Windows 2000, Linux FreeS/WAN and PGPNet interoperatibility using x509v3 certificates

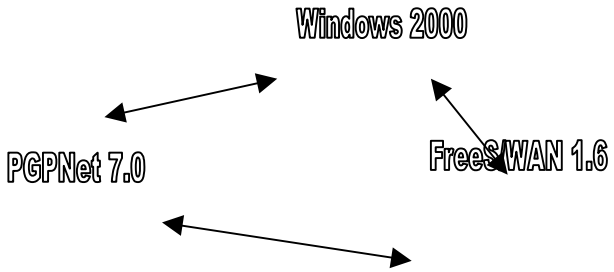
Oscar Delgado
oscar@dilmun.ls.fi.upm.es

CriptoLab Research Group
Polytechnical University of Madrid

1. Introduction

This guide describes, step-by-step, how manage to get full IPSec communication interoperability between different operative. The three different cases considered are:

- Windows 2000 with its IPSec build-in support.
- Windows NT 4.0 with PGPNet version 7.0 installed
- Linux RedHat 7.0 with FreeS/WAN 1.6.



Next we will see how compatibility is possible with certificates issued by different CA's, in particular we have studied that with the Microsoft Test-CA and those created with OpenSSL.

2. The common part: certificates creation.

2.1. Do-it-yourself. Have you installed OpenSSL?

The first step is to get the proper OpenSSL distribution and install it. We have used OpenSSL version 0.9.6, but whatever above 0.9.2 will work. You can always retrieve the latest version at http://www.openssl.org. As example, let's create our own CA, called *CriptoLabCA*. This can be done using the following in-line command:

```
$ sh CA.sh -newca
```

Now, let's go with the tricky part. OpenSSL has a curious feature, which will allow you to get a CA signing certificates whose validity period of time is not inside the CA's certificate validity dates. Windows 2000 IPSec IKE's usage of CAPI APIs requires that the certificate validity periods must nest (as it would be in general!). Thus, if Issuer is valid from 1/1/00 through 12/1/00, then the subject's validity period must >= 1/1/00 and <= 12/1/00[1].

[1] William Dixon, from the Ipsec Interop Test Team, revealed us that the certificate validity time nesting is not required by IKE in the next release of Windows 2000.

By default, the OpenSSL configuration file, *openssl.cnf*, assigns 365 days to the validity date for the CA certificates. As OpenSSL use always the same configuration file, it does not distinguish signatures when they are for CA (self)certification or as issuer of CA signed certificates. OpenSSL does the same each time signs a certificate without considering any kind of "semantic" in that operation. If you create the CA and inmediatly sign a certificate, the notAfter date for the second certificate is higher than the equivalent date in the CA certificate and you get a validity time overrun.

We will no discuss here about the advantages and risks that you get when you follow this procedure, but be aware that Windows 2000 will no accept this certificates for negotiating IPSec. To fix this rejection, extend the CA certificate notAfter date by typing:

```
$ openssl x509 -in newca.pem -days 1024 -out cacert.pem -signkey
CriptoLabCA/private/cakey.pem
```

Following our proposed example, now make sure you replace the previous CA certificate with this and we can start signing certificates. The first step is making an issue request:

```
$ sh CA.sh –newreq
```

then our CA has to sign this request:

```
$ sh CA.sh –sign
```

The certificate remains in *newcert.pem* and the private key in *newreq.pem* and its useful to rename those files with a bit more descriptive names: *Win2000cert.pem* and *Win2000privatekey.pem* for example. In order to be able to install this certificate and private key in other machines, we need to create a PKCS#12 file typing something similar to this:

```
$ openssl pkcs12 -export -in Win2000cert.pem -inkey newreq.pem -certfile
CriptoLabCA/cacert.pem -out Win2000.p12
```

this file will be later imported into Windows 2000 or PGPNet.. Certificates for PGPNet or Linux are created in the same way, except that, for Linux, we will not not need the PKCS#12 file.

