguration der Netzwerkkarten dieses Rechners an das Skript übergeben werden. In diesem Fall müssen auf dem Firewall-Rechner folgende Befehle ausgeführt werden:

```
fw:~ # ifconfig > fw-ifconfig.txt
fw:~ # iptables -L -v -n > fw-config.txt
```

Nun ruft man das Skript `ipt2bro` mit folgenden Parametern auf:

```
bro:~ # ./ipt2bro -i fw-ifconfig.txt -f fw-config.txt
```

Das Skript gibt eine Statusmeldung über seine Konfiguration aus. Die Ausgabe sollte unbedingt kontrolliert werden, v.a. ob die Namen der Netzwerkkarten und die dazugehörigen IP-Adressen richtig sind, da sonst Fehler in der Umsetzung auftreten können.

Der zusätzliche Parameter `-d` bewirkt, dass alle Verbindungen zu einem Eintrag in das Logfile führen.

Alle angegebenen Parameter werden bei der Ausführung an die Perl-Skripte `ipt2ids.pl` und `fire2bro.pl` weitergereicht. Diese werden in den folgenden zwei Abschnitten erklärt.

### 3.1 Das Skript `ipt2ids.pl`

Das Skript `ipt2ids.pl` formatiert die Ausgabe von `iptables` neu. Es korrigiert unsinnige oder zur Weiterverarbeitung ungeeignete Angaben ohne jedoch Regeln zu löschen. Zu diesen Angaben gehören

- leerer `state`
- leeres Targetfeld

Diese Regeln werden zwar an `fire2bro.pl` weitergereicht, dort jedoch ignoriert. Zusätzlich wird der `state` als eigenes Feld deklariert. Alle Felder die keiner Auswertung bedürfen, werden mit "\*" belegt. Das nachfolgende Textsegment zeigt einen Ausschnitt aus einer Ausgabe von `iptables`:

```
...
pkts  bytes  target  prot  opt  in  out  source  destination
0      0      tcp     --   *   *   0.0.0.0/0  0.0.0.0/0  tcp dpt:22
0      0      ACCEPT  tcp  --   *   *   0.0.0.0/0  0.0.0.0/0  state NEW
...
```

Diese Eingabe führt zu folgender Ausgabe:

```
...
target  prot  opt  in  out  source  destination  state  options
*      tcp  *   *   *   *       *       *       *       dpt(s) 22
ACCEPT  tcp  *   *   *   *       *       *       N
...
```

Zur Umwandlung in Policy-Skripte für Bro wird diese Ausgabe in die Pipe des Skriptes `fire2bro.pl` umgeleitet.

### 3.2 Das Skript `fire2bro.pl`

Die Hauptarbeit übernimmt das Skript `fire2bro.pl`. Es liest die Netzwerkkonfiguration ein und erstellt für jedes Interface drei Dateien. Die Datei `eth0.bro` für den Adapter `eth0` ist das Policy-Skript, welches beim Start von Bro geladen werden muss. Es setzt den Capture-Filter, also welche Protokolle analysiert werden sollen, auf ( `TCP or UDP or ICMP`). Zudem werden neue Alarme deklariert. Der erste zeigt an, wenn Fehler im Skript aufgetreten sind. Der zweite Alarm tritt auf, sobald Unterschiede zwischen Konfiguration und Aktion der Firewall erkannt werden. In dieser Datei ist die Funktion `exec_action(...)` deklariert. Dieser wird zur Laufzeit das Ereignis eines Pakets, das auf der Firewall ausgeführt wurde bzw. wird, mitgeteilt. Die Funktion entscheidet dann, ob das Paket in dem beobachteten Segment auftreten darf und generiert daraufhin einen Alarm, einen Log-Eintrag oder nichts, falls eine Regel existiert, die dieses Paket akzeptiert. Falls in der Firewall-Konfiguration Chains selbst definiert wurden, so sind diese ebenfalls in dieser Datei als Funktion eingetragen.

Die anderen Dateien (im Beispiel: `eth0.input.bro` und `eth0.output.bro`) beschreiben alle Regeln, die eingehenden Verkehr, bzw. ausgehenden Verkehr betreffen. D.h. alle `INPUT`-Regeln und `FORWARD`-Regeln, in deren Menge von `INPUT`-

Interfaces `eth0` enthalten ist, werden in `eth0.input.bro` übernommen. Analog dazu ist die Datei `eth0.output.bro` aufgebaut.

## 4 Umsetzung der Firewall-Regeln

In diesem Kapitel soll die Umsetzung der Regeln, sowie die Implementierung in `Bro` genauer betrachtet werden. Die nachfolgenden Abschnitte erläutern den allgemeinen Ablauf einer Paketüberprüfung, sowie die Analyse der unterschiedlichen Protokolle. Danach erklärt ein weiterer Abschnitt, wie selbstdefinierte Chains umgesetzt wurden.

