

Configuring IPsec/IKE on Solaris Part Three

Ido Dubrawsky 2002-09-23

This is the third article in a three-part series on configuring IPsec and the Internet Key Exchange (IKE) on Solaris hosts. The [first article](#) covered the basics of IPsec and IKE. The [second article](#) focused on configuring IPsec to protect traffic between two Solaris hosts. This article will discuss the configuration of an IPsec VPN tunnel between two Solaris hosts.

Package Installation

During the lab work for this article an annoying bug was discovered in the Solaris IPsec utilities. The configuration of ESP in the ipseckey utility is not possible in the default installation of Solaris. In order to use the ESP protocol defined in IPsec the following Solaris packages must be installed on the VPN gateways: SUNWcry and SUNWcryrx. These packages are available for [download](#) from the [Sun](#) Web site at no cost for registered users. No reference is made to these additional packages in Sun's on-line documentation and the only way to identify the problem is to do a search on [SunSolve](#) with a valid SunSolve account.

Without these packages no encryption can be done to support ESP. Use the `ndd` command to determine whether the packages are already installed on the gateway hosts. If the encryption packages are not installed, the value of the "Encryption Algorithms" field in the `ndd` query will be set to 1 as shown below:

```
[root@dhcp-7-11 /]
2 # ndd /dev/ipsecesp ipsecesp_status
ESP status
-----
Authentication algorithms          = 2
Encryption Algorithms              = 1 <----- without ESP
Packets passing authentication     = 0
Packets failing authentication     = 0
Packets apparently decrypting badly = 0
Packets failing replay checks      = 0
Packets failing early replay checks = 0
Failed inbound SA lookups          = 0
Inbound PF_KEY messages           = 0
Inbound ESP packets               = 0
Outbound ESP requests             = 0
PF_KEY ACQUIRE messages          = 0
Expired associations (# of bytes)  = 0
Discarded inbound packets         = 0
Discarded outbound packets        = 0
[root@dhcp-7-11 /]
```

3 #

If the encryption packages are installed, the value of "Encryption Algorithms" will be 3:

```
[root@vpn2 /]
# ndd /dev/ipsecesp ipsecesp_status
ESP status
-----
Authentication algorithms          = 2
Encryption Algorithms              = 3 <----- with ESP
Packets passing authentication     = 186
Packets failing authentication     = 0
Packets apparently decrypting badly = 0
Packets failing replay checks      = 0
Packets failing early replay checks = 0
Failed inbound SA lookups         = 0
Inbound PF_KEY messages           = 15
Inbound ESP packets               = 186
Outbound ESP requests              = 151
PF_KEY ACQUIRE messages          = 2
Expired associations (# of bytes)  = 0
Discarded inbound packets         = 0
Discarded outbound packets        = 0
[root@vpn2 /]
#
```

SA Configuration

With the encryption packages installed, the configuration of the IPsec security association (SA) can proceed. The [second article](#) in this series covered the syntax of the ipseckey command. The ipsec SA information can be put into a file and loaded using the -f option to ipseckey as shown below:

```
[root@vpn1 /]
# cat /etc/inet/ipseckey
add esp spi 5346 src 10.89.144.243 dst 10.89.144.244 authalg HMAC-MD5
authkey AF123BCDE89C53482AA4221CF89E343D encrkey AFBCDEFA12345678
add esp spi 6435 src 10.89.144.244 dst 10.89.144.243 authalg HMAC-MD5
authkey F123BCE6583132CF68DACB9FC8339D4B encrkey AFBCDEFA12345678

[root@vpn1 /]
# ipseckey -f /etc/inet/ipseckey
```

```
[root@vpn1 /]
# ipseckey dump
Base message (version 2) type DUMP, SA type ESP.
Message length 152 bytes, seq=1, pid=8146.
SA: SADB_ASSOC spi=0x1923, replay=0, state=MATURE
SA: Authentication algorithm = HMAC-MD5
SA: Encryption algorithm = DES-CBC
SA: flags=0xc0000000 < X_USED X_UNIQUE >
SRC: Source address (proto=0/)
SRC: AF_INET:  port = 0, 10.89.144.244 (vpn2).
DST: Destination address (proto=0/)
DST: AF_INET:  port = 0, 10.89.144.243 (vpn1).
AKY: Authentication key.
AKY: f123bce6583132cf68dacb9fc8339d4b/128
EKY: Encryption key.
EKY: aeabcdfffb13345779/64
  LT: Lifetime information
CLT: 6536 bytes protected, 0 allocations used.
CLT: SA added at time Wed Sep 11 01:29:39 2002
CLT: SA first used at time Wed Sep 11 01:35:09 2002
CLT: Time now is Wed Sep 11 22:14:09 2002
```

```
Base message (version 2) type DUMP, SA type ESP.
Message length 152 bytes, seq=1, pid=8146.
SA: SADB_ASSOC spi=0x14e2, replay=0, state=MATURE
SA: Authentication algorithm = HMAC-MD5
SA: Encryption algorithm = DES-CBC
SA: flags=0x80000000 < X_USED >
SRC: Source address (proto=0/)
SRC: AF_INET:  port = 0, 10.89.144.243 (vpn1).
DST: Destination address (proto=0/)
DST: AF_INET:  port = 0, 10.89.144.244 (vpn2).
AKY: Authentication key.
AKY: af123bcde89c53482aa4221cf89e343d/128
EKY: Encryption key.
EKY: aeabcdfffb13345779/64
  LT: Lifetime information
CLT: 7864 bytes protected, 0 allocations used.
CLT: SA added at time Wed Sep 11 01:29:39 2002
CLT: SA first used at time Wed Sep 11 01:35:09 2002
CLT: Time now is Wed Sep 11 22:14:09 2002
```

```
Dump succeeded for SA type 0.
```

```
[root@vpn1 /]
```

```
#
```

