

IDS/IPS mit Snort und Iptables als transparente Bridge

Author: Norbert Klein, www.infosecprojects.net,
01.02.2010

Einleitung

Ausgangssituation

Als Sicherheitsmaßnahme auf Paket-Ebene (ab Layer 3 (IP) aufwärts) soll ein Intrusion Prevention System realisiert werden. Dies führt zu einem starken Schutz vor Angriffen, die mit Hilfe einzelner oder weniger Netzwerk-Pakete durchgeführt werden können. Dadurch entsteht eine dritte Sicherheits-Stufe zwischen der Firewall (Port-Blockierung) und dem Gateway (Schutz vor Angriffen auf Ebene der Anwendungs-schicht (http)).

Überblick

Eine Minimal-Installation eines SLES 10 (ohne Desktop-Umgebung) wird soweit gehärtet, sodass ein ausreichender Schutz im vorliegenden Szenario gegeben ist:

- * Unbenötigte Dienste (siehe /etc/init.d) und Ports werden abgeschaltet.
- * Compiler und Assembler werden deinstalliert.
- * Unbenötigte Benutzer und Gruppen werden gelöscht.
- * Benutzer von Diensten werden gelockt und erhalten eine false-shell.
- * SSH-Zugang nur mit Public Key und als non-root.
- * su nur für Benutzer in der Gruppe wheel (mindestens ein User)
- * Konfiguration der lokalen Firewall (/etc/sysconfig/SuSEfirewall2)
- * Alle Dateisysteme in der fstab außer /dev erhalten 'nodev'
- * Prüfen, dass keine world-writeable Dateien vorhanden sind
- * Prüfen auf gesetzte suid-Bits und sgid-Bits.
- * Prüfen auf ungültige UIDs/GIDs.
- * Sensible Informationene aus /etc/issue, /etc/issue.net und /etc/motd entfernen.

Weiterhin können denyhosts, logwatch und rkhunter eingesetzt werden. Sinnvoll aber nicht zwingend wäre auch der Einsatz von Samhain.

Es erfolgt keine vollständige Härtung des Betriebssystems, da das Risiko in keinem Verhältnis mehr zum erforderlichen Aufwand steht. Also z.B. keine Härtung des Kernels, Apparmor, SELinux, usw.

Das System hat drei Netzwerkinterfaces, wovon 2 als Bestandteile einer transparenten Bridge konfiguriert werden, sie bekommen somit auch keine IP-Adresse und sind damit auf Layer 3 "unsichtbar".

Als IDS/IPS kommt Snort in Verbindung mit iptables zum Einsatz. Es werden die aktuellsten Snort-Rules verwendet, die für eine geringe Jahresgebühr in real-time erhältlich sind. Eine kostenlose Registrierung erlaubt den Zugriff auf die neuesten Regeln mit 30 Tagen Verzögerung, was für den Produktiv-Einsatz jedoch eine viel zu große Zeitspanne ist.

Sobald ein Paket mit verdächtigem Inhalt oder Format versucht die Bridge zu passieren,

kommt es zu einer Regelverletzung in Snort (IDS), worauf die lokale iptables-Firewall das Paket verwirft und gegebenenfalls die Verbindung zurücksetzt (IPS). Dadurch werden konkrete Angriffe noch bevor sie sich auswirken können geblockt.

Die Konfiguration des Systems erfolgt mittels eines dritten Netzwerkinterfaces, welches im Gegensatz zu den ersten beiden eine IP-Adresse erhält.

Auf einem zweiten identischen System im Lan können Änderungen an der Konfiguration und Regelupdates durchgeführt und getestet werden. Dieses Testsystem erhält alle grafischen Front-Ends, die für die Arbeit notwendig sind: z.B. Snorby oder BASE für eine grafische Analyse der Snort-Alerts über den Browser. Auf Windows-Clients kann der IDS-Policy-Manager zum Konfigurieren von Snort verwendet werden.

Eine Übertragung der neuen Regelsätze und/oder Konfiguration erfolgt dann über das zuvor angesprochene dritte Netzwerk-Interface. Diese Übertragung muss verschlüsselt erfolgen.

Die Logauswertung soll ebenfalls auf einem separaten System stattfinden. Per stunnel wird eine permanente, verschlüsselte Verbindung von der Bridge zum Log-Host errichtet und die Logs dort direkt in eine MySQL-Datenbank geschrieben. Dazu kann das Testsystem oder ein weiteres System verwendet werden. Eigentlich sollte das System wo die Logs liegen auch gut gehärtet werden, damit Angreifer keine Änderungen an den Logs vornehmen können. Wichtig ist, dass die Systemzeiten stimmen (ntpd). Die Logs sollten langfristig aufbewahrt werden, da Angriffsversuche oft über lange Zeiträume durchgeführt werden.

Die aktuellen Snort-Regeln können auf einem Update-Server in der DMZ bereitgehalten werden. Alternativ wäre auch denkbar per Oinkmaster das Testsystem automatisch mit den neuesten Regeln zu versorgen. Mindestens ein Syntax-Test muss vor dem Kopieren auf das Produktivsystem erfolgen, um zu verhindern, dass sich Snort nicht neustarten lässt.

Bei Verletzung einer Snort-Regel soll eine Alert-Notification per Email und SMS versendet werden. Dies geschieht z.B. mit Hilfe von syslog-ng. Per cron kann minütlich geprüft werden ob der Snort-Daemon noch läuft und eine Nachricht an Nagios gesendet werden. Dies sollte nicht mit Nagios-Mitteln direkt gemacht werden, da diese unter Umständen neue Einfalltüren darstellen.

Wichtig ist, die Bandbreite der Netzwerk-Leitung zu beachten, da sie sich auf die notwendige Performance des IDS auswirkt. Snort kann ohne weiteres auf einer 10MBit-Leitung betrieben werden.

Es muss sichergestellt werden, dass der Traffic nur über die Bridge das Gateway erreicht. Die Bridge darf nicht umgangen werden können! Es darf also nur einen physikalischen Weg von der Firewall zum Gateway geben und dazwischen muss die Bridge aufgestellt werden.

Begründung des Set-ups

Snort ist das meistverbreitetste frei verfügbare IDS. Es wird in sehr vielen Umgebungen erfolgreich eingesetzt und stellt ein mächtiges Werkzeug dar. Man erhält einen deutlichen Sicherheits-Gewinn. Neben der Prüfung auf verdächtige Paket-Inhalte werden auch Aktivitäten im Vorfeld eines Angriffes aufgespürt, wie z.B. Portscans oder OS-Fingerprinting. Einige kommerzielle Intrusion-Detection-Systeme verwenden intern Snort.

Snort ist auch für Windows erhältlich.

Durch die transparente Bridge ist der Server für das Netzwerk praktisch unsichtbar und somit für potentielle Angreifer kaum zu identifizieren oder anzugreifen. Die beteiligten NICs besitzen keine IP-Adressen. Das System agiert auf der gleichen Protokoll-Ebene wie ein Switch (Layer 2).

Kommerzielle Systeme sind zwar verfügbar, z.B. vom Marktführer TippingPoint. Die Preise sind jedoch exorbitant und bewegen sich zwischen 15.000\$ und weit über 100.000\$.

Snort ist weit verbreitet und man setzt dabei auf etablierte und bewährte Technologie. Die Installation und Administration von Snort ist ungewöhnlich einfach. Bei Bedarf kann die Architektur erweitert werden, indem weitere Snort-Instanzen z.B. an den Monitoring-Port von relevanten Switches angeschlossen werden.

Aufwand

Set-up

Einrichtung von SLES als Bridge auf dem Produktivsystem und Härtung. Installation von Snort. Einrichtung von SLES als Test/Log-Host, keine Bridge, mit Snort und allen benötigten grafischen Front-Ends und MySQL. Einrichtung von stunnel zum verschlüsselten Transfer der Logs vom Produktivsystem auf den Log-Host. Einrichtung der Email- und SMS-Notifications. Anfangs-Tuning/Testen der Regeln bei nichtblockierendem Betrieb. Test des Zugriffs auf das Testsystem von Windows aus mit IDS Policy Manager.

Laufender Betrieb

Anpassen der Regeln im laufenden Betrieb bei false positives. Administration von Snort.

Beschreibung zum Set-up

Siehe dazu Diagramm auf <http://www.infosecprojects.net/images/snort.png>

Für den beschriebenen Setup wird ein physischer Server (Snort-Sensor) mit drei Netzwerkkarten benötigt. 2 Netzwerkkarten sind Bestandteil der Bridge. Das heist auf dem Diagramm von links kommt die Leitung aus der externen Firewall bzw. aus einem danach folgenden Switch und wird in die erste Netzwerkkarte geführt (NIC1). Aus der zweiten Netzwerkkarte (NIC2) verläuft die Leitung hinaus in den Eingang des folgenden Switches, sodass alle Pakete auf dem Weg zum LAN den Snort-Sensor passieren müssen.

Der Snort-Sensor darf nur vom Log-Host (LAN) über die dritte Netzwerkkarte(NIC3) erreichbar sein und nur per ssh. Neue Regeln werden damit vom Log-Host auf den Sensor übertragen. Die Verwaltung der Regeln per IDS Policy Manager erfolgt auf dem Log-Host.

Mit Hilfe von iptables werden alle Paketdetails im User-Space verfügbar gemacht, sodass sie von Snort untersucht werden können. Die Log-Daten von Snort werden dann vom Snort-Sensor zum Log-Host per stunnel übertragen. Es werden darüber sowohl die Daten für das grafische Front-End als auch die Alerts für die Notifications übertragen.

