

Version 1.0

Ken Bantoft (ken@xelerance.com)

© 2004 Ken Bantoft, Xelerance Corporation

Licensed under the Gnu Public License v2

Abstract: The world of IPsec on Linux is a confusing one at the moment. With Kernel 2.6 came a new IPsec kernel stack, combined with 2 userland toolkits (one of which has 3 forks) has lead to mass confusion at the end user level.

All of this leaves users without a clue on where to start, which packages to use, and what features they need to get the job done.

This paper compares and contrasts the 2 main userland toolkits for the 2.6 stack, as well as differences in the new kernel stack. It will also go into detail into some of the more recent features added into the userland tools, including NAT-Traversal, and talk about current major developments in the IPsec on Linux world.

The Future of IPsec on Linux

History of IPsec on Linux

In 1997, John Gilmore (Sun employee #5, and one of the founders of Cygnus) started the FreeS/WAN (Free, Secure Wide Area Network) to create an IPsec stack for Linux. John's political agenda hindered the project throughout it's entire lifetime, with issues like 'no US code would be accepted' and posts from US citizens with patches were ignored in order to keep the codebase 'US free', due to US Cryptographic export regulations at the time.

FreeS/WAN provided an stable, secure IPsec stack on Linux for the 2.0, 2.2, 2.4 and now 2.6 series kernels.

In September of 2000, Ken Bantoft forked FreeS/WAN 1.99 into the 'Super FreeS/WAN' tree, and merged all outstanding major patches (including ones from US citizens). This package quickly became the most popular IPsec on Linux package for Kernels 2.2 and 2.4, and made it's way into many other projects (WOLK, LEAF/bering, etc...) and several commercial distributions (Astaro, ImageStream, NIT to name a few).

As mid 2.5 kernels came out, it emerged that another IPsec stack was being written, led by David Miller from RedHat. David, along with a few others had always been in conflict with the FreeS/WAN project. Their view was that there is no way code that can't be maintained by US citizens would ever go into the

Linux Kernel, so they needed something different.

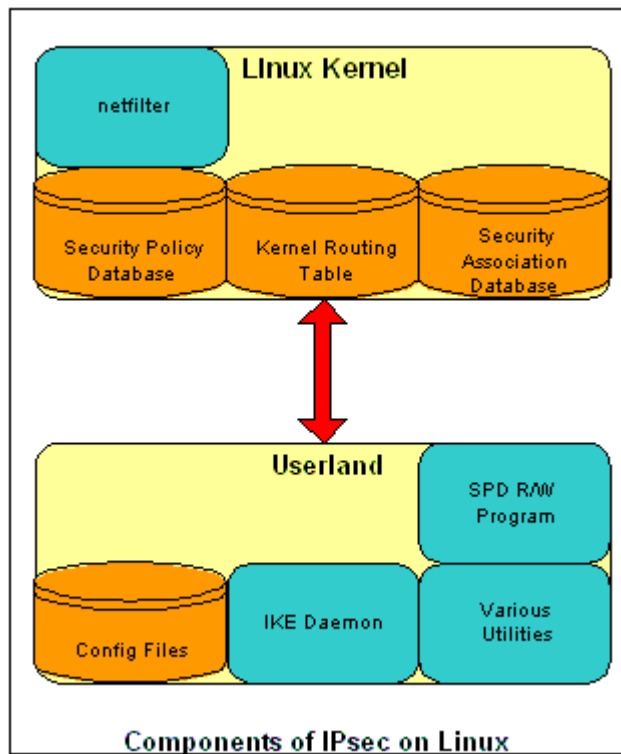
At the time, US citizens could not export cryptographic code. It was a very serious issue at the time, however most were unaware of it. As such, many said that David Miller could not write the code, nor could he maintain it.