The SAs shown above use an HMAC-MD5 as the authentication algorithm and DES as the encryption algorithm. The authentication and the encryption keys are stored in this file and, therefore, this file should *not* be world readable. This file should be treated in a similar manner as the `/etc/shadow` file is treated.

Once the SAs have been established on both hosts, the next step is to configure the gateway tunnel. The configuration of the lab used for this article is shown in Figure 1 below. In this configuration the two VPN gateways were connected to a switch and resided within the same /24 subnet. While this is not likely to reflect a "real-world" configuration with regards to the connectivity between the two hosts, it does not affect the configurations needed on the Solaris hosts. A discussion of additional considerations when the two VPN gateways are separated by a WAN cloud is provided at the end of this article.

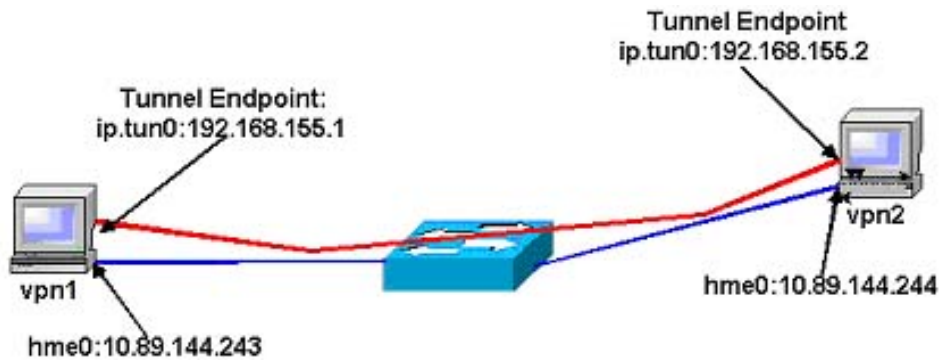


Figure 1 - IPsec Tunnel

Given the fact that a VPN gateway host will most likely be placed outside of a firewall in an enterprise's DMZ the system must first be hardened so that it will not be compromised by an attacker. There are several tools available on the Internet to help lock down Solaris systems. One of the more mature tools is [Titan](#). A very simple (and certainly not a comprehensive) list of tasks to complete to harden a Solaris host is provided below.

- Turn off "small" services such as `chargen`, `discard`, `time`, and `echo` - this can be accomplished by commenting out these services in the `/etc/inet/inetd.conf` file.
- If the host is not to be used as a mail server turn off `Sendmail`
- Turn off any unnecessary services - these can include the ToolTalk daemon, `sadmind`, `rpc.cmsd`, `fs` (font server), `finger`, etc. Again these can be turned off by commenting them out in the `/etc/inet/inetd.conf` file.
- The host should be a stand-alone system and not mount any NFS volumes or export any NFS filesystems. Disable NFS. Consequently also disable `rpcbind`.
- Turn off `telnet` and `FTP` by commenting these services out of `/etc/inet/inetd.conf`

- Install Secure Shell for secure remote management

The list above only begins to scratch the surface of hardening Solaris in preparation for use as a VPN gateway. Sun provides significantly more information on the subject in the [Sun Blueprint](#) series and there are additional articles available elsewhere throughout the Internet on hardening Solaris.

