Remote user access VPN with IPsec

Emmanuel Dreyfus

October 24, 2005

Abstract

IPsec is a set of Internet Protocol (IP) extensions used to bring secure communication to the network level. IPsec can be used in various Virtual Private Network (VPN) scenarios such as bridging private networks or user remote access to a private network.

Remote User access VPN is an area where the the IPsec tools available on BSD systems were a bit frustrating. In this paper, we describe some requirements for a remote user access solution and how the existing solutions based on IPsec did not fully satisfy the requirements.

We then have a look to the IPsec extensions implemented by other vendors and how they would match our goals if we had them. The end of the paper tells how these extensions have been added to NetBSD IPsec stack, and how it led NetBSD to integrate software from the ipsec-tools project.

It is assumed that the reader is knowledgeable with TCP/IP networking.

1 Virtual Private Networks

A Virtual Private Network (VPN) is a link between two private networks, which are usually connected through the Internet.

The link is secured so that no one can easily eavesdrop or alter the traffic. This is why the VPN is said to be private. It also maintains the illusion of a single private network, without the Internet in between, and this is why we speak about a virtual network.

Maintaining the security of the VPN is not trivial. Of course the network traffic must be encrypted and checked against modifications, but the endpoints must also authenticate each other. If this mutual authentication is not done, the VPN is left vulnerable to Man in the Middle (MiM) attacks, where an attacker will impersonate each VPN endpoint and will be able to tamper with the network traffic.

The VPN can be a set of links between various sites of the same enterprise. In this situation the network administrator only has to deal with the secure communication between the border VPN gateways. This situation is quite comfortable, since it only deals with mutually authenticating machines. Many tools

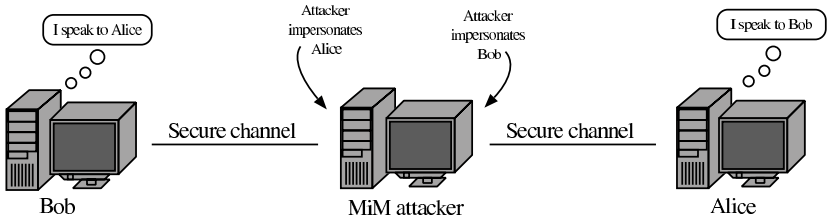

Figure 1: MiM attack

are available in BSD systems to get the work done.

The remote user access VPN is another scenario where one of the VPN endpoints is not a network but a single machine. It can be a road warrior accessing the private network from home or from a conference. In this situation, the network administrator is not likely to be the mobile machine administrator, so this is not just a problem of authenticating a machine – we need to authenticate the remote user instead.

2 Requirements for a remote user access VPN

Let us define the requirement we could have for a remote user access VPN. We want our remote user access VPN to

- be secure
- use login and passwords for user authentication
- be as simple as possible for users to configure
- use free software on the server
- be compatible with as many client Operating Systems (OSes) as possible

Here is a deeper review of the requirements.

2.1 Security

We want a secure VPN. We do not want attackers to eavesdrop or modify the traffic, and we do not want an unauthorized user to gain access to our private network.

2.2 Login and password authentication

There are a few ways of authenticating users. The weakest way is using a group password, one which is shared by all users. This is lower security because a shared password does not remain secret very long, as the administrator has no way to know who disclosed it if it gets disclosed.

Login and password is a bit better, because at least they are personal. We get a better idea of who is doing what. This is not highest security because passwords can be guessed or cracked.

A better user authentication is to use digital certificates. This is the highest security, but it may be impossible to manage in the real world. Certificate enrollment is not a trivial task and users may not be knowledgeable enough to manage their digital certificates.

Depending on the situation, login and passwords may be the best balance between security and usability. We will assume this is the situation here. We target a security level equivalent to SSHv2 (Secure SHell version 2) using password authentication: passwords cannot be eavesdropped, but they could be guessed, so we need to regularly check that user passwords are strong enough.

2.3 User friendliness

We want a solution as easy as possible to manage for users. In an ideal world, the user would only have to configure a VPN gateway IP, a login and a password, and the VPN connection should be made.