The turning point was the Bernstein case (funded in part by John Gilmore and several others via both the FSF, and the EFF). The Bernstein case was not a win - the US government settled out of court. The result is the current BXA "notify us" system, which many US developers think is a win. Is it? Gilmore thinks it is not, that that they are seriously imperiling themselves. Time will tell, as more and more Open Source software contains cryptographic code, and is being exported daily.

David Miller, Derek Atkins and a few others chose to port the various pieces of the USAGI IPsec stack (as they have IPsec for IPv6 working), *BSD's Kame + Racoon codebase to Linux. The kernel portion was merged late in the 2.5 Kernels, and is now included in 2.6. The userland tools, now known as ipsec-tools contain setkey and racoon - the SPD manipulation tool, and the IKE daemon.

In mid-April 2004, the FreeS/WAN project was declared over, with 2.06 being the final release. At least 2 new projects have been started based on recent FreeS/WAN code - Openswan and Strongswan. Both share the same heritage, and have nearly identical features - this collection of IPsec stacks that use Pluto as the IKE daemon are collectively referred to as *swan.

One interesting thing that FreeS/WAN 2.06 contains is an implementation of KLIPS for the 2.6 Kernel, so perhaps KLIPS will become more of an option for 2.6 users.



IPsec is a complicated protocol, and requires many interactions between different parts of the Kernel, as well as userland tools to make it work securely.

The userland portion deals with managing configuration files, keys/passwords, and an IKE (Internet Key Exchange) daemon to communicate between peers. The IKE daemon talks to the SPD via some mechanism, currently either by calling the setkey tool directly, or via a kernel socket (*swan).

The SPD contains all of the security policies, which dictate what traffic is to be encrypted/decrypted, by which keys, and where to send it. The Security Association Database (SAD) contains all of the currently active SA's. On Kernel 2.4, the IKE daemon manages the Kernel Routing Table by installing routes as needed via the 'route' or 'ip' commands. In 2.6, this is no longer required as the SPD is directly in line on the path packets take through the networking stack.

Contrasts in Kernel Land

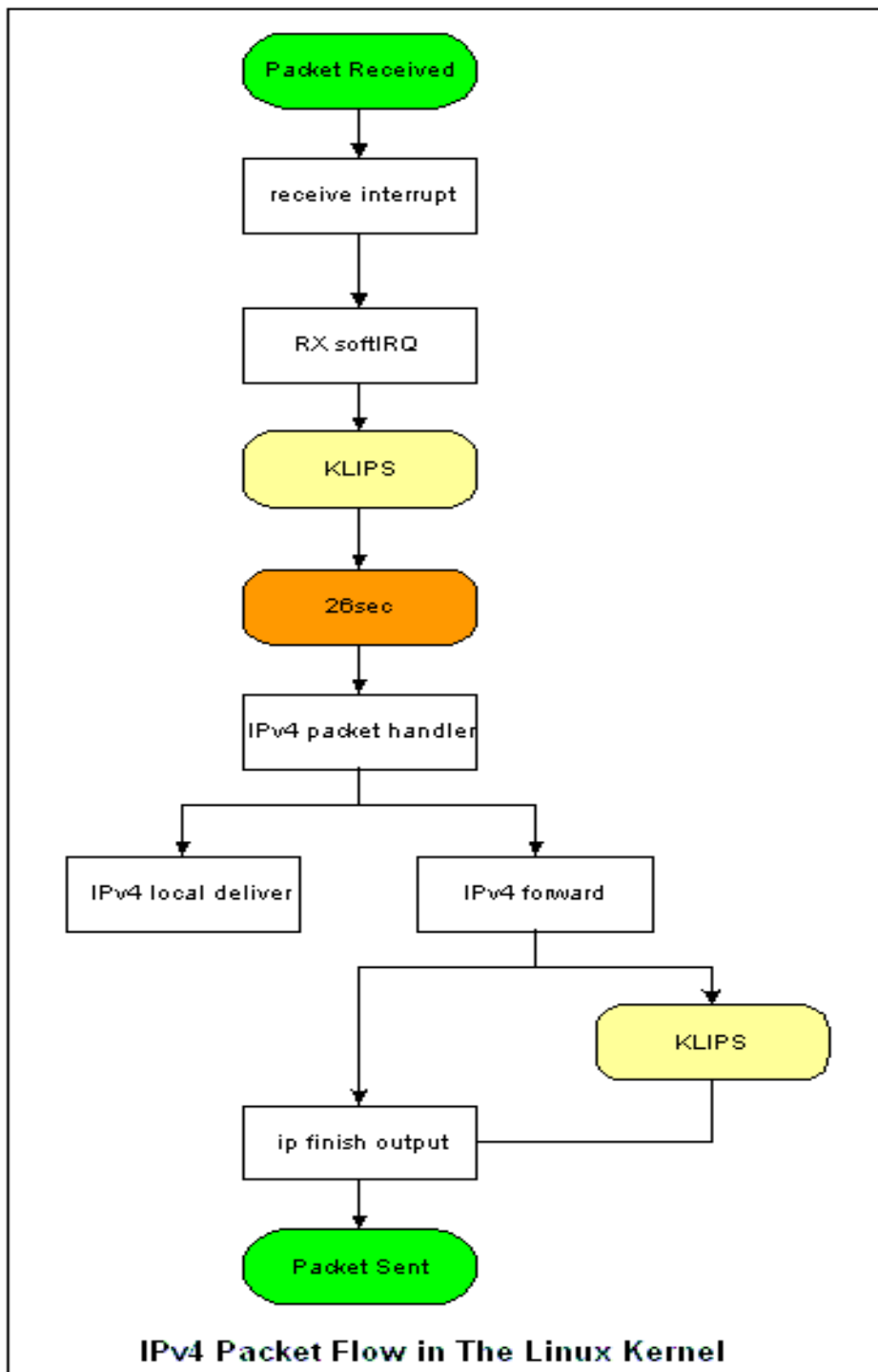
*swan users are used to having a virtual network device provided by KLIPS. All IPsec traffic is passed through these 'ipsec#' interfaces. This was quite useful for many purposes, a few of which are listed here:

1. Firewalling is simple, because if it comes in ipsec0, it's already been decrypted and you can assume it's a safe packet.
2. Debugging simple, as if you tcpdump -i ipsec0, and see the packets before encryption, and after decryption.

3. Routing - each IPsec tunnel appeared as a route in the Kernel routing table.
4. Fun network tricks – you could layer GRE tunnels inside IPsec tunnels, or manually route traffic to the ipsec# devices. You could also do NAT, policy routing, bind services only to ipsec# and many other tricks using the virtual device.

The 2.6 implementation does not have a virtual network device, meaning some additional work is needed to do firewalling and debugging, and many of the tricks and hacks no longer work.

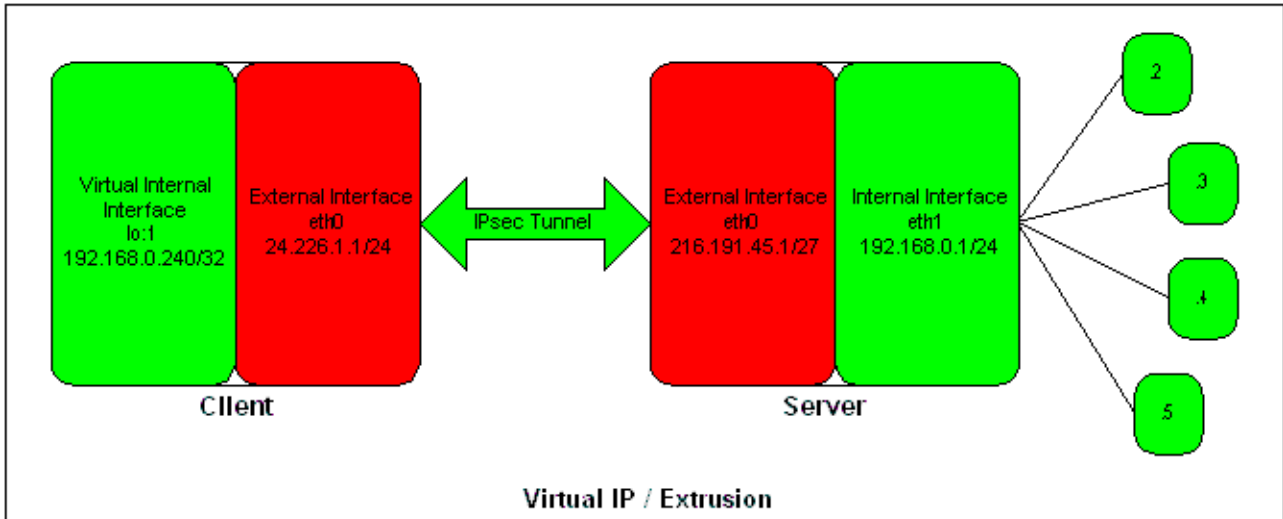
Another area of major difference is the location of the SPD (Security Policy Database). In KLIPS, the SPD lookups are after the routing process, which is why many networking tricks (like pointing network routes to ipsec0) work. This is commonly known as the 'Bump in the stack' technique, as you insert your code into the networking stack and interrupt **all** traffic to see if it is IPsec related, and the process as required.



In 26sec, all SPD interaction is before the routing process. This is the Kame/*BSD model, which basically a direct implementation of RFC2401. It means the encrypt or decrypt happens before the packets hit the routing process, so netfilter firewalling is inherently more complicated, and debugging can be much harder. KLIPS on the other hand really hooks into two separate places, doing the decrypt before the routing process, and the encrypt after.

The 26sec stack so far has created a number of current limitations:

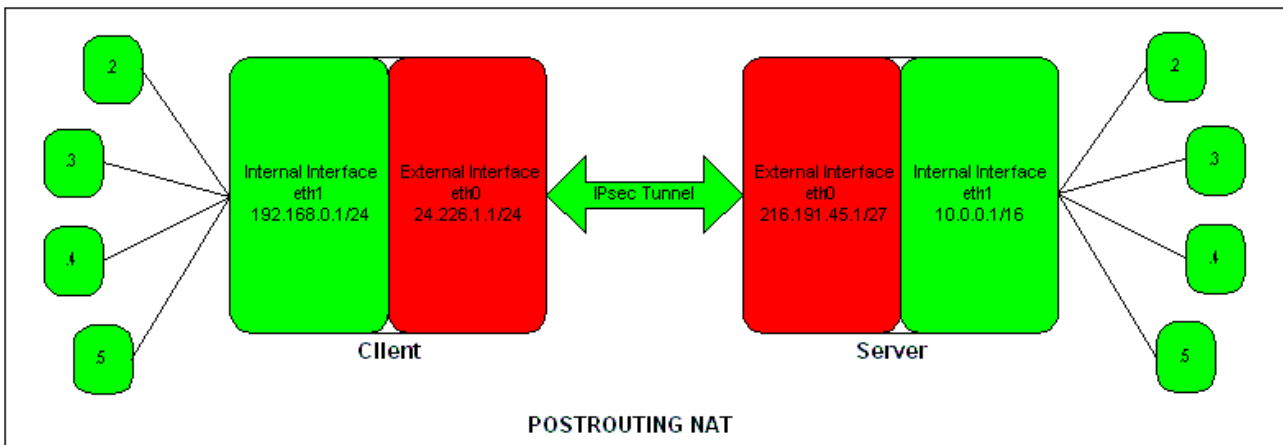
1. You cannot 'extrude' an IP or subnet:



The 26sec SPD does not currently allow you to 'exempt' a portion of an SPD entry from the tunnel. Configuring a setup like this, even if you assign the 192.168.0.240/32 extrusion to the lo:1 interface, won't work. While the SA is established, you won't even be able to ping locally.

