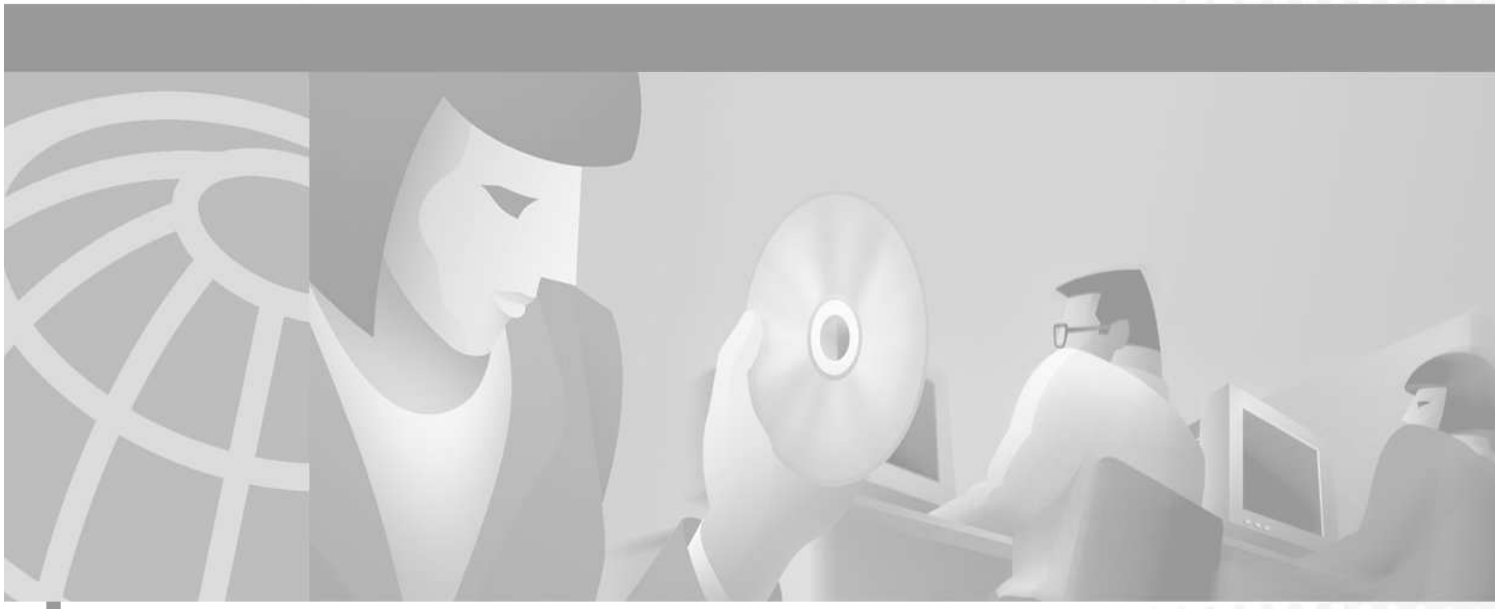


Cisco Systems Advanced Services

SSL Operation on CSS11500

Version 3.0

Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

Contents

Document Control

Author: Yi Xue, Zahoor Khan
Advanced Services
Document Number: AS-59897

History

Table 1 Revision History

Version No.	Issue Date	Status	Reason for Change
0.1	17-October-2003	Draft	Initial Draft
1.0	31-October-2003	Version 1	Incorporated comments from BU TME
2.0	22-February-2005	Version 2	Added operational procedures
3.0	5-May-2005	Version 3	Added Client Auth, SSL Init, SSL Keepalive, and HTT header insert

Review

Table 2 Revision Review

Reviewer's Details	Version No.	Date
Derek Huckaby	0.1	22-October-2003
Jay Cedrone	0.1	28-October-2003
Derek Huckaby	2.0	23-February-2005

Change Forecast: Low

This document will be kept under revision control.

Introduction

CSS11500 supports internal SSL acceleration modules that can be used to decrypt client traffic for better load balancing decision (front-end SSL termination). Using the CSS to offload SSL from the servers significantly increases server performance and allows traffic to be better distributed to backend applications. The CSS11500 can also re-encrypt and send traffic back to the backend SSL server (back-end SSL acceleration). This is necessary for environments requiring secure client to server communication and advanced server load balancing. The integrated SSL capabilities of the CSS allow it to make content aware decisions to ensure the data is sent to the correct application, while maintaining data encryption throughout the network.

