

How to use X.509v3 certificates for authentication with isakmpd.

Copyright 2000, Jörgen Granstam

This document is still under construction and the information is partly untested and might be completely wrong. Consider yourself warned.

Preface

This document is made available in the belief that it might be useful to someone. The author is not responsible for any consequences that use or misuse of this information might cause. The information is not guaranteed to be correct or even tested at all. You are allowed to copy, redistribute, modify and reuse this document in whole or in part in any way you want (including commercial distribution) as long as you do not remove my copyright notice or these conditions and clearly mark any significant modifications you do as made by you.

By "significant modification" I mean anything except correcting spelling errors or similar minor changes. If you don't like these conditions, ask me to make an exception in your particular case. It will generally be granted if I can not find some strong reason against. Oh, and if you're considering any kind of non-online distribution (i.e. on CD, printed on paper), please check the facts, we don't want to spread misinformation, do we?

Please tell me if you find any errors in this document and I will correct them as soon as possible. Also please ask questions. Not long ago, I myself had not managed to configure isakmpd. The manuals seemed incomplete or at least unclear. Now that I know how to do this, the manuals makes perfectly sense (although, the manuals would need more detailed information in some cases).

Also thanks to the authors of isakmpd in general and especially to Niklas Hallqvist och Angelos Keromytis who helped me by correcting bugs in isakmpd and by helping me understand more about how isakmpd works. Thanks also to Patrick Ethier whos documentation on using isakmpd with preshared keys was of great assistance and also thanks to everyone else on the OpenBSD misc mailinglist that helped figuring this out.

[This document is now also available as PDF](#) thanks to Mohammad Rizal Othman who made the conversion.

The configurations in this document have not yet been fully tested please tell me if you find any errors.

Jörgen Granstam <Jorgen.Granstam@abc.se>

Introduction

Isakmpd now also supports the use of X.509 certificates for authentication and not just preshared keys. Using certificates can be a great advantage compared to using pre-shared keys in many cases.

Setting up isakmpd to use certificates instead of pre-shared keys is not really that much harder and in a big network with many untrusted peers it might actually simplify configuration and perhaps most important, the key management.

Updating software.

To get isakmpd to work with certificates it is currently (that is for users of OpenBSD 2.6) necessary to update some of the software. There were a few bugs in the isakmpd shipped with OpenBSD 2.6 that makes it impossible to use certificates for authentication with that version. Thus, get the most recent version of isakmpd from CVS.

Also, there is a problem with the isakmpd linking to libcrypto. Niklas Hallqvist described this problem in some detail on the misc mailinglist. The date and time of his mail was 01/25/2000 02:28:50. Search the archives if you need the details.

To get isakmpd working it is however enough to know that isakmpd must be recompiled *after* the libcrypto libraries have been upgraded to an RSA-enabled version. Thus, install the updated libcrypto, get the most recent isakmpd source from CVS (versions after 01/31/2000 should work, for the basic configuration, but more bugs have been found since, some of which may be bad for security in some cases) and recompile and install isakmpd.

Isakmpd also uses the keynote library, libkeynote, and although not absolutely necessary to get isakmpd working it is thus also be a good idea to get the latest libkeynote code and recompile and install it too before compiling isakmpd. Bugs have been found in libkeynote since the release of OpenBSD 2.6, it might still work with some configurations in this document, but not with all. Certpatch that also had bugs in OpenBSD 2.6, is part of the

isakmpd source and will thus also be updated to a corrected version when isakmpd is updated.

Isakmpd configuration (without certificates).

The first step for anyone who wants to use isakmpd should be to get isakmpd to work at all with a simple configuration. The simplest case is to use preshared keys. There is an excellent description by Patrick Ethier about how to set up isakmpd with preshared keys at:

<http://www.secureops.com/resources/vpn/>

It is assumed that anyone reading this document already have an isakmpd configuration that work when using preshared keys (or at least that the reader already know enough about isakmpd configuration to set up isakmpd in such a configuration). Also check the isakmpd.conf and the isakmpd.policy man-pages.

Now lets go on and get isakmpd working with certificates also.

Generating certificates.

There is a good description of how to generate keys and certificates in the README.PKI file in the isakmpd source directory. You need to have a CA key, a corresponding CA X.509 certificate, one private key for each computer on the network that will use isakmpd and one X.509 certificate for each such key.

The X.509 certificates need to have a Subject Alternative Name (SubjectAltName) extension describing the certificate holder. How to set a SubjectAltName extension using certpatch for a certificate is also described in README.PKI for the case of setting an IP address as the SubjectAltName. Using an IP address here is also the default behaviour for isakmpd.