2.2. Using the Microsoft SecTestCA.

OpenSSL is a good choice for making you own certificates, but is not the only one. You could also use the Microsoft testing CA that you can find at http://sectestca1.rte.microsoft.com/. By the way, you need Internet Explorer for getting everything right. The concrete steps to follow are:

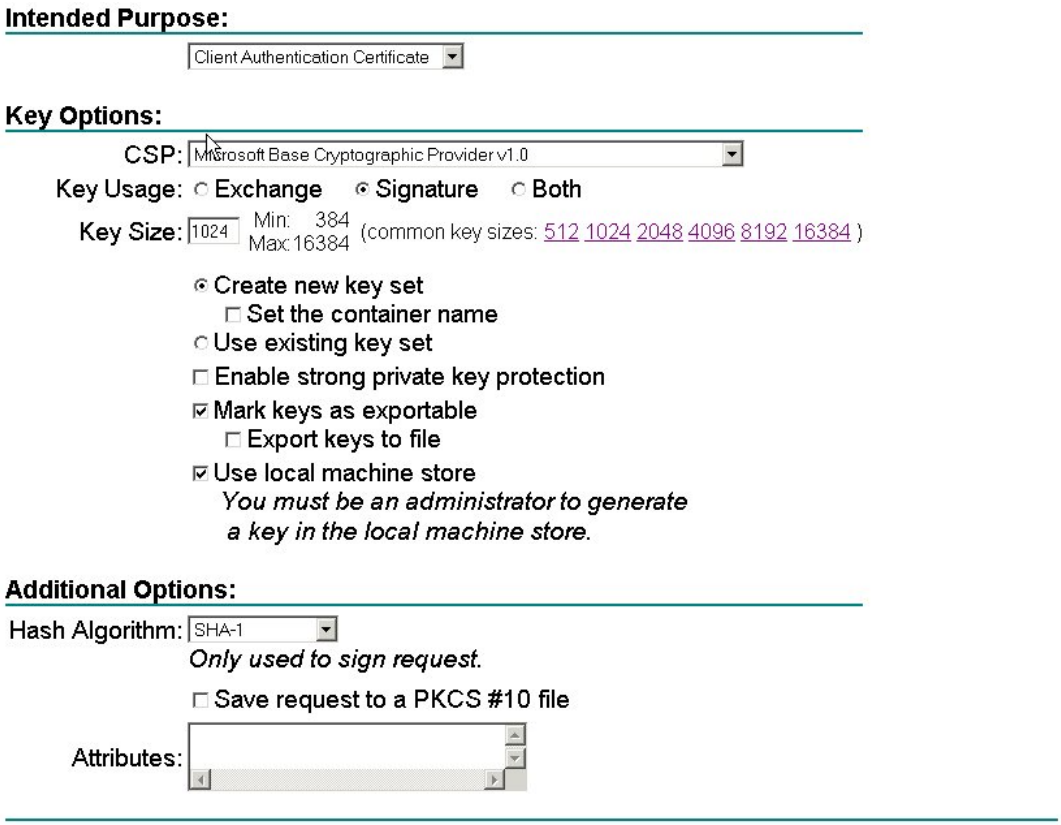
1. Select **CA Standalone Root** (RSA 2048).
2. For requesting select **Request a Certificate**, and click on **Next**.
3. Select **Advanced Request**, and **Next**.
4. Use a form, so select **Submit a Certificate Request Using a Form**.
5. In the **Advanced Certificate Request**, choose the following answers:
   - Fill the identification information fields, name, e-mail, etc.
   - **Intended Purpose**: choose *Client Authentication*. There is an option for creating an IPSec certificate, that works too, but it is no necessary. It is supposed to guarantee the compatibility with other IPSec implementations.
   - **Cryptographic Service Provider**: Microsoft Base Cryptographic Provider v1.0
   - **Key Usage**: Signature.

     Note: This election show the private key can be only used for signing. Don't forget Windows 2000 IKE will reject a certificate which private key is not marked as suitable for signing. If you choose (Both), signing and encrypting, the certificate is OK.
   - **Key Size**: 1024. You can choose a higher value if selected *Microsoft Enhanced Cryptographic Provider*. But, even in that case, the request may fail because Windows 2000 is not fortified with the *High Encryption Pack* (we will discuss more about this later).

- **Create A New Key Set**: activate it.
- **Mark keys as exportable**: activate it. This is an important option because it will allow us to create a PKCS#12 file that we can carry to the Linux box and use the certificate there.
- **Use local machine store**: activate it.

6. Make the request. Click on **Submit** and you will get a message stating that you have received the certificate you requested.
7. Click on **Install this certificate**.

At this point, your screen must seems as this:



After these, we should have the CA certificate installed in our machine and to be sure let's check it opening the IPSec administration console and selecting the *Certificates* section.