2.4 Free software on the server

For a lot of good reasons, we have a strong bias for using free software. It is easier to debug, it can be more easily enhanced, and it is free as in beer. So we would like our VPN gateway to run on a free OS.

2.5 Compatible with as many client systems as possible

We cannot rule out proprietary software on the mobile host, as we assumed that the network administrator was not managing it. So our solution must be compatible with Windows systems, for instance. But we also want to avoid locking out users of free OSes.

3 A short survey of remote user access VPN techniques

3.1 PPP over SSH

Point to Point Protocol (PPP) over SSH is about running a PPP session over an SSH tunnel. Security is handled by SSH, and PPP allows us to build a virtual network.

This solution meets the security and login authentication requirement. It is easy to implement on a BSD system as all the tools are available in the base system. It still has major drawbacks.

The first drawback is performance. The network stack when a web page is fetched through this kind of VPN would be

We can see that Transfer Control Protocol (TCP) is used twice in the stack. The two TCP layers will fight each other when trying to deal with network bandwidth, resulting in very poor performances.

Second, it is not easy at all to configure for a client running a Unix-like system, and third, if it is even possible at all, it would be very difficult to configure for a client running a Windows system.

So PPP over SSH looks like a quick and dirty solution for knowledgeable user, but it does not fulfill our requirements.

3.2 PPTP

Point to Point Tunneling Protocol (PPTP) is a VPN protocol designed by Microsoft. Built-in support for it has been available since Windows NT 4.0, and a few free-software implementation are available for Unix-like systems. Apple also provides a built-in PPTP capability since MacOS X.3.

PPTP has a long and scary history of security flaws [PPTP], and Microsoft seems to be adopting Layer 2 Tunneling Protocol (L2TP) over IPsec now. PPTP does not appear to be a good candidate to build a new VPN solution today.

3.3 Tunneling over SSL

Secure Socket Layer (SSL) is a security layer used on the top of TCP to secure application layer traffic. It uses digital certificate for mutual authentication and has provisions for the situation where only the server side uses a certificate. In that situation, a secure communication can take place where the client knows it speaks to the server, and the server does not know who it is speaking with.

At that stage, the client can authenticate itself through the secure channel. This may be done with login and passwords, and the password cannot be eavesdropped easily.

OpenVPN [OpenVPN] is a free software project that proposes a VPN solution using tunneling over SSL. It supports authentication using login and passwords, while the server uses a certificate. This meets our security objective. It is reasonably easy to use and even has a graphical user interface for Windows and MacOS X. It also runs on a lot of different Operating Systems.

Finally, OpenVPN uses User Datagram Protocol (UDP) as its transport layer, so it does not suffers the performance issue we described for PPP over SSH.

OpenVPN fulfills our requirements, but we would like some alternatives. Let us look in the direction of IPsec.

3.4 Plain IPsec

As we said before, IPsec is a set of IP extensions to bring security to the network layer. On the network protocol front, here is what it introduces:

- An Authentication Header (AH) for IP packets. This IP option is used to authenticate the host that sent an IP packet and to guarantee the data and IP header integrity.
- the Encapsulating Security Payload (ESP). This is a layer 4 protocol to carry encrypted data. The sending host is authenticated and the data integrity is guaranteed. ESP can be used in transport mode, where it encapsulates TCP or UDP, or in tunnel mode, where it encapsulates IP. For a VPN setup, the tunnel mode is used in order to build a virtual network. Because ESP in tunnel mode guarantee the integrity of the inner IP header, IPsec VPN do not need AH.
- The IP compression protocol (IPcomp). This is another layer 4 protocol used to reduce the overload of encryption.

In order to do their cryptographic job, ESP and AH need a shared key on the sending and receiving host. This key, and the various informations needed to use it (algorithms, key length...) are known as an IPsec Security Association (SA). It is stored inside the kernel, in the Security Association DataBase (SAD). The SAD can be set up manually (using the setkey(8) command for instance), or key exchange can be handled by a userland daemon. In that situation, the who hosts willing to establish an IPsec SA needs to have key exchange daemons that speak the same protocol. Today's protocol for IPsec key exchange is known as Internet Key Exchange (IKE) protocol, and it is defined in RFC 2409 [IKE].

