Cisco Group Encrypted Transport VPN (GET VPN) and LISP Interaction

SDU DCI Validation and Configuration Notes

June 19, 2012

Group Encrypted Transport VPN (GET VPN) and LISP Interaction

Cisco ASR and ISR series routers can be configured for both GET VPN encryption and LISP, therefore a decision must be made regarding what (if any) LISP traffic to encrypt when GET VPN is already deployed, or will be deployed, prior to the introduction of LISP.

Note This document assumes a working knowledge of GET VPN. An ACL should be prepared ahead of time and be ready to combine with the information in this document. GET VPN documentation, including the *Cisco Group Encrypted Transport VPN* Configuration Guide can be found here: http://www.cisco.com/en/US/docs/ios-xml/ios/sec_conn_getvpn/configuration/xe-3s/sec-get-vpn.html

Options

GET VPN functionality can be selectively used to encrypt none, some or all of the LISP traffic transiting between two or more routers in a GDOI[1] domain. Since GET VPN uses an ACL to determine which traffic to encrypt or leave unencrypted, it's possible to encrypt all LISP data and/or control traffic by use of explicit permit statements in the ACL. When a port is permitted in a GET VPN ACL, packets sent to the port will be *encrypted* which will provide the benefit of secure data and/or control information exchanges between the GDOI member routers. Packets that are encrypted by a GDOI domain member can only be unencrypted by another member router, therefore if a GDOI member router suffers a fault or misconfiguration affecting GET VPN, any encrypted LISP packets normally transiting the affected router would be affected.

Conversely, ACL deny statements can be used to *prevent* encryption to data and/or control traffic between GDOI group members. Using deny statements for LISP data or control ports can prevent an unforeseen impact to LISP due to a GET VPN session failure or misconfiguration by modifying the ACL used in the configuration to deny the data and/or control port numbers. When a port is denied in a GET VPN ACL, packets sent to the port are *ignored* and will not be encrypted. Configuring the ACL to ignore the control or both control and data ports used by LISP will ensure that if the GET VPN session were to go down or if a misconfiguration were introduced affecting the GET VPN settings, any LISP traffic that matched a deny ACL entry would be unaffected by the GET VPN session failure.

Lastly there are two hybrid options: data packets can be encrypted, but control packets can be left unencrypted, or the reverse where control is encrypted but data is unencrypted.

Assuming that a best practice would be to use a deterministic approach to explicitly deny or permit LISP control and data traffic, the possibilities are in Table 1-1.

1. Group Domain of Interpretation – see RFC 6407: http://tools.ietf.org/html/rfc6407

Table 1-1 Deny or Permit Options for LISP Control and Data Traffic

Data Port	Control Port
Deny	Deny
Deny	Permit
Permit	Deny
Permit	Permit

Careful planning and design considerations should be implemented when creating or editing an ACL to have the desired effect on the traffic that could be seen and acted on by the ACL, according to the needs and priorities for a given network.

LISP Traffic and GET VPN ACL Considerations

In the LISP whitepaper "Locator/ID Separation Protocol", available on CCO[1], we see that both the port numbers and traffic type used by LISP are specified:

"Both data and control messages use User Datagram Protocol (UDP) transport to facilitate passage through firewalls and distribution across parallel link aggregation group (LAG) paths through the Internet. **Encapsulated user** *data packets* **are transported using UDP port** *4341***, and LISP** *control packets* **are transported using UDP port** *4342***.**"

When a port is denied in a GET VPN ACL, packets using the port are ignored and will not be encrypted. Since ports 4341 and 4342 are used by LISP, including both those ports in deny statements in the ACL used by GET VPN will ensure that the LISP traffic will never be encrypted by the GET VPN session. Alternatively, if it is desired that LISP traffic be encrypted, both the control and data ports could be permitted, thus allowing LISP state information and data packets to be transmitted and received in encrypted format. The hybrid option of permitting one type of traffic and denying the other could also be a valid choice. The examples to follow detail how to modify an existing ACL for all three approaches.

Modifying an Existing GET VPN ACL to Deny All LISP Traffic

To modify an existing ACL to *deny* LISP control packets using port 4342, and data packets using port 4341, follow the configuration steps in the next section.

Note The following steps are based on the assumption that GET VPN is already in use and the ACL is available to modify or prepare for the introduction of LISP traffic.