Sockserv ist ein Dienst, der die Log-Daten von Snort von einem Socket entgegennimmt und über das Netz an servsock überträgt. Servsock schreibt sie dann gemäß dem Snort-Datenbankschema in eine MySQL-Datenbank. Ab einem bestimmten Prioritätslevel wird über einen Socket eine Nachricht an den Dienst alert gegeben, welcher eine Email zur Benachrichtigung absendet.

Als Front-End kommt Snorby (snorby.org) zum Einsatz. Es ist das modernste und zugleich übersichtlichste grafische Front-End für Snort. Der alternative Set-up mit BASE wird ebenfalls beschrieben.

Technische Umsetzung

Snort-Loghost

Über Yast NTP-Client einrichten

Firewall

Damit man sich nicht selbst aussperrt kann man temporär per cron alle 5 Min. die Regeln löschen lassen:

```
* /5 * * * * root /root/firewall_off.sh
```

```
vi /root/firewall_off.sh
```

```
#!/bin/sh
```

```
iptables=/usr/sbin/iptables
```

```
$iptables -P INPUT ACCEPT
```

```
$iptables -P FORWARD ACCEPT
```

```
$iptables -P OUTPUT ACCEPT
```

```
$iptables -F -t filter
```

```
$iptables -F -t nat
```

```
$iptables -X -t filter
```

```
$iptables -X -t nat
```

```
echo "FIREWALL OFF !!!"
```

```
chmod u+x /root/firewall_off.sh
```

```
vi /etc/init.d/firewall
```

```
#!/bin/sh
```

```
iptables=/usr/sbin/iptables
```

```
# interfaces
```

```
LOCALHOST='lo'
```

```
NIC1='eth0'
```

```
SENSOR='192.168.1.35'
```

```
LOGHOST='192.168.1.40'
```

```
DNSERVER='192.168.1.1'
```

```
LAN='192.168.1.0/24'
```

```
# activate IP-Forwarding in the kernel
```

```
/bin/echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
case "$1" in
```

```
start)
```

```
# flush all chains in the filter table (deleting rules)
```

```
# flush all chains in the nat table (deleting rules)
```

```
# delete every non-builtin chain (must be empty of rules) in the filter table
# delete every non-builtin chain (must be empty of rules) in the nat table
$IPTABLES -F -t filter
$IPTABLES -F -t nat
$IPTABLES -X -t filter
$IPTABLES -X -t nat

# set the policies for the built-in chains
$IPTABLES -P INPUT DROP
$IPTABLES -P FORWARD DROP
$IPTABLES -P OUTPUT DROP

# allow all for localhost
$IPTABLES -A INPUT -i $LOCALHOST -j ACCEPT
$IPTABLES -A OUTPUT -o $LOCALHOST -j ACCEPT

# allow outgoing traffic
$IPTABLES -A OUTPUT -o $NIC1 -m state --state NEW,RELATED,ESTABLISHED -j
ACCEPT

# allow responses
$IPTABLES -A INPUT -i $NIC1 -m state --state ESTABLISHED,RELATED -j ACCEPT

# OPEN PORTS
# icmp (ping)
$IPTABLES -A INPUT -i $NIC1 -s $LAN -p icmp --icmp-type destination-unreachable -j
ACCEPT
$IPTABLES -A INPUT -i $NIC1 -s $LAN -p icmp --icmp-type time-exceeded -j ACCEPT
$IPTABLES -A INPUT -i $NIC1 -s $LAN -p icmp --icmp-type echo-reply -j ACCEPT
$IPTABLES -A INPUT -i $NIC1 -s $LAN -p icmp --icmp-type echo-request -j ACCEPT
# ssh
$IPTABLES -A INPUT -i $NIC1 -s $LAN -m state --state NEW --protocol tcp --dport 22 -j
ACCEPT
# www
$IPTABLES -A INPUT -i $NIC1 -p tcp --dport 80 -j ACCEPT
$IPTABLES -A INPUT -i $NIC1 -p tcp --dport 443 -j ACCEPT
# stunnel server
$IPTABLES -A INPUT -i $NIC1 -s $SENSOR -m state --state NEW --protocol tcp --dport
10440 -j ACCEPT

# Log the rest
$IPTABLES -A INPUT -j LOG --log-prefix="IPTABLES-INPUT: "
$IPTABLES -A OUTPUT -j LOG --log-prefix="IPTABLES-OUTPUT: "
$IPTABLES -A FORWARD -j LOG --log-prefix="IPTABLES-FORWARD: "
;;
stop)
$IPTABLES -F -t filter
$IPTABLES -F -t nat
$IPTABLES -X -t filter
$IPTABLES -X -t nat

# set the policies for the built-in chains
```

```
$IPTABLES -P INPUT ACCEPT
$IPTABLES -P FORWARD ACCEPT
$IPTABLES -P OUTPUT ACCEPT
;;
restart)
    $0 stop
    $0 start
    ;;
*)
    echo "Usage: $0 {start|stop}"
    exit 1
esac

exit 0
```

```
/etc/init.d/firewall start
```

```
chmod 0755 /etc/init.d/firewall
chkconfig firewall 345
```

Sobald die Firewall funktioniert den Cron-Eintrag für /root/firewall_off.sh auskommentieren !!!

chmod 0000 /root/firewall_off.sh !!!

/etc/init.d/firewall.sh start !!!

stunnel

Mit yast stunnel installieren.

```
vi /etc/stunnel/stunnel.conf
```

```
client = no
```

```
cert = /etc/stunnel/stunnel.pem
```

```
[mysql]
```

```
    accept = 192.168.1.40:10440
```

```
    connect = 127.0.0.1:10440
```

```
openssl req -new -x509 -days 4000 -nodes -out /etc/stunnel/stunnel.pem -keyout \
/etc/stunnel/stunnel.pem
```

```
chmod 600 /etc/stunnel/stunnel.pem
```

```
chkconfig stunnel 345
```

FLoP

(für FLoP muss configure in den Snort-Sourcen ausgeführt worden sein, daher werden die Snort-Sourcen downgeloaded und configure darin ausgeführt.)

```
mkdir /home/test/build
```

```
cd /home/test/build
```

```
wget http://dl.snort.org/snort-current/snort-2.8.5.tar.gz
```

```
wget ftp://ftp.uni-kassel.de/Mirrors/ftp.fhh.opensource-mirror.de/gentoo/distfiles/libnet-1.0.2a.tar.gz
```

```
wget http://www.geschke-online.de/FLoP/src/FLoP-1.6.1.tar.gz
```

Für alle Sourcen den MD5-Check durchführen.

```
tar xfv libnet-1.0.2a.tar.gz
tar xfv snort-2.8.5.tar.gz
tar xfv FLoP-1.6.1.tar.gz
```

```
cp FLoP-1.6.1/patches/snort-2.8.5_patch snort-2.8.5
cd snort-2.8.5
patch -p1 < ./snort-2.8.5_patch
```

Mit Yast pcre-devel, mysql-devel, iptables-devel, mysql installieren.
Der Pfad zum libnet-prefix muss in der PATH-Variablen des Benutzers stehen, der configure ausführt:

```
vi ~/.bashrc
PATH=$PATH:/opt/libnet/bin
. ~/.bashrc
```

```
./configure --prefix="/opt/snort" --with-libnet-includes="/home/test/build/Libnet-1.0.2a/include" --with-libnet-libraries="/opt/libnet/lib" --enable-inline --with-mysql --enable-decoder-preprocessor-rules
```

```
cd ../FLoP-1.6.1/
CPPFLAGS=-I/usr/include/mysql ./configure --prefix="/opt/flop" --with-snort=/home/test/build/snort-2.8.5 --with-mysql=/usr --with-libpcap --enable-getpacket
make
make install
```

```
vi /opt/flop/etc/flop/servsock.conf
DBuser: snort
DBpassword: "[SNORTPW]"
DBname: snort
DBType: MySQL
ServerName: '127.0.0.1'
ServerPort: "10440"
SocketName: 127.0.0.1:3306
AlertSocket: /var/run/alert.socket
# -2 bedeutet: es werden nur Alerts gemailt, deren numerischer Wert geringer ist als der #absolute Wert der UnixSocketPriority, das heist also nur Alerts mit Prio 1.
UnixSocketPriority: -2
DaemonMode: 1
...
```

```
cp /home/test/build/FLoP-1.6.1/contrib/servsock.init /etc/init.d/servsock
vi /etc/init.d/servsock
#!/bin/sh
```

```
case "$1" in
  start)
    echo -n "Starting servsock: "
```

```

/opt/flop/bin/servsock -c /opt/flop/etc/flop/servsock.conf
touch /var/lock/subsys/servsock
echo
;;
stop)
echo -n "Stopping servsock: "
killproc servsock
rm -f /var/run/servsock.pid
rm -f /var/lock/subsys/servsock
echo
;;
reload)
echo -n "Sending HUP to servsock process(es): "
killproc servsock -HUP
echo
;;
restart)
$0 stop
$0 start
;;
condrestart)
if [ "x`pidof servsock`" != x ]; then
    $0 stop
    $0 start
fi
;;
status)
status servsock
RETVAL=$?
exit $RETVAL
;;
*)
echo "Usage: $0 {start|stop|reload|restart|status}"
exit 1
esac

exit 0

