BNSS Documentation

Intrusion detection

We installed snort on VM1 for intrusion detection. We added a simple test rule

```
alert icmp any any -> any any (msg: "IDS Demonstration: ICMP Packet found"; itype:8; sid:1;dsize:>64;)
```

to trigger an alarm message as soon as a larger ICMP echo request is detected and included this rule together with the default rules in /etc/snort/snort.conf.

```
include $RULE_PATH/test.rules
```

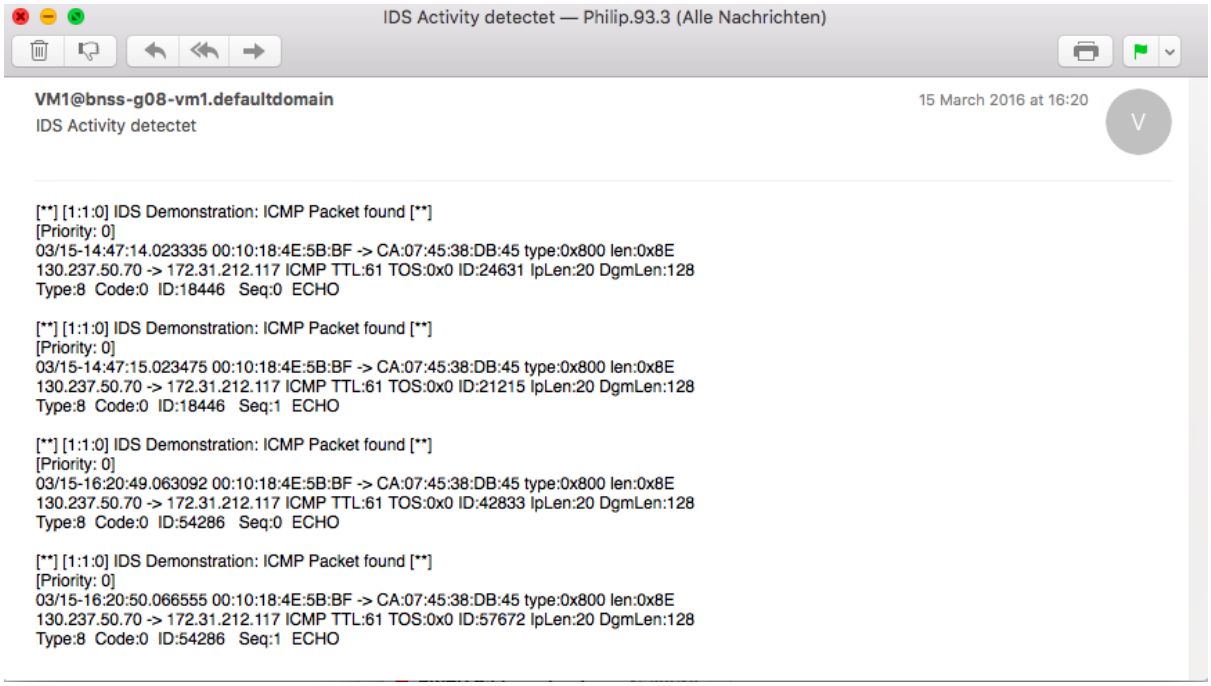
For email notification of changes we wrote a simple bash script to check the log file (/var/log/snort/alert) for changes and send an email containing the last entries of the log file.

```
root@bnss-g08-vm1:/# cat /etc/snort/mail.sh
#!/bin/bash
S1=`cat /var/log/snort/alert.timestamp`
S2=`ls -lah /var/log/snort/alert | awk '{ print $6 $7 $8 }'`
if [ $S1 == $S2 ]
then
        echo "No Email"
else
        echo "Log File has changed - sending email"
        echo "$S1"
        echo "$S2"
        sendmail syafiq132@gmail.com,philip.93.3@gmail.com,antoinehonor@gmail.com -i <<EOF
From:VM1
Subject:IDS Activity detectet
$(tail -n 24 /var/log/snort/alert)
EOF
```

Using crontab this script is executed every 5 minutes. The 5 minute interval seemed a reasonable tradeoff between performance and reaction time.