This means you cannot assign local LAN IP addresses to roadwarriors, unless you use a separate subnet, and don't include a tunnel for it (this also means you won't be able to have RW to RW communication)

2. Network Address Translation via netfilter doesn't work as in 2.4



Using *swan on Kernel 2.[024] you could setup a tunnel for only the external IP (24.226.1.1) to be permitted over IPsec. You could then NAT the 192.168.0.0/24 subnet to the 24.226.1.1 IP address, and tunnel the entire LAN via NAT/Masquerading.

With the 26sec code, because the SPD lookups are before the kernel

routing table, the packets are encapsulated before they hit any of the netfilter PRE/POSTROUTING hooks. This means SNAT/DNAT won't work, so this configuration will no longer work. There are fixes in development for this - see the New Developments – netfilter+ipsec section.

3. Debugging is made more difficult.

With the KLIPS stack's ipsec# device, administrators could run “tcpdump -i ipsec#”, and see the outbound packets before encryption, and inbound packets after encryption. This made debugging quite easy, as there was a clear view into the IPsec tunnel. In 26sec, this is no longer possible as there is no virtual device. Instead, tcpdump is only able to show ESP packets coming and going - not the content of those packets.

Setkey allows some basic debugging, but is more useful for looking at the SPD & SAD - not the contents of the packets.

Michael Richardson has done some work to help fix this, by extending tcpdump to include support for decrypting both 3DES and now AES. However, both depend on the administrator knowing the encrypt/decrypt keys - something that's only possible when running in a manually keyed mode - which rarely done anymore.

Contrasts in Userland

The largest visible differences in the two implementations are in the area of userland tools.

The userland tools packages need to have an IKE daemon, which is responsible for negotiating IPsec tunnels, updating the SPD, and key management.

	*swan	ipsec-tools
Keying daemon:	Pluto	Racoon
SPD Control:	setkey ^[1]	setkey

When comparing the two different approaches on the 2.6 Kernel, it is important to note that the userland tools are the only area of difference - the kernel code is the same^[2]. This is also where the largest difference in features occurs:

Feature/Spec	pluto	racoona
IPsec(rfc2409):	yes	yes
Support for 2.4:	yes	some ^[3]
Support for 2.6:	yes ^[4]	yes
Pre Shared Keys:	yes	yes
RSA Keys:	yes	?
X.509 Certs:	yes	yes
1DES:	no	yes
3DES:	yes	yes
AES:	yes	yes
NAT-Traversal:	yes	yes
Aggressive Mode:	no	yes
DPD (RFC3706):	yes	no
XAUTH:	some ^[5]	no
OSCP:	some ^[6]	no
CRL 'Fetching':	yes	no
IPv6:	some ^[7]	yes
Opportunistic Encryption:	yes	no

^[1] To be merged into Openswan 2.3.0

^[2] Ignoring for the moment FreeS/WAN 2.06's KLIPS2.6

^[3] need 26sec backport as well

^[4] need setkey

^[5] In Openswan, but not Strongswan

^[6] In Strongswan, but not Openswan(yet)

^[7] Pluto configuration subsystem is not yet fully IPv6 aware

New Developments

netfilter+ipsec modules

Patrick McHardy is currently developing some new netfilter code for the 2.6 Kernel which deals with firewalling IPsec packets much differently. This will allow administrators to 'hook' into the IPsec process much better, and allow better control of IPsec traffic. It will also fix the NAT issues identified earlier, by changing the path of IPsec traffic within the network stack.

From one of Patrick's posts, explaining how netfilter+ipsec patches feed the packets back into the PRE/POST Routing chains:

input looks like this:

```
(encrypted) PRE_ROUTING -> LOCAL_IN ->  
(plain) PRE_ROUTING -> LOCAL_IN/FORWARD
```

output looks like this:

```
(plain) FORWARD/LOCAL_OUT -> POST_ROUTING ->  
(encrypted) LOCAL_OUT -> POST_ROUTING
```

This is the same as with FreeS/WAN, only without the ipsec devices, the policy match can be used as a easy replacement for them (-m policy --pol ipsec).

