1. Strong swan Installation on Ubuntu

Modified from: http://xenostudy.tistory.com/470

```
sudo apt-get install strongswan
```

After installing, try to check the directory and files. You should create a certificate on each directory.

```
root@ubuntu:/etc/ipsec.d# ls -al
total 56
drwxr-xr-x  12 root root  4096 Oct 29 13:02 .
drwxr-xr-x 163 root root 12288 Oct 29 13:27 ..
drwxr-xr-x   2 root root  4096 Oct 19  2011 aacerts
drwxr-xr-x   2 root root  4096 Oct 19  2011 acerts
drwxr-xr-x   2 root root  4096 Oct 19  2011 cacerts
drwxr-xr-x   2 root root  4096 Oct 19  2011 certs
drwxr-xr-x   2 root root  4096 Oct 19  2011 crls
drwxr-xr-x   2 root root  4096 Oct 29 12:57 examples
drwxr-xr-x   2 root root  4096 Oct 19  2011 ocspcerts
drwxr-xr-x   2 root root  4096 Oct 29 12:57 policies
drwx------   2 root root  4096 Oct 19  2011 private
drwxr-xr-x   2 root root  4096 Oct 19  2011 reqs
```

2. Create Certificate Authority (CA) Certificate

(see: http://pluieglaciale.wordpress.com/2010/11/09/how-to-setup-strongswan-proxy-on-single-ip-vps-for-windows-7-client/)

2.1 Generate the private key

```
openssl genrsa -des3 -out ca.key 4096
```

```
# mv ca.key ca8.key
# openssl rsa -in ca8.key -out ca.key
```

2.2 Generate the certificate request. The answers to the questions aren't relevant. Common name (CN) is generally displayed so give that a useful name.

```
openssl req -new -key ca.key -out ca.csr
```

2.3 Sign the certificate request with the private key, in essence, creating the certificate. This also adds on other information, such as expiration time.

```
openssl x509 -req -days 365 -in ca.csr -signkey ca.key -out ca.crt
```

3. Create VPN Server Certificate

3.1 The first two steps are the same. Create private key.

```
openssl genrsa -des3 -out server.key 4096
```

```
# mv server.key server8.key
# openssl rsa -in server8.key -out server.key
```

3.2 Create the certificate request. This time, the CN must be the FQDN of the VPN host.

```
openssl req -new -key server.key -out server.csr
```

3.3 The VPN certificate must have some special attributes set in order for Windows 7 over accept it. Create gateway.conf with the following contents, with “<FQDN>” replaced with the FQDN, the same as the CN from above. For instance, "subjectAltName = DNS:<us.as.tempe.asu.test.build>"

```
extendedKeyUsage = serverAuth, 1.3.6.1.5.5.8.2.2
subjectAltName = DNS:<FQDN>
```

3.4 Sign the certificate request with the CA private key and certificate. This also adds on the additional info including that from the gateway.conf above.

```
openssl x509 -req -days 365 -in server.csr -CA ca.crt ₩
 -CAkey ca.key -set_serial 01 -out server.crt -extfile gateway.conf
```

The file names used in the config need to be replaced with the ones generated above. vpnCert.pem is server.crt, vpnKey.pem is server.key. Copy those two files into the locations (/etc/ipsec.d/) described by strongSwan's instruction.

4. Update ipsec.conf as follows

(see: http://wiki.strongswan.org/projects/strongswan/wiki/Win7EapMultipleConfig)

```
# ipsec.conf - strongSwan IPsec configuration file

config setup
    plutostart=no

conn %default
    keyexchange=ikev2
    ike=aes256-sha1-mode1024,aes128-sha256-mode2048,aes128-sha1-mode1024
    esp=aes256-sha1!
    dpdaction=clear
    dpddelay=300s
    rekey=no

conn win7
    left=%any                   # left is myself
    leftsubnet=0.0.0.0/0
    leftauth=pubkey
    leftcert=vpnCert.pem
    leftid=@vpn.strongswan.org
    right=%any
    rightsourceip=10.10.3.0/24   #right is the destination
                                 # you should modify it with your network address.
    rightauth=eap-mschapv2
    #rightsendcert=never   # see note
    eap_identity=%any
    auto=add
```

5. Start the service

```
ipsec restart
```

6. See debug log files

```
tail -f /var/log/syslogd &
```

(see: https://wiki.strongswan.org/projects/strongswan/wiki/LoggerConfiguration)