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
*/5 *   * * *   root    /etc/snort/mail.sh
#
```

By pinging VM1 with larger ICMP packets we generated alerts and an exemplary email looks like:


IDS Activity detectet — Philip.93.3 (Alle Nachrichten)
VM1@bnss-g08-vm1.defaultdomain
15 March 2016 at 16:20
IDS Activity detectet
V
[**] [1:1:0] IDS Demonstration: ICMP Packet found [**]
[Priority: 0]
03/15-14:47:14.023335 00:10:18:4E:5B:BF -> CA:07:45:38:DB:45 type:0x800 len:0x8E
130.237.50.70 -> 172.31.212.117 ICMP TTL:61 TOS:0x0 ID:24631 lpLen:20 DgmLen:128
Type:8 Code:0 ID:18446 Seq:0 ECHO
[**] [1:1:0] IDS Demonstration: ICMP Packet found [**]
[Priority: 0]
03/15-14:47:15.023475 00:10:18:4E:5B:BF -> CA:07:45:38:DB:45 type:0x800 len:0x8E
130.237.50.70 -> 172.31.212.117 ICMP TTL:61 TOS:0x0 ID:21215 lpLen:20 DgmLen:128
Type:8 Code:0 ID:18446 Seq:1 ECHO
[**] [1:1:0] IDS Demonstration: ICMP Packet found [**]
[Priority: 0]
03/15-16:20:49.063092 00:10:18:4E:5B:BF -> CA:07:45:38:DB:45 type:0x800 len:0x8E
130.237.50.70 -> 172.31.212.117 ICMP TTL:61 TOS:0x0 ID:42833 lpLen:20 DgmLen:128
Type:8 Code:0 ID:54286 Seq:0 ECHO
[**] [1:1:0] IDS Demonstration: ICMP Packet found [**]
[Priority: 0]
03/15-16:20:50.066555 00:10:18:4E:5B:BF -> CA:07:45:38:DB:45 type:0x800 len:0x8E
130.237.50.70 -> 172.31.212.117 ICMP TTL:61 TOS:0x0 ID:57672 lpLen:20 DgmLen:128
Type:8 Code:0 ID:54286 Seq:1 ECHO

VM1

OpenVPN

Installation :

```
root@bnss-g08-vm1:/home/atiiq# apt-get install openvpn
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Configuration :

```
root@bnss-g08-vm1:/etc# more /etc/openvpn/tun_netgear/server.conf

########SERVER CONF#########
port 1195
proto udp
dev tun
secret netgear_vm1.key
ifconfig 10.8.0.1 10.8.0.2
push "route 192.168.100.0 255.255.255.0"
keepalive 10 120
comp-lzo
persist-key
persist-tun
;status openvpn-status.log
;log         openvpn.log
;log-append  openvpn.log
verb 3
;mute 20
ping-timer-rem
daemon
```

Starting openvpn daemon :

```
root@bnss-g08-vm1:/etc# openvpn /etc/openvpn/tun_netgear/server.conf
```

Log can be turned on or off, located in `/etc/openvpn/tun_netgear/openvpn.log`

Freeradius

Installation :

```
root@bnss-g08-vm1:/home/atiiq# apt-get install freeradius
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Configuration :

Main configuration :
`/etc/freeradius/eap.conf`
`/etc/freeradius/radiusd.conf`
`/etc/freeradius/clients.conf`

Other configuration (leave as default) :
`/etc/freeradius/*.conf`

Certs :
Mainly all, the certs are located in `/var/certs/freeradius/`

Some important files are :
`/var/certs/freeradius/ca.cnf`
`/var/certs/freeradius/client.cnf`
`/var/certs/freeradius/server.cnf`

Starting freeradius service :
`root@bnss-g08-vm1:/home/atiiq# freeradius -X` (with debugging mode)

Create certificate for server :
```
make /var/certs/freeradius/ca.pem
make /var/certs/freeradius/ca.der
make /var/certs/freeradius/printca
```

Before that, we need to change the server.cnf files as follows :