```

```

chmod 0700 /etc/init.d/servsock
chkconfig servsock 345

```

```

vi /etc/my.cnf
[mysqld]
bind-address = 127.0.0.1
port          = 3306

```

```
chkconfig mysql 345
/etc/init.d/mysql start
```

```
mysql -u root
update mysql.user set password=password('PASSWORD') where user='root';
update mysql.user set password=password('PASSWORD') where user='snort'
flush privileges;
grant all privileges on snort.* to 'snort'@'localhost' identified by 'PASSWORD';
grant all privileges on snort_archive.* to 'snort'@'localhost' identified by 'PASSWORD';
```

Base

über yast apache2, php5-pear, apache2-mod_php5, php5-gd, php5-mysql, php5-mysqli installieren.

```
cd /home/test/build
pear5 channel-update pear.php.net
mkdir /opt/pear
cd /opt/pear
wget http://pear.php.net/go-pear
MD5-Check durchführen.
In /etc/php5/cli/php.ini das memory_limit auf 128M setzen
php5 go-pear
(alles installieren, include path anpassen lassen)
/opt/pear/bin/pear install --alldeps Image_Graph-alpha
In /etc/php5/apache2/php.ini den include_path mit /opt/pear/PEAR erweitern.
```

```
pear5 config-show | grep "PEAR directory"
Den ausgegebenen Pfad für das folgende Kommando nehmen:
```

```
cd /home/test/build
wget http://downloads.sourceforge.net/project/secureideas/BASE/base-1.4.4/base-1.4.4.tar.gz?use_mirror=mesh
MD5-Check
tar xfv base-1.4.4.tar.gz
```

```
cp /home/test/build/base-1.4.4/world_map6.* /opt/pear/PEAR/Image/Graph/Images/Maps/
```

```
mv base-1.4.4 /srv/www/htdocs/base
```

Adodb (für BASE)

```
cd /home/test/build
wget http://downloads.sourceforge.net/project/adodb/adodb-php5-only/adodb-509a-for-php5/adodb509a.tgz?use_mirror=mesh
MD5-Check durchführen.
tar xfv adodb509a.tgz
mv adodb5 /opt
```

```
cd /home/test/build
wget http://search.cpan.org/CPAN/authors/id/N/NW/NWETTERS/IP-Country-2.27.tar.gz
```

```
MD5-Check durchführen.  
tar xvf IP-Country-2.27.tar.gz
```

```
wget http://search.cpan.org/CPAN/authors/id/A/AB/ABIGAIL/Geography-Countries-  
2009041301.tar.gz  
MD5-Check durchführen.  
tar xfv Geography-Countries-2009041301.tar.gz  
cd Geography-Countries-2009041301  
perl Makefile.PL  
make  
make test  
make install
```

```
cd ../IP-Country-2.27  
perl Makefile.PL  
make  
make test  
make install
```

Die snort- und snort_archive-Datenbank manuell erstellen.

```
cd /home/test/build  
mysql -uroot -p snort < FLoP-1.6.1/contrib/create_mysql  
mysql -uroot -p snort_archive < FLoP-1.6.1/contrib/create_mysql
```

Signaturen und Referenzen importieren:
cd /home/test/build/FLoP-1.6.1/contrib/
cp rules.pl.conf rules_archive.pl.conf

```
vi rules.pl.conf  
$dbtype="mysql";  
$dbname="snort";  
$dbuser="snort";  
$dbpass="snort";  
$dbhost="localhost";  
$dbport="3306";  
$ClassFile="/home/test/build/snort-2.8.5/etc/classification.config";  
$ReferenceFile="/home/test/build/snort-2.8.5/etc/reference.config";  
$PrioRange=3;  
1; # needed to return true to require
```

```
vi rules_archive.pl.conf  
$dbtype="mysql";  
$dbname="snort_archive";  
$dbuser="snort";  
$dbpass="snort";  
$dbhost="localhost";  
$dbport="3306";  
$ClassFile="/home/test/build/snort-2.8.5/etc/classification.config";  
$ReferenceFile="/home/test/build/snort-2.8.5/etc/reference.config";  
$PrioRange=3;  
1; # needed to return true to require
```

```
./rules.pl -c ./rules.pl.conf  
./rules.pl -c ./rules_archive.pl.conf
```

```
chown wwwrun /srv/www/htdocs/base
```

```
vi /etc/php5/apache2/php.ini  
von  
error_reporting = E_ALL  
nach  
error_reporting = E_ALL & ~E_NOTICE
```

Apache neustarten

```
http://snort-loghost/base/setup/  
(Path to ADODB: /opt/adodb5)
```

```
cd /srv/www/htdocs/base  
vi base_conf.php
```

```
$BASE_Language = 'german';  
keine Authentifizierung, besser über .htaccess  
$BASE_urlpath = '/base';  
$BASE_installID = '01';  
$DBlib_path = '/opt/adodb5';
```

```
$alert_dbname = 'snort';  
$alert_host = 'localhost';  
$alert_port = '3306';  
$alert_user = 'snort';  
$alert_password = '[SNORTPW]';
```

```
$archive_exists = 1;  
$archive_dbname = 'snort_archive';  
$archive_host = 'localhost';  
$archive_port = '3306';  
$archive_user = 'snort';  
$archive_password = '[SNORTPW]';
```

Email-Konfiguration

```
$use_sig_list = 2; (benötigt Javascript)  
$resolve_IP = 1;
```

```
$IP2CC = "/usr/bin/ip2cc";
```

Eine .htaccess anlegen:
htpasswd -c /srv/www/.htpasswd snortteam
vi /srv/www/htdocs/base/.htaccess
AuthType Basic
AuthName "Welcome to BASE"
AuthUserFile /srv/www/.htpasswd
require valid-user

vi /etc/apache2/default-server.conf
Folgenden Abschnitt entsprechend ändern:
<Directory "/srv/www/htdocs">

```
...  
    AllowOverride AuthConfig  
...  
</Directory>
```

vi /etc/apache2/listen.conf
Listen 192.168.1.40:80
#Listen 443

```
<IfDefine SSL>  
  <IfDefine !NOSSL>  
    <IfModule mod_ssl.c>  
  
        Listen 192.168.1.40:443  
  
    </IfModule>  
  </IfDefine>  
</IfDefine>
```

chkconfig apache2 345

/etc/init.d/apache2 restart
/etc/init.d/stunnel start
/etc/init.d/sockd start

netstat -tulpen

REBOOT

Snorby/Apache

Mit Yast openssl-devel installieren.

```
cd /home/test/build
wget ftp://ftp.ruby-lang.org/pub/ruby/stable/ruby-1.8.7-p173.tar.gz
MD5-Check durchführen.
tar xfv ruby-1.8.7-p173.tar.gz
cd ruby-1.8.7-p173
./configure --prefix="/opt/ruby" --disable-install-doc
make
make install
(erst bei Ruby 1.9 wird rubygems mit installiert)
```

Snorby benötigt Ruby 1.8.7, erst Snorby 2.0 läuft mit Ruby 1.9!

```
cd /home/test/build
wget http://rubyforge.org/frs/download.php/60718/rubygems-1.3.5.tgz
MD5-Check
tar xfv rubygems-1.3.5.tgz
cd rubygems-1.3.5
/opt/ruby/bin/ruby setup.rb
/opt/ruby/bin/gem install rake prawn --no-ri --no-rdoc
/opt/ruby/bin/gem install -v=2.3.2 rails --no-ri --no-rdoc
/opt/ruby/bin/gem install dbd-mysql --no-ri --no-rdoc
/opt/ruby/bin/gem install passenger --no-ri --no-rdoc
```

```
vi /opt/ruby/lib/ruby/gems/1.8/gems/passenger-2.2.5/bin/passenger-install-apache2-
module
Erste Zeile ändern auf:
#!/usr/bin/env /opt/ruby/bin/ruby
```

Mit Yast apache2-devel und gcc-c++ installieren.

```
/opt/ruby/lib/ruby/gems/1.8/gems/passenger-2.2.5/bin/passenger-install-apache2-module
```

Gemäß Output:

```
cp /opt/ruby/lib/ruby/gems/1.8/gems/passenger-2.2.5/ext/apache2/mod_passenger.so /usr/
lib/apache2/
ln -s /usr/lib/apache2/mod_passenger.so /usr/lib/apache2-prefork/mod_passenger.so
```

```
vi /etc/sysconfig/apache2
APACHE_MODULES="... passenger"
```