ACL Configuration Procedure

The following procedure, using the appropriate corresponding IOS commands, allows you to modify an existing ACL to deny LISP control packets on port 4342 and data packets on 4341.

1. http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6554/ps6599/ps10800/white_paper_c11-652502.html

Step 1 Enter global configuration mode.

configure terminal

Step 2 Modify an existing extended ACL used for GET VPN.

ip access-list extended *ACL_NAME*

Step 3 By prepending a line number to an entry into the ACL, we can control where it appears in the ACL, which can affect how likely the entry will match the condition. Here a UDP packet sent from any IP to any IP using destination UDP 4342 will trigger a match on the ACL for LISP control packets.

line_number **deny udp any any eq 4342**

Step 4 Here a UDP packet sent from any IP to any IP using destination UDP 4341 will trigger a match on the ACL for LISP data packets.

line_number **deny udp any any eq 4341**

Step 5 After inserting the lines into the ACL, we can resequence them, selecting the starting number and also the step increment value for each line in the ACL.

ip access-list resequence *ACL_NAME starting_number step_increment_value*

Step 6 Exit configuration mode.

exit

Example Configuration Changes

The following examples illustrate an ACL configuration procedure. The ACL being modified on the router called key-server is named GET VPN_EXAMPLE. Entries for port 4342 and 4341 will be added to the ACL. Note that the router *key-server* functions as the key server[1] for the GDOI group, and the ACL is stored and maintained on this router. Copies of the ACL are sent to each of the GDOI member routers so that all have the same ACL information to use to deny or permit encryption.

Note For brevity, the following procedure shows a partial configuration of an example ACL and does not convey, expressed or implied, a complete ACL, nor should it be used as an ACL in a production network.

GET VPN ACL Before Modification (ACL Shortened for Brevity)

```
key-server#show ip access-lists GET VPN_EXAMPLE
Extended IP access list GET VPN_EXAMPLE
    10 deny udp any eq 848 any
    20 deny udp any any eq 848
    30 deny tcp any any eq tacacs
    40 deny tcp any eq tacacs any
    50 deny tcp any any eq bgp
    60 deny tcp any eq bgp any
    70 deny ospf any any
    80 deny eigrp any any
    90 deny udp any any eq ntp
    100 deny udp any eq ntp any
```

1. Key Server: Among other duties, the Key Server router maintains the group policy (ACL), creates and maintains the keys used by the group and also manages group membership.

Sample Modification Procedure

The following simplified modification process exemplifies the noted ACL configuration changes.

Step 1 Insert lines 25 and 26.

```
key-server#configure terminal
key-server(config)#ip access-list extended GET VPN_EXAMPLE
key-server(config-ext-nacl)#25 deny udp any any eq 4342
key-server(config-ext-nacl)#26 deny udp any any eq 4341
key-server(config-ext-nacl)#end
key-server#
```

Step 2 Verify changes made to the ACL.

```
key-server#show ip access-list GET VPN_EXAMPLE
Extended IP access list GET VPN_EXAMPLE
10 deny udp any eq 848 any
20 deny udp any any eq 848
25 deny udp any any eq 4342
26 deny udp any any eq 4341
30 deny tcp any any eq tacacs
40 deny tcp any eq tacacs any
50 deny tcp any any eq bgp
60 deny tcp any eq bgp any
70 deny ospf any any
80 deny eigrp any any
90 deny udp any any eq ntp
100 deny udp any eq ntp any
```

Step 3 Resequence the ACL.

```
key-server#configure terminal
key-server(config)#ip access-list resequence GET VPN_EXAMPLE 10 10
key-server(config)#end
key-server#
```

GET VPN ACL After Modification (ACL Shortened for Brevity)

```
key-server#show ip access-list GET VPN_EXAMPLE
Extended IP access list GET VPN_EXAMPLE
    10 deny udp any eq 848 any
    20 deny udp any any eq 848
    30 deny udp any any eq 4342
    40 deny udp any any eq 4341
    50 deny tcp any any eq tacacs
    60 deny tcp any eq tacacs any
    70 deny tcp any any eq bgp
    80 deny tcp any eq bgp any
    90 deny ospf any any
    100 deny eigrp any any
    110 deny udp any any eq ntp
    120 deny udp any eq ntp any
```