Now, we should be able to view our just requested certificate into *Personal Certificates*. Select it and verify these points:

1. The certificate and the trust chain are valid up to the root certificate.
2. The root certificate is auto-signed, valid and appears as a *Trusted Root Certification Authority*,
3. And the most important point: check that as you select the certificate appears the following message below: "**You have a private key that corresponds to this certificate**". If it doesn't, the certificate is not suitable for signing and the IKE negotiation will not work. The most probable reason is you didn't select '**Use local machine store**' when you made the certificate request.

If everything is worked right, we have only to export the certificate in PKCS#12 format for being able to use it in other machines.

3. Connecting Windows 2000-Linux FreeS/WAN 1.6

Ok, once we got the certificates, let's start with the specific configuration for each system: Windows 2000 and Linux.

3.1. Configuring Linux.

The first requirement is getting the patch that enables the use of certificates. We have used the patch developed by Andreas Steffen, that you can find at http://www.strongsec.com/freeswan/. The patch ships well documented and you don't need additional information for installing it. It is recommendable that, from the same location, you get the Andreas Gruenbacher tool called *fswcert*, which will allow you to save effort and make a more graceful installation.

After installing these both tools, we have to configure and install the certificates we created. If you use OpenSSL you should have two files, named *Windows2000cert.pem* and *Freeswancert.pem*, then follow these steps:

1. Copy the certificates files to */etc/ipsec.d*
2. The Linux box certificate must be in */etc* too, and DER encoded. Place in */etc/ipsec.d* and type:

```
$ openssl x509 -in Freeswancert.pem -outform DER -out ../x509cert.der
```

doing this, you will transform the certificate to DER format and place it in the correct place.

3. Now we have to install the private key corresponding to our certificate in the */etc/ipsec.secrets* file. We'll use *fswcert* for this. Place where you installed it and type:

```
$ ./fswcert -k Win2000privatekey.pem
```

After typing the password required for decrypting the key, you will get an output you will be able to cut & paste directly into the */etc/ipsec.secrets* file. That operation should look more or less like this:

```
# This file holds shared secrets or RSA private keys for inter-Pluto
# authentication.  See ipsec_pluto(8) manpage, and HTML documentation.
# Shared secret (an arbitrary character string, which should be both long
# and hard to guess, enclosed in quotes) for a pair of negotiating hosts.
# Must be same on both; generate on one and copy to the other.

# RSA private key for this host, authenticating it to any other host
# which knows the public part.  Put ONLY the "pubkey" part into connection
# descriptions on the other host(s); it need not be kept secret.
: rsa  {
       # RSA 1024 bits   neuromante.ls.fi.upm.es   Tue Oct 17 17:56:07
       # (0x4200 = auth-only host-level, 4 = IPSec, 1 = RSA)
       Modulus:
0xB34C8AAB70798E7D1314665B903A73D9DCFAF0E1B9F84E101C5C23CB495F183B175457CF0436
2330F160C40A3B932B23F543ED7537559188E19B5BCB049528D6DDE42F2FA0FEAF25201EB0BD62
6049F27A102888BDEC93B49BED175D03CC988203E8374DCC05108B6D9CA0101B156A31260D4D12
6EC37B48DAD73D407F73AF63
       PublicExponent: 0x010001
       PrivateExponent:
0x13AA4D7B4836CCDE02FDB5CC4D215C1B851530DE511B5ED47D87CD…
       Prime1: 0xDADB1520F264D6E8829ED0E577CCC9C22E64C6FDC73D03D53DE5D3B…
       Prime2: 0xD1BAC84E688EA1B854AB5C493DCC31D90FE90C5BB0BEA23261EF38EE…
       Exponent1: 0xAB16B3A8C2F543F01614D3975F950F70A6D60F9DDAC07F1B97BCF0…
       Exponent2: 0x3B4BC4D02E4D8D39916EB573DBADFCB5F3029FC4D8AEBD2AE377803…
       Coefficient: 0xADF19CB3D176576862ACA89E1FC63D3EE3810691A88922933D181CC4…
       }
```

Now we need to configured */etc/ipsec.conf* file to point to the authentication type to use and where the certificates can be found:

```
# /etc/ipsec.conf - FreeS/WAN IPSEC configuration file
# sample connection
conn Win2000
        # RSA authentication with certificates.
        authby=rsasig
        # Left security gateway, subnet behind it, next hop toward right.
        left=138.100.10.147
        leftsubnet=138.100.10.147/32
        leftcert=Win2000cert.pem
        # Right security gateway, subnet behind it, next hop toward left.
        right=138.100.10.244
        rightsubnet=138.100.10.244/32
        rightcert=Freeswancert.pem
        # Authorize this connection, but don't actually start it, at startup.
        auto=add
       pfs=no
```

If you use the Microsfot CA, you must have a .p12 file that contains your certificate, your private key and the CA certificate. Ok, use OpenSSL for do it:

```
$ openssl pkcs12 -in Windows2000.p12 -info
```

From the output, you can cut & paste the elements you need: the private key and the certificate, into two separate files. The process from here is identical to the described before.