SUSEconfig

```
vi /etc/apache2/default-server.conf
# Snorby
PassengerRoot /opt/ruby/lib/ruby/gems/1.8/gems/passenger-2.2.5
PassengerRuby /opt/ruby/bin/ruby
```

```
cd /home/test/build
```

```
wget http://kernel.org/pub/software/scm/git/git-1.6.5.tar.gz
MD5-Check
tar xfv git-1.6.5.tar.gz
cd git-1.6.5
./configure --prefix="/opt/git"
make
make install
```

```
cd /home/test/build
/opt/git/bin/git clone git://git.kernel.org/pub/scm/git/git.git
```

```
cd /srv/www/htdocs
/opt/git/bin/git clone git://github.com/mephux/Snorby.git
mv Snorby snorby
cd snorby
cp config/database.yml.example config/database.yml
cp config/email.yml.example config/email.yml
```

```
vi config/database.yml
production:
  adapter: mysql
  database: snort
  username: snorby
  password: [SNORBYPW]
  host: localhost
```

```
vi config/email.yml
production:
  :address: smtp.domain.com
  :domain: localhost
```

```
cd /srv/www/htdocs/snorby
/opt/ruby/bin/rake gems:install
/opt/ruby/bin/rake snorby:setup RAILS_ENV=production
```

```
mysql>
ALTER TABLE data ADD COLUMN data_header TEXT;
ALTER TABLE data ADD COLUMN pcap_header TEXT;
ALTER TABLE `schema` ADD COLUMN full_payload SMALLINT;
UPDATE `schema` SET full_payload=1;
```

```
ALTER TABLE event ADD COLUMN reference INT8;
ALTER TABLE `schema` ADD COLUMN reference SMALLINT;
UPDATE `schema` SET reference=1;
```

```
grant all on snort.* to snort@localhost identified by 'snort';
```

```
Signaturen und Referenzen importieren:
cd /home/test/build/FLoP-1.6.1/contrib/
cp rules.pl.conf rules_archive.pl.conf
vi rules.pl.conf
$dbtype="mysql";
```

```
$dbname="snort";  
$dbuser="snort";  
$dbpass="snort";  
$dbhost="localhost";  
$dbport="3306";  
$ClassFile="/home/test/build/snort-2.8.5/etc/classification.config";  
$ReferenceFile="/home/test/build/snort-2.8.5/etc/reference.config";  
$PrioRange=3;  
1; # needed to return true to require
```

```
./rules.pl -c ./rules.pl.conf
```

```
chown wwwrun /srv/www/htdocs/snorby
```

```
echo '<a href="/snorby/public">Snorby</a>' > /srv/www/htdocs/index.html
```

```
vi /etc/apache2/default-server.conf
```

```
DocumentRoot "/srv/www/htdocs/snorby/public"
```

Unten bei # snorby noch anfügen:

```
RailsBaseURI /  
<directory "/srv/www/htdocs/snorby/public">  
AllowOverride AuthConfig  
Order deny,allow  
Allow from 192.168.1.33  
</directory>
```

```
/etc/init.d/apache2 restart
```

Snorby-Admin-Passwort ändern!

REBOOT

Snort-Sensor

Mit yast das Paket bridge-utils installieren.

Härtung

Yast -> Sicherheit und Benutzer -> Einstellungen zur Sicherheit

Benutzerdefinierte Einstellungen

Strg-Alt-Del ignorieren

Niemand

Dateiberechtigungen: Sicher

Benutzer der updatedb starten soll: nobody

Unbenötigte Dienste und Ports abgeschalten

```
ps auxfwww;netstat -tulpen
```

```
/etc/init.d/postfix stop
```

```
insserv -r postfix
```

```
/etc/init.d/resmgr stop
```

```
insserv -r resmgr
```

```
/etc/init.d/slpd stop
```

```
insserv -r slpd
```

```
/etc/init.d/nscd stop
```

```
insserv -r nscd
```

```
/etc/init.d/portmap stop
```

```
insserv -rf portmap
```

```
/etc/init.d/novell-zmd stop
```

```
insserv -r novell-zmd
```

Compiler und Assembler deinstallieren

(In Yast in der Beschreibung nach "compiler" und "assembler" suchen)

Unbenötigte Benutzer deaktivieren

```
#!/bin/bash
```

```
for NAME in `cut -d: -f1 /etc/passwd`;
```

```
do
```

```
  MyUID=`id -u $NAME`
```

```
  if [ $MyUID -lt 500 -a $NAME != 'root' ];
```

```
  then
```

```
    usermod -L -s /bin/false $NAME
```

```
  fi
```

```
done
```

Alle zusätzlichen Benutzer (neben root) aus zusätzlichen Gruppen entfernen (/etc/group)

SSH-Zugang nur mit Public Key und als non-root.

```
mkdir /home/test/.ssh
```

Auf dem Snort-LOGHOST erstellen:

```
ssh-keygen -t rsa -b 2048 -f ~/.ssh/snort-loghost_rsa
```

und den Public-key von dort auf den Sensor kopieren:

```
scp ~/.ssh/snort-loghost_rsa.pub user@snort-sensor:/home/test/.ssh
```

Snort-Sensor:

```
touch /home/test/.ssh/authorized_keys
```

```
cd /home/test
```

```
mkdir .ssh
```

```
cat ./snort-loghost_rsa.pub > /home/test/.ssh/authorized_keys
```

```
rm ./snort-loghost_rsa.pub
```

```
vi /etc/ssh/sshd_config
```

```
ListenAddress 192.168.1.35
```

```
PermitRootLogin no
```

```
RSAAuthentication no
```

```
PubkeyAuthentication yes
```

```
AuthorizedKeysFile .ssh/authorized_keys
```

```
PasswordAuthentication no
```

```
PermitEmptyPasswords no
```

```
UsePAM no
```

```
/etc/init.d/sshd restart
```

Benutzer test in die Gruppe wheel eintragen!

su nur für Benutzer in der Gruppe wheel

```
vi /etc/pam.d/su
```

```
auth required pam_wheel.so nach der letzten auth-Zeile einfügen.
```

Mindestens einen Benutzer in die Gruppe wheel eintragen und testen!

Konfiguration der lokalen Firewall

Damit man sich nicht selbst aussperrt kann man temporär per cron alle 5 Min. die Regeln löschen lassen:

```
*/5 * * * * root /root/firewall_off.sh
```

```
vi /root/firewall_off.sh
```

```
#!/bin/sh
```

```
iptables -P INPUT ACCEPT
```

```
iptables -P FORWARD ACCEPT
```

```
iptables -P OUTPUT ACCEPT
```

```
iptables -F -t filter
```

```
iptables -F -t nat
```

```
iptables -X -t filter
```

```
iptables -X -t nat
echo "FIREWALL OFF !!!"
```

```
chmod u+x /root/firewall_off.sh
```

Für Snort inline ist in iptables "-j QUEUE" erforderlich, welches wiederum durch ein Kernelmodul ermöglicht wird:

```
vi /etc/sysconfig/kernel
MODULES_LOADED_ON_BOOT="ip_queue"
SuSEconfig
```

```
vi /etc/init.d/bridge
#!/bin/sh
```

```
# interfaces
LOCALHOST='lo'
BR1='br0'
NIC1='eth2'
NIC2='eth1'
NIC3='eth0'
```

```
SENSOR='192.168.1.35'
LOGHOST='192.168.1.40'
DNSSERVER='192.168.1.1'
```

```
LAN='192.168.1.0/24'
```

```
IPTABLES=/usr/sbin/iptables
```

```
# deactivate IP-Forwarding in the kernel because we run a bridge
/bin/echo 0 > /proc/sys/net/ipv4/ip_forward
```

```
case "$1" in
  start)
```

```
echo "bridge start" >> /var/log/messages
```

```
# Bridge setup
#####
ifconfig $NIC1 0.0.0.0 up
ifconfig $NIC2 0.0.0.0 up
brctl addbr $BR1 > /dev/null 2>&1
brctl addif $BR1 $NIC1 >/dev/null 2>&1
brctl addif $BR1 $NIC2 >/dev/null 2>&1
ifconfig $BR1 0.0.0.0 up
brctl stp $BR1 off
brctl show $BR1
#####
```

```
;;
stop)
echo "bridge stop" >> /var/log/messages
brctl delif $BR1 $NIC1
brctl delif $BR1 $NIC2
ifconfig $BR1 down
brctl delbr $BR1
;;
```

```
restart)
    $0 stop
    $0 start
    ;;
```

```
*)
    echo "Usage: $0 {start|stop|restart}"
    exit 1
```

```
esac
```

```
exit 0
```

```
chmod 0700 /etc/init.d/bridge
chkconfig bridge 345
```

```
vi /etc/init.d/firewall
```

```
#!/bin/sh
```

```
# interfaces
LOCALHOST='lo'
BR1='br0'
```

```
#Achtung, damit hier immer die richtige Zuordnung erfolgt muss per udev geregelt werden, welches NIC eine IP bekommt.
```

```
NIC1='eth2'
NIC2='eth1'
NIC3='eth0'
```

```
SENSOR='192.168.1.35'
LOGHOST='192.168.1.40'
DNSSERVER='192.168.1.1'
```

```
LAN='192.168.1.0/24'
```

```
IPTABLES=/usr/sbin/iptables
```

```
# deactivate IP-Forwarding in the kernel because we run a bridge
/bin/echo 0 > /proc/sys/net/ipv4/ip_forward
```

```
case "$1" in
start)
```

```
echo "firewall start" >> /var/log/messages
```

```
# flush all chains in the filter table (deleting rules)
# flush all chains in the nat table (deleting rules)
# delete every non-builtin chain (must be empty of rules) in the filter table
# delete every non-builtin chain (must be empty of rules) in the nat table
$IPTABLES -F -t filter
$IPTABLES -F -t nat
$IPTABLES -X -t filter
$IPTABLES -X -t nat
```

```
# set the policies for the built-in chains
$IPTABLES -P INPUT ACCEPT
$IPTABLES -P FORWARD DROP
$IPTABLES -P OUTPUT ACCEPT
```

```
# allow all for localhost
$IPTABLES -A INPUT -i $LOCALHOST -j ACCEPT
$IPTABLES -A OUTPUT -o $LOCALHOST -j ACCEPT
```

```
# allow outgoing traffic
$IPTABLES -A OUTPUT -o $NIC3 -m state --state NEW,RELATED,ESTABLISHED -j
ACCEPT
```

```
# allow responses
$IPTABLES -A INPUT -i $NIC3 -m state --state ESTABLISHED,RELATED -j ACCEPT
```

```
# OPEN PORTS
```

```
# icmp (ping)
```

```
$IPTABLES -A INPUT -i $NIC3 -s $LOGHOST -p icmp --icmp-type destination-
unreachable -j ACCEPT
```

```
$IPTABLES -A INPUT -i $NIC3 -s $LOGHOST -p icmp --icmp-type time-exceeded -j
ACCEPT
```

```
$IPTABLES -A INPUT -i $NIC3 -s $LOGHOST -p icmp --icmp-type echo-reply -j ACCEPT
```

```
$IPTABLES -A INPUT -i $NIC3 -s $LOGHOST -p icmp --icmp-type echo-request -j
ACCEPT
```

```
# ssh
```

```
$IPTABLES -A INPUT -i $NIC3 -s $LOGHOST -m state --state NEW --protocol tcp --dport
22 -j ACCEPT
```

```
# stunnel
```

```
$IPTABLES -A OUTPUT -o $NIC3 -s $SENSOR -m state --state NEW --protocol tcp
--dport 10440 -j ACCEPT
```

```
# Log the rest
```

```
$IPTABLES -A INPUT -j LOG --log-prefix="IPTABLES-INPUT: "
```

```
$IPTABLES -A OUTPUT -j LOG --log-prefix="IPTABLES-OUTPUT: "
```

```
$IPTABLES -A FORWARD -j LOG --log-prefix="IPTABLES-FORWARD: "
```

```
::
```

```
stop)
```

```
echo "firewall stop" >> /var/log/messages
```

```

$IPTABLES -F -t filter
$IPTABLES -F -t nat
$IPTABLES -X -t filter
$IPTABLES -X -t nat

# set the policies for the built-in chains
$IPTABLES -P INPUT ACCEPT
$IPTABLES -P FORWARD ACCEPT
$IPTABLES -P OUTPUT ACCEPT

;;
restart)
    $0 stop
    $0 start
    ;;

*)
    echo "Usage: $0 {start|stop|restart}"
    exit 1
esac

exit 0

/etc/init.d/firewall

chmod 0700 /etc/init.d/firewall
chkconfig firewall 345