The ACL now has deny entries for the LISP control and data ports, and has been uniformly resequenced. Any LISP packets will be sent in unencrypted format no matter what the state of the GET VPN session. If we examine the packets using a sniffer device, we can see that the control packets using port 4342 and data packets using port 4341 are sent unencrypted. See Figure 1-1 and Figure 1-2 for an example sniffer capture showing both control and data packets that were sent without encryption:

Figure 1-1 Sniffer Trace of a LISP Control Packet Sent to Port 4342

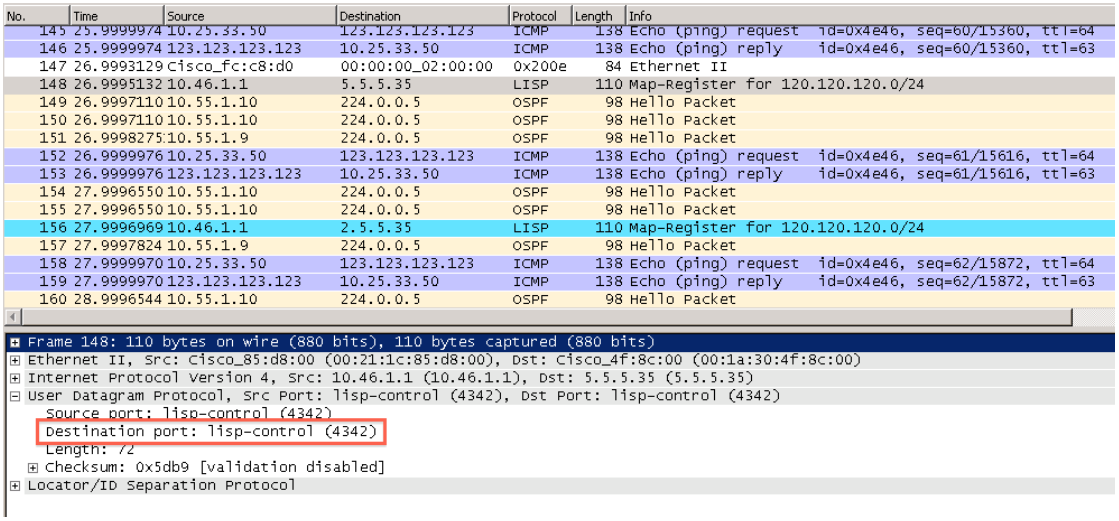
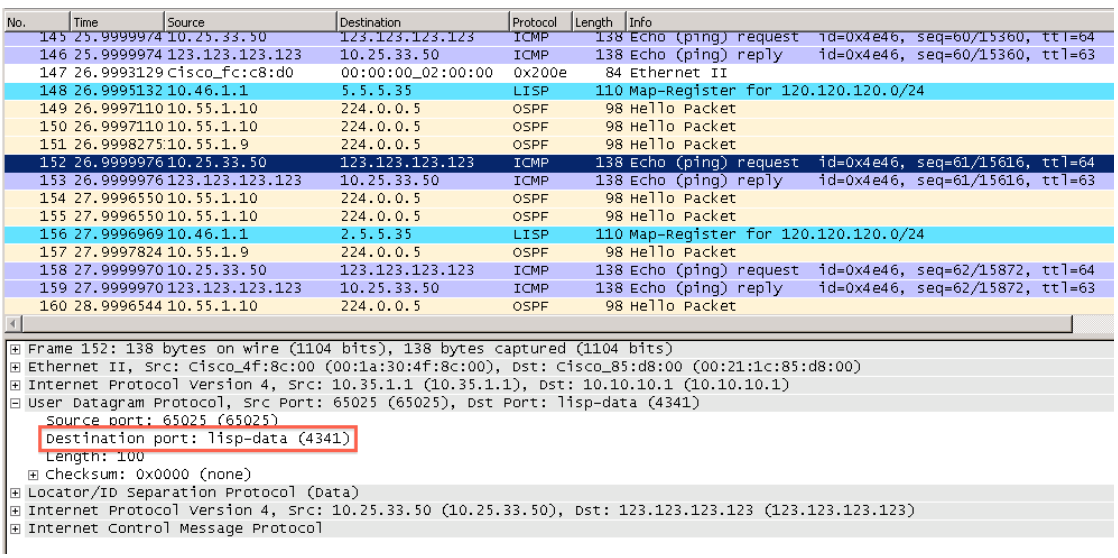


Figure 1-2 Sniffer Trace of a LISP Control Packet Sent to Port 4341



With both LISP ports denied in the GETVPV ACL, all LISP packets would be sent unencrypted, and would be unaffected if there were any problem with the GET VPN ACL or any of the member routers. However, the traffic would be unencrypted, and hence would be less secure and could possibly be captured and used to create crafted packets or for other undesirable purposes.