Certpatch also support using either a FQDN (Fully Qualified Domain Name) or a UFQDN (User FQDN). An example of an FQDN might be www.openbsd.org, an example of an UFQDN would be an email address. Something like Jorgen.Granstam@abc.se for example.

In this howto document I am going to use FQDNs as SubjectAltNames. Using IP addresses would be a bit easier since that is the default behaviour of isakmpd but it is not much of a difference as we soon will see.

To insert an FQDN SubjectAltName into a certificate one would do something like this:

```
/usr/sbin/certpatch -t fqdn -i home.mysite.se -k ca.key \
                    originalcert.crt newcert.crt
```

Here the ca.key is the private key of the CA, thus this can only be done by whoever has access to the CA private key. The (fictional) home.mysite.se is the FQDN to be inserted into the certificate. The originalcert.crt and newcert.crt filenames may be the same name in which case the original file will be overwritten by the new moified certificate.

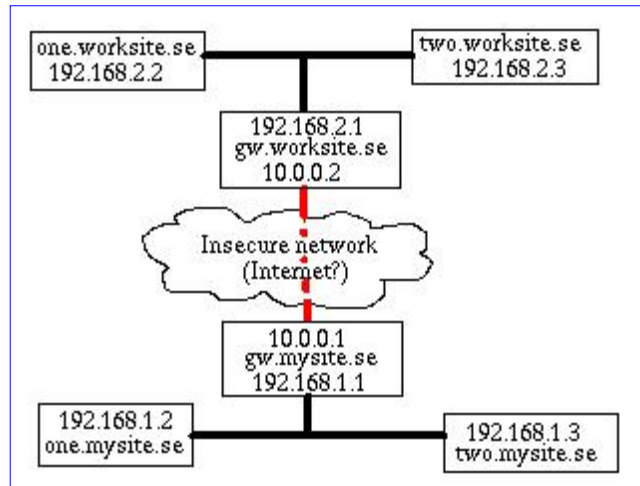
Put the keys and certificates in the directories as deccribed at the end of README.PKI. The CA key (ca.key) should be kept in some safe place if the keys are really to be seriously used.

Configuration of isakmpd

Lets now look at the /etc/isakmpd/isakmpd.conf configuration file. It was originally taken from the example file in the isakmpd.conf manpage but have been heavily modified. I will also use a much shorter file here than the example file in the man-page, I have removed most parts from that file that is not needed in this setup. In part to simplify the understanding of this setup. I have also added some commenting (some comments are left as in the manpage) and changed some names. None of the domain names used here exists as far as I know.

Actually since everyone who reads this already have a working configuration for the preshared keys case (right? :-)) there won't be many surprises in this file. I won't explain this in every detail, check the isakmpd.conf manpage for descriptions of the parts I don't comment on.

Lets assume our setup looks something like this



That is, two networks that should be connected using an IPSec tunnel over an otherwise insecure network. Ignore the fact that I am using IP addresses reserved for private Internets here (RFC1918), I have to use something. I wont explain how to use isakmpd in combination with NAT or similar (because I haven't tried that myself).