```

Wenn man die Bridge freigeben möchte, das heist nichts kontrollieren möchte, kann man folgendes eingeben, Snort muss dann auch nicht laufen. (Die Regeln für das NIC3 bleiben bestehen.)

```

/usr/sbin/iptables -P FORWARD ACCEPT
/usr/sbin/iptables -D FORWARD 2

```

Es reicht also die default-Policy in FORWARD von DROP auf ACCEPT zu stellen und die Regel für das Queueing zu entfernen.

Testen ob ntp- und dns-Abfragen funktionieren:

```
dig localdomain.com
```

```
ntpdate 0.de.pool.ntp.org
```

```
**** root ntpdate 0.de.pool.ntp.org
```

(ntpdate jeweils mit lokalem Zeitserver, die Einrichtung von ntp-client ist nicht notwendig)

Anpassungen in der /etc/fstab

```
/tmp                zusätzlich: nosuid,nodev,noexec
```

```
/dev/sdaX /home      zusätzlich: nosuid,nodev
```

```
/dev/sdaX /rest          zusätzlich: nodev
```

```
mount -o remount /tmp
mount -o remount /home
mount -o remount /
```

Prüfen, dass keine world-writable Dateien vorhanden sind

```
vi /root/worldwritables.sh
#!/bin/bash
for PART in `awk '($6 != "0") { print $2 }' /etc/fstab`;
do
    echo $PART
    find $PART -xdev -perm /o+w -type f ! -type l
    echo "---"
done

chmod 0700 /root/worldwritables.sh
/root/worldwritables.sh
```

Prüfen auf gesetzte suid-Bits und sgid-Bits.

```
vi /root/suidsgid.sh
#!/bin/bash
for PART in `awk '($6 != "0") { print $2 }' /etc/fstab`;
do
    echo $PART
    echo "suid bit:"
    find $PART -xdev -perm /u+s -type f ! -type l
    echo "sgid bit:"
    find $PART -xdev -perm /g+s -type f ! -type l
    echo "---"
done

chmod 0700 /root/suidsgid.sh
/root/suidsgid.sh
```

Prüfen auf ungültige UIDs/GIDs.

```
vi /root/invaliduidgid.sh
#!/bin/bash
for PART in `awk '($6 != "0") { print $2 }' /etc/fstab`;
do
    find $PART -nouser -o -nogroup
done

chmod 0700 /root/invaliduidgid.sh
/root/invaliduidgid.sh
```

Sensible Informationen aus /etc/issue, /etc/issue.net und /etc/motd entfernen.

REBOOT

Snort

```
mkdir /home/test/build  
cd /home/test/build
```

```
wget http://dl.snort.org/snort-current/snort-2.8.5.tar.gz  
wget http://dl.snort.org/reg-rules/snortrules-snapshot-CURRENT.tar.gz  
wget ftp://ftp.uni-kassel.de/Mirrors/ftp.fhh.opensource-mirror.de/gentoo/distfiles/libnet-1.0.2a.tar.gz  
wget http://www.geschke-online.de/FLoP/src/FLoP-1.6.1.tar.gz
```

Für alle MD5-Check durchführen.

Mit Yast stunnel, pcre-devel, iptables-devel, mysql-devel installieren.

```
tar xfv libnet-1.0.2a.tar.gz  
cd Libnet-1.0.2a/  
./configure --prefix="/opt/libnet"  
make  
make install
```

```
cd ..  
tar xfv snort-2.8.5.tar.gz  
tar xfv FLoP-1.6.1.tar.gz  
cp FLoP-1.6.1/patches/snort-2.8.5_patch snort-2.8.5  
cd snort-2.8.5  
patch -p1 < ./snort-2.8.5_patch
```

Der Pfad zum libnet-prefix muss in die PATH-Variable des Benutzers, der configure ausführt:

```
vi ~/.bashrc  
PATH=$PATH:/opt/libnet/bin  
. ~/.bashrc
```

```
./configure --prefix="/opt/snort" --with-libnet-includes="/home/test/build/Libnet-1.0.2a/include" --with-libnet-libraries="/opt/libnet/lib" --enable-inline --with-mysql --enable-decoder-preprocessor-rules  
make  
make install
```

```
cd ../FLoP-1.6.1/  
CPPFLAGS=-I/usr/include/mysql ./configure --prefix="/opt/flop" --with-snort=/home/test/build/snort-2.8.5 --with-mysql=/usr --with-libpcap --enable-getpacket  
make  
make install
```

```
cd ..  
mkdir /opt/snort/conf  
cp snortrules-snapshot-CURRENT.tar.gz /opt/snort/conf  
cd /opt/snort/conf  
tar xfv snortrules-snapshot-CURRENT.tar.gz  
rm snortrules-snapshot-CURRENT.tar.gz
```

Mit folgendem Skript eine Kopie aller Regeln anlegen, sodass sie drop statt alert haben, für IPS- und IDS-Betrieb.

```
#!/bin/bash
for f in `ls /opt/snort/conf/rules/*.rules` ;
do
  sed s/^alert/drop/g $f > ${f}.new;
  mv ${f}.new $f;
done
```

```
cd /opt/snort
rm -r src/
vi /etc/stunnel/stunnel.conf
client = yes
#cert = ...
[mysql]
  accept = 127.0.0.1:10440
  connect = 192.168.1.40:10440
chkconfig stunnel 345
```

```
cd /opt/snort/conf
vi rules/mytest.rules
drop tcp 192.168.1.33 any -> 192.168.1.34 22 (classtype:attempted-user; msg:"Port 22 Connection Initiated"; sid:99001;)
```

```
drop icmp 192.168.1.33 any <> 192.168.1.34 any (classtype:attempted-user; msg:"ICMP Echo Request"; icode:0; itype:8; sid:99002;)
```

```
drop udp 192.168.1.33 any <> 192.168.1.34 any (classtype:attempted-user; msg:"DNS Request"; sid:99003;)
```

```
drop ip 192.168.1.33 any -> 192.168.1.34 any (msg:"SHELLCODE x86 inc ecx NOOP"; content:"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"; classtype:shellcode-detect; sid:99004; rev:10;)
```

```
drop tcp any any <> any 80 (uricontent:"listener"; msg:"uricontent test"; sid:500005;)
drop tcp any any -> any 80 (content:"test"; msg:"content test"; sid:500015;)
```

```
mkdir /opt/snort/conf/rules/preproc_rules
mkdir /var/log/snort
```

Hinweise:

Bei Snort inline kann man die uid und gid nicht ändern, es muss als root laufen. Zuerst nur als IDS laufen lassen um zu testen ob alles geht, also nur mit alert statt drop.

```
vi etc/snort.conf (Siehe auch Ende des Documents für eine vollständige snort.conf, hier sind nur die Änderungen der mitinstallierten snort.conf angegeben)
var HOME_NET 192.168.1.0/24
var EXTERNAL_NET any
portvar HTTP_PORTS [80,443]
```

```

var RULE_PATH /opt/snort/conf/rules
preprocessor stream5_udp:
Bei detection engine alle auskommentieren
Bei dynamicdetectionf file alle auskommentieren
# falls später reject-Regeln verwendet werden:
config layer2resets: 00:01:02:03:04:05
# IDS-Betrieb:
#config enable_decode_oversized_alerts
# IPS-Betrieb:
config enable_decode_oversized_drops
config checksum_mode: all
#config disable_decode_alerts
#config disable_tcpopt_experimental_alerts
#config disable_tcpopt_obsolete_alerts
#config disable_ttcp_alerts
#config disable_tcpopt_alerts
#config disable_ipopt_alerts
#config disable_decode_drops

dynamicengine /opt/snort/lib/snort_dynamicengine/libsf_engine.so
Überall die Pfade von /usr/local nach /opt/snort anpassen.
#output log_tcpdump: tcpdump.log
output alert_unixsock_db: /var/run/snort.socket, all
preprocessor sfportscan: proto { all } \
    scan_type { all } \
    memcap { 10000000 } \
    sense_level { low }
preprocessor http_inspect: global iis_unicode_map /opt/snort/conf/etc/unicode.map 1252
preprocessor http_inspect_server: \
    server default \
    ports { 80 443 } \
    apache_whitespace no \
    ascii no \
    bare_byte no \
    chunk_length 500000 \
    flow_depth 1460 \
    directory no \
    double_decode no \
    iis_backslash no \
    iis_delimiter no \
    iis_unicode no \
    multi_slash no \
    non_strict \
    oversize_dir_length 500 \
    u_encode yes \
    non_rfc_char { 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 } \
    webroot no

