ACME Documentation

Martin Borek, Joel Gartner, Miguel Gordo, Yashu Sharma
Group V

March 21, 2016

1 File transfer

The file transfer is using SSL connections do ensure its security. The actual implementation of this in java uses the SSLContext class. This is initially created with `SSLContext.getInstance("SSL");`. This initial instance does however not keep track of our own certificates and we thus need to initialize it with these. This is done by using Java Keystores and Truststores to specify which keys to use to identify our self and what keys to trust. The files containing the keystore and truststores were created with portecle, which is a graphical application for managing keystores. This could however been done just as well with the java keytool. With this tool we created one keystore containing the CA certificate which is to be the truststore used in the application. It was also used to create one keystore per user containing that users private key. From this private key a certificate signing request was exported and sent to the certificate authority and the certificate reply was imported back into this keystore.

Loading these keystores into the program as the trust and keystores they can be used to initialize the SSLContext. This will allow SSLSocketFactories and SSLServerSocketFactories to be created from the SSLContext. These will have the information about the trusted CA certificate as well as about the users private key. These factories can be used to create SSLSockets and SSLServerSockets which can then be used as ordinary Sockets in java. The created sockets should however be modified so that they require client authentication with the method `setNeedClientAuth` as otherwise only the user

acting as server in the transfer will need to authenticate himself. Using these sockets a connection between the users is created which is used to actually transfer the file.

The first version of the file transfer established the connection between users by simply having one set himself to be the server in the transfer and the other entering that persons IP-address and creating a direct connection to that user over which the file transfer could take place. This was however not very user friendly and a more advanced version was developed in which the other users IP address was not necessary. In order to make this a possibility a server was required to be running and establish the connection between the users. The sockets used in this server was created in the same way as the sockets in the previous version. As such the server created SSLServerSockets which the users could connect to as the servers IP address was fixed and known to the clients. This creates a secure SSL connection between the user and the server.

The created connection was then used to connect the users together. This was done by the client first sending a connect message to the server informing the server of who the user is. This was done primitively on the user side by simply taking an alias from the keystore which contains the users key. When the server receives such a message it sends a message to all its connected users which contains an updated list of the users connected to the server, identified by the alias. The same type of list is also sent to all connected users when a user disconnects from the server. This allows all users to keep track of which other users are connected to the server. The messages are encoded in the way which was deemed the simplest to implement, which was to simply wrap the input and output streams from the sockets in java ObjectInput and ObjectOutput streams and simply transmit java objects over the sockets. As such the encoding issue was dealt with by java serialization of objects.

The approach of identifying a user by alias is probably not the optimal way to uniquely identify the users. A better approach would probably be to have the server extract information from the certificate which the user sends to identify himself. This would make it impossible for one user to impersonate another user on the file transfer server. In the current version it is however trivial to change the alias in the keystore to anything you would like and thus allowing you to choose your name freely on the file exchange server.

Even if it would be impossible for the user to change the alias in the keystore it would still be possible for him to choose the identifier on the server freely as the server has no way of actually making sure that the alias the user sends is the one indicated in the keystore.

When the user actually wants to send a file to another user he have to create a connection to this user. This is done by sending a message to the file transfer server indicating who you want to send a file to and containing the name of the file. The server will then transmit this message to the user who should receive the file who gets a prompt for whether or not to accept the file. The response to this is sent to the server. If the file transfer was rejected the response is simply forwarded to the user who wanted to send the file. If the transfer was accepted however, the file transfer server must establish a connection between the users.

The first idea for how to establish the connection was to simply send the IP-address of the user which the file should be sent to back to the user who wanted to send a file. This approach was however not implemented as it would not allow a connection to be established between users in two different private networks. This would not have been a problem in our use case but might have been one in other cases. It was also a problem when testing with two android virtual machines as both machines are located in their own private network with addresses in the 10.0.2.0/24 range. As such there is no simple way for the server to get an IP address which one of the virtual machines could use to connect to the other virtual machine. Because of this another approach was taken. The users are still given a message with a port number and an IP address. The users are also still expected to connect to this address and port. The given address is however not the other users address but is instead the servers own address. The server has created a standard server socket on this port which the user connects to with a SSLSocket. This would cause a problem if something was sent over the socket which was created from this ordinary server socket as SSLSocket is expecting an SSL handshake. This is however not how this ordinary socket is used and instead it is set up so that everything which comes to this socket is sent to the other socket which was created by the other user involved in this file transfer, as well as the other way around. One of the users also set their SSLSocket to not use client mode and it will thus act as the server in the SSL handshake. This will allow the SSL handshake to be performed between the two users

and thus they can validate that the user they send their file to have had their certificate signed by the CA. The result of this is thus that a SSL connection is established between the users with all traffic passing by the server. This connection is then used for the actual transfer of the file.