Now, lets look at the configuration file. This is the file for the security gateway gw.mysite.se:

```
# *****
# ***** Start of the gw.mysite.se isakmpd.conf *****
# *****

# A configuration sample for the isakmpd ISAKMP/Oakley (aka IKE) daemon.
[General]
Policy-File=          /etc/isakmpd/policy
Retransmits=         5
Exchange-max-time=   120
Listen-on=           10.0.0.1

# The name work-gw here is used just as a section name and a tag for
# use in this configuration file below and need not actually be the
# real hostname or domain name of the peer (but it could be). The IP
# address however needs to be correct. Phase 1 as you might already
# know is to negotiate an ISAKMP security association (SA). There
# should ofcourse be one IP and name for each peer we want to
# communicate with.
[Phase 1]
10.0.0.2=             work-gw

# Now phase 2 is negotiating IPsec SAs. As in phase 1, the name here
# is a section name to be used later. Actually, it can be a comma
# separated list of section names here. Thus if traffic from many
# networks (or individual hosts) should be forwarded through this
# tunnel, more sectionnames would be added (and ofcourse corresponding
# new sections further down).
[Phase 2]
Connections=         work-gw-my-gw

# Now, here are some parameters for the ISAKMP SA negotiations. Almost
# self documenting. The section name is from [Phase 1] above. The most
# interesting tag might be the ID tag. The ID tag is set to the name
# of the section where the identity information about this host that
```

```
# will be presented to connecting peers, can be found. If the ID tag
# is not available, isakmpd will assume that it will identify itself
# using the IP address. You might also notice that there is no longer
# any authentication tag here in this configuration. The authentication
# data is currently used only in the preshared key case.
[work-gw]
Phase=                1
Transport=            udp
Local-address=        10.0.0.1          # Local address
Address=              10.0.0.2          # Peer address
ID=                  my-ID
Configuration=        Default-main-mode

# This is the identity data. ID-type may also be IPV4_ADDR (the
# default), IPV4_ADDR_SUBNET or UFQDN. The Name tag is used for
# FQDN and UFQDN, for IPV4_ADDR an Address tag would be used instead.
# For IPV4_ADDR_SUBNET a Network and a Netmask tag would be used.
[my-ID]
ID-type=              FQDN
Name=                 gw.mysite.se

# This is the section for the IPSec connection. The section name is
# from the list in the [Phase 2] section above. The ISAKMP-peer is
# ofcourse the tag of our peer from section [Phase 1] above. The
# Local-ID and Remote-ID tags should be section names describing which
# packages should be forwarded over the IPSec tunnel to the remote
# network.
[work-gw-my-gw]
Phase=                2
ISAKMP-peer=          work-gw
Configuration=        Default-quick-mode
Local-ID=              Net-west
Remote-ID=             Net-east

# Any packet originating from a computer on the network described
# here...
[Net-west]
ID-type=              IPV4_ADDR_SUBNET
Network=              192.168.1.0
Netmask=              255.255.255.0

# ... and with a destination matching the network described here,
# will be encrypted and forwarded over the IPSec tunnel to the remote
# system.
[Net-east]
ID-type=              IPV4_ADDR_SUBNET
Network=              192.168.2.0
Netmask=              255.255.255.0

# Main mode descriptions

# Here are the data for main mode. Using DES here for real purposes
# are not very smart since DES is no longer considered a secure
# encryption algorithm. 3DES is generally considered to have much better
# security since it has has enough bits in the key to be considered secure.
# Transforms is a list of tags describing main mode transforms. In
# this example we have only one.
```

```
[Default-main-mode]
DOI=                IPSEC
EXCHANGE_TYPE=     ID_PROT
Transforms=        3DES-MD5

# Certificates stored in PEM format
# This is important when using certificates. The CA certificates should
# be in the CA-directory (but not the CA private key ofcourse).
# The Cert-directory should have at least the certificate for the
# local host but other certificates are also allowed. The private key
# should be the private key of the local host.
[X509-certificates]
CA-directory=       /etc/isakmpd/ca/
Cert-directory=     /etc/isakmpd/certs/
Private-key=        /etc/isakmpd/private/local.key

# Main mode transforms
#####

# Here is our main mode transform the important thing here is to use
# RSA_SIG as authentication method when using certificates. It is the
# only method supported when using certificates so far. Commercial
# entities in the US will thus have to wait until September 2000 to
# use this due to the RSA patent. Luckily, I am not living in the US.
# Also important is the GROUP_DESCRIPTION tag. It must match the
# GROUP_DESCRIPTION tag in the Quick mode transforms further down.
# The Life tag here could possibly be modified. The LIFE_60_SECS might
# be shorter than necessary for normal use.

[3DES-MD5]
ENCRYPTION_ALGORITHM= 3DES_CBC
HASH_ALGORITHM=       MD5
AUTHENTICATION_METHOD= RSA_SIG
GROUP_DESCRIPTION=    MODP_1024
Life=                 LIFE_60_SECS,LIFE_1000_KB

# Quick mode description
#####

[Default-quick-mode]
DOI=                IPSEC
EXCHANGE_TYPE=     QUICK_MODE
Suites=            QM-ESP-3DES-MD5-PFS-SUITE

# Quick mode protection suites
#####
# 3DES

[QM-ESP-3DES-MD5-PFS-SUITE]
Protocols=          QM-ESP-3DES-MD5-PFS

# 3DES

[QM-ESP-3DES-MD5-PFS]
PROTOCOL_ID=        IPSEC_ESP
Transforms=         QM-ESP-3DES-MD5-PFS-XF

# Quick mode transforms
```

```

# Don't forget. The GROUP_DESCRIPTION must match the GROUP_DESCRIPTION
# in main mode above. For forwarding packets between two networks (or
# from a host to a network, we use TUNNEL mode. Between two hosts we
# may also use TRANSPORT mode instead.
[QM-ESP-3DES-MD5-PFS-XF]
TRANSFORM_ID=          3DES
ENCAPSULATION_MODE=    TUNNEL
AUTHENTICATION_ALGORITHM=  HMAC_MD5
GROUP_DESCRIPTION=     MODP_1024
Life=                  LIFE_60_SECS

# As we know from the isakmpd.config manpage The LIFE_DURATION here is
# an offer value (60), a minimum acceptable value (45) and a maximum
# acceptable value. The isakmpd.conf example has this set to
# 600,450/720 instead. That might be a better value for normal use.
[LIFE_60_SECS]
LIFE_TYPE=              SECONDS
LIFE_DURATION=          60,45:72

[LIFE_1000_KB]
LIFE_TYPE=              KILOBYTES
LIFE_DURATION=          1000,768:1536

# *****
# ***** End of the gw.mysite.se isakmpd.conf *****
# *****