```

(Die IP ist die vom nachfolgenden Webserver. Parameter wie multi_slash etc. bedeuten, dass sie angewendet werden, no bedeutet kein alert, yes bedeutet mit alert)

Alle includes aktivieren, außer die experimental-rules.
include \$RULE_PATH/mytest.rules

Alle Zeilen mit
dynamicpreprocessor <full path to ...**so**>
einkommentieren und den Pfad anpassen
Alle Zeilen mit
dynamicpreprocessor <full path to ...**dylib**>
auskommentieren

```
cd /home/test/build/  
cp FLoP-1.6.1/contrib/sockserv.init /etc/init.d/sockserv
```

```
vi /etc/init.d/sockserv  
#!/bin/sh
```

```
#####  
# Now to the real heart of the matter:
```

```
# See how we were called.
```

```
case "$1" in  
start)  
    echo -n "Starting sockserv: "  
    /opt/flop/bin/sockserv -S 127.0.0.1 -p 10440 -P /var/run/sockserv.pid -s  
/var/run/snort.socket -b  
    touch /var/lock/subsys/sockserv  
    echo  
    ;;  
stop)  
    echo -n "Stopping sockserv: "  
    killproc sockserv  
    rm -f /var/run/sockserv.pid  
    rm -f /var/lock/subsys/sockserv  
    echo  
    ;;  
reload)  
    echo -n "Sending HUP to sockserv process(es): "  
    killproc sockserv -HUP  
    echo  
    ;;  
restart)  
    $0 stop  
    $0 start  
    ;;  
condrestart)  
    if [ "x`pidof sockserv`" != x ]; then  
        $0 stop  
        $0 start  
    fi  
    ;;  
status)  
    status sockserv  
    ;;
```

```
*)
    echo "Usage: $0 {start|stop|reload|restart|status}"
    exit 1
esac

exit 0
```

```
chmod 0700 /etc/init.d/sockserv
chkconfig sockserv 345
```

Hinweis: Der Socket (/var/run/snort.socket) wird von sockserv angelegt, nicht von Snort.

```
vi /etc/init.d/snort
#!/bin/sh
```

```
#####
```

```
# Now to the real heart of the matter:
```

```
# See how we were called.
```

```
case "$1" in
```

```
start)
```

```
    # paket-informationen in den user space hochreichen. Direktes bridging
    # unterbinden, sodass Snort bei Bedarf Pakete verwerfen kann.
```

```
    /usr/sbin/iptables -P FORWARD DROP
```

```
    /usr/sbin/iptables -A FORWARD -j QUEUE
```

```
    echo -n "Starting snort: "
```

```
    /opt/snort/bin/snort -Q -c /opt/snort/conf/etc/snort.conf -D
```

```
    touch /var/lock/subsys/snort
```

```
    echo
```

```
    ;;
```

```
stop)
```

```
    echo -n "Stopping snort: "
```

```
    killproc snort
```

```
    rm -f /var/run/snort.pid
```

```
    rm -f /var/lock/subsys/snort
```

```
    echo
```

```
    ;;
```

```
reload)
```

```
    echo -n "Sending HUP to snort process(es): "
```

```
    killproc snort -HUP
```

```
    echo
```

```
    ;;
```

```
restart)
```

```
    $0 stop
```

```
    $0 start
```

```
    ;;
```

```

condrestart)
    if [ "x`pidof snort`" != x ]; then
        $0 stop
        $0 start
    fi
    ;;
status)
    status snort
    ;;
*)
    echo "Usage: $0 {start|stop|reload|restart|status}"
    exit 1
esac

exit 0

```

```

chmod 0700 /etc/init.d/snort

```

Sobald der Log-Host eingerichtet ist:

```

/etc/init.d/stunnel start
/etc/init.d/sockserv start      (horcht auf keinem Port)
/etc/init.d/snort start

```

```

chkconfig snort 345
find /etc/rc.d/ -iname "*snort*"
mv /etc/rc.d/rc5.d/S01snort /etc/rc.d/rc5.d/S99snort
mv /etc/rc.d/rc5.d/K21snort /etc/rc.d/rc5.d/K99snort

```

snort und sockserv in den Runleveln auf K99 K98 S99 S98 verlinken

/var/log/messages durchsehen nach Fehlern.

Wenn Dienste beim Booten nicht automatisch starten auch in
/var/log/boot.omsg nachsehen.

Sobald erste Events in Sorby zu sehen sind, manuell das Interface setzen:

Auf dem Loghost in mysql:

```

update snort.sensor set interface='br0' where sid=1;

```

alert

```
cp /home/test/build/FLoP-1.6.1/contrib/alert.init /etc/init.d/alert
chmod 0700 /etc/init.d/alert
chkconfig alert 345
```

```
vi /etc/init.d/alert
```

```
#!/bin/sh
```

```
#####  
# Now to the real heart of the matter:
```

```
# See how we were called.
```

```
case "$1" in
```

```
start)
```

```
    echo -n "Starting alert: "  
    /opt/flop/bin/alert -c /opt/flop/etc/flop/alert.conf  
    touch /var/lock/subsys/alert  
    echo  
    ;;
```

```
stop)
```

```
    echo -n "Stopping alert: "  
    killproc alert  
    rm -f /var/run/alert.pid  
    rm -f /var/lock/subsys/alert  
    echo  
    ;;
```

```
reload)
```

```
    echo -n "Sending HUP to alert process(es): "  
    killproc alert -HUP  
    echo  
    ;;
```

```
restart)
```

```
    $0 stop  
    $0 start  
    ;;
```

```
condrestart)
```

```
    if [ "x`pidof alert`" != x ]; then  
        $0 stop  
        $0 start  
    fi  
    ;;
```

```
status)
```

```
    status alert  
    ;;
```

```
*)
```

```
    echo "Usage: $0 {start|stop|reload|restart|status}"  
    exit 1
```

```
esac
```

```
exit 0
```

vi /opt/flop/etc/flop/alert.conf

MailServer: localhost
MailRecipient: root
MailSender: alert@snort-loghost.homenet
MailDomain: snort-loghost.homenet
MailPort: 25
SocketName: /var/run/alert.socket
AlarmDelay: 1
AlarmLevel: 0
MaxCount: 10
FQNNames: 0
DaemonMode: 1
PIDFile: /var/run/alert.pid
Debug: all,0

**Den Cron-Eintrag für /root/firewall_off.sh auskommentieren !!!
chmod 0000 /root/firewall_off.sh !!!
/etc/init.d/firewall.sh start !!!**

Umstellen auf IPS: drop statt alert.

Zuerst im Vordergrund starten und die gesamte Ausgabe anschauen.

Das include für die Testrules in der snort.conf auskommentieren.

