TaoSecurity

Richard Bejtlich's blog on digital security, strategic thought, and military history.

Monday, January 03, 2005

IPSec Tunnels with FreeBSD

Although the FreeBSD Handbook offers a VPN over IPSec section, it doesn't describe the scenario I face when deploying network security monitoring sensors. That document also references commands that no longer exist in FreeBSD 5.3, like 'gifconfig.' My architecture looks like this (all IP addresses are obfuscated):

```
   remote sensor
     'fedorov'
   -------------
   interface em0
public management IP
   18.235.153.37
         |
      Internet
         |
  VPN concentrator
  and NAT gateway
    'forsberg'
   -------------
   interface em0
external public IP
   78.172.25.27
        ---
   interface em1
internal private IP
   192.168.1.1
         |
       switch
         |
monitoring backend
------------------
   interface em0
internal private IP
   192.168.1.10
```

Blogging Since 8 Jan 2003

About Me

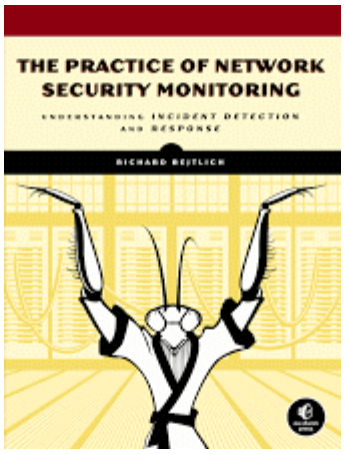
RICHARD BEJTLICH

View my complete profile

twitter.com/taosecurity

Follow @taosecurity

Practice of NSM On Sale Now!


THE PRACTICE OF NETWORK SECURITY MONITORING
UNDERSTANDING INCIDENT DETECTION AND RESPONSE
RICHARD BEJTLICH

Save 30% With Code NSM101

I need to encrypt communications from the sensor to the monitoring backend. This can involve multiple individual sockets. I don't like to use OpenSSH port forwarding or Stunnel because I must set up a separate port forwarding or tunnel session for each channel. I would much rather use IPSec, since that can carry any communications between the sensor and the backend.

Complicating matters, I need to communicate between a sensor with a public management IP and a backend with an internal private IP address. That backend internal private IP address is transformed using NAT on the VPN concentrator and NAT gateway. All boxes in this scenario run FreeBSD 5.3 RELEASE.

One answer to this problem, and the approach I use, is to create a virtual tunnel from the sensor to the gateway, through which traffic to and from the backend can pass. I will use the gif facility in FreeBSD. This will create an IP-in-IP tunnel, which I will then wrap inside IPSec ESP.

Here is the architecture, with gif tunnels added:

```
   remote sensor
     'fedorov'
   -------------
   interface gif0
 private tunnel IP
    10.4.12.10
        ---
   interface em0
public management IP
   18.235.153.37
         |
      Internet
         |
 VPN concentrator
 and NAT gateway
    'forsberg'
   -------------
   interface em0
external public IP
   78.172.25.27
        ---
   interface gif0
 private tunnel IP
    10.4.12.1
        ---
   interface em1
internal private IP
```

Featured Post

My Federal Government Security Crash Program

In the wake of recent intrusions into government systems, multiple parties have been asking for my recommended courses of action. In 200...

Bejtlich Media Appearances

Select Videos at YouTube

Books by Richard Bejtlich

Detect and respond to intrusions using tools and techniques that work. Plenty of free excerpts online.

Blog Archive

► 2016 (6)

► 2015 (55)

► 2014 (23)

► 2013 (18)

```
   192.168.1.1
        |
       switch
        |
monitoring backend
------------------
internal private IP
   192.168.1.10
```

The monitoring backend will communicate with 10.4.12.10 when it needs to talk to the sensor. The sensor will communicate with 192.168.1.10 when it needs to talk to the backend. The gateway will take care of connecting the two endpoints.

The first step is to recompile the kernels of the sensor and gateway to suit their roles. Here is what I add to the sensor's kernel config file before recompiling the kernel:

```
options   FAST_IPSEC
device    crypto
```

Here is what I add to the gateway's kernel config file before recompiling the kernel. The last two lines are completely optional, but the IPFIREWALL_DEFAULT_TO_ACCEPT means I don't need to add rules to permit later traffic:

```
options   FAST_IPSEC
device    crypto
options   IPFIREWALL
options   IPDIVERT
options   IPFIREWALL_DEFAULT_TO_ACCEPT
options   IPFIREWALL_VERBOSE
```