IKE defines two phases. In phase 1, the peers authenticate each other and set up a phase 1 Security Association (also known as an ISAKMP SA). The phase 1 SA is a shared key stored in memory by the IKE daemons. It is used to periodically run a phase 2, which uses the phase 1 SA to produce IPsec keys (also known as IPsec SA or phase 2 SA). This two-layer system enables periodical re-keying, where the IPsec SA can be periodically renewed in order to make the attacker's job more difficult.

Static keys may seem easier to manage but if we use them, we lose the re-keying feature. It is much better to use IKE, and this is what we are going to do. Let us now focus on IKE phase 1 authentication.

IKE phase 1 authentication can be done by two ways: shared secret or digital certificate. Both peers have to use the same method. We said that we did not want user digital certificates, and a shared secret is not the same thing as a login/password pair. The lack of user credential management tools seems to always drive network administrators to use a group password as the

shared secret. The real problem is that IKE phase 1 was mostly designed to authenticate hosts and not users.

An additional problem is that the configuration on the client side is not easy, as proper routing entries shall be created to get the private traffic going through the ESP tunnel.

For those reasons, plain IPsec does not seem to meet our goals.

3.5 L2TP over IPsec

Layer 2 Tunneling Protocol (L2TP) provides the ability to perform a user authentication through a login/password, and it is able to carry several point to point links on the top of ESP in transport mode. The multiple links can be used to carry a multi-protocol VPN, doing both IP and AppleTalk, for instance. L2TP is defined by RFC 3931 [L2TP].

L2TP over IPsec has a login/password authentication, and we have built-in clients in Windows XP and MacOS X.3. It looks like a nice solution to our problem but a closer look will show it is not.

The problem is that the L2TP completely relies on IPsec for securing the user authentication. If the IPsec SA is insecure, then authentication credentials may be stolen by an attacker. And the IPsec SA is secure only if the IKE phase 1 SA is secure. And the IKE phase 1 SA is secured by an authentication which can only use a shared secret or certificates.

Therefore even if L2TP has a user authentication using a login/password, it can only be made secure using certificates for users. This is not what we want.

4 Some Ipsec extensions

The temporary conclusion to our remote user access VPN survey is that there is no IPsec solution that meet our requirements. Let us see what other vendors have done to improve IPsec so that it can get the job done.

4.1 Xauth

Xauth (which has nothing to do with X Window Xauth) is an IKE extension that introduces a user authentication step between IKE phase 1 and IKE phase 2. The user authentication can be done through a login and a password. This solution has exactly the same drawback as L2TP over IPsec: the user authentication is secured by IKE phase 1, and IKE phase 1 can only be secured by a shared secret or certificates.

It is worth noting that a lot of vendors recommend using Xauth (or L2TP over IPsec) with a group password as the IKE phase 1 shared key because this is easier to manage. Such a setup leads to a weak security: anyone that knows the shared secret can eavesdrop other users' traffic and steal their user credentials.

Xauth adds a user authentication to IKE, but it does it in a way where a user certificate is still required in order to have a decent level of security. That is not very helpful.

Xauth is documented in a dead IETF draft [Xauth], but it is used in various VPN solutions, such as Cisco VPN.

4.2 Hybrid authentication

Hybrid authentication is another IKE extension that makes the phase 1 asymmetric: the VPN gateway authenticate to the mobile host by using a certificate, and the mobile host does not authenticate in phase 1.

At the end of phase 1 we get a secure channel where the VPN gateway does not know who it is speaking with, and the mobile host knows it speaks with the VPN gateway. In this secured channel, a user authentication through Xauth can safely take place.

Hybrid authentication with Xauth gives us the security level of SSHv2 with passwords, and this is what we were looking for. We only have to manage a server certificate, which is easy to do, and users will authenticate using login and passwords: no user certificate, no group password.

Hybrid authentication is documented in an IETF draft [Hybrid].

4.3 ISAKMP mode config

ISAKMP mode config is another IKE extension used by a mobile host to pull the network configuration from the VPN gateway. It can also be used by the VPN gateway to push the network configuration. This extension makes the user's life much more easier, as it enable VPN auto-configuration.