The resulting way to establish a connection had the downside that it forced firewall rules to be less strict. The reason for this was that the port numbers which was chosen on the server for the user to connect to was chosen automatically. This made it so that no specific port could be opened in the firewall rules to enable the file transfer users to connect. The optimal solution would have been to allow related traffic to access the server and in that way enable the second connection the user makes to the server to pass the firewall. This did however not seem to work without further configurations and thus the solution instead became to allow traffic to all ports on VM3 if it comes from the ACME Stockholm or London office or the VPN connection. A better solution might have been to use a fixed port for the second connection to the server, or possibly a fixed range of ports.

The solution implemented would possibly have worked with OCSP revocations by simply setting the flag `ocsp.enable` to true in java with the command `Security.setProperty("ocsp.enable", "true");` as well as possibly some other flags. This would hopefully have made java do all the hard work of actually checking the status of the certificates as the information of how to obtain this information is available in the certificate under the OCSPUri. If this didn't work it would still have been possible to implement your own implementation of trust manager which ensured that the certificates are verified with OCSP information. Exactly how this should have been implemented was however never tested as we didn't actually get OCSP revocations working and thus there would have been no way to test if any implementation was working.

2 Intrusion detection

[1] For intrusion detection snort was used. It is running on virtual machine 1 and listens on interface eth0 on this machine. This is the outward fac-

[1]Configuration mostly followed instructions given on `https://www.snort.org/documents/snort-2-9-8-x-on-ubuntu-12-lts-and-14-lts-and-15`

ing interface which faces the internet and is such were most attacks on the system would probably originate. Not much configuration was performed for snort itself, although a user and group by the name snort was created which is the user which is responsible for snort. The output which comes directly from snort like this is however not very readable as it is outputted to a binary file which is specified in `/etc/snort/snort.conf`. In order to get better user readable output another program called barnyard2 was installed to interpret the binary output. This program was configured with the file `/etc/snort/barnyard2.conf` where it was specified what it should do with the output. The chosen output was as a MySql database called snorby.

To just get the data logged to this database is however still not very user friendly. Because of this snorby was also installed, which is a web interface for showing the events in the database. This was configured to get data from this same database which barnyard2 was writing to and also got its own MySql user named snorby which it uses to get the data from the database. To actually use the web interface you have to sign in, which is done with the standard user `snorby@snorby.com` with the standard password `snorby`.

At this point the intrusion detection is configured and a web interface is set up to display any detected events. There has however not been any rules specified for what snort should detect. This could be done manually but this would probably result in outdated rules. As such another program called pulledpork was installed. This is used to download new snort rules automatically based on a configuration file located at `/etc/snort/pulledpork.conf`. In order to get most of the rules an API key is needed. This key, called an Oinkcode is available if creating a free account at `http://snort.org`. This was however not done in the actual implementation and as such only the community rules which didn't require an Oinkcode were downloaded with pulledpork. The rules were saved to a file called `/etc/snort/rules/snort.rules` were all rules downloaded via pulledpork should be stored. In the snort configuration file this file was then added as a source for rules and the newly downloaded rules are then used in snort.

The recommended setup is to only use the rules downloaded via pulledpork and thus remove all other sources of rules indicated in the snort configuration file. This was initially done but due to pulledpork only downloading the free community ruleset this didn't result in any visible warnings which

could be used for demonstration. Because of this the standard configuration of rules was reinserted to the configuration file of snort and the rules which snort actually uses are thus the standard rules which came when snort was installed as well ass the free community rule set which was installed with pulledpork. For future use I would recomend to create a `snort.org` account and insert the oinkcode into the pulledpork configuration file and get more rules via pulledpork. This would then allow the standard rules coming with snort to be removed which would allow constantly up to date rules. I would also recomend to schedule pulledpork to run regularly, for example once a day, so that the newest rules are always used in the intrusion detection system.

The actual commands used for running the

3 VPN

The VPN server is deployed at the VM1, being at the edge of the Stockholm headquarters. It uses OpenVPN[2]. The address of the VPN server is 172.31.212.109. The VPN server is installed with the Access Server. Thus, it can be accessed, configured and monitored via a web interface. The address to open the web interface is `https://172.31.212.109:943/admin/` because the VPN server runs on the port 943. The free license for the OpenVPN allows only two users connected simultaneously; one of them being the machine VM2 as the end of the tunnel for the London office.

Since we could not use VM2 as mentioned in the Report, we used a personal laptop instead. However, it does not change the depicted topology. Only the machine name is different. The OpenVPN client is running on this machine to establish a tunnel with the VM1. A configuration file for the client account had to be downloaded from the VPN web interface for clients `https://172.31.212.109:943/` that is accessible after logging in with the client's credentials. The OpenVPN client is started by running:

```
openvpn --config /home/mborekcz/vm2.ovpn
```

When an employee wants to connect to the ACME VPN from home, he/she needs to access the VPN client website `https://172.31.212.109:943/` at first. This website offers downloading the OpenVPN client together

[2] `https://openvpn.net/`

with the configuration file. When logged in for the first time, the employee is prompted to scan a QR code with his/her phone to link it with his/her account. After downloading the OpenVPN client, the employee is asked to provide his/her credentials (username and password verified by the RADIUS). With correct credentials, the employee is asked for the Google Authentication token as the second factor. The two-factor authentication is described in the section below.