```

So far the configuration for the local system. The remote system is configured just the same way only opposite. Thus only the first part of the isakmpd.conf file differs. Lets just look at that first part of the isakmpd.conf file for the security gateway gw.worksite.se:

```

# *****
# ***** Start of the gw.worksite.se isakmpd.conf *****
# *****

[General]
Policy-File=            /etc/isakmpd/policy
Retransmits=            5
Exchange-max-time=     120
Listen-on=              10.0.0.2

[Phase 1]
10.0.0.1=               my-gw

[Phase 2]
Connections=            work-gw-my-gw

[my-gw]
Phase=                  1
Transport=              udp
Local-address=          10.0.0.2          # Local address
Address=                10.0.0.1          # Peer address
ID=                     work-ID
Configuration=          Default-main-mode

[work-ID]
ID-type=                FQDN
Name=                   gw.worksite.se

```

```
[work-gw-my-gw]
Phase=                2
ISAKMP-peer=         my-gw
Configuration=       Default-quick-mode
Local-ID=             Net-east
Remote-ID=           Net-west

# *****
# ***** .. to be continued *****
# *****
```

Now that wasn't so hard, just a bit booring to read perhaps. A slightly more interesting part next.

The policy file.

Actually, the policy file might slightly confusing for anyone who has not used it before, especially if things doesn't work as expected. The `isakmpd.policy` man-page is not really that bad. It might perhaps be a little bit unclear in some parts but generally it's good.

The simplest possible working policy file would containg just a single line:

```
authorizer: "POLICY"
```

This basically means that that there is no policy limitations on who would be allowed to connect. Thus not a very secure setup. The authorizer tag here means the one who has the authorization to decide the policy. The special authorizer "POLICY" has the ultimate and unlimited authority on policy. Any other authorizer must first be authorized by "POLICY" to have any authority here.

There can also be a set of conditions for what is allowed. The following policy thus would mean that only someone using the ESP protocol with some real encryption would be authorized (oh well, someone using DES would also be authorized here although DES could almost be considered snakeoil today, it is left as an exercise for the reader to change this policy into not allowing DES either). Note that anyone who does encrypts with ESP would still be allowed.

```
authorizer: "POLICY"
conditions: app_domain == "IPsec policy" &&
            esp_present == "yes" &&
            esp_enc_alg != "null" -> "true";
```

It is also possible to "sublicense" authority to someone else (might be one or more entities). The simple case would be the preshared key case. In that case, anyone who knows the preshared passphrase is authorized. Thus:

```
authorizer: "POLICY"
licensees:  "passphrase:something really secret"
conditions: app_domain == "IPsec policy" &&
            esp_present == "yes" &&
            esp_enc_alg != "null" -> "true";
```

This would authorize anyone who knows this passphrase to connect and comply with the conditions (but remember that the passphrase must also be set in the Authentication tag in `isakmpd.conf`).

Nothing difficult so far. Now to the interesting stuff. First there can be many licensees, although all must be authorized by "POLICY". Further, authorized licencees can sublicense to other licensees. A licensee can be just a string in case it is further described in the policy file:

```
authorizer: "POLICY"
licensees:  "subpolicyAH" || "subpolicyESP"
conditions: app_domain == "IPsec policy" -> "true";
```

```

authorizer: "subpolicyESP"
licensees: "passphrase:something more secret"
conditions: esp_present == "yes" -> "true";

```

```

authorizer: "subpolicyAH"
licensees: "passphrase:something really secret"
conditions: ah_present == "yes" -> "true";