Next I modify the /etc/rc.conf on each system to add support for IPSec and the gif tunnel. Here is the sensor's additions to /etc/rc.conf:

```
gif_interfaces="gif0"
gifconfig_gif0="18.235.153.37 78.172.25.27"
ifconfig_gif0="10.4.12.10 10.4.12.1 netmask 255.255.255.2
static_routes="gif0_0"
route_gif0_0="192.168.1.0/24 -interface gif0"
#ipsec_enable="YES"
```

The gifconfig statement defines the public IPs used as the

- ▶ 2012 (60)
- ▶ 2011 (108)
- ▶ 2010 (193)
- ▶ 2009 (213)
- ▶ 2008 (253)
- ▶ 2007 (358)
- ▶ 2006 (493)
- ▼ 2005 (501)
  - ▶ December (45)
  - ▶ November (33)
  - ▶ October (32)
  - ▶ September (36)
  - ▶ August (61)
  - ▶ July (54)
  - ▶ June (42)
  - ▶ May (36)
  - ▶ April (45)
  - ▶ March (41)
  - ▶ February (45)
  - ▼ January (31)
    - Review of Forensic Discovery Posted
    - Creating favicon.ico Files
    - In-Depth Book Review at LinuxSecurity.com
    - Flowgrep: Flow-oriented Content Matching
    - SANS Confuses Threats with

tunnel endpoints. The ifconfig_gif0 statement sets up the tunnel, with 10.4.12.10 as the local endpoint and 10.4.12.1 as the remote endpoint. The static_routes and route_gif0_0 statements tell the sensor how to reach the backend network.

Here is the gateway's additions to /etc/rc.conf:

```
gateway_enable="YES"
firewall_enable="YES"
firewall_type="open"
natd_enable="YES"
natd_interface="em0"
#natd_flags="-redirect_port tcp 192.168.1.10:8080 8080"
gif_interfaces="gif0"
gifconfig_gif0="78.172.25.27 18.235.153.37"
ifconfig_gif0="10.4.12.1 10.4.12.10 netmask 255.255.255.2
#ipsec_enable="YES"
```

First I tell the gateway to act as a gateway, and I enable the firewall. I also enable NAT, with em0 being the Internet-facing interface with the external public IP address. I have commented out a natd_flags line showing how to do port forwarding. For example, a connection to port 8080 TCP on the gateway's external IP would be sent to the internal system 192.168.1.10.

Next I set up the gif interfaces for this end of the tunnel. They are mirror images of the entries for the sensor.

Note that in both cases I have commented out ipsec_enable="YES" for the moment. It is important to get a working IPSec configuration before one enables ipsec_enable="YES" in /etc/rc.conf. If you reboot a system with ipsec_enable="YES" uncommented, and your /etc/ipsec.conf configuration file is faulty, the system will not completely boot up. You will end up needing physical access to the system or remote serial access to fix the problem.

We have done enough at this point to try sending traffic without using IPSec, but with the gif tunnel. To create the gif interface manually on the sensor, use syntax like this:

```
ifconfig gif0 create
ifconfig gif0 tunnel 18.235.153.37 78.172.25.27
ifconfig gif0 10.4.12.10 10.4.12.1 netmask 255.255.255.25
```

To manually create the static route for the 192.168.1.0/24 network on the sensor, use this syntax.

```
route add 192.168.1.0/24 -interface gif0
```

Checking ifconfig, you will see a result like this:

```
fedorov:/root# ifconfig gif0
gif0: flags=8051 mtu 1280
  tunnel inet 18.235.153.37 --> 78.172.25.27
  inet6 fe80::2c0:9fff:fe40:1818%gif0 prefixlen 64 scopei
  inet 10.4.12.10 --> 10.4.12.1 netmask 0xffffffff
```

To create the gif interface manually on the gateway, use syntax like this:

```
ifconfig gif0 create
ifconfig gif0 tunnel 78.172.25.27 18.235.153.37
ifconfig gif0 10.4.12.1 10.4.12.10 netmask 255.255.255.25
```

You will see a result like this:

```
forsberg:/root# ifconfig gif0
gif0: flags=8051 mtu 1280
  tunnel inet 78.172.25.27 --> 18.235.153.37
  inet6 fe80::2c0:9fff:fe3f:4fc4%gif0 prefixlen 64 scopei
  inet 10.4.12.1 --> 10.4.12.10 netmask 0xffffffff
```

With these interfaces live, you can use them. Here we ping the gateway from the sensor:

```
fedorov:/root# ping 10.4.12.1
PING 10.4.12.1 (10.4.12.1): 56 data bytes
64 bytes from 10.4.12.1: icmp_seq=0 ttl=64 time=8.395 ms
```