All employees connected to the VPN get an IP address from within the ACME network to be allowed access to all ACME servers. Also, all VPN clients are allowed to communicate with each other. The VPN is set on the Layer 3 of the ISO/OSI model and the authentication used is RADIUS with the Google Authenticator.

`as.json`:

```
# OpenVPN AS 1.1 configuration file
# enable AS Connect functionality
AS_CONNECT=true

# temporary directory
tmp_dir=~/tmp

lic.dir=~/licenses

# run_start retries
run_start_retry.give_up=60
run_start_retry.resample=10

# enable client gateway
sa.show_c2s_routes=true

# certificates database
certs_db=sqlite:///~/db/certs.db

# user properties DB
user_prop_db=sqlite:///~/db/userprop.db

# configuration DB
config_db=sqlite:///~/db/config.db

# log DB
log_db=sqlite:///~/db/log.db

# wait this many seconds between failed retries
db_retry.interval=1

# how many retries to attempt before failing
db_retry.n_attempts=6

# bootstrap authentication via PAM -- allows
# admin to log into web UI before authentication
# system has been configured.  Configure PAM users
```

```
# allowed to access via the bootstrap auth mechanism.
boot_pam_service=openvpnas
boot_pam_users.0=openvpn
# boot_pam_users.1=
# boot_pam_users.2=
# boot_pam_users.3=
# boot_pam_users.4=

# System users that are allowed to access the server agent XML API.
# The user that the web server will run as should be in this list.
system_users_local.0=root
system_users_local.1=openvpn_as

# The user/group that the web server will run as
cs.user=openvpn_as
cs.group=openvpn_as

# socket directory
general.sock_dir=~/sock

# source directory for OpenVPN Windows executable
# (Must have been built with MultiFileExtract)
sa.win_exe_dir=~/exe

# The company name will be shown in the UI
sa.company_name=OpenVPN Technologies, Inc.

# server agent socket
sa.sock=~/sock/sagent

# If enabled, automatically generate a client configuration
# when a client logs into the site and successfully authenticates
cs.auto_generate=true
# files for web server (PEM format)
cs.ca_bundle=~/web-ssl/ca.crt
cs.priv_key=~/web-ssl/server.key
cs.cert=~/web-ssl/server.crt

# web server will use three consecutive ports starting at this
# address, for use with the OpenVPN port share feature
cs.dynamic_port_base=870

# which service groups should be started during
# server agent initialization
sa.initial_run_groups.0=web_group

# use this twisted reactor
sa.reactor=epoll

# The unit number of this particular AS configuration.
# Normally set to 0.  If you have multiple, independent AS instances
# running on the same machine, each should have a unique unit number.
sa.unit=0

# If true, open up web ports on the firewall using iptables
iptables.web=true
```

```
vpn.server.user=openvpn_as
vpn.server.group=openvpn_as
```

config.json:


```
{
  "Default": {
    "admin_ui.https.ip_address": "eth0",
    "admin_ui.https.port": "943",
    "auth.ldap.0.name": "My LDAP servers",
    "auth.ldap.0.ssl_verify": "never",
    "auth.ldap.0.timeout": "4",
    "auth.ldap.0.use_ssl": "never",
    "auth.module.type": "pam",
    "auth.pam.0.service": "openvpnas",
    "auth.radius.0.acct_enable": "false",
    "auth.radius.0.name": "My Radius servers",
    "cs.cws_proto_v2": "true",
    "cs.https.ip_address": "eth0",
    "cs.https.port": "943",
    "cs.prof_sign_web": "true",
    "cs.ssl_method": "SSLv3",
    "cs.tls_version_min": "1.0",
    "host.name": "172.31.212.109",
    "sa.initial_run_groups.0": "web_group",
    "sa.initial_run_groups.1": "openvpn_group",
    "vpn.client.routing.inter_client": "false",
    "vpn.client.routing.reroute_dns": "true",
    "vpn.client.routing.reroute_gw": "true",
    "vpn.daemon.0.client.netmask_bits": "20",
    "vpn.daemon.0.client.network": "172.27.224.0",
    "vpn.daemon.0.listen.ip_address": "eth0",
    "vpn.daemon.0.listen.port": "443",
    "vpn.daemon.0.listen.protocol": "tcp",
    "vpn.daemon.0.server.ip_address": "eth0",
    "vpn.server.daemon.enable": "true",
    "vpn.server.daemon.tcp.n_daemons": 2,
    "vpn.server.daemon.tcp.port": "443",
    "vpn.server.daemon.udp.n_daemons": 2,
    "vpn.server.daemon.udp.port": "1194",
    "vpn.server.group_pool.0": "172.27.240.0/20",
    "vpn.server.port_share.enable": "true",
    "vpn.server.port_share.ip_address": "1.2.3.4",
    "vpn.server.port_share.port": "1234",
    "vpn.server.port_share.service": "admin+client",
    "vpn.server.routing.private_access": "nat",
    "vpn.server.routing.private_network.0": "172.31.212.0/24",
    "vpn.server.routing.private_network.1": "192.168.0.0/24",
    "vpn.tls_refresh.do_reauth": "true",
    "vpn.tls_refresh.interval": "360"
  },
  "_INTERNAL": {
    "run_api.active_profile": "Default",
    "webui.edit_profile": "Default"
  }
}
```