```

And now to what everyone has been waiting for. Policy can also be sublicensed or delegated to a key. In this case it is usually a X.509 certificate. The simple use of certificates would be to use them like the passphrases. Just insert individual users certificates in the policy file:

```

keynote-version: 2
comment: This is an example of a policy delegating to a key.
authorizer: "POLICY"
licensees: "x509-base64:\
    MIIIBStCCARoCAQAwdQYJKoZIhvcNAQEEBQAwwITELMAkGA1UEBhMCc2UxEjAQBGNV\
    BAMTCUllRUxBQIBDQTAeFw0wMDAxMjg5NzQyMTZaFw0wMTAxMjcxNzQyMTZaMCEX\
    CZA          This is would be a user certificate          AQEB\
    BQA          IUuz\
    eOW8P5UGJUH2JVkiaA2CTDryFf0CHYwd2P003dtVYw5RvET7XLMpRZiCcWtBdxneW\
    ct+016zUBP/cQMM1+KownxAUq9ezA8GvTyUWC97SOMOgoVj/QR3FHMEjPui3AgMB\
    AAEwDQYJKoZIhvcNAQEEBQAwwITELMAkGA1UEBhMCc2UxEjAQBGNV\
    iWLV5q9H3wickVGN0NPerq0YLwx/VA9WaecYN8V+ALtNKYPuDiT11zwwE8GQeaa\
    NuzgmQ9hh3GifEgN9VEic3j4kTytonKr0Q+vTLM7xYzheOxvrtUErRwZ9Xs1KzHe\
    yiXHSU8="
conditions: app_domain == "IPsec policy" &&
            esp_present == "yes" &&
            esp_enc_alg != "null" -> "true";

```

Now, this is obviously a stupid idea if there are a lot of users. The certificates that isakmpd reads from the CA- and Certificate directories, and the certificates received from the peer is converted into pseudo credentials. Such certificates converted into pseudo credentials essentially would look something like:

```

authorizer: "x509-base64:\
    MIIIBStCCARoCAQAwdQYJKoZIhvcNAQEEBQAwwITELMAkGA1UEBhMCc2UxEjAQBGNV\
    BAMTCUllRUxBQIBDQTAeFw0wMDAxMjg5NzQyMTZaFw0wMTAxMjcxNzQyMTZaMCEX\
    CZA          This is would be the public key/certificate of the          AQEB\
    BQA          signer of the user certificate (i.e. the CA certificate)IUuz\
    eOW8P5UGJUH2JVkiaA2CTDryFf0CHYwd2P003dtVYw5RvET7XLMpRZiCcWtBdxneW\
    ct+016zUBP/cQMM1+KownxAUq9ezA8GvTyUWC97SOMOgoVj/QR3FHMEjPui3AgMB\
    AAEwDQYJKoZIhvcNAQEEBQAwwITELMAkGA1UEBhMCc2UxEjAQBGNV\
    iWLV5q9H3wickVGN0NPerq0YLwx/VA9WaecYN8V+ALtNKYPuDiT11zwwE8GQeaa\
    NuzgmQ9hh3GifEgN9VEic3j4kTytonKr0Q+vTLM7xYzheOxvrtUErRwZ9Xs1KzHe\
    yiXHSU8="
licensees: "x509-base64:\
    MIIIBStCCARoCAQAwdQYJKoZIhvcNAQEEBQAwwITELMAkGA1UEBhMCc2UxEjAQBGNV\
    BAMTCUllRUxBQIBDQTAeFw0wMDAxMjg5NzQyMTZaFw0wMTAxMjcxNzQyMTZaMCEX\
    CZA          This is would be the key of the subject of the          AQEB\
    BQA          certificate          IUuz\
    eOW8P5UGJUH2JVkiaA2CTDryFf0CHYwd2P003dtVYw5RvET7XLMpRZiCcWtBdxneW\
    ct+016zUBP/cQMM1+KownxAUq9ezA8GvTyUWC97SOMOgoVj/QR3FHMEjPui3AgMB\
    AAEwDQYJKoZIhvcNAQEEBQAwwITELMAkGA1UEBhMCc2UxEjAQBGNV\
    iWLV5q9H3wickVGN0NPerq0YLwx/VA9WaecYN8V+ALtNKYPuDiT11zwwE8GQeaa\
    NuzgmQ9hh3GifEgN9VEic3j4kTytonKr0Q+vTLM7xYzheOxvrtUErRwZ9Xs1KzHe\
    yiXHSU8="
conditions: app_domain == "IPsec policy" -> "true";