This document describes the SSL traffic flow, from the client, to the CSS, and to the backend server. It provides configuration walk through, different implementation scenarios, and operational procedure which offers no interruption to the current SSL production environment that are running pre-WebNS 7.5 code versions. In addition, configuration samples for Client Authentication, SSL initiation, and SSL keepalives are also included.

A section on encryption key and certificate generation procedure is enclosed as reference.

SSL Flow

IP Addresses Used

Following are the IP addresses used in this example:

Client IP: 10.21.80.170

VIP (Public, HTTPS) on the CSS that the client is connecting to: 10.87.99.78:443

VIP (HTTP) on the CSS that's used for load balancing decrypted traffic: 192.191.134.165:80

Real backend server IP addresses: 192.191.134.170, 192.191.134.171

Traffic Flow

Here are the steps involved in a complete client – server SSL connection:

A. Client connects to the public VIP (HTTPS) of a SSL content rule.

B. CSS sends this packet to the SSL module configured (as a service) in the "content SSL-VIP" rule. If multiple SSL modules are configured in the rule, load balance will also be performed. See "Design Considerations" for more details.

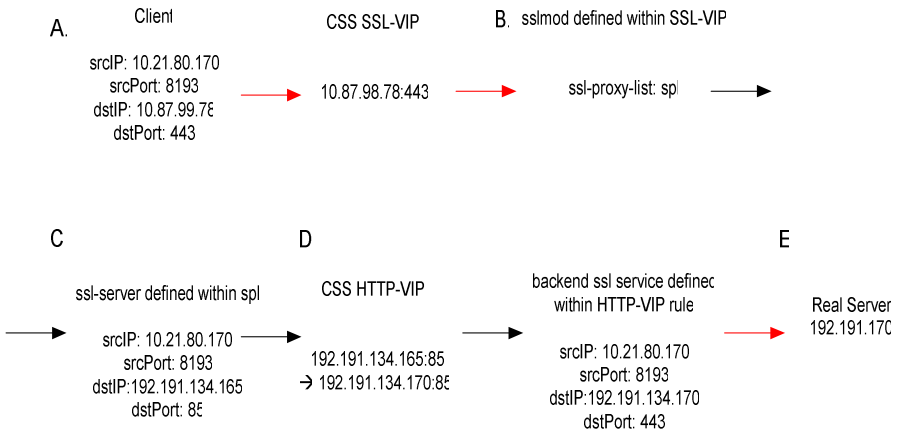
C. According to the ssl-proxy-list configuration for the SSL module, the SSL module first matches the proper ssl-server via the public VIP, then decrypts the packet and sends it to VIP (this VIP could be the same as the public facing VIP in step A, but also could be a bogus private IP if the customer needs to conserve his public IPs and/or for security purposes) by matching the HTTP content rule. This rule will perform load balancing between the real backend servers. In the case of front-end SSL only (SSL traffic terminates at the SSL module), the backend servers are configured as normal local services, and the forward direction flow ends here.

D. If "ssl-accel-backend" is configured on the backend servers' services, it causes the packet to be sent back to the SSL module for re-encryption.

E. Encrypted packet now gets sent to the real backend server.

F. Return traffic follows the reverse of the above, matching the appropriate FCBs.

Note: In the diagram below, red colour arrows represent encrypted traffic. Italics lines in the Sample Configuration represent default settings which will not show up in "show run".



Sample Configuration:

!********** SSL PROXY LIST **********

ssl-proxy-list spl

ssl-server 100

ssl-server 100 rsakey mykeypair

ssl-server 100 rsacert mycert

ssl-server 100 vip address 10.87.99.78

ssl-server 100 port 443 ← must match port configured in SSL-VIP content rule

ssl-server 100 cipher rsa-with-rc4-128-md5 192.191.134.165 85 <- must match port configured in HTTP-VIP content rule

backend-server 10

backend-server 10 type backend-ssl

backend-server 10 ip address 192.191.134.170

backend-server 10 port 86 ← must match port configured in the service

backend-server 10 server-ip 192.191.134.170

backend-server 10 server-port 443 ← must match HTTPS service port running on the real server