4 Two-factor authentication

To connect to the VPN from home, employees are required to authenticate with their Google Authenticator tokens beside authenticating to the RADIUS server. To enable Google Authenticator for all accounts in the OpenVPN, `sacli` commands were used to configure the OpenVPN server. `Sacli` being located at `/usr/local/openvpn_as/scripts/sacli` at the VM1.

```
$ ./sacli --key vpn.server.google_auth.enable --value true ConfigPut
$ ./sacli start
```

Since Google Authenticator should not be required for creating the tunnel from the VM2, the Google Authenticator is disabled for this specific user:

```
$ ./sacli --user vm2 --key prop_google_auth --value true UserPropPut
```

All ACME employees get a phone with pre-installed Google Authenticator. This phone is matched with their VPN (RADIUS) account. This is achieved by scanning a QR code (with the corporate phone) that is displayed the first time clients insert their credentials to connect to the VPN server. After that, tokens are automatically generated every minute and are displayed only in the Google Authenticator app installed on the employees' phone. When an employee wants to connect from home, he/she is prompted to insert the valid token together with his/her user credentials. Upon inserting all these valid information, a VPN connection is successfully established.

5 Public Key Infrastructure

Public Key Infrastructure is used for the authentication as a mechanism for trusting identities online. CFSSL PKI toolkit is used for the implementation[3]. PKI CA is built in Virtual Machine 3.

First of all, a private key and a self-signed certificate is created for the CA using gencert command with initca as option. The input to this command was a file in json format. This file contains the name of the CA as Common Name, the domain for which the certificate is valid for as hosts. This command generates three files: key and certificate in PEM format and CSR which can be used if this CA needs to be cross-signed by another CA. Also a configuration file is created for the CA in json format which determines the signing policy (validity period, key usages, etc.)

[3] `https://cfssl.org/`

```
{
 "signing": {
   "default": {
     "auth_key": "key1",
     "expiry": "8760h",
     "ocsp_url": "http://192.168.0.3:8888",
     "usages": [
         "signing",
         "key encipherment",
         "server auth",
         "client auth"
     ]
    }
  },
  "auth_keys": {
        "key1": {
          "key": "00000000000000000000000000000000",
          "type": "standard"
        }
  }
}
```

This authentication key should be kept private. The API key is a basic authentication mechanism that prevents unauthorized parties from requesting certificates.

In order to request a certificate from a CA, Certificate Signing Request (CSR) needs to be sent to CA. The CSR contains following information:

1. Information about the requestor organization

2. Requestor's public key

3. A digital signature by the requestors private key

genkey command is used to create requestor's private key and CSR. It takes again a json fromat file containing similar information as in the case of CA. Once the CA receives the CSR, it checks the CSR's signature to verify if requestor has control over the associated private key. After this, CA creates and signs the certificate using sign command with the CSR as input. For signing, CA takes its configuration file and database configuration file in json format as input.

The DB config file contains the information about connection to the database:

```
{"driver":"sqlite3","data_source":"certs.db"}
```

An entry is added to Certificates tables in certs.db for the signed certificate.We have used goose db migration scripts for SQlite3 as DB backend. To start a SQLite DB using goose:

```
goose -path $GOPATH/src/github.com/cloudflare/cfssl/certdb/sqlite up'
```

For revocation in case of a private key being compromised, revoke command is used. It takes the serial number of the certificate and revocation reason as input. The status of the certificate will be marked as revoked in Certificates table. In order to check the status of a certificate online, OCSP was used. ocsprefresh command refreshes the ocspresponse table for all unexpired certificates and ocspdump command concatenates all these responses into a response file in base64-encoded format. This response file is used by ocspserve command to respond to OCSP requests. Here the ocsp url mentioned in configuration file of CA is used. The certificates signed with the configuration file as input use this url to make OCSP requests.

6 RADIUS

Install FreeRadius on VM3:

```
sudo apt-get install freeradius
```

Modify `/etc/freeradius/clients.conf` to add authenticators that can connect to the radius server.