```

Now note that these are not authorized by "POLICY" and thus won't have any effect without a policy authorizing them somehow. Further, this showed what happens to certificates internally. The above credential is thus not seen

in the policy file. However, it is possible to sublicense to such credentials. Remember sublicensing above. It is thus possible to license all certificates that are signed by a certain CA by putting the CA certificate as a licensee to "POLICY":

```
keynote-version: 2
comment: This is an example of a policy delegating to a key.
authorizer: "POLICY"
licensees: "x509-base64:\
    MIIBStCCARoCAQAwDQYJKoZIhvcNAQEEBQAwwITELMAkGA1UEBhMCc2UxEjAQBgNV\
    BAMTCUllRUxBQIBDQTAeFw0wMDAxMjg0NzQyMTZaFw0wMTAxMjg0NzQyMTZaMCEX\
    CZA          This would be the CA certificate          AQEB\
    BQA          IUuz\
    eOW8P5UGJUH2JVkiaA2CTDryFf0CHYwd2P003dtVYw5RvET7XLMpRZiCcWtBdxneW\
    ct+016zUBP/cQMMl+KownxAUq9ezA8GvTyUWC97SOMOgoVj/QR3FHMEjpUi3AgMB\
    AAEwDQYJKoZIhvcNAQEEBQAwwITELMAkGA1UEBhMCc2UxEjAQBgNV\
    iWLV5q9H3wickVGN0NPerq0YLwx/VA9WaecYN8V+ALtNKYPuDiT1lzwvE8GQeaa\
    NuzgmQ9hh3GifEgN9VEiC3j4kTytonKr0Q+vTLM7xYzheOxvrtUErRwZ9Xs1KzHe\
    yiXHSU8="
conditions: app_domain == "IPsec policy" &&
             esp_present == "yes" &&
             esp_enc_alg != "null" -> "true";
```

Thus the above policy is a simple example of a policy that delegates to a CA. Thus any user that has a certificate that is signed by the CA that has this certificate and otherwise comply to other conditions set by the policy and the configuration file would be authorized.

Almost secure, is not secure.

Now, to be really safe, this is not enough unfortunately. There are ways to attack a security gateway that is configured in this way. If you really don't want the details, skip a to the next section now. To everyone else, lets try to understand why this is not secure. It's not hard.

Consider what information one of the isakmpds has access to at this stage. From the configuraton file, isakmpd knows which IP-address its peer will send from (in the [Phase 1] section). From the information it gets at the phase 1 negotiation it knows the ID that the peer presents itself with and the certificate it gets from the peer proves that the peer really have this ID. Looks fine so far?

Well, if the ID information was the IP (the default situation if we do not provide a phase 1 ID section) everything would be fine. The CA would have tied the IP to the cert and the IP in the configuration would be all information we need. It would be possible for an imposter to use the same IP from another computer in some cases (i.e. if both computers were on the same local network and the computer that usually have this IP is down for some reason). It should however not be possible for an imposter to be able to have a certificate and a corresponding private key that (falsely) proves that this IP belongs to the imposter.

If that ever happened, the imposter have managed to either steal the private key from the real owner of the IP, or the imposter has managed to fool the CA into issuing a certificate containing false information somehow. If any of these things happens, then either the private key have not been protected well enough, the CA have failed to check the identity of the imposter well enough (or the ID info for the cert) or the CA private key has net been well enough protected. Since all these are prerequisites for security to work at all, none of these situations can be allowed to ever occur.

Now, in our example the situation is different. Here we actually have an FQDN in the certificate instead of an IP address. Since we still have an IP address in the [Phase 1] section this will result in a possible security problem. What now would happen during an ISAKMP phase 1 negotiation would be that we could check that the peer was sending from the expected IP (but as explained earlier that could possibly be forged in some situations). We could check that the ID our peer presents actually belongs to our peer. But what we can not check now is if that ID really is the ID we expect our peer to have, because isakmpd have never been told what ID that should be.

Someone now might say that the DNS system ties the IP to the FQDN for the host. That is true, however todays DNS system is not secure and can in under some circumstances be fooled to give out false information (or, it could be subjected to a denial of service (DoS) attack by an attacker, and the attackers computer might be able to fake the DNS servers answer). Secure DNS will come in the future, but it is not here yet (at least most DNS

servers are not secure yet), thus today, using DNS to check if the FQDN in the cert corresponds to the expected IP is no guarantee. In fact isakmpd does not check this with DNS. Even if DNS was secure, checking this would not help in the case of using an UFQDN.