Without ISAKMP mode config, the peers must agree some way on the private addresses to use for tunneling. This leaves no other choice than manual IP configuration for each VPN user, which is not very convenient.

ISAKMP mode config mechanism is also used by Xauth for requesting and submitting the user credentials.

ISAKMP mode config is documented in an IETF draft [mode-cfg].

4.4 Nat-Traversal

IPsec VPNs are based on ESP, and ESP has trouble going through firewalls and Network Address Translators (NAT). There are two problems: first most network administrators block any traffic and allow only things they know about. ESP is not widespread enough to be allowed everywhere, and in fact it is blocked nearly anywhere.

The other problem is that ESP has no ports like TCP and UDP, which makes it difficult to handle for a network address translator. Most NAT will just not let it get through.

The solution to this is NAT-Traversal, a set IPsec extensions used to encapsulate ESP in UDP datagrams.

There have been many NAT-Traversal drafts, but the final RFC works that way: IKE starts on standard UDP port 500. After the first exchange, port 4500 is used. After IKE phase 2 is done, the ESP packets are encapsulated in UDP

packets. The UDP ports used for IKE are used for ESP over UDP. This ensures that NAT state installed by the IKE exchange can reused by ESP over UDP.

NAT-Traversal also require that keep-alive packets gets transmitted regularly to avoid NAT state timeout.

NAT-Traversal is defined by RFC 3947 [NAT-T] and RFC 3948 [ESP-over-UDP]. Microsoft claimed to hold a US patent on it [MS-IPR], but it is not clear that NAT-Traversal is really encumbered.

4.5 Dead Peer Detection (DPD)

Remote users accessing through a VPN may experience dangling connections, and it is important to detect when a remote user gets disconnected. Unfortunately, IPsec has no good built-in mechanism to discover that the remote host crashed or has gone off-line. Such an event is only detected at re-keying time. If phase 2 cannot complete, the IPsec SA gets killed. But it is not convenient to use a very short phase 2 lifetime just to detect dead peers.

DPD is yet another IKE extension used to monitor the peer and quickly detect when it gets unreachable. It works by exchanging probe packets, and if the peer does not answer for some time, the security associations are killed.

DPD is documented by RFC 3706 [DPD].

4.6 IKE fragmentation

Many network appliances such as DSL routers or home firewalls consider that UDP is only for DNS and NTP. Big UDP packets are not expected and are not allowed to pass through. This is a problem for our VPN since IKE tend to produce big UDP packets.

Of course we could use IP fragmentation to send smaller packets, but unfortunately some broken appliance will also consider UDP fragment as an evil thing that must be blocked.

IKE fragmentation is one more IKE extension used to split a big IKE packets into smaller fragments, so that they can pass through any network appliance.

IKE fragmentation seems to be a proprietary extension from Cisco and it seems it is not documented anywhere.

4.7 ESP fragmentation

ESP over UDP suffers exactly the same problem as IKE with network appliances. This can be fixed without any IPsec extensions, by using a simple trick. For our VPN, we use ESP in tunnel mode. The packets look this way on the wire:

```
MAC header:IP:UDP:ESP:IP:TCP:HTTP
```

And fragmented packets look like this:

```
MAC header:frag(IP:UDP:ESP:IP:TCP:HTTP)
MAC header:frag(IP:UDP:ESP:IP:TCP:HTTP)
```

We know that some network appliance will block big UDP packets or fragmented UDP packets. The idea is to fragment packets at the IP level before ESP encapsulation takes place. The fragmented packets look like this on the wire:

```
MAC header:IP:UDP:ESP:frag(IP:TCP:HTTP)
MAC header:IP:UDP:ESP:frag(IP:TCP:HTTP)
```

Because the fragmentation takes place inside ESP payload, network devices in between the mobile host and the VPN gateway have no way to see that the packet was fragmented.

ESP fragmentation ensures that packets of any size can be sent through the VPN. There might still be some problems with TCP being unable to perform Path Maximum Transmission Unit (PMTU). This is addressed by using a well known fix known as Maximum Segment Size (MSS) clamping.