Modifying Existing GET VPN ACL to Permit All LISP Traffic

To modify an existing ACL to permit (encrypt) LISP control packets using port 4342 and data packets using port 4341, follow the configuration steps in the next section .

Note The following steps are based on the assumption that GET VPN is already in use and the ACL is available to modify to prepare for the introduction of LISP traffic.

ACL Configuration Procedure

The following procedure, using the appropriate corresponding IOS commands, allows you to modify an existing ACL to permit LISP control packets on port 4342 and data packets on 4341.

Step 1 Enter global configuration mode.

configure terminal

Step 2 Modify an existing extended ACL used for GET VPN.

ip access-list extended *ACL_NAME*

Step 3 By prepending a line number to an entry into the ACL, we can control where it appears in the ACL, which can affect how likely the entry will match the condition. Here a UDP packet sent from any IP to any IP using destination UDP 4342 will trigger a match on the ACL.

line_number **permit udp any any eq 4342**

Step 4 A second line is inserted into the ACL. Here a UDP packet sent from any IP to any IP using destination UDP 4341 will trigger a match on the ACL.

line_number **permit udp any any eq 4341**

Step 5 After inserting the lines into the ACL, we can resequence them, selecting the starting number and also the step increment value for each line in the ACL.

ip access-list resequence *ACL_NAME starting_number step_increment_value*

Step 6 Exit configuration mode.

exit

Example Configuration Changes

The following examples illustrate the ACL. The ACL being modified on the router called *key-server* is named GET VPN_EXAMPLE. Entries for port 4342 and 4341 will be added to the ACL.

Note For brevity, the following procedure shows a partial configuration of an example ACL and does not convey, expressed or implied, a complete ACL, nor should it be used as an ACL in a production network.

GET VPN ACL Before Modification (ACL Shortened for Brevity)

```
key-server#show ip access-lists GET VPN_EXAMPLE
Extended IP access list GET VPN_EXAMPLE
    10 deny udp any eq 848 any
    20 deny udp any any eq 848
    30 deny tcp any any eq tacacs
    40 deny tcp any eq tacacs any
    50 deny tcp any any eq bgp
    60 deny tcp any eq bgp any
    70 deny ospf any any
    80 deny eigrp any any
    90 deny udp any any eq ntp
```

Sample Modification Procedure

The following simplified modification process exemplifies the noted ACL configuration changes.

Step 1 Insert lines 25 and 26.

```
key-server#configure terminal
key-server(config)#ip access-list extended GET VPN_EXAMPLE
key-server(config-ext-nacl)#25 permit udp any any eq 4342
key-server(config-ext-nacl)#26 permit udp any any eq 4341
key-server(config-ext-nacl)#end
key-server#
```

Step 2 Verify changes made to the ACL

```
key-server#show ip access-list GET VPN_EXAMPLE
Extended IP access list GET VPN_EXAMPLE
10 deny udp any eq 848 any
20 deny udp any any eq 848
25 permit udp any any eq 4342
26 permit udp any any eq 4341
30 deny tcp any any eq tacacs
40 deny tcp any eq tacacs any
50 deny tcp any any eq bgp
60 deny tcp any eq bgp any
70 deny ospf any any
80 deny eigrp any any
90 deny udp any any eq ntp
100 deny udp any eq ntp any
```

Step 3 Resequence the ACL.

```
key-server#configure terminal
key-server(config)#ip access-list resequence GET VPN_EXAMPLE 10 10
key-server(config)#end
key-server#
```