Thus in the case of having an FQDN as ID, it could be possible for an attacker to get an own private key and having this key signed by the same CA that we use (but with the attackers own FQDN ofcourse). Then launch a DoS attack on our peer so that it goes down (in fact, there are some flaws in the ISAKMP protocol itself that possibly could be used to launch a remote DoS against the peer and make it go down, although I don't know how sensitive isakmpd is to those attacks). Then the attacker could configure its own computer in the same way as our peer, connect it to our peers network and try to connect using its own ID, private key and certificate.

Since our own isakmpd has not been informed about what ID to expect, all it can do is to check that the IP address is the expected IP of our peer (and it is because the attacker is identically configured as our peer besides the certificate, ID and private key). Further our isakmpd can check that the certificate was signed by the same CA (but most CAs sign lots of certs, a cert might not be hard to get), and that the presented ID is the same as the ID in the cert. However it would not with the configuration presented so far check that this ID is the expected ID. Thus the attacker would be allowed to connect.

Preventing the attack.

The question now thus is, how can we inform isakmpd about what ID to expect? This is fortunately easy, and documented in the isakmpd.policy man-page. We must do the check in the policy. Like this:

```
keynote-version: 2
comment: This is an example of a policy delegating to a key.
authorizer: "POLICY"
licensees: "x509-base64:\
    MIIBsTCCARoCAQAwDQYJKoZIhvcNAQEEBQAwITELMAkGA1UEBhMCc2UxEjAQBGNV\
    BAMTCUllRUxBQIBDQTAeFw0wMDAxMjg5NzQyMTZaFw0wMTAxMjg5NzQyMTZaMCEX\
    CzA          This would be the CA certificate          AQEB\
    BQA          IUuz\
    eOW8P5UGJUH2JVkiA2CTDryFf0CHYwd2P003dtVYw5RvET7XLMpRZiCcWtBdxneW\
    ct+016zUBP/cQMMl+KownxAUq9ezA8GvTyUWC97SOMOgoVj/QR3FHMEjpUi3AgMB\
    AAEwDQYJKoZIhvcNAQEEBQADgYEALGShaAxHvGncev0iFnKrJI4x5T4vlaMPlad+\
    iWLV5q9H3wickVGN0NPerq0YLwx/VA9WaecYN8V+ALtNKYPuDiT11zwwE8GQeaa\
    NuzgmQ9hh3GifEgN9VEiC3j4kTytonKr0Q+vTLM7xYzheOxvrtUErRwZ9Xs1KzHe\
    yiXHSU8="
conditions: app_domain == "IPsec policy" &&
            esp_present == "yes" &&
            esp_enc_alg != "null" &&
            remote_id == "gw.worksite.se" -> "true";
```

Now only gw.worksite.se should be able to get an IPsec connection. More allowed IDs could easily be added by adding more alternative remote_id checks, i.e. by having conditions like these in the policy:

```
conditions: app_domain == "IPsec policy" &&
            esp_present == "yes" &&
            esp_enc_alg != "null" &&
            (remote_id == "gw.worksite.se" ||
             remote_id == "gw.somesite.se" ||
             remote_id == "gw.whatsite.se") -> "true";
```

With this policy either of gw.worksite.se, gw.somesite.se or gw.whatsite.se could connect.

Some might say that is unfortunate that there has to be entire certificates inserted in the policy. It requires some work to reformat the certificates into the format in the policy and it makes the policy rather unreadable. If someone by mistake replaced a user certificate with the corresponding CA certificate somewhere in a complex policy it might cause unauthorized users to be allowed to connect in some cases and worse, it would not be easily detectable by reading through the policy file (since X.509 certificates are not in a human readable format).

Now, for the really bleeding edge people out there a solution to this problem is available. It is now possible to use the certificate Distinguished Name (DN) instead of a certificate in the policy (the corresponding certificate must

ofcourse be available from the certs or ca directories on disk so that isakmpd can find it). With this format the policy above might look like something like this instead:

```
keynote-version: 2
comment: This is an example of a policy delegating to a key.
authorizer: "POLICY"
licensees: "DN:\C=se\CN=IKELAB CA"
conditions: app_domain == "IPsec policy" &&
             esp_present == "yes" &&
             esp_enc_alg != "null" &&
             (remote_id == "gw.worksite.se" ||
              remote_id == "gw.somesite.se" ||
              remote_id == "gw.whatsite.se") -> "true";
```

Much more readable, isn't it? The information about what the exact DN for a certificate is can be found by looking at the certificate using the openssl utility. Something like:

```
openssl x509 -text < ca.crt
```

More complicated policy configurations are ofcourse possible but this is a start anyway, and there are another example in the next section.