VPN Configuration

The purpose of the gateways is to provide a private connection between two hosts or two networks across an untrusted public network. To do this, the two hosts must not forward packets between physical interfaces but should instead forward packets from a physical interface through a logical interface. The `ndd` command can be used to stop the Solaris kernel from forwarding packets between physical interfaces. The two commands to do this are:

```
[root@vpn1 /]
# ndd -set /dev/hme0 ip_forwarding 0
[root@vpn1 /]
# ndd -set /dev/hme0 ip_strict_dst_multihoming 0
```

The first command turns off IP forwarding between interfaces and the second command simply ensures that packets arriving on one of the physical interfaces is destined for the address assigned to that interface.

Once the forwarding of packets between physical interfaces has been disabled the gateway tunnel endpoints can be configured. The `ifconfig` command is used to configure the VPN tunnel endpoints. However, unlike the normal syntax for configuring physical interfaces, the syntax for virtual interfaces is:

```
ifconfig <virtual interface name> <tunnel src address> <tunnel dst address>
      tsrc <system1 address> tdst <system2 address> encr_alg <encryption algorithm>
      encr_auth_alg <authentication algorithm>
```

To configure the tunnel endpoints on the hosts requires that the interfaces be first plumbed and then configured in two steps. On the host `vpn1`:

```
[root@vpn1 /]
# ifconfig ip.tun0 plumb
[root@vpn1 /]
# ifconfig ip.tun0 192.168.155.1 192.168.155.2 tsrc 10.89.144.243 tdst 10.89.144.244
encr_algs
      des encr_auth_algs md5
[root@vpn1 /]
# ifconfig ip.tun0 up
[root@vpn1 /]
```

```
# ifconfig -a
lo0: flags=1000849 mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
hme0: flags=1000843 mtu 1500 index 2
    inet 10.89.144.243 netmask ffffffff80 broadcast 10.89.144.255
    ether 8:0:20:a4:f2:f8
ip.tun0: flags=10008d1 mtu 1480 index 4
    inet tunnel src 10.89.144.243    tunnel dst 10.89.144.244
    tunnel security settings  esp (des-cbc/hmac-md5)
    inet 192.168.155.1 --> 192.168.155.2 netmask ffffffff00
```

And similarly on the host vpn2:

```
[root@vpn2 /]
# ifconfig ip.tun0 plumb
[root@vpn2 /]
# ifconfig ip.tun0 192.168.155.2 192.168.155.1 tsrc 10.89.144.244 tdst 10.89.144.243
encr_algs
    des encr_auth_algs md5
[root@vpn2 /]
# ifconfig ip.tun0 up
[root@vpn 2/]
# ifconfig -a
lo0: flags=1000849 mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
hme0: flags=1000843 mtu 1500 index 2
    inet 10.89.144.244 netmask ffffffff80 broadcast 10.89.144.255
    ether 8:0:20:9c:79:a0
ip.tun0: flags=10008d1 mtu 1480 index 3
    inet tunnel src 10.89.144.244    tunnel dst 10.89.144.243
    tunnel security settings  esp (des-cbc/hmac-md5)
    inet 192.168.155.2 --> 192.168.155.1 netmask ffffffff00
```

Successfully pinging between the two endpoints indicates that the VPN tunnel has been established:

```
[root@vpn2 /]
# ping -s 192.168.155.1
PING 192.168.155.1: 56 data bytes
64 bytes from 192.168.155.1: icmp_seq=0. time=1. ms
64 bytes from 192.168.155.1: icmp_seq=1. time=1. ms
```

```
64 bytes from 192.168.155.1: icmp_seq=2. time=1. ms
```

```
^C
```

```
----192.168.155.1 PING Statistics----
```

```
3 packets transmitted, 3 packets received, 0% packet loss
```

```
round-trip (ms)  min/avg/max = 1/1/1
```

```
[root@vpn2 /]
```

```
#
```

To allow traffic to be forwarded through the tunnel both the gateway's internal system interface and the tunnel interface should have IP forwarding turned on:

```
[root@vpn1 /]
```

```
# ndd -set /dev/ip ip.tun0:ip_forwarding 1
```

```
[root@vpn1 /]
```

```
# ndd -set /dev/ip hme1:ip_forwarding 1
```

Routing Packets Through the Tunnel

With the VPN tunnel established, the next step is to ensure that the gateway will properly route packets from other interfaces through the tunnel and not out the external interface. However, the routing protocol should not advertise to other systems that the external interface (hme0) is a valid interface for forwarding packets. To resolve this problem simply make the external interface a private interface.

```
[root@vpn1 /]
```

```
# ifconfig hme0 private
```

Next ensure that the external interface is the default interface for the host. While all traffic will be transported through the VPN tunnel the host itself still needs to have a default gateway to send the IPsec packets so that they will reach the VPN peer.