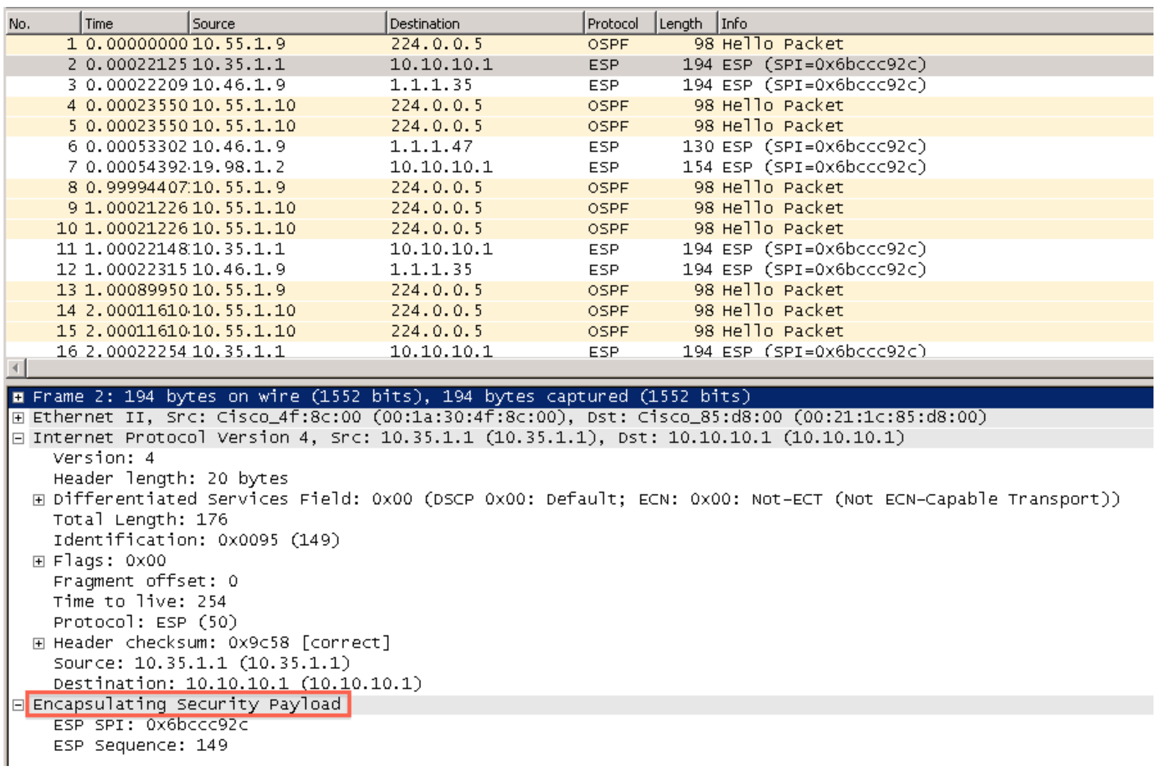
GET VPN ACL After Modification (ACL Shortened for Revity)

```
key-server#show ip access-list GET VPN_EXAMPLE
Extended IP access list GET VPN_EXAMPLE
    10 deny udp any eq 848 any
    20 deny udp any any eq 848
    30 permit udp any any eq 4342
```

```
40 permit udp any any eq 4341
50 deny tcp any any eq tacacs
60 deny tcp any eq tacacs any
70 deny tcp any any eq bgp
80 deny tcp any eq bgp any
90 deny ospf any any
100 deny eigrp any any
110 deny udp any any eq ntp
120 deny udp any eq ntp any
```

The ACL now has permit entries for the LISP control and data ports, and has been resequenced. All LISP control and data packets are encrypted and are sent as Encapsulating Security Payload (ESP) packets. If we examine the packets using a sniffer device, we see no references to ports 4342 or 4341 since the contents of all LISP packets are encrypted. Figure 1-3 exemplifies a sniffer capture showing a sample ESP packet decode. Note that with the exception of OSPF hello packets, all other packets seen in the sniffer trace are encrypted ESP packets:

Figure 1-3 sniffer Capture Showing a Sample ESP Packet Decode



In this situation, the highest level of security would be maintained, since both control and data LISP packets are encrypted and no information that the packets are used for LISP is available in the sniffer trace. However, though this is the most secure approach, there is also some risk in that the control and data information is encrypted, so if there is a GET VPN failure on one or more of the GDOI group member routers, or a corrupt ACL is pushed from the key server to the GDOI members, all LISP traffic would be unavailable until the issue affecting GET VPN were resolved.

A Hybrid Approach—Modifying an Existing GET VPN ACL to Deny LISP Control Traffic and Permit LISP Data Traffic

To modify an existing ACL to deny (leave unencrypted) LISP control packets using port 4342 and permit (encrypt) LISP data packets using port 4341, follow the configuration steps in the next section.

Note The following steps are based on the assumption that GET VPN is already in use and the ACL is available to modify to prepare for the introduction of LISP traffic.