This should provide the necessary information about the certificate in ca.crt. Now this is good as long as we have a small or at least resonably small policy. It is however still not too great if we have a gigantic site with lots of users that should be allowed to connect.

Multiuser configurations and/or centrally managed authorisation.

Now, lets look at some really cool features of isakmpd. Previously we assumed that the expected peer was well known and had a static IP address. This is not always the case. Lots of people use dynamically assigned IPs or use many different computers. In other cases (like for a server) we might not know for sure who wants to connect.

Therefor one very nice feature of isakmpd is the ability to use a default tag instead of an IP in the [Phase 1] section, thus allowing isakmpd negotiations from any IP. This might thus look something like this:

```
[phase 1]
Default=          work-gw
```

First, it should be said that this configuration might not be secure from DoS attacks. As said before, there are some flaws in the ISAKMP/IKE protocols. Anyway, using a default [phase 1] section also enable us to use a kind of "authorisation certificates" instead.

Consider the case where we have a lot of authorized users but when we would not accept just any user. Like at a company. We would like company employees to be allowed to connect but nobody else. Now, imagine a big company where there might be thousands of employees. We might like them to all be able to connect from any computer (not only from within the company network) but everyone should not be allowed to do anything. It should now be possible to write a policy like this:

```
keynote-version: 2
authorizer: "POLICY"
licensees: "telnet@work" || "telnet@lab" || "pop3@work"
conditions: app_domain == "IPsec policy" &&
             esp_present == "yes" &&
             esp_enc_alg != "null" &&
             remote_id_type == "UFQDN" &&
             (remote_id == "telnet@worksite.se" ||
              remote_id == "pop3@worksite.se" ||
              remote_id == "telnet@lab.worksite.se") -> "true";
```

```
authorizer: "telnet@work"
licensees: "DN:\C=se\CN=IKELAB CA"
conditions: remote_id == "telnet@worksite.se" &&
             local_filter_type == "IPv4 address" &&
             local_filter_port == "23" &&
             local_filter == "192.168.002.003" -> "true";

authorizer: "telnet@lab"
licensees: "DN:\C=se\CN=IKELAB CA"
conditions: remote_id == "telnet@lab.worksite.se" &&
             local_filter_type == "IPv4 address" &&
             local_filter_port == "23" &&
             local_filter == "192.168.002.002" -> "true";

authorizer: "pop3@work"
licensees: "DN:\C=se\CN=IKELAB CA"
conditions: local_filter_type == "IPv4 address" &&
             local_filter_port == "110" &&
             local_filter == "192.168.002.003" &&
             remote_id == "telnet@worksite.se" -> "true";
```

This might not be exactly how it should be. This is as far as I know completely untested (in fact, these filter conditions might not work at all as I expect). Also, a policy such as this one (in fact any with default as peer IP), would require rewrites of the isakmpd.conf file too. This would have some security implications too. Further, for this kind of connections where anyone should be allowed to connect, It would probably be desirable to log the DN of anyone who connected. Isakmpd does not yet support that to my knowledge. Also this probably could have other security implications. Your on your own, you have been warned. The basic idea should be clear anyway.

Just in case someone missed the really interesting possibilities this would have. If all computers using ISAKMP/IKE this way had a standard set of conditions for all services the users might like to use from remote, the CA could actually authorize users by just putting the right SubjectAltName extensions in their certificates. Further, the expiration time for such certificates could be set to expire relatively often although the users would be able to download new reissued certificates when their current certificate is getting old. If the users misuse their authorizations, just stop reissuing the certificates and they wont get in more after it has expired. No need to change policy files on all computers just because an employee i.e. quits their job. The same scheme should work for other purposes than ISAKMP/IPsec too (including authorization for off-line systems!) although that would require special software. In any case, any organisation doing this would probably want to be their own CA.

Multiusers configurations (mobile users) like these are possible with pre-shared keys too, but then it is required that AGGRESSIVE mode is used instead of ID_PROT mode since we then must be able to choose the right password phrase based on ID since we do not know that from what the IP is in this case (in AGGRESSIVE mode the ID is sent over at an earlier stage of the negotiation, but it is sent unencrypted, thus AGGRESSIVE mode is fast because it needs fewer message exchanges, but it also is a bit less secure since the ID is sent in clear).

Happy IKEing!

Jörgen Granstam <Jorgen.Granstam@abc.se>