### 4.1 Ablaufdiagramm eines Pakets

Nachfolgende Abbildung zeigt den Aufbau der Dateien und den Ablauf, sobald ein Event generiert wurde. Als erstes wird überprüft, in welche Chain (`INPUT`, `OUTPUT`, `FORWARD`) des Paketfilters es eingeordnet werden kann. Danach werden die Funktionen `tcp/udp/icmp_eth0.input/output` aufgerufen, um das Paket mit den Regeln zu vergleichen. Entweder liefert diese Funktion direkt das anzuwendende Target oder es werden weitere selbstdefinierte Chains aufgerufen. Nachdem das Event die Nummer des Targets erhalten hat, wird die Funktion `exec_action` aufgerufen, die sich nun um die Ausführung des Targets kümmert.

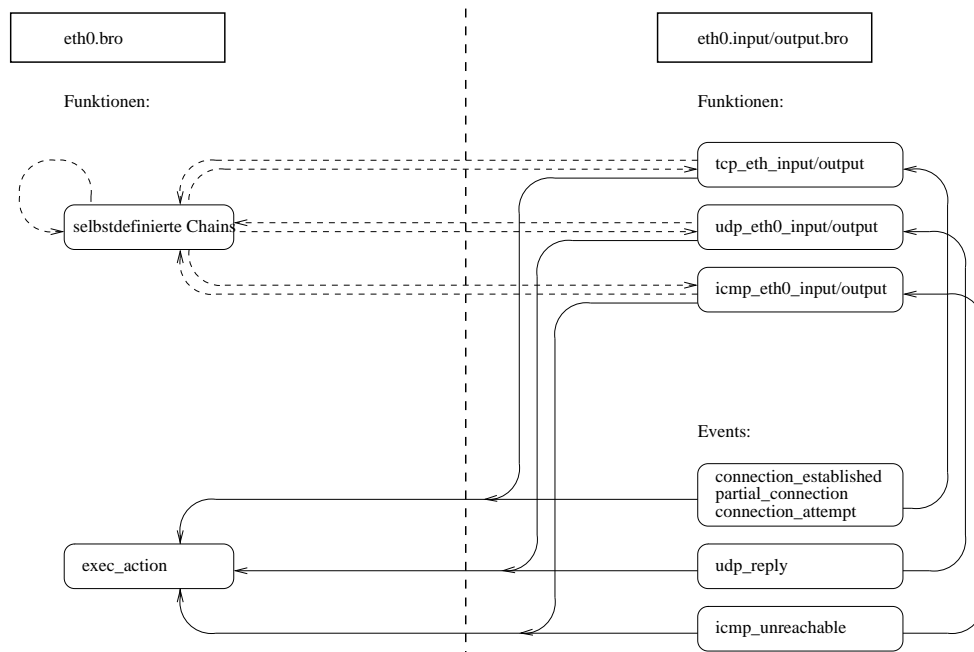


Abbildung 2: Ablaufdiagramm in `Bro`

### 4.2 TCP-Regeln

Die Umsetzung der TCP-Regeln basiert auf drei Events in `Bro`. Für Verbindungen, bei denen der `Three-Way-Handshake` beobachtet wird, wird das Event

`connection_established` verwendet.

Bei denjenigen, wo der Verbindungsaufbau nicht mitverfolgt wurde, also Verbindungen, die bereits vor dem Start von Bro existierten, wird das Event `partial_connection` aufgerufen.

In die letzte Kategorie fallen zurückgewiesene Verbindungen. D.h. es wird nur ein SYN-Paket versendet, allerdings erhält der Sender keine Antwort oder ein `destination-unreachable`-Paket zurück. Für diesen Fall existiert in Bro das Event `connection_attempt`. Die umgesetzte Firewall unterstützt für TCP folgende Optionen:

- Source- und Destination-Ports
- Multiport-Option
- state NEW,ESTABLISHED,RELATED
- Flags SYN,ACK

Diese Optionen werden zusätzlich zu den Bedingungen einer Regel (Protokoll, Chain, IP-Adressen) in `tcp_eth0_input/output` eingebaut, welche das Paket auf Übereinstimmung prüft. Bei etablierten TCP-Verbindungen werden immer vier Pakete getestet:

- SYN-Paket des Originators (state NEW)
- SYN,ACK-Paket des Responders (state ESTABLISHED)
- ACK-Paket des Originators (state ESTABLISHED)
- ACK-Paket des Responders (state ESTABLISHED)

Das bedeutet, es wird ebenfalls überprüft, ob über diese Verbindung Daten ausgetauscht werden dürfen, auch wenn dies nicht genutzt wird. Das lässt sich jedoch in einer Verbindungsanalyse nicht unterscheiden.