```
[ CA_default ]
..
default_days = 1825
default_md = sha1
..
[ req ]
..
default_bits = 4096
..
```

With the same way, we can generate client certificate by changing client.cnf files, and execute the following :
```
make client.pem
make client_android.p12
```
Revoking Certificate :
```
openssl ca -revoke CERT_TO_REVOKE.pem -keyfile ca.key -cert ca.pem -config ./ca.cnf

openssl ca -gencrl -keyfile ca.key -cert ca.pem -out mycrl.pem -config ca.cnf
cat ca.pem mycrl.pem > ca_and_crl.pem
```

`ca_and_crl.pem` should be the new certificate, and be sure that `eap.conf` pointing the pem file to the new generated one.

`killall -HUP radius` (or just restart freeradius service)

Source :
Installation : http://www.ossramblings.com/using-freeradius-ubuntu-server-wifi
Revoke CA : http://www.linuxjournal.com/article/8151

IPTables

Rules

```
# Generated by iptables-save v1.4.12 on Thu Mar 17 22:52:47 2016
*nat
:PREROUTING ACCEPT [8271:629972]
```

```
:INPUT ACCEPT [2720:251730]
:OUTPUT ACCEPT [8308:547615]
:POSTROUTING ACCEPT [13652:905651]
COMMIT
# Completed on Thu Mar 17 22:52:47 2016
# Generated by iptables-save v1.4.12 on Thu Mar 17 22:52:47 2016
*filter
:INPUT DROP [1054:39496]
:FORWARD ACCEPT [11256:782622]
:OUTPUT ACCEPT [30527:23171514]
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 443 -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -s 10.8.0.2/32 -p icmp -j ACCEPT
-A INPUT -s 172.31.212.0/24 -j ACCEPT
-A INPUT -s 130.237.0.0/16 -j ACCEPT
-A INPUT -i eth1 -j ACCEPT
-A FORWARD -i tun0 -j ACCEPT
-A FORWARD -i tun0 -o eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i eth0 -o tun0 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -p icmp -m icmp --icmp-type 0 -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
COMMIT
```

Allow VM1 to do packet forwarding :

```
root@bnss-g08-vm1:/home/atiiq# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Webserver VM1

Basic apache installation, with redirection to https for every http request.

Installation :

```
root@bnss-g08-vm1:/etc/apache2# apt-get install apache2
Reading package lists... Done
Building dependency tree
```

```
Reading state information... Done

root@bnss-g08-vm1:/etc/apache2# aptitude install -y libapache2-mod-proxy-html
libxml2-dev
```

Configuration :
/etc/apache2/apache2.conf
/etc/apache2/sites-enabled/000-default

Additional module :
mod_proxy [1] used as reverse proxy, to forward HTTP request to backend webserver (VM4) without having real route to the backend network. Users from the rest of the world can only reach and know VM1 as their webserver. On the other hand, user from stockholm and london have direct access to VM4.

Reverse proxy config :

```
root@bnss-g08-vm1:/home/atiiq# more /etc/apache2/sites-enabled/000-default

<VirtualHost *:443>
…
        ProxyPass                   /vm4    http://192.168.100.52/
        ProxyPassReverse            /vm4    http://192.168.100.52/
</VirtualHost>
```