ACL Configuration Procedure

The following procedure, using the appropriate corresponding IOS commands, allows you to modify an existing ACL to deny LISP control packets on port 4342 and permit LISP data packets using port 4341.

Step 1 Enter global configuration mode.

configure terminal

Step 2 Modify an existing extended ACL used for GET VPN.

ip access-list extended *ACL_NAME*

Step 3 By prepending a line number to an entry into the ACL, we can control where it appears in the ACL, which can affect how likely the entry will match the condition. Here a UDP packet sent from any IP to any IP using destination UDP 4342 will trigger a match on the ACL, resulting in the packets being left unencrypted.

line_number **deny udp any any eq 4342**

Step 4 Here a UDP packet sent from any IP to any IP using destination UDP 4341 will trigger a match on the ACL, causing the packets to be encrypted.

line_number **permit udp any any eq 4341**

Step 5 After inserting the lines into the ACL, we can resequence them, selecting the starting number and also the step increment value for each line in the ACL.

ip access-list resequence *ACL_NAME starting_number step_increment_value*

Step 6 Exit configuration mode.

exit

Example Configuration Changes

The following examples illustrate the ACL. The ACL being modified on the router called *key-server* is named GET VPN_EXAMPLE. Entries for port 4342 and 4341 will be added to the ACL.

For brevity, the following procedure shows a partial configuration of an example ACL and does not convey, expressed or implied, a complete ACL, nor should it be used as an ACL in a production network.GET VPN ACL Before modification (ACL shortened for brevity)

```
key-server#show ip access-lists GET VPN_EXAMPLE
Extended IP access list GET VPN_EXAMPLE
```

```
10 deny udp any eq 848 any
20 deny udp any any eq 848
30 deny tcp any any eq tacacs
40 deny tcp any eq tacacs any
50 deny tcp any any eq bgp
60 deny tcp any eq bgp any
70 deny ospf any any
80 deny eigrp any any
90 deny udp any any eq ntp
```

Sample Modification Procedure

The following simplified modification process exemplifies the noted ACL configuration changes.

Step 1 Insert lines 25 and 26.

```
key-server#configure terminal
key-server(config)#ip access-list extended GET VPN_EXAMPLE
key-server(config-ext-nacl)#25 deny udp any any eq 4342
key-server(config-ext-nacl)#26 permit udp any any eq 4341
key-server(config-ext-nacl)#end
key-server#
```

Step 2 Verify changes made to the ACL.

```
key-server#show ip access-list GET VPN_EXAMPLE
Extended IP access list GET VPN_EXAMPLE
10 deny udp any eq 848 any
20 deny udp any any eq 848
25 deny udp any any eq 4342
26 permit udp any any eq 4341
30 deny tcp any any eq tacacs
40 deny tcp any eq tacacs any
50 deny tcp any any eq bgp
60 deny tcp any eq bgp any
70 deny ospf any any
80 deny eigrp any any
90 deny udp any any eq ntp
100 deny udp any eq ntp any
```

Step 3 Resequence the ACL.

```
key-server#configure terminal
key-server(config)#ip access-list resequence GET VPN_EXAMPLE 10 10
key-server(config)#end
key-server#
```

GET VPN ACL After Modification (ACL Shortened for Brevity)