Add into `clients.conf`:

```
        client 192.168.1.2{
                ipaddr=192.168.1.2
                secret=SharedSecret99
}
        client 192.168.0.2{
                ipaddr=192.168.0.2
                secret=SharedSecret99
}
        client 172.31.212.109{
                ipaddr=172.31.212.109
                secret=SharedSecret99
}
```

Client 192.168.1.2 is the London router. The other two are the addresses of the two interfaces of VM1 (authentication for OpenVPN).

Modify users in `/etc/freeradius`, and add users as needed. For example:

```
mumble  Cleartext-Password :=  h e l l o  , Pool-Name :=  m a i n _ p o o l
                Service-Type = Framed-User,
                Framed-Ip-Address = 192.168.1.20,
                Framed-Ip-Netmask = 255.255.255.0
                Framed-Routing = Broadcast-Listen,
                Framed-MTU = 1500
```

To start server:

```
sudo freeradius -X &. Kill all processes when a stop is required
```

7 WiFi Router

- Set wireless security settings to WPA2 Enterprise
- Set RADIUS IP address to `192.168.0.3`
- Set static IP to `192.168.1.2`
- Set default gateway to `192.168.1.1` (Vm2 - Personal Laptop)
- Ports for RADIUS are `1812` and `1813`
- Shared Secret should be set to `SharedSecret99`
- Internal addresses in London office will have addresses in `10.0.0.1/16` network.
- Set NAT settings to Open.
- Connect internet port to ethernet port of laptop (VM2)

8 Firewalls

The firewalls automatically got quite a lot of rules when setting up the openVPN server. In order to ensure that the VPN was still working we did as little changes to these as possible. We did however change the policies on the chains `INPUT` and `FORWARD` to be drop instead and thus we only allowed traffic which was explicitly accepted by the VPN or our rules to access VM1 and the internal network.

By changing the policy almost everything was blocked from VM1 and the internal network, which wasn't desired. As such some new rules were required to allow accepted traffic to access the network. First of all we allowed SSH traffic to access the internal network if from within the ACME corporate network and also SSH traffic to VM1 from anywhere. This was done so that remote configuration of the virtual machines would still be possible even when the firewall was in place.

Traffic was also allowed to access VM4 if it was on port 80 or 443 and if it was originating from within the ACME corporate network. The purpose of this rule was to allow HTTP and HTTPS traffic destined to the web server. As the router in the London office must access VM3 in the Stockholm office for radius authentication another rule was also added which allowed radius traffic to VM3 from within the ACME network. In order for the connections to be allowed in both directions after they were established, a rule was also added to allow related and established traffic through the firewall. The file transfer server on VM3 also required several ports in the firewall to be opened in order for it to function. For the initial connection only port 2183 was required to be opened in the firewall. For the second connection over which the actual file transfer takes place there was however not a well defined port used. An optimal solution would have been if this was registered as related traffic and thus allowed automatically with the previously defined rules. This did however not work without further configuration. As such the solution became to allow all traffic from within the corporate network to access VM3, no matter which port was used to access the server.