Patrick's code basically catches IPsec packets and sends them back through the netfilter process after they've been encrypted or decrypted. During this 2nd pass, they can be mangled/NAT'd/etc... like normal packets.

KLIPS for Linux Kernel 2.6

KLIPS for Kernel 2.6 was released in the final version of FreeS/WAN, 2.06. It remains to be seen if this work will be integrated into either of the *swan forks, or the Kernel itself. As the project is officially over, there are no longer restrictions preventing US citizens from working on the code.

NAT-Traversal

NAT-Traversal is the name given to a set of IETF drafts (on the way to becoming an RFC) that permits IPsec peers to tunnel traffic through devices that do NAT. Essentially, it permits both peers of an IPsec tunnel to detect if there are one (or more) NAT devices in between them, and then switch to

tunneling ESP packets into UDP packets. One major benefit of this is that it allows IPsec traffic to pass through a NAT device without any additional configuration of the NAT device.

NAT-T has been tested on many different networks, including GPRS providers, various WiFi hotspots as well as broken ISPs.

Mathieu Lafon from Arkoon Network Security wrote the original NAT-Traversal implementation for FreeS/WAN, which included a patch for the 2.2 and 2.4 kernels to deal with the ESPinUDP packet handling. This portion of NAT-T has to be handled in the kernel IP stack, somewhat like this:

```
udp.c:
if(is_esp_header_inside(udp_packet)) {
    esp_packet = strip_udp_header(udp_packet);
    process_esp_packet(esp_packet);
}
```

The patch for 2.4 was never accepted, but in 2.6 a similar but incompatible version was put into the kernel.

Currently, only Openswan (thanks to Herbert Xu's patches) and Racoon (v0.3, released 2004-04-14) currently support NAT-Traversal on the 2.6 Kernel.

X.509 & PKI

PKI (Public Key Infrastructure) is starting to regain momentum, as more companies spend money on security. As such, technologies like X.509 Digital Certificates and SmartCards are becoming more commonplace. While X.509 support for IPsec has been around for a few years, recently support for SmartCards has been added to *swan, and Strongswan now even supports OCSP (Online Certificate Status Protocol – RFC 2560). Some of this technology is not yet standardized, but work is being done by the PKI for IPsec working group, and RFC's are expected this year.

Conclusion

IPsec on Linux is here to stay, and users now have 2+ choices for how to secure their networks. The only major question remaining is, will this speed up the adoption of IPsec?

With both mainstream Wireless encryption methods (WEP, LEAP) having been proven insecure, and tools released to help script kiddies speed up their attacks, the best solution still remains IPsec over Wireless. Yet nearly all vendors of Wireless AP's and Gateways still don't ship IPsec enabled devices. Those that do, do it only on the external interface, so you can setup a tunnel to another site, but not between your laptop and the AP.

References

1. Bernstein vs. United States, <http://cr.yip.to/export.html>
2. ipsec-tools, <http://ipsec-tools.sourceforge.net>
3. WOLK, Working Overloaded Linux Kernel, <http://wolk.sourceforge.net>
4. LEAF, Linux Embedded Appliance Firewall, <http://leaf.sourceforge.net>
5. USAGI Project, IPv6 for Linux, <http://www.linux-ipv6.org>
6. Netfilter, <http://www.netfilter.org>
7. Netfilter-ipsec posts/discussions from Patrick Hardy
<http://lists.netfilter.org/pipermail/netfilter-devel/2004-March/014470.html>
8. The Linux FreeS/WAN Project, <http://www.freeswan.org>
9. Openswan, <http://www.openswan.org>
10. Strongswan, <http://www.strongswan.org>
11. VPN Consortium, <http://www.vpnc.org>
12. The RFC Editor Site, <http://www.rfc-editor.org>
13. TCPDUMP, <http://www.tcpdump.org>