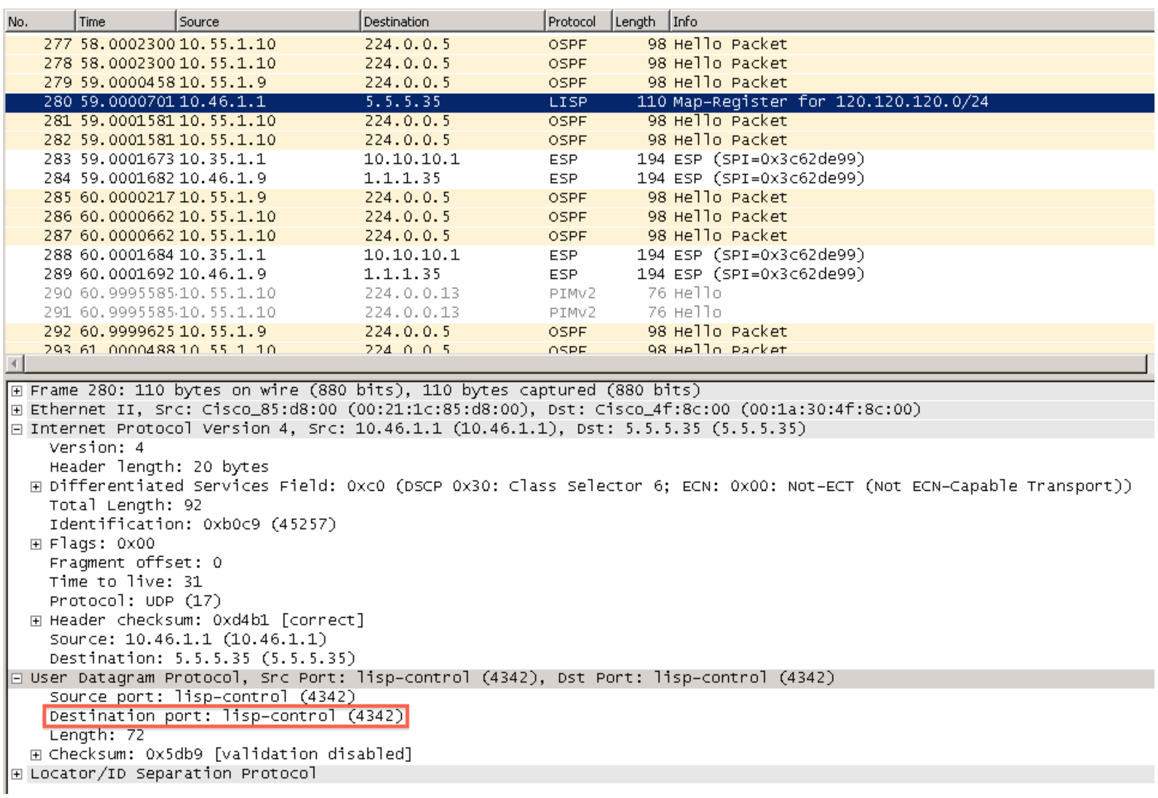
```
key-server#show ip access-list GET VPN_EXAMPLE
Extended IP access list GET VPN_EXAMPLE
    10 deny udp any eq 848 any
    20 deny udp any any eq 848
    30 deny udp any any eq 4342
    40 permit udp any any eq 4341
    50 deny tcp any any eq tacacs
    60 deny tcp any eq tacacs any
    70 deny tcp any any eq bgp
    80 deny tcp any eq bgp any
    90 deny ospf any any
```

```
100 deny eigrp any any
110 deny udp any any eq ntp
120 deny udp any eq ntp any
```

The ACL now has a deny entry for the LISP control port and a permit entry for the LISP data port, and has been resequenced. All LISP control packets are sent unencrypted while data packets are sent as Encapsulating Security Payload (ESP) packets. If we examine the packets using a sniffer device, we see packets sent to port 4342, but no references to port 4341 since the contents of the packets are encrypted. Figure 1-4 exemplifies sniffer captures showing both unencrypted and ESP encrypted decodes.

Packet number 280 captured during a sniffer trace: It's a Map-Register LISP control packet destined for port 4342 that is unencrypted:

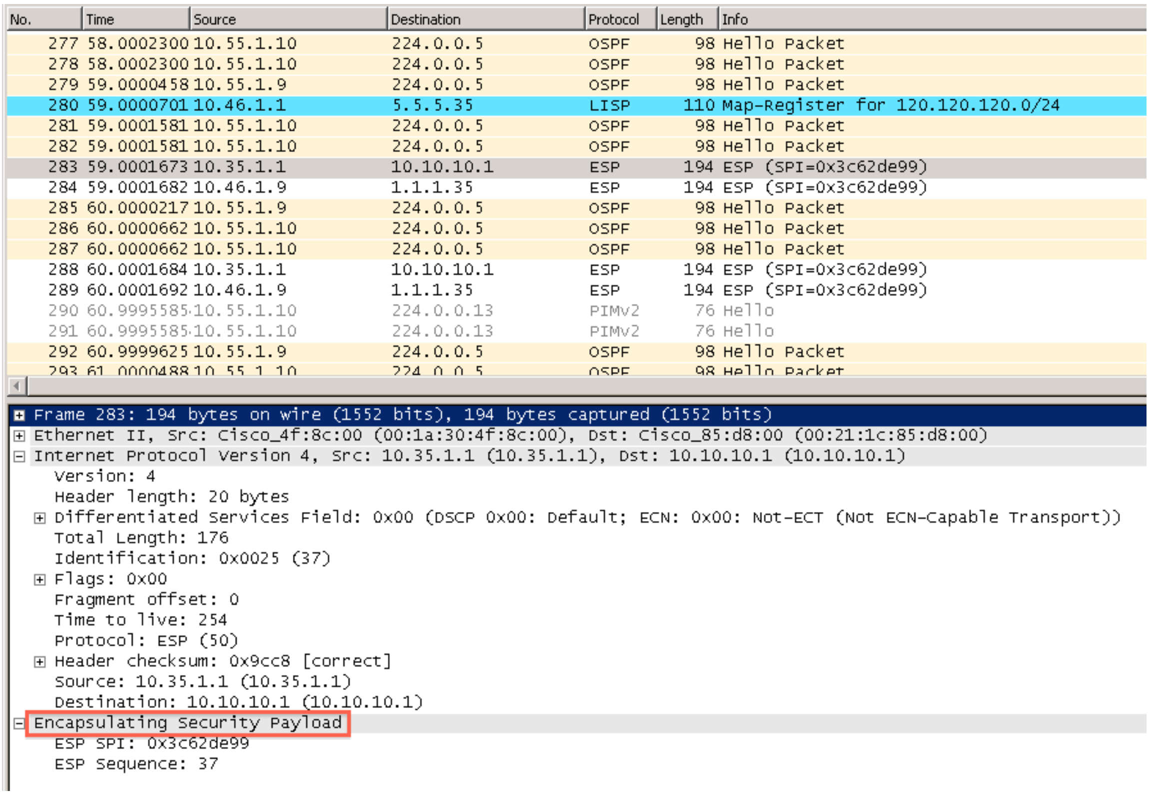
Figure 1-4 Sniffer Captures Showing Both Unencrypted and ESP Encrypted Decodes



Note that with the exception of the OSPF and PIM Hello packets, the other packets seen are Encapsulating Security Payload – ESP packets - representing the LISP data packets sent when the sniffer trace was taken. Because the LISP data is encrypted, we cannot see the destination port number 4341 to which it was sent.

Figure 1-5 shows another packet from the same sniffer trace example, now showing packet number 283 which is an encrypted ping packet:

Figure 1-5 Encrypted Ping Packet



Because the packet is encrypted we cannot see what port it was destined to, however because pings were being sent during the capture period when the sniffer trace was active, we can infer that it was most likely an ICMP ping packet, followed by packet number 284, which was an ICMP ping reply.

In this hybrid case, control packets are sent unencrypted, so LISP state information exchanged between the GDOI group routers would be unaffected by any GET VPN malfunctions. The LISP data would be encrypted, however, so it would be secure when transiting between routers, but would be vulnerable to any GET VPN malfunctions.

For brevity, the other hybrid possibility – control encrypted and data unencrypted – is not shown. To implement such a design, simply switch the port numbers used in the hybrid example information.

Summary

Cisco ASR and ISR series routers provide a wide latitude of flexibility for configuring GET VPN with LISP. GET VPN functionality can be selectively used to encrypt some, all or none of the LISP traffic, depending on the needs in a given network. Careful planning and design considerations should be implemented when creating or editing an ACL to have the desired effect on the traffic that could be seen and acted on by the ACL.

For more about GET VPN, browse to: http://www.cisco.com/go/getvpn

To learn more about LISP, go to http://www.cisco.com/go/lisp and http://lisp.cisco.com

Glossary

The following terms are identified for clarification:

- **ACL—**Access Control List
- **ESP**—Encapsulating Security Payload
- **GDOI—**Group Domain of Interpretation
- **GETVPN—**Group Encrypted Transport Virtual Private Network
- **Key Server—**Among other duties, the Key Server router maintains the policy (ACL), creates and maintains the keys used by the group and manages group membership
- **LISP—**Locator/ID Separation Protocol

Phil Conoly

DCI Systems Engineer, Systems Development Unit, Cisco Systems

Phil Conoly is a Systems testing engineer in SDU

His background in the networking industry spans over 12 years in various roles at Cisco. Phil has experience in a wide range of technologies including Routing, Switching, Storage and DCI related technologies such as OTV and LISP. In his current role, Phil focuses on DCI Systems releases for the enterprise customer. His most recent collaborative Cisco solution testing and validation effort is Cisco LISP VM-Mobility & Path Optimization Solution with EMC VPLEX.