5 Implementing what we need

With all these new features, we get closer to having what we are looking for. Using hybrid auth, we get our login and password authentication, and it is as secure as SSHv2 using login and passwords. Using ISAKMP mode config and DPD, we get something easy to use for the end-user. NAT-Traversal, IKE fragmentation and ESP fragmentation make the VPN usable when users connect from behind a NAT.

But now we want some real software. On the client side, Cisco makes Cisco VPN client, which implement all the nice IPsec extension we talked about. It works on Windows and MacOS X, and Cisco gives it for free if you buy a Cisco VPN 3000, a network device that does the VPN server side.

The Cisco VPN 3000 configuration is something really heavyweight that relies on a web interface or text-base menus. We said we wanted a VPN gateway running free software for various reasons (which may also include not using the VPN 3000 bloated web interface anymore). So let us now have a look on how NetBSD was enhanced to replace a Cisco VPN 3000.

5.1 The KAME project's IPsec stack

NetBSD and FreeBSD IPsec stacks were obtained from the KAME project. KAME's goal was to provide an IPv6 stack for BSD systems. Because IPv6 contains IPsec, KAME also provided IPsec.

The IPsec stack is split into a few components:

- incoming and outgoing packet handing in the kernel
- key management in the kernel.
- an IKE daemon in userland called racoon
- the setkey command used to manipulate SAD and SPD manually

- the libipsec library, which is a layer between racoon and the kernel. The userland/kernel key management interface is done through special sockets of type PF_KEY. The interface itself is also called PF_KEY, and it is defined in RFC 2367 [PF_KEY].

Implementing ISAKMP mode config, Xauth, and hybrid authentication in racoon was not very difficult. All the protocols are documented by IETF drafts, so most of the work was done by trying to connect to racoon using the Cisco VPN client, and filling the gaps each time something was causing a failure.

IKE fragmentation has been more difficult because it was not documented. Some reverse engineering was required. Fortunately the protocol was rather simple: each IKE fragment had a small header containing the fragment length, the fragment index, and a flag for the last fragment.

Once the Cisco VPN client was able to establish an IPsec SA with a NetBSD machine, it was obvious that the project was going somewhere, and, therefore, the question of integrating these racoon changes raised. NetBSD racoon was only an import of KAME racoon, and hence the goal was to integrate the changes in KAME and import the newer racoon in NetBSD.

Unfortunately, the KAME project has not really been interested by this work. It was impossible to get it integrated, so the time had come to think about doing a fork. But managing a fork is never a pleasant plan, because it means a lot of work, merging future fixes from the original project with code added in the forked version.

Fortunately, a KAME fork already existed: ipsec-tools

5.2 IPsec-tools

Some time ago, it has been decided that KAME racoon was the way to go for Linux. Some contributions where done to KAME racoon for adding Linux support, but that turned into a racoon fork known as ipsec-tools.

ipsec-tools is only a fork of the userland part of the KAME IPsec stack: it contains setkey, libipsec and racoon. The project accepted the contributions to restore NetBSD build, and to add ISAKMP mode config, Xauth, Hybrid authentication, and IKE fragmentation.

ipsec-tools was later further improved to better fit in the remote user VPN scenario. Support was added to behave as a VPN client for hybrid authentication, and on the server side, things such as RADIUS and PAM authentication for Xauth logins were added. Another contributor added DPD.

For a documentation about how to configure a NetBSD system as a VPN gateway or a VPN client using hybrid authentication, see the how-to in the NetBSD documentation [HOW-TO].

5.3 NAT-Traversal in NetBSD

ipsec-tools has support for NAT-Traversal in tunnel mode, but that requires kernel support which was only available on Linux. Some work had to be done in the NetBSD kernel to get NAT-Traversal working on NetBSD.

The new code is ifdef'ed by the `IPSEC_NAT_T` option in the NetBSD kernel. It sits in four locations:

- in the setsockopt(2), where the IKE daemon tells the kernel that a given socket can be used for ESP over UDP.
- in the PF_KEY interface: the IKE daemon gives the UDP port number that is being used by IKE, and that will be used for ESP over UDP.
- in the UDP input function: for sockets tagged as using ESP over UDP, the UDP payload is transmitted to the ESP input function or to the IKE daemon in userland. A non-IKE marker is used at the beginning of the UDP payload to distinguish ESP and IKE packets
- in the ESP output function, data that is to be handled by an IPsec SA using ESP over UDP is sent to the UDP output function.