backend-server 10 cipher rsa-with-rc4-128-md5

backend-server 20

backend-server 20 type backend-ssl

backend-server 20 ip address 192.191.134.171

backend-server 20 port 86

backend-server 20 server-ip 192.191.134.171

backend-server 20 server-port 443

backend-server 20 cipher rsa-with-rc4-128-md5

active

!********** SERVICE **********

service bessl-svc1

type ssl-accel-backend

ip address 192.191.134.170

port 86

```
  keepalive type ssl
  keepalive port 443
  add ssl-proxy-list spl
active

service bessl-svc2
  type ssl-accel-backend
  ip address 192.191.134.171
  port 86
  keepalive type ssl
  keepalive port 443
  add ssl-proxy-list spl
  active

service sslmod
  type ssl-accel
  keepalive type none
  slot 2
  add ssl-proxy-list spl
  active

!****************** OWNER ******************
owner cisco

  content  HTTP-VIP
    vip address 192.191.134.165
    protocol tcp
     port 85
     add service bessl-svc1
     add service bessl-svc2
     active

  content SSL-VIP
     vip address 10.87.99.78
     protocol tcp
     port 443
     add service sslmod
     active
```

SSL Configuration

Configuration Consideration:

1. Multiple SSL modules can share one single ssl-proxy-list. Or, each can use its own proxy list, which reduces the impact of having to take a module out of service for maintenance or reconfiguration. **However, a single SSL module is limited to one ssl-proxy-list**.
2. If multiple SSL modules are implemented, sticky configuration might be needed.
    a. application ssl and advanced_balance ssl – Stickiness is provided based on the SSL session ID. Aside from the known IE SSL renegotiate (every 2 minutes) issue, the CSS also must proxy the connection searching for the SSL session ID, hence adds latency to the connection.
    b. srcIP sticky – This option works fine for sticking a given user connection to the same SSL module except that if the user's ISP uses Mega proxy, then you will have a group of user being stuck to the same SSL module, hence might swamp a particular SSL module.
3. Ciphers for ssl-server and backend-server do not have to be the same. One may choose to use stronger encryption on the ssl-server (front end to the user) for better security protection, and weaker encryption on the backend-server (between the CSS and the real servers) for better performance. Note: The CSS will utilize "session key re-use" for increased performance for backend SSL sessions.
4. The "ssl-server urlrewrite" configuration under ssl-proxy-list is only necessary if you are terminating SSL at the CSS and would like to prevent a clear text redirect issued by the backend HTTP server. For more information on URL Rewrite on CSS11500 see the document titled "URL Rewrite on the CSS11500".

Configuration Variations

Here is a scenario that there are different server groups that need SSL support on the CSS would each has a different public VIP to represent the group, and each has different cipher requirement. In the following configuration sample, a single SSL module / ssl-proxy-list with multiple ssl-server configurations will fulfil the above requirement.

Note: In this example, since different encryption ciphers are configured for the two VIP / web sites, the same user browser would not be able to access both sites at the same time.

```
!********************** SSL PROXY LIST **********************
ssl-proxy-list spl
  ssl-server 100
  ssl-server 100 rsakey mykeypair1
  ssl-server 100 rsacert mycert1
```

```
 ssl-server 100 vip address 10.87.99.78
 ssl-server 100  port 443
 ssl-server 100 cipher rsa-with-rc4-128-md5 192.191.134.165 80
 ssl-server 200
 ssl-server 200 raskey mykeypair2
 ssl-server  200 rsacert mycert2
 ssl-server 200 vip address 10.87.99.80
 ssl-server 200 port 443
 ssl-server 200 cipher rsa-with-rc4-128-sha  192.191.134.170 80
 backend-server 10
 backend-server 10 type backend-ssl
 backend-server 10 ip address 192.191.134.170
 backend-server 10 port 80
 backend-server 10 server-ip 192.191.134.170
 backend-server 10 server-port 443
 backend-server 10 cipher rsa-with-rc4-128-md5
 backend-server 20
 backend-server 20 type backend-ssl
 backend-server 20 ip address 192.191.134.171
 backend-server 20 port 80
 backend-server 20 server-ip 192.191.134.171
 backend-server 20 server-port 443
 backend-server 20 cipher rsa-with-rc4-128-md5
 active

!***************** SERVICE *****************
service bessl-svc1
 type ssl-accel-backend
 ip address 192.191.134.170
 port 80
 keepalive type ssl
 keepalive port 443
 add ssl-proxy-list spl
active

service bessl-svc2
 type ssl-accel-backend
 ip address 192.191.134.171
 port 80
 keepalive type ssl
 keepalive port 443
 add ssl-proxy-list spl
```

```
  active

service sslmod
  type ssl-accel
  keepalive type none
  slot 2
  add ssl-proxy-list spl
  active

!****************** OWNER ******************
owner cisco

  content HTTP-VIP-1
    protocol tcp
    vip address 192.191.134.165
    port 80
    add service bessl-svc1
    active

   content HTTP-VIP-2
    protocol tcp
    vip address 192.191.134.170
    port 80
    add service bessl-svc2
   active

  content SSL-VIP-1
    vip address 10.87.99.78
    protocol tcp
    port 443
    add service sslmod
    active

  content SSL-VIP-2
    vip address 10.87.99.80
    protocol tcp
    port 443
    add service sslmod
    active
```