Here is what the traffic looks like:

```
16:01:20.112213 IP 18.235.153.37 > 78.172.25.27:
 IP 10.4.12.10 > 10.4.12.1: icmp 64: echo request seq 0 (
16:01:20.120540 IP 78.172.25.27 > 18.235.153.37:
 IP 10.4.12.1 > 10.4.12.10: icmp 64: echo reply seq 0 (ip
```

As it passes on the Internet, this traffic is not TCP (IP protocol 6). It is IP-in-IP, or IP protocol 4. Tcpdump reports it correctly as 'ipip-proto-4). Your /etc/protocols file shows it to be this:

```
ipencap 4   IP-ENCAP  # IP encapsulated in IP (officially
```

If you add the routes necessary for the sensor to know how to reach the 192.168.1.0/24 network, you can ping the backend:

```
fedorov:/root# ping 192.168.1.10
PING 192.168.1.10 (192.168.1.10): 56 data bytes
64 bytes from 192.168.1.10: icmp_seq=0 ttl=63 time=8.800
```

Again, the traffic:

```
16:04:59.618465 IP 18.235.153.37 > 78.172.25.27:
 IP 10.4.12.10 > 192.168.1.10: icmp 64: echo request seq
16:04:59.627196 IP 78.172.25.27 > 18.235.153.37:
 IP 192.168.1.10 > 10.4.12.10: icmp 64: echo reply seq 0
```

Now that the traffic is being passed appropriately, we need to apply IPSec ESP to it. We create the following /etc/ipsec.conf file on the sensor. All of the spdadd statements should occupy a single unbroken line:

```
flush;
spdflush;

spdadd 10.4.12.10 10.4.12.1 any -P out ipsec
 esp/tunnel/18.235.153.37-78.172.25.27/require;

spdadd 10.4.12.1 10.4.12.10 any -P in ipsec
 esp/tunnel/78.172.25.27-18.235.153.37/require;

spdadd 10.4.12.10 192.168.1.0/24 any -P out ipsec
 esp/tunnel/18.235.153.37-78.172.25.27/require;

spdadd 192.168.1.0/24 10.4.12.10 any -P in ipsec
 esp/tunnel/78.172.25.27-18.235.153.37/require;
```

The first two lines flush IPSec Security Association Database (SAD) entries and Security Policy Database (SPD) entries. The first spdadd statement says traffic sent out from 10.4.12.10 to 10.4.12.1 should go via the IPSec tunnel from 18.235.153.37 to 78.172.25.27.

The second spdadd statement says traffic sent in from 10.4.12.1 to 10.4.12.10 should go via the IPSec tunnel from 78.172.25.27 to 18.235.153.37. These two entries are

► 2003 (375)

Labels

afcert (31)
Air Force (24)
aix (1)
analysis (1)
announcement (2)
appsec (4)
apt (57)
attacks (2)
attribution (16)
bestbook (7)
birthday (11)
blackhat (33)
books (9)
boyd (1)
breakers (4)
bro (7)
bruins (1)
business (24)
campaigns (1)
ccna (6)
cdm (4)
certification (26)
china (91)
cisco (30)
cissp (7)
cloud (4)
clowns (12)
commodore (6)
conferences (34)
containment (1)

enough to protect traffic sent between the sensor and gateway.

The third spdadd statement says traffic sent out from 10.4.12.10 to the 192.168.1.0/24 network should go via the IPSec tunnel from 18.235.153.37 to 78.172.25.27.

The fourth spdadd statement says traffic sent in from the 192.168.1.0/24 network to 10.4.12.10 should go via the IPSec tunnel from 78.172.25.27 to 18.235.153.37. These two entries protect traffic sent between the sensor and the backend.

The /etc/ipsec.conf file on the gateway is a mirror image of the sensor's /etc/ipsec.conf:

```
flush;
spdflush;

spdadd 10.4.12.10 10.4.12.1 any -P in ipsec
 esp/tunnel/18.235.153.37-78.172.25.27/require;

spdadd 10.4.12.1 10.4.12.10 any -P out ipsec
 esp/tunnel/78.172.25.27-18.235.153.37/require;

spdadd 10.4.12.10 192.168.1.0/24 any -P in ipsec
 esp/tunnel/18.235.153.37-78.172.25.27/require;

spdadd 192.168.1.0/24 10.4.12.10 any -P out ipsec
 esp/tunnel/78.172.25.27-18.235.153.37/require;
```