The output from the `iptables-save` command is shown below. This contains all the rules which were present in the firewall and can be used with the `iptables-restore` command to set the current rules in the firewall to the ones saved.

```
# Generated by iptables-save v1.4.12 on Tue Mar 15 14:42:50 2016
*security
:INPUT ACCEPT [28116:7546119]
:FORWARD ACCEPT [427:75018]
:OUTPUT ACCEPT [23514:4488384]
COMMIT
# Completed on Tue Mar 15 14:42:50 2016
# Generated by iptables-save v1.4.12 on Tue Mar 15 14:42:50 2016
*raw
:PREROUTING ACCEPT [28606:7626487]
:OUTPUT ACCEPT [23558:4492401]
COMMIT
# Completed on Tue Mar 15 14:42:50 2016
```

```
# Generated by iptables-save v1.4.12 on Tue Mar 15 14:42:50 2016
*mangle
:PREROUTING ACCEPT [15138:2511522]
:INPUT ACCEPT [61930:21152026]
:FORWARD ACCEPT [16584:12006411]
:OUTPUT ACCEPT [46416:11423978]
:POSTROUTING ACCEPT [63000:23430389]
:AS0_MANGLE_PRE_REL_EST - [0:0]
:AS0_MANGLE_TUN - [0:0]
-A PREROUTING -m state --state RELATED,ESTABLISHED -j AS0_MANGLE_PRE_REL_EST
-A PREROUTING -i as0t+ -j AS0_MANGLE_TUN
-A AS0_MANGLE_PRE_REL_EST -j ACCEPT
-A AS0_MANGLE_TUN -j MARK --set-xmark 0x2000000/0xffffffff
-A AS0_MANGLE_TUN -j ACCEPT
COMMIT
# Completed on Tue Mar 15 14:42:50 2016
# Generated by iptables-save v1.4.12 on Tue Mar 15 14:42:50 2016
*nat
:PREROUTING ACCEPT [4324:264099]
:INPUT ACCEPT [4152:253278]
:OUTPUT ACCEPT [813:64119]
:POSTROUTING ACCEPT [13:846]
:AS0_DPFWD_TCP - [0:0]
:AS0_DPFWD_UDP - [0:0]
:AS0_NAT - [0:0]
:AS0_NAT_POST_REL_EST - [0:0]
:AS0_NAT_PRE - [0:0]
:AS0_NAT_PRE_REL_EST - [0:0]
:AS0_NAT_TEST - [0:0]
-A PREROUTING -m state --state RELATED,ESTABLISHED -j AS0_NAT_PRE_REL_EST
-A PREROUTING -d 172.31.212.109/32 -p udp -m udp --dport 1194 -m state --
    state NEW -j AS0_DPFWD_UDP
-A PREROUTING -d 172.31.212.109/32 -p tcp -m tcp --dport 443 -m state --
    state NEW -j AS0_DPFWD_TCP
-A POSTROUTING -m state --state RELATED,ESTABLISHED -j AS0_NAT_POST_REL_EST
-A POSTROUTING -m mark --mark 0x2000000/0x2000000 -j AS0_NAT_PRE
-A POSTROUTING -o eth0 -j MASQUERADE
-A AS0_DPFWD_TCP -p tcp -j DNAT --to-destination 172.31.212.109:914
-A AS0_DPFWD_TCP -j ACCEPT
-A AS0_DPFWD_UDP -p udp -j DNAT --to-destination 172.31.212.109:917
-A AS0_DPFWD_UDP -j ACCEPT
-A AS0_NAT -o eth0 -j SNAT --to-source 172.31.212.109
-A AS0_NAT -j ACCEPT
-A AS0_NAT_POST_REL_EST -j ACCEPT
-A AS0_NAT_PRE -m mark --mark 0x8000000/0x8000000 -j AS0_NAT
-A AS0_NAT_PRE -d 169.254.0.0/16 -j AS0_NAT_TEST
-A AS0_NAT_PRE -d 192.168.0.0/16 -j AS0_NAT_TEST
-A AS0_NAT_PRE -d 172.16.0.0/12 -j AS0_NAT_TEST
-A AS0_NAT_PRE -d 10.0.0.0/8 -j AS0_NAT_TEST
-A AS0_NAT_PRE -j AS0_NAT
-A AS0_NAT_PRE_REL_EST -j ACCEPT
-A AS0_NAT_TEST -o as0t+ -j ACCEPT
-A AS0_NAT_TEST -m mark --mark 0x4000000/0x4000000 -j ACCEPT
-A AS0_NAT_TEST -d 192.168.1.0/25 -j ACCEPT
-A AS0_NAT_TEST -d 192.168.1.128/25 -j ACCEPT
-A AS0_NAT_TEST -d 192.168.2.0/24 -j ACCEPT
-A AS0_NAT_TEST -j AS0_NAT
```

```
COMMIT
# Completed on Tue Mar 15 14:42:50 2016
# Generated by iptables-save v1.4.12 on Tue Mar 15 14:42:50 2016
*filter
:INPUT DROP [11528:2262663]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [41447:10587657]
:AS0_ACCEPT - [0:0]
:AS0_DNS - [0:0]
:AS0_IN - [0:0]
:AS0_IN_NAT - [0:0]
:AS0_IN_POST - [0:0]
:AS0_IN_PRE - [0:0]
:AS0_IN_ROUTE - [0:0]
:AS0_OUT - [0:0]
:AS0_OUT_LOCAL - [0:0]
:AS0_OUT_POST - [0:0]
:AS0_OUT_S2C - [0:0]
:AS0_U_AB_OUT - [0:0]
:AS0_U_NEW2_OUT - [0:0]
:AS0_U_NEW_OUT - [0:0]
:AS0_U_TEST1_OUT - [0:0]
:AS0_U_TEST2_OUT - [0:0]
:AS0_U_VM2_OUT - [0:0]
:AS0_WEBACCEPT - [0:0]
:FORWARDING_RULES - [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j AS0_ACCEPT
-A INPUT -i lo -j AS0_ACCEPT
#-A INPUT -m mark --mark 0x2000000/0x2000000 -j AS0_IN_PRE
#-A INPUT -m mark --mark 0x2000000/0x2000000 -j AS0_IN_PRE
-A INPUT -d 172.31.212.109/32 -p tcp -m state --state NEW -m tcp --dport 915
     -j AS0_ACCEPT
-A INPUT -d 172.31.212.109/32 -p tcp -m state --state NEW -m tcp --dport 914
     -j AS0_ACCEPT
-A INPUT -d 172.31.212.109/32 -p udp -m state --state NEW -m udp --dport 917
     -j AS0_ACCEPT
-A INPUT -d 172.31.212.109/32 -p udp -m state --state NEW -m udp --dport 916
     -j AS0_ACCEPT
#-A INPUT -m state --state RELATED,ESTABLISHED -j AS0_WEBACCEPT
#-A INPUT -m state --state RELATED,ESTABLISHED -j AS0_WEBACCEPT
-A INPUT -d 172.31.212.109/32 -p tcp -m state --state NEW,ESTABLISHED -m tcp
     --dport 443 -j ACCEPT
-A INPUT -d 172.31.212.109/32 -p tcp -m state --state NEW -m tcp --dport 943