Vollständige snort.conf

```
var HOME_NET 192.168.1.34
var EXTERNAL_NET any
```

```
var DNS_SERVERS $HOME_NET
var SMTP_SERVERS $HOME_NET
var HTTP_SERVERS $HOME_NET
var SQL_SERVERS $HOME_NET
var TELNET_SERVERS $HOME_NET
var SNMP_SERVERS $HOME_NET
var FTP_SERVERS $HOME_NET
var SSH_SERVERS $HOME_NET
var POP_SERVERS $HOME_NET
var IMAP_SERVERS $HOME_NET
var RPC_SERVERS $HOME_NET
var WWW_SERVERS $HOME_NET
var AIM_SERVERS
```

```
[64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.188.3.0/24,205.188.5.0/24,205.188.7.0/24,205.188.9.0/24,205.188.153.0/24,205.188.179.0/24,205.188.248.0/24]
```

```
portvar HTTP_PORTS [80,2301,3128,7777,7779,8000,8008,8028,8080,8180,8888,9999]
portvar SHELLCODE_PORTS any
portvar ORACLE_PORTS 1024:
portvar AUTH_PORTS 113
portvar DNS_PORTS 53
portvar FINGER_PORTS 79
portvar FTP_PORTS 21
portvar IMAP_PORTS 143
portvar IRC_PORTS [6665,6666,6667,6668,6669,7000]
portvar MSSQL_PORTS 1433
portvar NNTP_PORTS 119
portvar POP2_PORTS 109
portvar POP3_PORTS 110
portvar SUNRPC_PORTS
[111,32770,32771,32772,32773,32774,32775,32776,32777,32778,32779]
portvar RLOGIN_PORTS 513
portvar RSH_PORTS 514
portvar SMB_PORTS [139,445]
portvar SMTP_PORTS 25
portvar SNMP_PORTS 161
portvar SSH_PORTS 22
portvar TELNET_PORTS 23
portvar MAIL_PORTS [25,143,465,691]
portvar SSL_PORTS [25,443,465,636,993,995]
portvar DCERPC_NCACN_IP_TCP [139,445]
portvar DCERPC_NCACN_IP_UDP [138,1024:]
portvar DCERPC_NCACN_IP_LONG [135,139,445,593,1024:]
portvar DCERPC_NCACN_UDP_LONG [135,1024:]
portvar DCERPC_NCACN_UDP_SHORT [135,593,1024:]
portvar DCERPC_NCACN_TCP [2103,2105,2107]
portvar DCERPC_BRIGHTSTORE [6503,6504]
```

```
var RULE_PATH /opt/snort/conf/rules
```

```
config enable_decode_oversized_drops
config checksum_mode: all
config pcre_match_limit: 1500
config pcre_match_limit_recursion: 1500
```

```
config layer2resets: 00:01:02:03:04:05
```

```
dynamicpreprocessor file /opt/snort/lib/snort_dynamicpreprocessor/libsf_dce2_preproc.so
dynamicpreprocessor file /opt/snort/lib/snort_dynamicpreprocessor/libsf_dns_preproc.so
dynamicpreprocessor file
/opt/snort/lib/snort_dynamicpreprocessor/libsf_ftptelnet_preproc.so
dynamicpreprocessor file /opt/snort/lib/snort_dynamicpreprocessor/libsf_smtp_preproc.so
dynamicpreprocessor file /opt/snort/lib/snort_dynamicpreprocessor/libsf_ssh_preproc.so
dynamicpreprocessor file /opt/snort/lib/snort_dynamicpreprocessor/libsf_ssl_preproc.so
dynamicengine /opt/snort/lib/snort_dynamicengine/libsf_engine.so
```

```
preprocessor frag3_global: max_fragments 65536
preprocessor frag3_engine: policy windows timeout 180
preprocessor stream5_global: max_tcp 8192, track_tcp yes, \
    track_udp yes
preprocessor stream5_tcp: policy windows, use_static_footprint_sizes, \
    ports client 21 22 23 25 42 53 79 80 109 110 111
113 119 135 136 137 139 143 110 111 161 445 513 514 691 1433 1521 2100 2301 3128
3306 6665 6666 6667 6668 6669 7000 8000 8080 8180 8888 32770 32771 32772 32773
32774 32775 32776 32777 32778 32779, \
    ports both 443 465 563 636 989 992 993 994 995
preprocessor stream5_udp:
preprocessor http_inspect: global iis_unicode_map /opt/snort/conf/etc/unicode.map 1252
preprocessor http_inspect_server: \
    server default \
    ports { 80 443 } \
    apache_whitespace no \
    ascii no \
    bare_byte no \
    chunk_length 500000 \
    flow_depth 1460 \
    directory no \
    double_decode no \
    iis_backslash no \
    iis_delimiter no \
    iis_unicode no \
    multi_slash no \
    non_strict \
    oversize_dir_length 500 \
    u_encode yes \
    non_rfc_char { 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 } \
    webroot no
preprocessor rpc_decode: 111 32770 32771 32772 32773 32774 32775 32776 32777
32778 32779
```

```

preprocessor bo
preprocessor ftp_telnet: \
  global \
  encrypted_traffic yes \
  check_encrypted \
  inspection_type stateful
preprocessor ftp_telnet_protocol: \
  telnet \
  ayt_attack_thresh 20 \
  normalize_ports { 23 } \
  detect_anomalies
preprocessor ftp_telnet_protocol: \
  ftp server default \
  def_max_param_len 100 \
  ports { 21 2100 } \
  ftp_cmds { USER PASS ACCT CWD SDUP SMNT QUIT REIN PORT PASV TYPE
STRU MODE } \
  ftp_cmds { RETR STOR STOU APPE ALLO REST RNFR RNT0 ABOR DELE RMD
MKD PWD } \
  ftp_cmds { LIST NLST SITE SYST STAT HELP NOOP } \
  ftp_cmds { AUTH ADAT PROT PBSZ CONF ENC } \
  ftp_cmds { FEAT OPTS CEL CMD MACB } \
  ftp_cmds { MDTM REST SIZE MLST MLSD } \
  ftp_cmds { XPWD XCWD XCUP XMKD XRMD TEST CLNT } \
  alt_max_param_len 0 { CDUP QUIT REIN PASV STOU ABOR PWD SYST NOOP } \
  alt_max_param_len 100 { MDTM CEL XCWD SITE USER PASS REST DELE RMD
SYST TEST STAT MACB EPSV CLNT LPRT } \
  alt_max_param_len 200 { XMKD NLST ALLO STOU APPE RETR STOR CMD RNFR
HELP } \
    alt_max_param_len 256 { RNT0 CWD } \
  alt_max_param_len 400 { PORT } \
    alt_max_param_len 512 { SIZE } \
  chk_str_fmt { USER PASS ACCT CWD SDUP SMNT PORT TYPE STRU MODE } \
  chk_str_fmt { RETR STOR STOU APPE ALLO REST RNFR RNT0 DELE RMD MKD } \
  chk_str_fmt { LIST NLST SITE SYST STAT HELP } \
  chk_str_fmt { AUTH ADAT PROT PBSZ CONF ENC } \
  chk_str_fmt { FEAT OPTS CEL CMD } \
  chk_str_fmt { MDTM REST SIZE MLST MLSD } \
  chk_str_fmt { XPWD XCWD XCUP XMKD XRMD TEST CLNT } \
  cmd_validity MODE < char ASBCZ > \
  cmd_validity STRU < char FRP > \
  cmd_validity ALLO < int [ char R int ] > \
  cmd_validity TYPE < { char AE [ char NTC ] | char I | char L [ number ] } > \
  cmd_validity MDTM < [ date nnnnnnnnnnnnnnn[n[n[n]]] ] string > \
  cmd_validity PORT < host_port >
preprocessor ftp_telnet_protocol: \
  ftp client default \
  max_resp_len 256 \
  bounce yes \
  telnet_cmds no
preprocessor SMTP: \
  ports { 25 465 691 } \

```

```
inspection_type stateful \
normalize cmds \
valid_cmds { MAIL RCPT HELP HELO ETRN EHLO EXPN VRFY ATRN SIZE BDAT
DEBUG EMAL ESAM ESND ESOM EVFY IDENT NOOP RSET SEND SAML SOML
AUTH TURN ETRN PIPELINING CHUNKING DATA DSN RSET QUIT ONEX QUEU
STARTTLS TICK TIME TURNME VERB X-EXPS X-LINK2STATE XADR XAUTH XCIR
XEXCH50 XGEN XLICENSE XQUEU XSTA XTRN XUSR } \
    normalize_cmds { MAIL RCPT HELP HELO ETRN EHLO EXPN VRFY ATRN SIZE
BDAT DEBUG EMAL ESAM ESND ESOM EVFY IDENT NOOP RSET SEND SAML SOML
AUTH TURN ETRN PIPELINING CHUNKING DATA DSN RSET QUIT ONEX QUEU
STARTTLS TICK TIME TURNME VERB X-EXPS X-LINK2STATE XADR XAUTH XCIR
XEXCH50 XGEN XLICENSE XQUEU XSTA XTRN XUSR } \
    max_header_line_len 1000 \
    max_response_line_len 512 \
    alt_max_command_line_len 260 { MAIL } \
    alt_max_command_line_len 300 { RCPT } \
    alt_max_command_line_len 500 { HELP HELO ETRN EHLO } \
    alt_max_command_line_len 255 { EXPN VRFY ATRN SIZE BDAT DEBUG EMAL
ESAM ESND ESOM EVFY IDENT NOOP RSET } \
    alt_max_command_line_len 246 { SEND SAML SOML AUTH TURN ETRN PIPELINING
CHUNKING DATA DSN RSET QUIT ONEX } \
    alt_max_command_line_len 246 { QUEU STARTTLS TICK TIME TURNME VERB X-
EXPS X-LINK2STATE XADR } \
    alt_max_command_line_len 246 { XAUTH XCIR XEXCH50 XGEN XLICENSE XQUEU
XSTA XTRN XUSR } \
    xlink2state { enable }
preprocessor arpspoof
preprocessor arpspoof_detect_host: 192.168.1.34 00:02:3f:18:59:0f
... arpspoof Liste vervollständigen ...
preprocessor ssl: ports { 443 465 563 636 989 992 993 994 995 }, trustservers,
noinspect_encrypted
preprocessor dcerpc2: memcap 102400, events [co ]
preprocessor dcerpc2_server: default, policy WinXP, \
    detect [smb [139,445], tcp 135, udp 135, rpc-over-http-server 593], \
    autodetect [tcp 1025:, udp 1025:, rpc-over-http-server 1025:], \
    smb_max_chain 3
preprocessor dns: ports { 53 } enable_rdata_overflow
preprocessor sfportscan: proto { all } \
    scan_type { all } \
    memcap { 10000000 } \
    sense_level { low }
#output log_tcpdump: tcpdump.log
output alert_unixsock_db: /var/run/snort.sock, all

include classification.config
include reference.config
include $RULE_PATH/local.rules
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/scan.rules
include $RULE_PATH/finger.rules
include $RULE_PATH/ftp.rules
```