Now that the /etc/ipsec.conf files are ready, the last step is to install a program to manage key negotiations. We'll use Racoon, which can be installed via the security/racoon port. First, I made this change to the /usr/local/etc/racoon /racoon.conf file on the sensor to tell Racoon where to listen for key exchange packets:

```
listen
{
        isakmp 18.235.153.37 [500];
}
```

On the gateway, the modification looks like this:

```
listen
```

controls (6)
correlation (5)
counterintelligence (18)
crime (13)
criticalstack (1)
cybercommand (3)
cyberspace (1)
cyberwar (22)
dan geer (1)
databases (1)
ddos (1)
detection (16)
dfm (1)
disasters (11)
dna (9)
dns (1)
doctrine (1)
dod (8)
donations (1)
dprk (1)
education (8)
engineering (12)
exploits (15)
favorites (260)
feds (18)
finance (2)
fisma (21)
football (10)
forensics (6)
fraud (3)
freebsd (434)
GE (11)
ge-cirt (10)

```
{
        isakmp 78.172.25.27 [500];
}
```

Now, both public IP endpoints are listening on port 500 UDP for key exchange traffic.

Next I enabled a secret key. On the sensor, /usr/local /etc/racoon/psk.txt looks like this, which says use the specified key with the gateway 78.172.25.27:

```
78.172.25.27  thisisabadsecret
```

On the gatewat, /usr/local/etc/racoon/psk.txt looks like this, which says use the specified key with the sensor 18.235.153.37:

```
18.235.153.37  thisisabadsecret
```

Make sure the permissions for the /usr/local/etc/racoon /psk.txt file are 600, or the Racoon daemon will complain.

Now we are ready to start up Racoon, and enable IPSec. I recommend starting Racoon on the sensor and gateway in separate windows, using the 'racoon -F' syntax to show Racoon running in the foreground. Next enable IPSec via 'setkey -f /etc/ipsec.conf' on each system.

You can test your IPSec tunnel by pinging the gateway's gif IP address from the sensor:

```
fedorov:/root# ping 10.4.12.1
PING 10.4.12.1 (10.4.12.1): 56 data bytes
64 bytes from 10.4.12.1: icmp_seq=1 ttl=64 time=7.428 ms
64 bytes from 10.4.12.1: icmp_seq=2 ttl=64 time=7.343 ms
^C
--- 10.4.12.1 ping statistics ---
3 packets transmitted, 2 packets received, 33% packet los
round-trip min/avg/max/stddev = 7.343/7.386/7.428/0.042 m
```

We lost the first packet during negotiation of the tunnel. Here is how the traffic looks. Notice first the tunnel set-up, and then the ESP IPSec packets:

```
16:25:22.918098 IP 18.235.153.37.500 > 78.172.25.27.500:
```

hakin9 (11)
hardware (20)
hiring (1)
history (5)
impressions (18)
incidents (36)
information warfare (1)
infrastructure (1)
insiders (10)
insurance (3)
intelligence (2)
internet (2)
interviews (5)
ips (10)
ipv6 (61)
ir (16)
iran (1)
IS (1)
itu (1)
kill chain (5)
law (19)
leadership (2)
legislation (1)
linux (16)
malware (15)
mandiant (6)
metasploit (5)
metrics (2)
microsoft (31)
mssp (1)
net optics (14)
network (5)
novasec (3)

```
 isakmp: phase 1 I agg
16:25:22.972442 IP 78.172.25.27.500 > 18.235.153.37.500:
 isakmp: phase 1 R agg
16:25:22.991453 IP 18.235.153.37.500 > 78.172.25.27.500:
 isakmp: phase 1 I agg
16:25:22.993060 IP 18.235.153.37.500 > 78.172.25.27.500:
 isakmp: phase 2/others I inf[E]
16:25:23.029160 IP 78.172.25.27.500 > 18.235.153.37.500:
 isakmp: phase 2/others R inf[E]
16:25:23.039564 IP 18.235.153.37.500 > 78.172.25.27.500:
 isakmp: phase 2/others I oakley-quick[E]
16:25:23.074885 IP 78.172.25.27.500 > 18.235.153.37.500:
 isakmp: phase 2/others R oakley-quick[E]
16:25:23.081259 IP 18.235.153.37.500 > 78.172.25.27.500:
 isakmp: phase 2/others I oakley-quick[E]
16:25:23.909566 IP 18.235.153.37 > 78.172.25.27:
 ESP(spi=0x02a91a40,seq=0x1)
16:25:23.916908 IP 78.172.25.27 > 18.235.153.37:
 ESP(spi=0x0b103e81,seq=0x1)
16:25:24.919570 IP 18.235.153.37 > 78.172.25.27:
 ESP(spi=0x02a91a40,seq=0x2)
16:25:24.926837 IP 78.172.25.27 > 18.235.153.37:
 ESP(spi=0x0b103e81,seq=0x2)
```

That's it. You will also be able to ping the backend (192.168.1.10) from the sensor, or ping the sensor (10.4.12.10) from the backend. It will all be encrypted via IPSec.