### 4.3 UDP-Regeln

Das Event `udp_reply` wurde definiert, um UDP-Pakete zu testen. Damit die mitgeführte Tabelle nicht übermässig gross wird, werden nur Pakete analysiert, die eine Antwort erhalten. Es gibt somit drei Arten von Paketen:

- UDP-Request mit state NEW
- UDP-Reply mit state ESTABLISHED
- UDP-Request mit state ESTABLISHED

Um zu überprüfen, ob einem UDP-Reply bereits Pakete vorausgingen, wird eine Tabelle benutzt, die die Information des Senders und des Empfängers trägt. Zum Anfang wird erst die Tabelle befragt, ob schon eine UDP-Verbindung zwischen den beiden Host existiert, falls nicht, werden der Request und der Reply mit der Firewallkonfiguration verglichen. Ansonsten wird nur noch das Paket in Richtung des Verbindungsaufbaus getestet, nun allerdings mit state ESTABLISHED. Es werden folgende Optionen für UDP unterstützt:

- Source- und Destination-Port
- Multiport-Option
- state NEW,ESTABLISHED

#### 4.4 ICMP-Regeln

Analysiert werden ICMP-Regeln mit Hilfe des Events `icmp_unreachable`. Es bietet die Möglichkeit die drei unterschiedlichen Arten von auftretenden ICMP-Paketen zu erkennen. Diese drei Grundtypen sind:

- kontextlose Pakete:
  - Typen 1,2,4-7,9-12
- Pakete nach dem Request-Reply-Prinzip:
  - Echo request (Type 8), Echo reply (Type 0)
  - Timestamp request (Type 13), Timestamp reply (Type 14)
  - Information request (Type 15), Information reply (Type 16)
  - Address mask request (Type 17), address mask reply (Type 18)
- Statusmeldungen, die aufgrund vorhergehender Pakete erzeugt wurden:
  - destination unreachable (Type 3)

Kontextlose Pakete werden aufgrund ihres Typs direkt auf zutreffende Regeln abgefragt.

Bei Paketen der zweiten Kategorie wird eine Tabelle mitgeführt, um überprüfen zu können, ob einem Reply auch wirklich ein Request vorausgegangen ist. Der Request durchläuft die umgesetzte Firewall und wird in der Tabelle eingetragen. Erscheint nun die Antwort, wird zuerst in der Tabelle nach diesem Eintrag gesucht. Ist dieser vorhanden, wird dem Paket der Verbindungsstatus `ESTABLISHED` zugewiesen. Im anderen Fall wird es mit `state NEW` versehen.

Eine Besonderheit bilden Destination-unreachable Pakete. Sie treten auf, wenn das Zielnetz, der Zielrechner oder der anzusprechende Dienst nicht erreichbar sind. In `iptables` ist hierfür der `state RELATED` zuständig. Das Policy-Skript sieht für den Fall, dass ein TCP-Verbindungsaufbau diese Meldung versendet hat, folgende Methode vor. ICMP-Pakete dieses Typs werden nicht sofort überprüft, sondern in einer weiteren Tabelle eingetragen. Erst wenn das Event `connection_attempt` aufgerufen wird, also nach einem Timeout des SYN-Pakets, wird die Tabelle durchsucht. Nachdem der Eintrag gefunden wurde, wird sowohl das TCP- als auch das ICMP-Paket analysiert.

#### 4.5 Selbstdefinierte Chains

`iptables` ermöglicht dem Administrator in der Firewall neue Chains anzulegen, um die Übersicht zu erhöhen. Dieses Feature wurde für Bro ebenfalls

verwirklicht. Der Unterschied zu einer terminierenden Regel ist, dass der Rückgabewert nicht einem Wert, wie `ACCEPT` oder `DROP`, entspricht, sondern einen weiteren Funktionsaufruf darstellt. Ist der Rückgabewert dieser Funktion nicht `RETURN`, `LOG` oder eine weitere selbstdefinierte Chain, so terminiert diese Funktion an dieser Stelle. Ist er hingegen `RETURN`, so wird wieder in die aufrufende Funktion zurückgesprungen. Eine genauere Erklärung der Targets befindet sich im nächsten Abschnitt.