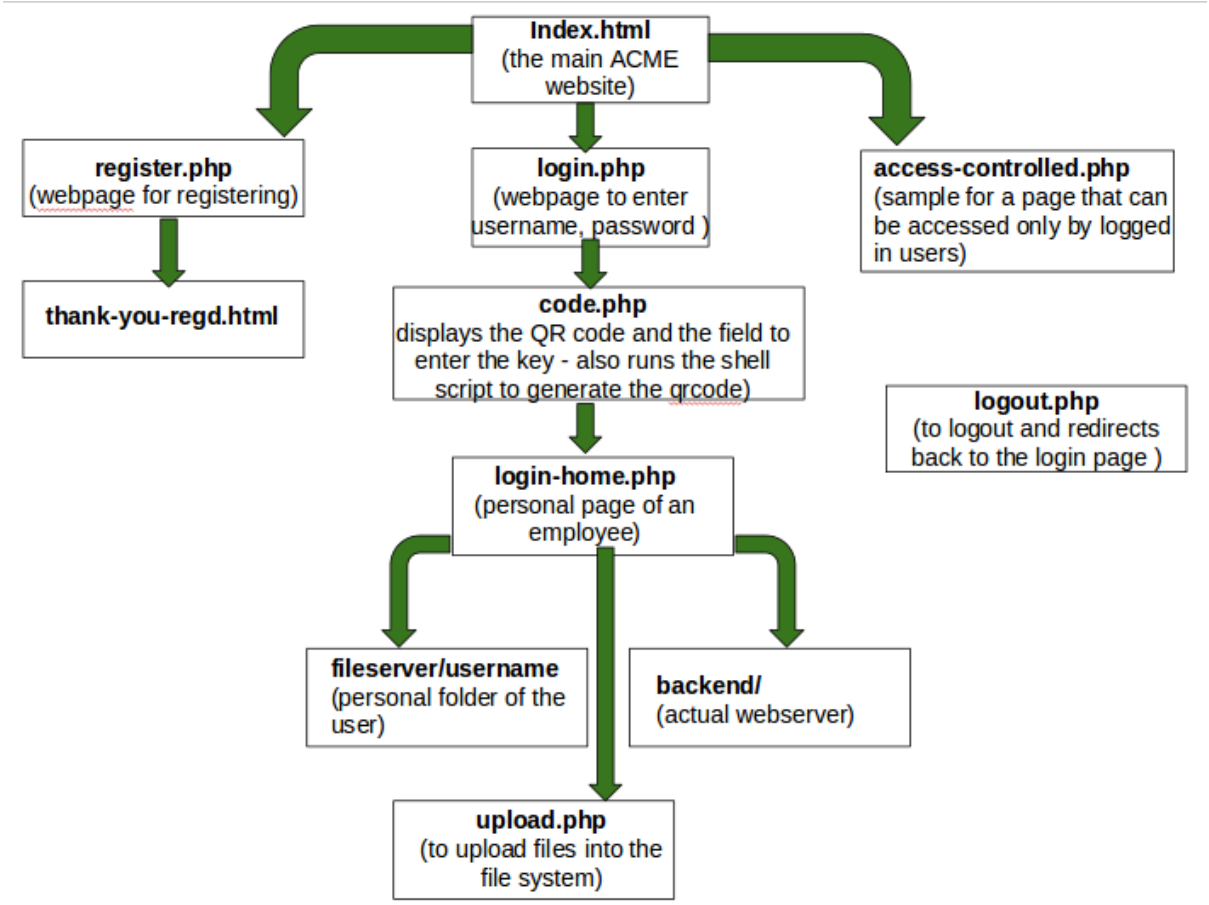
[1]. https://httpd.apache.org/docs/2.4/mod/mod_proxy.html

Two Factor Authentication :

- ACME Login System :

  Created webpages using php/html/javascript coding (which can be found in /var/www/ on VM1) that has been included in the webserver-fileserver-DB/var/www/ folder. To store the user info, a mySQL database was used and it's configuration file can be found in webserver-fileserver-DB/etc/mysql/ . The passwords of the employees were hashed and then stored in the database. This database was then connected with the apache server on VM1. The webserver mainly allows the employees to create accounts (following which the php code takes the input and inserts them into the DB and the code for this lies in the /var/www/include/ folder), access their file system, upload files, access their personal webpage etc.. As a part of the two step authentication to validate the employees working from home, first they have to login using their username and password following which the QR code of that employee (which is linked to their employee ID that is unknown to them) pops up. Using their trusted device they have to scan this QR code and enter the key in the provided field. This will be compared with the actual key (which gets generated in /var/www/otp/) and if match is found, they get logged in successfully. Conditions and checks to ensure that only logged in users can access the webpages are also taken care of. User sessions are also created to ensure one user cannot not access the pages of another user.