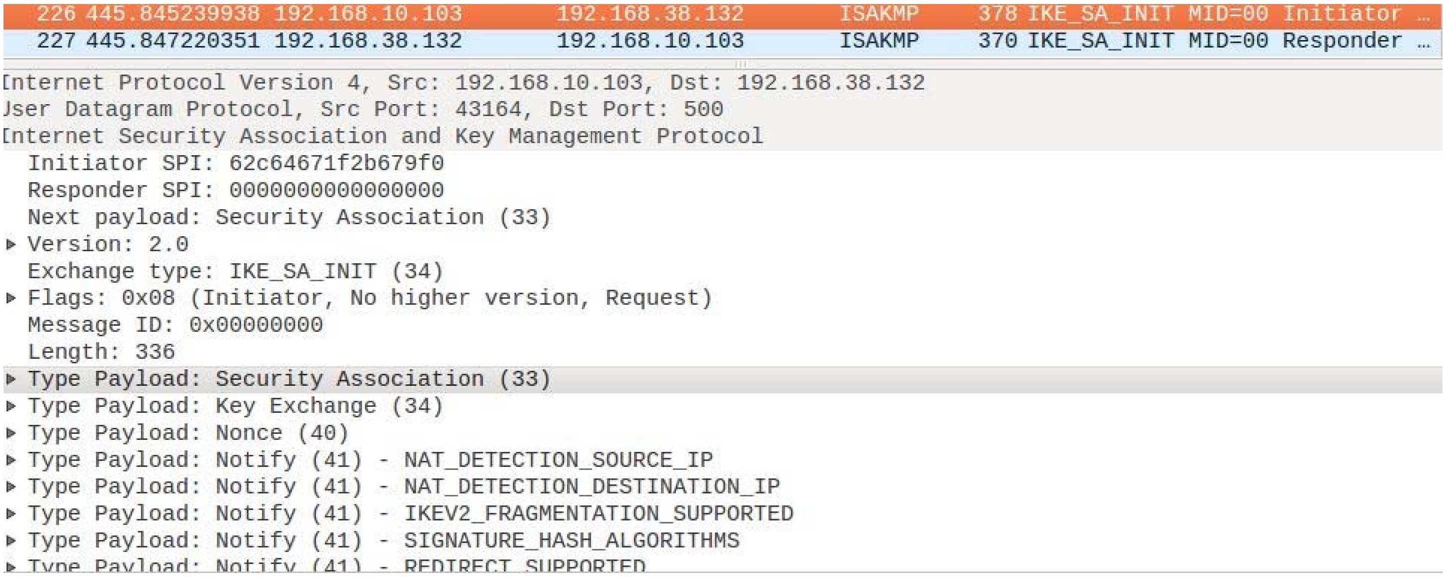
According to upper link, we should modify logger configuration properly in /etc/strongswan.conf.
The IKE daemon knows different numerical levels of logging, ranging from -1 to 4:

4: Also include sensitive material in dumps, e.g. keys

```
charon {
    # to defined file loggers
    filelog {
        /var/log/charon.log {
            # add a timestamp prefix
            time_format = %b %e %T
            # prepend connection name, simplifies grepping
            ike_name = yes
            # overwrite existing files
            append = no
            # increase default loglevel for all daemon subsystems
            default = 2
            # flush each line to disk
            flush_line = yes
        }
        stderr {
            # more detailed loglevel for a specific subsystem, overriding the
            # default loglevel.
            ike = 4
            knl = 3
        }
    }
    # and two loggers using syslog
    syslog {
        # prefix for each log message
        identifier = charon-custom
        # use default settings to log to the LOG_DAEMON facility
        daemon {
        }
        # very minimalistic IKE auditing logs to LOG_AUTHPRIV
        auth {
            default = -1
            ike = 4
        }
    }
}
```