3.2. Configuring Windows 2000.

In order to be able to negotiate with Linux, Windows 2000 must be "fortified" because it only supports simple DES as encryption algorithm. This algorithm is considered insecure by FreeS/WAN developers and many other sources, so Linux will reject any proposal that contains it. So, the common encryption algorithm must be 3DES. To "fortify" Windows 2000 you need, therefore, the *High Encryption Pack*, which can be found at:

http://www.microsoft.com/WINDOWS2000/downloads/recommended/encryption/default.asp

Ok, imagine that we have our Windows 2000 box ready, now you have to install some certificates if you used OpenSSL a issuing tool. The process is as simple as opening the Certificates console and importing them. Place the CA certificate into *Trusted Root Certification Authorities* and check the trust chain verifies. If you have used the Microsoft Test CA, you should have the certificates installed and ready to work. The tunnel configuration is not different from a preshared secret connection[2], excepting, of course, the authentication method. Choose *Certificate Authentication* and pick the appropriate root CA.

Be aware of Windows 2000, unlike Linux, does not need the peer certificate for initiating a negotiation. The only thing you have to indicate is which will be your common CA. Later, Windows will verify that the received certificate is signed by the CA you choose or, extending the trust chain, reach at a CA which you positively rely on. Moreover, you don't have to explicitly indicate which certificate you want Windows 2000 to use; whether you have several certificates, Windows will choose the first valid stored in *Personal Certificates*.

It could be very useful enable the built-in IPSec logging Windows 2000 ships. For doing that, find (if it exists, and, if don't, create it) the key in the registry:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\PolicyAgent\Oakley
```

Add a REG_DWORD value named **EnableLogging** and assign it a value of 1. Since then, you can glance at a file called *oakley.log*, which will be created in `%SystemRoot%\debug`. All this information could be found in detail at:

http://www.microsoft.com/windows2000/library/planning/security/ipsecsteps.asp

[2] You can find an example in the FreeS/WAN Compatibility Guide documentation.

4. Connecting Windows 2000-PGPNet 7.0

We have already learned how to solve the most important difficulties: we created the certificates, didn't we? So, the only problem from here is the configuration.

Don't you have imported the certificates into PGPNet yet? Ok, no problem. Whether you made them with OpenSSL or Microsoft CA, you can import the .p12 file into the pubkey ring. Open PGPKeys and then Keys→Import.

Then, open PGPNet and choose x509 certificates as authentication method. Be aware of only PGP Personal Privacy will you allow to use certificates. With evaluation version you are only able to use preshared keys and establish host-to-host SA, not between subnets.

5. Connecting PGPNet 7.0-Linux FreeS/WAN 1.6

The mater here is the same than before, then only handicap you have to face is the famous "1500 bytes limit" all the people is talking about in the mail list.

The issue is that there seems to be a problem because PGPNet is not able to reorder the fragments of the 5th IKE Phase 1 message which Linux sends in reverse order. "Reverse order" means the smaller fragment is sent first. And we say "there seems to be" because there are opinions against this. Our personal experience is that we haven't had problems with this. Even with huge keys and therefore huge certificates, the negotiations finished correctly every time we tried. So, our advice is give it a try. If you find problems, glance at the mail list and ask for help.

6. Conclusions

We have seen how the Windows 2000, PGPNet 7.0 and FreeS/WAN 1.6 interoperatibility is possible. The only secret is to make sure the validity dates of the CA and machine certificates nest. Other important issue to bear in mind is your certificate must have a private key for being capable to sign. Check this with the certificate viewer in your Windows 2000 box. Take a look at the IPSec log generated by Windows but don't rely too much on it: it's far from be accurate. I've spend almost a week trying Windows to accept my OpenSSL-generated certificate. The problem was the validity date didn't nest as we have seen, but the log gave another reason: "Trust failed" so I was verifying the CA sign over the certificate, the own CA certificate and other many meaningless and paranoid things for a while.