Flow chart of the code files (covers the main code files used) :



Netgear OpenWRT

Openvpn

Installation :

```
opkg install openvpn
```

Configuration :

```
# more /etc/openvpn/client.conf

dev tun
proto udp
remote 172.31.212.117 1195
```

```
ifconfig 10.8.0.2 10.8.0.1
persist-key
persist-tun
secret netgear_vm1.key
comp-lzo
verb 3
keepalive 10 120
ping-timer-rem
Daemon
```

Starting Daemon

```
# openvpn /etc/openvpn/client.conf
```

Add route from local network to internal network stockholm :

```
# route add -net 192.168.100.0 netmask 255.255.255.0 gw 10.8.0.1
```

WPA2-Enterprise Config

```
# more /etc/config/wireless

config wifi-device 'radio0'
       option type 'mac80211'
       option channel '11'
       option hwmode '11g'
       option path 'platform/ar934x_wmac'
       option htmode 'HT20'
       option txpower '22'
       option country 'US'
       option disabled '0'

config wifi-iface
       option device 'radio0'
```

```
option mode 'ap'
option network 'lan'
option server '172.31.212.117'
option key 'toto!2x06'
option ssid 'ACME_London_OpenWRT'
option encryption 'wpa2'
```

VM4

FileServer
SCP fileserver with scp restricted user using rssh software

rssh config :
`/etc/rssh.conf`

Webserver VM4

Basic apache installation, with redirection to https for every http request.

Installation :

```
root@bnss-g08-vm1:/etc/apache2# apt-get install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done

root@bnss-g08-vm1:/etc/apache2# aptitude install -y libapache2-mod-proxy-html
libxml2-dev
```

Configuration :
/etc/apache2/apache2.conf
/etc/apache2/sites-enabled/000-default

No additional module.

Android Apps for two factor auth

==========

Title: Two ways authentication, App and scripts

Author: BNSS - Groupe 8 -

Description: This archive contains the android app to perform the 2 ways authentication and the scripts we use to handle the keys and OTP.

==========

--------Android App--------

- The minimum SDK version is 15. The app cannot be installed with device running a SDK version lower than 15.

- This archive contains a file 'client.pvk' this file should be kept secret and can be put in any directory on the device. The default location is /sdcard. The location can be modified in the file app/src/main/res/values/strings.xml.
The line to modify is <string name="path_privatekey">/sdcard/client.pvk</string>
-This key is associated with an account when trying to connect on 172.31.212.117
username: bono
pass: 1234
- Some other buttons are also displayed on the main page but do not concern the project.

To create more keys, first use the authentication server main page to register and then use the following script to create a new public/private key pair.

The App is divided into 3 activities:
- BlankActivity is the welcome page and contains buttons. One should click on "Scan QRcode" to access the second activity.
- CaptureActivity is the scanning activity. This part of the app uses a opensource code to scan and decode the QRcode.
The compiled additional package is the following : 'me.dm7.barcodescanner:zxing:1.8.4'*.
After a successful scanning, the byte stream is stored in an intent and the third activity is called.
- DisplayActivity is the decrypting part of the App. This part uses opensource code to decrypt a byte stream with RSA key.
The compiled additional packages are the following:*
'com.madgag.spongycastle:core:1.51.0.0'
'com.madgag.spongycastle:prov:1.51.0.0',
'com.madgag.spongycastle:pkix:1.51.0.0' and
'com.madgag.spongycastle:pg:1.51.0.0'.

*: These packages should be compiled in the app by adding the line '' compile 'package' '' in the dependencies of the project. (file build.gradle in Android-Studio).

----/etc/pki/keygen.sh-----

The script should be called with a number as argument every time a new user registers. This number corresponds to the userID the key will be generated for. The number of the client is associated to a username and the association is made in login.php regarding the SQL database.

---/var/www/qrcodegen.sh---

The script is called in code.php with userID as argument. It generates an 8 digits OTP in otp/client$userID, encrypts this OTP with the public key of the user, and encode the encrypted password in base64.