Making Changes to SSL Configuration without Disrupting Services

Problem – If you are running a WebNS code prior to 7.5, when a change needs to be implemented on any ssl operation, the following sequence is normally followed.

- Suspend the ssl-proxy-list
- Make changes in the ssl-proxy list
- Suspend all services associated with that ssl-proxy-list: ssl modules and back end servers
- Activate the ssl-proxy-list
- Activate all services associated with the ssl-proxy-list that were suspended in the previous step

This is not only complicated and error prune, but also interrupts all SSL services. Thus any SSL changes causes a hit to the production network.

Following is a procedure that would eliminate the interruption of services, while making SSL changes.

The first recommendation is to create two separate ssl-proxy-lists – One for backend and one for front end servers. Although this is not required when you are making backend-only changes, it is recommended for a cleaner operation procedure because front-end and back-end server configuration have different change requirement. The reasons will become obvious later on.

The following was tested on CSS 11503 with two SSL modules running 7.20 build 203.

Two Windows NT servers with HTTPs daemon running.

A test tool that is capable of opening continunuous HTTPs connections.

Ethereal sniffer.

Server1:	10.10.20.111
Server2:	10.10.20.222
Secure VIP:	172.18.87.100
CSS client vlan:	172.18.87.98
CSS server vlan:	10.10.20.1

Base Configuration

```
!********************** SSL PROXY LIST **********************

ssl-proxy-list frontend Note that this proxy list has only front end servers
 ssl-server 100
 ssl-server 100 rsakey zakrsa
 ssl-server 100 rsacert zak-cert-ass
 ssl-server 100 vip address 172.18.87.100
 ssl-server 100 cipher rsa-with-rc4-128-md5 172.18.87.100 9999
 active
!
ssl-proxy-list backend          Note that this proxy-list has only backend servers
 backend-server 111
 backend-server 111 ip address 10.10.20.111
 backend-server 111 server-ip 10.10.20.111
 backend-server 111 cipher rsa-with-rc4-128-md5
 backend-server 111 port 85
 backend-server 222
 backend-server 222 ip address 10.10.20.222
 backend-server 222 port 85
 backend-server 222 server-ip 10.10.20.222
 backend-server 222 cipher rsa-with-rc4-128-md5
 active

*************************** SERVICE ***************************
service backendssl-scv1
 type ssl-accel-backend
 ip address 10.10.20.111
 port 85
 keepalive type ssl
 keepalive port 443
 add ssl-proxy-list backend
 active

 service backendssl-scv2
 ip address 10.10.20.222
 type ssl-accel-backend
 keepalive port 443
 keepalive type ssl
 port 85
 add ssl-proxy-list backend
```

```
  active

service ssl2
  slot 2
  type ssl-accel
  keepalive type none
  add ssl-proxy-list frontend
  active

service ssl3
  type ssl-accel
  keepalive type none
  slot 3
  add ssl-proxy-list frontend
  active

!**************************** OWNER ****************************
owner www.testbackendssl.com

  content HTTP-VIP
    vip address 172.18.87.100
    protocol tcp
    port 9999
    add service backendssl-scv1
    add service backendssl-scv2
    active

  content SSL-VIP
    vip address 172.18.87.100
    port 443
    protocol tcp
    add service ssl3
    add service ssl2
    active
```

Backend SSL Change Procedure:

1. Backend Changes:
   First, create a duplicate backend SPL adding or deleting the required changes. In this case we have taken out cipher rsa-with-rc4-128-md5 and added cipher rsa-with-rc4-128-sha, and the backend server port has been changed from 85 to 86. Note: this port is used only internal to the CSS to handle clear text traffic.

   Second, duplicate all backend services that are associated with the SPL, with a different port, in this case, port 86. Associate the service with the new backend SPL.