If you've been trying to deploy IPSec on FreeBSD, and have followed certain threads, you'll see I encountered no issues with enabling FAST_IPSEC and INET6 in the kernel. I also did not have to exempt port 500 UDP key exchange traffic in my /etc/ipsec.conf file. Those two problems seem to have been ironed out with FreeBSD 5.3 RELEASE.

Posted by Richard Bejtlich at 15:19

Labels: favorites, freebsd, nsm

8 comments:

Anonymous said...

I've tried IPSEC, but found it difficult to setup, especially when some IP addresses are dynamic or behind other private networks (which may block IPSEC).

Instead, OpenVPN (/usr/ports/security/openvpn) proved to be a much easier and reliable solution. It handles certificate verification, key exchange, authentication, encryption and tunneling. It uses UDP (or optionally TCP) for all communication.

10:29 PM

Anonymous said...

I think you have an error in IP address on gateway. On these ASCII 'pictures' it is 78.172.25.30 and in text it is 78.172.25.27

4:54 PM

Richard Bejtlich said...

Regarding the IPs in the ASCII art -- thanks, I just fixed those.

Also, the gifconfig commands I referenced from the FreeBSD Handbook no longer exist, so I updated the blog entry with the new syntax using ifconfig.

5:39 PM

Richard Bejtlich said...

I'm using FreeBSD 6.0 right now, where crypto, ipfw, and ipdivert are all available as kernel modules:

```
orr:/home/richard$ kldstat
Id Refs Address Size Name
1 10 0xc0400000 63072c kernel
2 2 0xc0a31000 74b0 snd_csa.ko
3 3 0xc0a39000 1d408 sound.ko
4 1 0xc0a57000 c3a4 r128.ko
5 2 0xc0a64000 eeec drm.ko
6 16 0xc0a73000 568dc acpi.ko
orr:/home/richard$ sudo kldload crypto
orr:/home/richard$ sudo kldload ipfw
orr:/home/richard$ sudo kldload ipdivert
orr:/home/richard$ kldstat
Id Refs Address Size Name
1 13 0xc0400000 63072c kernel
2 2 0xc0a31000 74b0 snd_csa.ko
3 3 0xc0a39000 1d408 sound.ko
4 1 0xc0a57000 c3a4 r128.ko
5 2 0xc0a64000 eeec drm.ko
6 16 0xc0a73000 568dc acpi.ko
7 1 0xc2032000 21000 crypto.ko
8 2 0xc2057000 c000 ipfw.ko
9 1 0xc2069000 4000 ipdivert.ko
```

Snort Report (22)

solaris (2)

south korea (1)

spying (4)

standards (8)

statistics (1)

stock (1)

strategy (6)

sysadmin (16)

threat model (11)

threats (100)

tools (43)

Traffic Talk (10)

training (51)

TTP (9)

tufte (10)

tv (1)

ubuntu (6)

usenix (1)

verizon (5)

virtualization (32)

vulnerabilities (37)

war (1)

warranties (1)

wisdom (13)

writing (44)

Powered by Blogger.

Say goodbye to kernel recompiles!

Keep in mind that this is the default ipfw rule loaded by the kernel module:

```
orr:/home/richard$ sudo ipfw list
65535 deny ip from any to any
```

To open things up, try this:

```
orr:/home/richard$ sudo ipfw add allow ip from any to
any
00100 allow ip from any to any
```

If you load the module remotely, you will cut off all access to the machine by default.

10:01 PM

B. de Bruin said...

I always wonder why people always initialize a gif-tunnel for ipsec traffic, while it functions perfectly without it.

There are only two reasons why you would use it: 1) FreeBSD does not support NAT-T (for example openbsd and linux both do) and you need NAT 2) routing, openbsd adds ipsec tunnels to its routing tables afaik

besides that, gif tunnel add overhead decreasing the mtu

4:19 AM

Richard Bejtlich said...

B. de Bruin, you answered your own question, at least in my scenario -- NAT.

7:48 AM

Anonymous said...

If you got more than one zone (i.e vlans) on the two fw's, you will also need the interface to get proper firewalling.

7:36 AM

dghnfgj said...

This comment has been removed by a blog administrator.
3:29 AM

Post a Comment

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)