After configured, ipsec restart

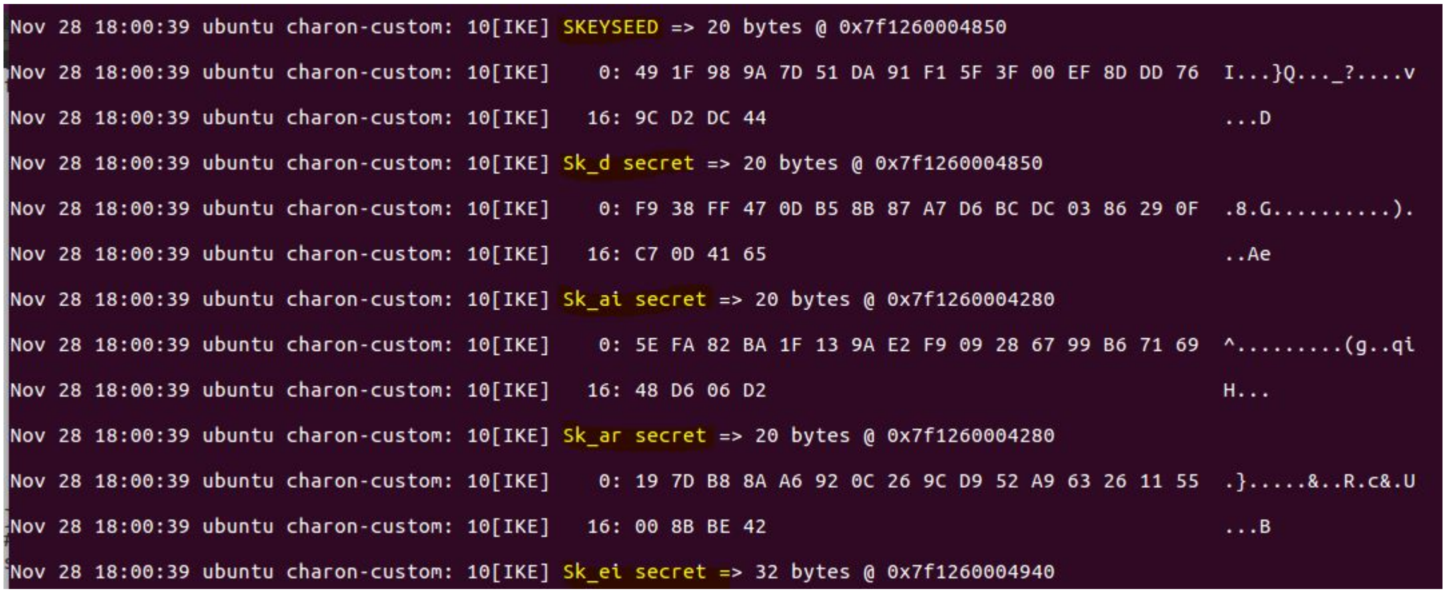
And try to send ISAKMP packet using Scopy to this server, and capture the response. (Fig 1)
Our goal is just get the response packet, so we don't need to get the real IPsec session with real accounts.



```
  226 445.845239938 192.168.10.103        192.168.38.132        ISAKMP     378 IKE_SA_INIT MID=00 Initiator …
  227 445.847220351 192.168.38.132        192.168.10.103        ISAKMP     370 IKE_SA_INIT MID=00 Responder …

Internet Protocol Version 4, Src: 192.168.10.103, Dst: 192.168.38.132
User Datagram Protocol, Src Port: 43164, Dst Port: 500
Internet Security Association and Key Management Protocol
  Initiator SPI: 62c64671f2b679f0
  Responder SPI: 0000000000000000
  Next payload: Security Association (33)
▸ Version: 2.0
  Exchange type: IKE_SA_INIT (34)
▸ Flags: 0x08 (Initiator, No higher version, Request)
  Message ID: 0x00000000
  Length: 336
▸ Type Payload: Security Association (33)
▸ Type Payload: Key Exchange (34)
▸ Type Payload: Nonce (40)
▸ Type Payload: Notify (41) - NAT_DETECTION_SOURCE_IP
▸ Type Payload: Notify (41) - NAT_DETECTION_DESTINATION_IP
▸ Type Payload: Notify (41) - IKEV2_FRAGMENTATION_SUPPORTED
▸ Type Payload: Notify (41) - SIGNATURE_HASH_ALGORITHMS
▸ Type Payload: Notify (41) - REDIRECT_SUPPORTED
```

Figure 1 When Scopy send mock packet “IKE_SA_INIT” REQUEST to Strongswan IPsec server, server sends its response immediately.



```
Nov 28 18:00:39 ubuntu charon-custom: 10[IKE] SKEYSEED => 20 bytes @ 0x7f1260004850
Nov 28 18:00:39 ubuntu charon-custom: 10[IKE]    0: 49 1F 98 9A 7D 51 DA 91 F1 5F 3F 00 EF 8D DD 76  I...}Q..._?....v
Nov 28 18:00:39 ubuntu charon-custom: 10[IKE]   16: 9C D2 DC 44                                      ...D
Nov 28 18:00:39 ubuntu charon-custom: 10[IKE] Sk_d secret => 20 bytes @ 0x7f1260004850
Nov 28 18:00:39 ubuntu charon-custom: 10[IKE]    0: F9 38 FF 47 0D B5 8B 87 A7 D6 BC DC 03 86 29 0F  .8.G.........).
Nov 28 18:00:39 ubuntu charon-custom: 10[IKE]   16: C7 0D 41 65                                      ..Ae
Nov 28 18:00:39 ubuntu charon-custom: 10[IKE] Sk_ai secret => 20 bytes @ 0x7f1260004280
Nov 28 18:00:39 ubuntu charon-custom: 10[IKE]    0: 5E FA 82 BA 1F 13 9A E2 F9 09 28 67 99 B6 71 69  ^.........(g..qi
Nov 28 18:00:39 ubuntu charon-custom: 10[IKE]   16: 48 D6 06 D2                                      H...
Nov 28 18:00:39 ubuntu charon-custom: 10[IKE] Sk_ar secret => 20 bytes @ 0x7f1260004280
Nov 28 18:00:39 ubuntu charon-custom: 10[IKE]    0: 19 7D B8 8A A6 92 0C 26 9C D9 52 A9 63 26 11 55  .}.....&..R.c&.U
Nov 28 18:00:39 ubuntu charon-custom: 10[IKE]   16: 00 8B BE 42                                      ...B
Nov 28 18:00:39 ubuntu charon-custom: 10[IKE] Sk_ei secret => 32 bytes @ 0x7f1260004940
```

Figure 2. When we adjusted the debugger level to 4, we can get the key material