     -j AS0_WEBACCEPT
-A FORWARD -m state --state RELATED,ESTABLISHED -j AS0_ACCEPT
#-A FORWARD -m mark --mark 0x2000000/0x2000000 -j AS0_IN_PRE
#-A FORWARD -m mark --mark 0x2000000/0x2000000 -j AS0_IN_PRE
-A FORWARD -o as0t+ -j AS0_OUT_S2C
-A FORWARD -i eth0 -o eth1 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i eth1 -o eth0 -j ACCEPT
-A OUTPUT -o as0t+ -j AS0_OUT_LOCAL
-A AS0_ACCEPT -j ACCEPT
-A AS0_ACCEPT -s 192.168.0.0/16 -d 192.168.0.0/16 -j ACCEPT
-A AS0_ACCEPT -s 130.0.0.0/32 -j ACCEPT
-A AS0_ACCEPT -s 172.0.0.0/32 -j ACCEPT
-A AS0_ACCEPT -j REJECT --reject-with icmp-port-unreachable
-A AS0_DNS -d 172.31.212.3/32 -j ACCEPT
```

```
-A AS0_DNS -d 192.168.2.1/32 -j ACCEPT
-A AS0_DNS -d 192.168.2.65/32 -j ACCEPT
-A AS0_DNS -d 192.168.2.129/32 -j ACCEPT
-A AS0_DNS -d 192.168.2.193/32 -j ACCEPT
-A AS0_DNS -j RETURN
-A AS0_IN -d 192.168.2.1/32 -j ACCEPT
-A AS0_IN -j AS0_IN_POST
-A AS0_IN_NAT -j MARK --set-xmark 0x8000000/0x8000000
-A AS0_IN_NAT -j ACCEPT
-A AS0_IN_POST -d 192.168.0.0/24 -j ACCEPT
-A AS0_IN_POST -o as0t+ -j AS0_OUT
-A AS0_IN_POST -j DROP
-A AS0_IN_PRE -p tcp -m state --state NEW -m tcp --dport 53 -j AS0_DNS
-A AS0_IN_PRE -p udp -m state --state NEW -m udp --dport 53 -j AS0_DNS
-A AS0_IN_PRE -d 169.254.0.0/16 -j AS0_IN
-A AS0_IN_PRE -d 192.168.0.0/16 -j AS0_IN
-A AS0_IN_PRE -d 172.16.0.0/12 -j AS0_IN
-A AS0_IN_PRE -d 10.0.0.0/8 -j AS0_IN
-A AS0_IN_PRE -j DROP
-A AS0_IN_ROUTE -j MARK --set-xmark 0x4000000/0x4000000
-A AS0_IN_ROUTE -j ACCEPT
-A AS0_OUT -d 0.0.0.0/32
-A AS0_OUT -d 0.0.0.0/32
-A AS0_OUT -d 0.0.0.0/32
-A AS0_OUT -d 0.0.0.0/32
-A AS0_OUT -d 0.0.0.0/32
-A AS0_OUT -d 192.168.1.254/32 -j AS0_U_VM2_OUT
-A AS0_OUT -d 192.168.1.0/25 -j AS0_U_VM2_OUT
-A AS0_OUT -j AS0_OUT_POST
-A AS0_OUT_LOCAL -p icmp -m icmp --icmp-type 5 -j DROP
-A AS0_OUT_LOCAL -j ACCEPT
-A AS0_OUT_POST -m mark --mark 0x2000000/0x2000000 -j ACCEPT
-A AS0_OUT_POST -j DROP
-A AS0_OUT_S2C -j AS0_OUT
-A AS0_U_AB_OUT -s 192.168.0.0/24 -j ACCEPT
-A AS0_U_AB_OUT -s 192.168.1.0/25 -j ACCEPT
-A AS0_U_AB_OUT -s 192.168.1.128/25 -j ACCEPT
-A AS0_U_AB_OUT -s 192.168.2.0/24 -j ACCEPT
-A AS0_U_AB_OUT -j AS0_OUT_POST
-A AS0_U_NEW2_OUT -s 192.168.0.0/24 -j ACCEPT
-A AS0_U_NEW2_OUT -s 192.168.1.0/25 -j ACCEPT
-A AS0_U_NEW2_OUT -s 192.168.1.128/25 -j ACCEPT
-A AS0_U_NEW2_OUT -s 192.168.2.0/24 -j ACCEPT
-A AS0_U_NEW2_OUT -j AS0_OUT_POST
-A AS0_U_NEW_OUT -s 192.168.0.0/24 -j ACCEPT
-A AS0_U_NEW_OUT -s 192.168.1.0/25 -j ACCEPT
-A AS0_U_NEW_OUT -s 192.168.1.128/25 -j ACCEPT
-A AS0_U_NEW_OUT -s 192.168.2.0/24 -j ACCEPT
-A AS0_U_NEW_OUT -j AS0_OUT_POST
-A AS0_U_TEST1_OUT -s 192.168.0.0/24 -j ACCEPT
-A AS0_U_TEST1_OUT -s 192.168.1.0/25 -j ACCEPT
-A AS0_U_TEST1_OUT -s 192.168.1.128/25 -j ACCEPT
-A AS0_U_TEST1_OUT -s 192.168.2.0/24 -j ACCEPT
-A AS0_U_TEST1_OUT -j AS0_OUT_POST
-A AS0_U_TEST2_OUT -s 192.168.0.0/24 -j ACCEPT
-A AS0_U_TEST2_OUT -s 192.168.1.0/25 -j ACCEPT
-A AS0_U_TEST2_OUT -s 192.168.1.128/25 -j ACCEPT
```

```
-A AS0_U_TEST2_OUT -s 192.168.2.0/24 -j ACCEPT
-A AS0_U_TEST2_OUT -j AS0_OUT_POST
-A AS0_U_VM2_OUT -s 192.168.0.0/24 -j ACCEPT
-A AS0_U_VM2_OUT -s 192.168.1.0/25 -j ACCEPT
-A AS0_U_VM2_OUT -s 192.168.1.128/25 -j ACCEPT
-A AS0_U_VM2_OUT -s 192.168.2.0/24 -j ACCEPT
-A AS0_U_VM2_OUT -j AS0_OUT_POST
-A AS0_WEBACCEPT -j ACCEPT
-A INPUT -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -s 192.168.0.0/22 --dport 80 -m state --state NEW,
    ESTABLISHED -j ACCEPT
-A FORWARD -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
-A FORWARD -p tcp -s 192.168.0.0/22 -d 192.168.0.4 --dport 80 -m state --
    state NEW,ESTABLISHED -j ACCEPT
-A FORWARD -p tcp -s 192.168.0.0/22 -d 192.168.0.4 --dport 443 -m state --
    state NEW,ESTABLISHED -j ACCEPT
-A FORWARD -p udp -s 192.168.0.0/22 -d 192.168.0.3 --dport 1812 -m state --
    state NEW,ESTABLISHED -j ACCEPT
-A FORWARD -p udp -s 192.168.0.0/22 -d 192.168.0.3 --dport 1813 -m state --
    state NEW,ESTABLISHED -j ACCEPT
-A FORWARD -p tcp -s 192.168.0.0/22 -d 192.168.0.3 --dport 2183 -m state --
    state NEW,ESTABLISHED -j ACCEPT
-A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A OUTPUT -p udp -d 192.168.0.3 --dport 1812 -m state --state NEW,
    ESTABLISHED -j ACCEPT
-A OUTPUT -p udp -d 192.168.0.3 --dport 1813 -m state --state NEW,
    ESTABLISHED -j ACCEPT
-A FORWARD -p tcp -s 192.168.0.0/22 -d 192.168.0.3  -m state --state NEW,
    ESTABLISHED -j ACCEPT
COMMIT
# Completed on Tue Mar 15 14:42:50 2016
```