The original NAT-Traversal support written for Linux was not able to cope with the situation where IPsec SA were installed with multiples peers behind a NAT. This was because the code only used the IP addresses in SPD and SAD. Because the port information was missing, there was no way of distinguishing the traffic coming from different machines. NAT-Traversal support in racoon and in the NetBSD kernel was improved to keep track of port information in order to fix that.

5.4 Switching from KAME to ipsec-tools

At that time it was clear that ipsec-tools had much more activity than KAME racoon. Many new features were being integrated into ipsec-tools, while KAME was too busy on other problems to integrate anything. This led to the decision to integrate ipsec-tools into NetBSD instead of KAME racoon. The switch was done in April of 2005. It caused a few minor regressions such as TCP-MD5 lossage and transport mode without NAT-Traversal being broken, but those problems have been fixed.

At the end of April 2005, the KAME project dropped support for racoon and advised users to use ipsec-tools racoon instead [KAME]. This decision was made because the primary goal of KAME is to develop IPv6 and not an IKE daemon, and because there is no point having two projects for racoon.

Conclusions

The need for a particular set of requirement has driven the integration of many new features in NetBSD IPsec stack. The solution described here is an average security solution, where some security is traded for usability. But stronger security setups, where digital certificates are used for user authentication, also benefited from the new features that were added, such as NAT-Traversal, DPD, ISAKMP mode config, or IKE fragmentation.

Racoon was also improved on some fronts unrelated to network communications. For instance, it is now able to run as an unprivileged user and within a chroot environment.

In the future, racoon should keep evolving towards being compatible with more IPsec VPN implementations. There are also some missing features that are frequently required: NAT-Traversal in transport mode and IKEv2 integration. As users frustration increase, the odd gets better that someone will implement these missing features as well

NAT-Traversal in transport mode will require support in the NetBSD kernel. There is also another point on the road map for NetBSD IPsec: `FAST_IPSEC`. This alternative in-kernel IPsec stack is designed to make cryptographic operation asynchronous so that hardware accelerators can be used. It has not been modified for NAT-Traversal, and this is another gap that has to be filled.

Thanks to John R. Shannon, Greg Troxel, Greg Oster, and D'Arcy J.M. Cain for reviewing this paper. Thanks also to Florence Henry for the help with LaTeX.

References

[PPTP] `http://www.schneier.com/paper-pptpv2.pdf`

[OpenVPN] `http://openvpn.net`

[IKE] `http://www.ietf.org/rfc/rfc2409.txt`

[L2TP] `http://www.ietf.org/rfc/rfc3931.txt`

[Xauth] `http://cvs.sourceforge.net/viewcvs.py/*checkout*/ipsec-tools/ipsec-tools/src/racoon/rfc/draft-beaulieu-ike-xauth-02.txt`

[Hybrid] `http://cvs.sourceforge.net/viewcvs.py/*checkout*/ipsec-tools/ipsec-tools/src/racoon/rfc/draft-ietf-ipsec-isakmp-hybrid-auth-05.txt`

[mode-cfg] `http://cvs.sourceforge.net/viewcvs.py/*checkout*/ipsec-tools/ipsec-tools/src/racoon/rfc/draft-dukes-ike-mode-cfg-02.txt`

[NAT-T] `http://www.ietf.org/rfc/rfc3947.txt`

[ESP-over-UDP] `http://www.ietf.org/rfc/rfc3948.txt`

[MS-IPR] `https://datatracker.ietf.org/public/ipr_detail_show.cgi?&ipr_id=78`

[DPD] `http://www.ietf.org/rfc/rfc3706.txt`

[PF_KEY] `http://www.ietf.org/rfc/rfc2367.txt`

[HOW-TO] `http://www.netbsd.org/Documentation/network/ipsec/rasvpn.html`

[KAME] `http://www.atm.tut.fi/list-archive/snap-users-2005/msg00105.html`