include \$RULE_PATH/telnet.rules
include \$RULE_PATH/rpc.rules
include \$RULE_PATH/rservices.rules
include \$RULE_PATH/dos.rules
include \$RULE_PATH/ddos.rules
include \$RULE_PATH/dns.rules
include \$RULE_PATH/tftp.rules
include \$RULE_PATH/web-cgi.rules
include \$RULE_PATH/web-coldfusion.rules
include \$RULE_PATH/web-iis.rules
include \$RULE_PATH/web-frontpage.rules
include \$RULE_PATH/web-misc.rules
include \$RULE_PATH/web-client.rules
include \$RULE_PATH/web-php.rules
include \$RULE_PATH/sql.rules
include \$RULE_PATH/x11.rules
include \$RULE_PATH/icmp.rules
include \$RULE_PATH/netbios.rules
include \$RULE_PATH/misc.rules
include \$RULE_PATH/attack-responses.rules
include \$RULE_PATH/oracle.rules
include \$RULE_PATH/mysql.rules
include \$RULE_PATH/snmp.rules
include \$RULE_PATH/smtp.rules
include \$RULE_PATH/imap.rules
include \$RULE_PATH/pop2.rules
include \$RULE_PATH/pop3.rules
include \$RULE_PATH/nntp.rules
include \$RULE_PATH/other-ids.rules
include \$RULE_PATH/web-attacks.rules
include \$RULE_PATH/backdoor.rules
include \$RULE_PATH/shellcode.rules
include \$RULE_PATH/policy.rules
include \$RULE_PATH/porn.rules
include \$RULE_PATH/info.rules
include \$RULE_PATH/icmp-info.rules
include \$RULE_PATH/virus.rules
include \$RULE_PATH/chat.rules
include \$RULE_PATH/multimedia.rules
include \$RULE_PATH/p2p.rules
include \$RULE_PATH/spyware-put.rules
include \$RULE_PATH/specific-threats.rules
include \$RULE_PATH/content-replace.rules
include \$RULE_PATH/voip.rules
include \$RULE_PATH/mytest.rules

http-Test

```
cd /home/test/build
wget http://www.cirt.net/nikto/nikto-2.03.tar.bz2
tar xjf nikto-2.03.tar.bz2
cd nikto...
vi nikto.pl, den absoluten Pfad für die config.txt setzen
$NIKTO{configfile} = "/home/test/build/nikto/config.txt";
./nikto.pl -update
```

Scan:

```
./nikto.pl -Cgdirs all --host 192.168.1.34 --port 80
./nikto.pl -Cgdirs all --host 192.168.1.34 --port 443
```

Ergänzende Informationen

Gut merken welches NIC der Bridge das interne und welches das externe ist. Dazu unbedingt die MAC-Adressen aufschreiben und in udev schauen welcher Devicename zugeordnet wird, da man das später remote schlecht oder garnicht herausfinden kann.

Der Zugriff mit stunnel auf den stunnel-Server ist geschützt durch iptables auf dem Sensor und auf dem Loghost und durch die Einträge in den stunnel.conf-Dateien. Zertifikate werden nicht verwendet.

Wenn man Snorby im LiveLook offen lässt kann es leider passieren, insbesondere wenn viele Events eintreffen, dass der automatische Update der Seite nicht hinterherkommt und quasi endlos läuft. Dies führt zu immer mehr Forks auf dem Loghost bis er lahmgelegt ist. Sobald man den LiveLook schließt normalisiert sich wieder alles. Also darauf achten, dass der LiveLook nicht dauernd offen ist, bis der Bug behoben ist.

Wie leert man die DB wenn sie zu voll wird, ohne dass es zu Inkonsistenzen kommt? Zuerst sockserv auf dem Loghost abschalten, damit keine Daten mehr in die DB gelangen. Dann ein Dump der DB machen, falls man Archivieren möchte. Anschließend die DB resetten, wie weiter unten beschrieben.

Die Datenbank resetten (Achtung, während dem Reset dürfen keine Daten in die DB geschrieben werden!)

```
cd /srv/www/htdocs/snorby
/opt/ruby/bin/rake snorby:reset RAILS_ENV=production
```

Alle Datenbankerweiterungen aus /FLoP-1.6.1/README.payload in MySQL ausführen. Achtung, schema muss in Backticks: `schema`
use snort;

```
ALTER TABLE data ADD COLUMN data_header TEXT;
ALTER TABLE data ADD COLUMN pcap_header TEXT;
ALTER TABLE `schema` ADD COLUMN full_payload SMALLINT;
UPDATE `schema` SET full_payload=1;
```

```
ALTER TABLE event ADD COLUMN reference INT8;
ALTER TABLE `schema` ADD COLUMN reference SMALLINT;
UPDATE `schema` SET reference=1;
```

Die Signaturen und Referenzen müssen auch wieder importiert werden:

```
cd /home/test/build/FLoP-1.6.1/contrib
./rules.pl -c ./rules.pl.conf
```

```
/etc/init.d/apache2 restart
```

Die zweite Regel (QUEUE) in iptables FORWARD löschen:
iptables -D FORWARD 2

Im IPS-Betrieb muss Snort laufen, sonst geht keinerlei Traffic durch die Bridge. Die Policy für iptables FORWARD im IPS-Betrieb muss IMMER auf DROP stehen und

QUEUE muss aktiviert sein, sonst geht der Traffic an Snort vorbei.

...

Warning: 'ignore_any_rules' option for Stream5 UDP disabled because of UDP rule with flow or flowbits option

Warning: flowbits key 'trojan.delf.post' is set but not ever checked.

Warning: flowbits key 'asp.upload' is set but not ever checked.

....

363 out of 512 flowbits in use.

Das bedeutet einfach nur, dass es Rules gibt in denen flowbits gesetzt ist, aber es keine Rules gibt wo diese auch verwendet werden, was aber kein Problem darstellt.

pcap-datei

```
vi /opt/flop/etc/flop/getpacket.conf
```

```
DBuser : snort # huhu
```

```
DBpassword: "snort"
```

```
DBname : snort
```

```
DBType: MySQL
```

```
SocketName: 127.0.0.1:3306
```

```
Reference: 1
```

```
Info: 1
```

```
VendorInfo: 1
```

```
/opt/flop/bin/getpacket -C 1 -S 1 -a -t -w /root/pcapfile /opt/flop/etc/flop/getpacket.conf
```

Was ist mit ebtables und arptables?

Wäre natürlich gut, aber es wird dann alles noch viel komplexer.

Wenn kein Traffic durch die Bridge geht, schauen ob auf dem davor liegenden Switch/Router z.B. statisches dhcp eingerichtet ist, oder ähnliches was Mac-Adressen betrifft.

```
arp -v
```

```
arp -d host
```

```
vi /etc/udev/rules.d/30-net_persistent...
```

UDP-Test

```
hping3 -2 -p 7 192.168.1.34 -d 50
```

Event-Priorität:

Die höchste Priorität, das heisst das kritischste, ist 1.

Sie dazu auch Datei classification.config oder DB-Tabelle sig_class.

Pflege des Systems

Dienste die laufen müssen:

Sensor:

iptables-Firewall/Bridge

Snort muss laufen, sonst geht kein Traffic durch die Bridge!

sockserv

stunnel

Loghost:

Snorby

MySQL

servsock

alert

stunnel

Was muss mindestens aktuell gehalten werden:

Snort

Snort-Regeln

FLoP (sockserv und servsock)

Snorby

Was muss man generell im Auge behalten:

RAM-Auslastung auf dem Sensor.

Festplattenauslastung/performance bzw. DB-Größe auf dem Loghost.

Firewall-Logs auf dem Sensor.

/var/log/messages auf dem Sensor.

Debugging

Wenn z.B. kein Traffic durch die Bridge geht oder es zu Problemen mit der Anwendung kommt prüfen:

Ist der Server hochgefahren?

Sind die NICs als Bridge konfiguriert?

Ist iptables FORWARD mit -j QUEUE konfiguriert?

Läuft Snort?

Möchte man temporär vorrangig den reibungslosen Durchgang des Traffics sicherstellen, (auf Kosten der Sicherheit) kann man Snort aus dem Szenario herausnehmen und alles ohne Prüfung direkt bridgen. Dazu einfach die zweite Regel (QUEUE) in iptables FORWARD löschen und die Policy in FORWARD auf ACCEPT stellen:

```
iptables -D FORWARD 2
```

```
iptables -P FORWARD ACCEPT
```

Wenn keine Daten auf dem Loghost ankommen:

Sicherstellen, dass alle beteiligten Dienste laufen.

Dann zuerst in der Tabelle event auf dem Loghost schauen ob es dort neue Einträge gibt.

Wenn nicht, mit tcpdump schauen ob stunnel auf dem Loghost Daten erhält.

Wenn nicht, mit tcpdump auf dem Sensor schauen ob stunnel dort Daten erhält.

Wenn nicht sockserv auf dem Sensor im Vordergrund mit Debugging laufen lassen, dazu -v,3 an den Aufruf anhängen.

Wenn auch hier nichts auffällt kann man noch Snort mit einem Standard-Outputplugin laufen lassen und Daten in eine lokale Textdatei schreiben lassen und schauen ob dort Daten ankommen.

Wenn Verdacht auf einen ernsthaften Einbruchsversuch besteht:

In Snorby die Source-IP-Adresse ermitteln. Per Firewall diese IP komplett sperren lassen.

Im alleräußersten Notfall die Bridge komplett sperren, indem man die Policy der FORWARD chain auf DROP stellt (sollte bereits der Fall sein) und die Regel in der FORWARD chain, die -j QUEUE enthält, löscht:

```
iptables -P FORWARD DROP
```

```
iptables --list
```

FORWARD sollte nur eine Regel enthalten, nämlich die besagte mit -j QUEUE.

Da es die erste Regel ist, mit 1 löschen:

```
iptables -D FORWARD 1
```

Achtung: Es hilft nichts wenn man versucht die INPUT oder OUTPUT chains für die Bridge-NICs zu sperren, da durch das Bridging diese iptables-Regeln für diese NICs garnicht angewendet werden!