## 4.6 Targets

Als Target wird in `iptables` die Aktion der Firewall bezeichnet, die bei Zutreffen einer Regel ausgeführt wird. Die umgesetzte Firewall kann je nach Topologie, d.h. vor bzw. hinter der Firewall und Verbindungsrichtung, entweder Voraussagen treffen oder entscheiden, ob das Paket an dieser Stelle noch auftreten darf oder nicht. Die nachfolgende Tabelle zeigt alle auftretenden Kombinationen und Aktionen, die durchgeführt werden:

Target	Src: !(Lok. Netz) Dest: Lok. Netz Bro(ext)	Src: !(Lok. Netz) Dest: Lok. Netz Bro(1..n)	Src: Lok. Netz Dest: !(Lok. Netz) Bro(ext)	Src: Lok. Netz Dest: !(Lok. Netz) Bro(1..n)
<code>ACCEPT</code>	;	;	;	;
<code>DROP</code>	<code>LOG</code>	<code>ALERT</code>	<code>ALERT</code>	<code>LOG</code>
<code>REJECT</code>	<code>LOG</code>	<code>ALERT</code>	<code>ALERT</code>	<code>LOG</code>
<code>QUEUE</code>	<code>LOG</code>	<code>ALERT</code>	<code>ALERT</code>	<code>LOG</code>
<code>MIRROR</code>	<code>LOG</code>	<code>LOG</code>	<code>LOG</code>	<code>LOG</code>
<code>LOG</code>	<code>LOG</code>	<code>LOG</code>	<code>LOG</code>	<code>LOG</code>
<code>Pol.-ACCEPT</code>	;	;	;	;
<code>Pol.-DROP</code>	;	<code>ALERT</code>	<code>ALERT</code>	;

Zur besseren Kontrolle wurde jedoch die Möglichkeit implementiert, alle Verbindungen aufzuzeichnen. Dazu muss das Skript `ipt2bro` mit dem zusätzlichen Parameter `-d` aufgerufen werden.

## 5 Vorschläge zur Weiterentwicklung

An dieser Stelle sei auf einige Mängel der Umsetzung und Möglichkeiten zur Erweiterung hingewiesen.

### 5.1 connection-state von iptables

`iptables` setzt für Verbindungen je nach ihrem Verbindungsstatus unterschiedliche Timeouts. Diese Timeouts legen die Zeit fest, wie lange eine Verbindung als offen und nachfolgende Pakete als dazugehörige (`ESTABLISHED`, `RELATED`) Pakete identifiziert werden. Diese Intervalle wurden in der Umsetzung unabhängig von ihrem aktuellen `state` gesetzt. TCP-Verbindungen richten sich nach dem von `Bro` zugewiesenen `state`. UDP-Pakete werden 30 Sekunden nach dem letzten Zugriff aus der Tabelle von etablierten Connections gelöscht. ICMP-Pakete verweilen sieben Minuten in der Tabelle. Der Grund für die lange Vorhaltezeit für ICMP-Pakete ist, dass `RELATED`-Pakete erst nach Auftreten des Events `connection_attempt` getestet werden können und dies zeitverzögert (sechs Minuten) zum SYN-Paket aufgerufen wird.

## 5.2 Target MIRROR

Das Target Mirror ist in iptables noch als experimental bezeichnet. Da dieses Target in den meisten Konfigurationen nicht auftritt, wurde es nicht in Bro umgesetzt. Dieses Target sorgt dafür, dass die IP-Adressen des Senders und des Empfängers vertauscht werden. Das ausgehende Paket wird jedoch nicht mehr durch die Firewall überprüft. Es müsste deswegen für jedes auftretende Paket vorher getestet werden, ob es sich nicht um eine Antwort dieses Targets handelt.

## 5.3 Erweiterbarkeit

Das Tool ist so ausgelegt, dass die Skripten `ipt2ids.pl` und `fire2bro.pl` ausgetauscht werden können. D.h. es können weitere Skripte erstellt werden, die es dem Tool erlauben, Konvertierungen für andere Firewalls oder ID-Systeme zu erzeugen.

# 6 Literaturverzeichnis

## Literatur

- [1] Vern Paxson's Webseite zu Bro:  
<http://www.icir.org/vern/bro-info.html>
- [2] Bro User Manual:  
<http://www.icir.org/vern/bro-manual/manual.pdf>
- [3] Dokumentation zu iptables:  
<http://www.iptables.org/>
- [4] Linux 2.4 Packet Filtering HOWTO:  
<http://www.iptables.org/documentation/HOWTO/de/packet-filtering-HOWTO.html>
- [5] Dokumentation zum Connection-Tracking-Modul von iptables:  
<http://www.sns.ias.edu/~jns/security/iptables/iptables.conntrack.html>
- [6] Perl Kochbuch  
Tom Christiansen, Nathan Torkington  
1999