The final step is to run a routing protocol on the gateway hosts so that they will advertise to the internal network the existence of the tunnel as well as the network behind it. This can be achieved by simply running Solaris' in.routed routing program.

```
[root@vpn2 /]
```

```
# netstat -rn
```

```
Routing Table: IPv4
```

Destination	Gateway	Flags Ref	Use	Interface
-------------	---------	-----------	-----	-----------

192.168.155.2	192.168.155.1	UGH	1	0
192.168.155.1	192.168.155.2	UH	1	1 ip.tun0
10.89.144.128	10.89.144.244	U	1	114 hme0
224.0.0.0	10.89.144.244	U	1	0 hme0
default	10.89.144.244	UG	1	225
127.0.0.1	127.0.0.1	UH	1	5858 lo0

Once all of these steps are done all traffic between the two hosts is encrypted by the VPN tunnel. Figures 2 and 3 show a connection being made between the two test hosts as well as the packet trace of the traffic as captured with Ethereal.

```
[idubraws@vpn1 idubraws]
6 $ ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ffffffff
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 10.89.144.243 netmask ffffffff broadcast 10.89.144.255
ip.tun0: flags=10008d1<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST,IPv4> mtu 1400 index 4
    inet tunnel src 10.89.144.243 tunnel dst 10.89.144.244
    tunnel security settings esp (des-cbc/hmac-md5)
    inet 192.168.155.1 -> 192.168.155.2 netmask ffffffff
[idubraws@vpn1 idubraws]
7 $ telnet 192.168.155.2
Trying 192.168.155.2...
Connected to 192.168.155.2.
Escape character is '^'.

SunOS 5.8

login: idubraws
Password:
Last login: Wed Sep 11 22:05:27 from 192.168.155.1
[idubraws@vpn2 idubraws]
1 $
```