where the rules which were added by us were:

```
-A INPUT -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -s 192.168.0.0/22 --dport 80 -m state --state NEW,
    ESTABLISHED -j ACCEPT
-A FORWARD -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT
-A FORWARD -p tcp -s 192.168.0.0/22 -d 192.168.0.4 --dport 80 -m state --
    state NEW,ESTABLISHED -j ACCEPT
-A FORWARD -p tcp -s 192.168.0.0/22 -d 192.168.0.4 --dport 443 -m state --
    state NEW,ESTABLISHED -j ACCEPT
-A FORWARD -p udp -s 192.168.0.0/22 -d 192.168.0.3 --dport 1812 -m state --
    state NEW,ESTABLISHED -j ACCEPT
-A FORWARD -p udp -s 192.168.0.0/22 -d 192.168.0.3 --dport 1813 -m state --
    state NEW,ESTABLISHED -j ACCEPT
-A FORWARD -p tcp -s 192.168.0.0/22 -d 192.168.0.3 --dport 2183 -m state --
    state NEW,ESTABLISHED -j ACCEPT
-A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A OUTPUT -p udp -d 192.168.0.3 --dport 1812 -m state --state NEW,
    ESTABLISHED -j ACCEPT
-A OUTPUT -p udp -d 192.168.0.3 --dport 1813 -m state --state NEW,
    ESTABLISHED -j ACCEPT
-A FORWARD -p tcp -s 192.168.0.0/22 -d 192.168.0.3  -m state --state NEW,
    ESTABLISHED -j ACCEPT
```