Figure 2 - Connection Through IPsec Tunnel

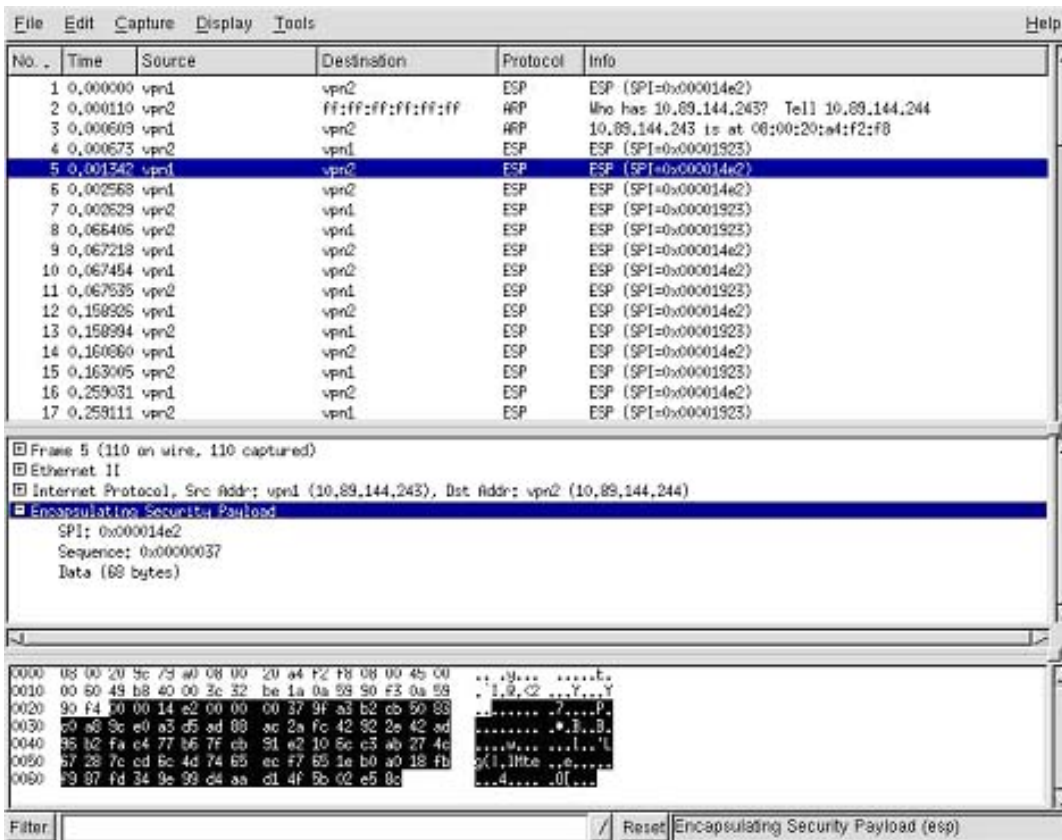


Figure 3 - Packet Trace of Connection Through IPsec Tunnel

Other Configurations

While the VPN gateways used in this example are connected to a common switch they could just as easily have been behind routers separated by a WAN cloud as shown in Figure 4.

The primary difference between the two configurations is the use of ACLs and NAT on the routers. In its current design, IPsec cannot pass through a NAT gateway because of the fact that the NATing system will try to make changes to various parts of the IP packet. These changes will then change the IP checksum of the packet but not the IPsec checksum of the packet. The receiving gateway will then conclude that since the IPsec checksum and the IP checksum do not match that the packet must have been tampered with during transit and is to be thrown away. Because of this several vendors have devised a workaround to the IPsec through NAT problem by encapsulating the traffic in a TCP or UDP header and having modify that header. The other side of the connection must also be able to handle the extra header and ensure that the packet arrives at the VPN peer with the proper destination address. The IETF is currently working on a more complete solution to this problem.

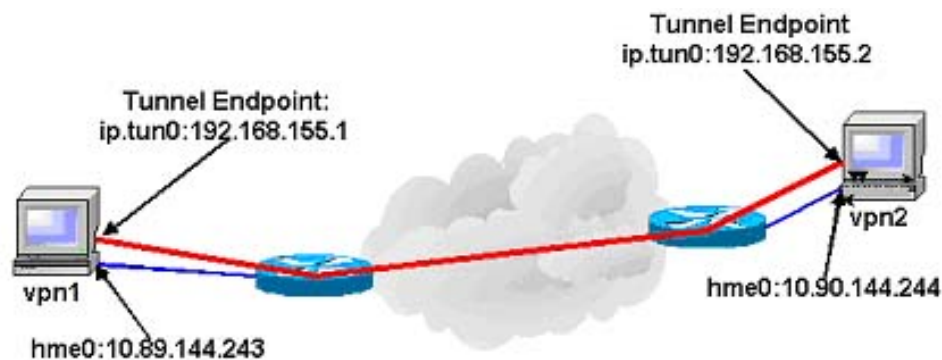


Figure 4 - IPsec Tunnel with WAN

Access Control Lists on routers also present a problem to IPsec but not nearly as much of a problem as NAT. ACLs on a router should be configured to permit AH and ESP traffic (protocols 50 and 51) through as well as open up port UDP 500 (for Internet Key Exchange traffic). An example snippet of an IPsec "friendly" ACL is shown below:

```
access-list 101 permit esp host 10.90.144.244 host 10.89.144.243
access-list 101 permit ahp host 10.90.144.244 host 10.89.144.243
access-list 101 permit udp host 10.90.144.244 host 10.89.144.243 eq isakmp
```

These ACLs permit ESP and AH traffic from the host 10.90.144.244 to the gateway host 10.89.144.243. Furthermore, the ACLs allow IKE traffic from host 10.90.144.244 to reach the gateway host 10.89.144.243 on port 500 (isakmp).

Solaris 9

Sun's inclusion of IPsec capabilities in Solaris is a significant step forward in broadening Solaris' functional roles. With Solaris 9 Sun has included additional features and utilities such as `ikeadm` and `in.iked` to help system administrators control and tune the Internet Key Exchange (IKE) more completely. The `in.iked` program provides automated key management for IPsec by implementing IKE authentication using either pre-shared keys, certificates or signatures; authentication protection and Diffie-Hellman key derivation.

Relevant Links

[Configuring IPsec and Ike on Solaris, Part One](#)
Ido Dubrawsky, securityFocus

[Configuring IPsec and Ike on Solaris Part Two](#)
Ido Dubrawsky, SecurityFocus