Last, add these newly created services to the content rule that handles the clear text HTTP load balance.

```
ssl-proxy-list backend2
  backend-server 111
  backend-server 111 ip address 10.10.20.111
  backend-server 111 server-ip 10.10.20.111
  ~~backend-server 111 cipher rsa-with-rc4-128-md5~~ deletion
  backend-server 111 cipher rsa-with-rc4-128-sha addition
  ~~backend-server 111 port 85~~ deletion
  backend-server 111 port 86 addition
  backend-server 222
  backend-server 222 ip address 10.10.20.222
  ~~backend-server 222 port 85~~ deletion
  backend-server 222 port 86 addition
  backend-server 222 server-ip 10.10.20.222
  ~~backend-server 222 cipher rsa-with-rc4-128-md5~~ deletion
  backend-server 222 cipher rsa-with-rc4-128-sha addition
  active

service backendssl-scv1-bak
  type ssl-accel-backend
  ip address 10.10.20.111
  port 86 Note the port number – It corrosponds to the new proxy list setting
  keepalive type ssl
  keepalive port 443
  add ssl-proxy-list backend2 Note the name of new SPL
  active

service backendssl-scv2-bak
  ip address 10.10.20.222
  port 86 Note the port number – It corrosponds to the new proxy list setting
  type ssl-accel-backend
  keepalive port 443
  keepalive type ssl
  add ssl-proxy-list backend2 Note the name of new SPL
  active

  content HTTP-VIP
    add service backendssl-scv1-bak addition
    add service backendssl-scv2-bak addition
```

As soon as you add the backend server services in the HTTP VIP, you will see the connections go through both backend SPLs. (port 85 and port 86). From the sniffer trace we checked that there are some backend connections that use the new cipher.

```
CSS11503-B# sh ssl flows

SSL Acceleration Flows for slot 2
      Virtual  Port TCP Proxy Flows  Active SSL Flows  SSL Flows in Handshake
--------------  ---- --------------  ----------------  ----------------------
 172.18.87.100  443         10              0                 0
  10.10.20.111  85           2              2                 0
  10.10.20.111  86           1              1                 0
  10.10.20.222  85           4              4                 0
  10.10.20.222  86           3              3                 0

SSL Acceleration Flows for slot 3
      Virtual  Port TCP Proxy Flows  Active SSL Flows  SSL Flows in Handshake
--------------  ---- --------------  ----------------  ----------------------
 172.18.87.100  443         10              0                 0
  10.10.20.111  85           3              3                 0
  10.10.20.111  86           4              4                 0
  10.10.20.222  85           1              1                 0
  10.10.20.222  86           2              2                 0
```

2. Gracefully shut down the original backend services: change the weight on the old backend services to zero.

```
service backendssl-scv1
 type ssl-accel-backend
 ip address 10.10.20.111
 port 85
 keepalive type ssl
 keepalive port 443
 add ssl-proxy-list backend
 weight 0
 active

service backendssl-scv2
 ip address 10.10.20.222
 type ssl-accel-backend
 keepalive port 443
 keepalive type ssl
 port 85
 add ssl-proxy-list backend
 weight 0
```

active

After some times there will be no flows on port 85.

#sh ssl flows

SSL Acceleration Flows for slot 2

Virtual	Port	TCP Proxy Flows	Active SSL Flows	SSL Flows in Handshake
172.18.87.100	443	11	1	0
10.10.20.111	85	0	0	0
10.10.20.111	86	0	0	0
10.10.20.222	85	0	0	0
10.10.20.222	86	12	11	0

SSL Acceleration Flows for slot 3

Virtual	Port	TCP Proxy Flows	Active SSL Flows	SSL Flows in Handshake
172.18.87.100	443	9	1	0
10.10.20.111	85	0	0	0
10.10.20.111	86	10	9	0
10.10.20.222	85	0	0	0
10.10.20.222	86	0	0	0

From the sniffer trace we made sure that new cipher is being used.

3. Now you can suspend all the old backend services and old SPL and subsequently delete them. Also you can remove the old backend services from the HTTP-VIP.

#Show ssl flows

SSL Acceleration Flows for slot 2

Virtual	Port	TCP Proxy Flows	Active SSL Flows	SSL Flows in Handshake
172.18.87.100	443	9	1	0
10.10.20.111	86	0	0	0
10.10.20.222	86	4	4	0

SSL Acceleration Flows for slot 3

Virtual	Port	TCP Proxy Flows	Active SSL Flows	SSL Flows in Handshake
172.18.87.100	443	11	0	0
10.10.20.111	86	4	3	0
10.10.20.222	86	0	0	0

Note that during all that process no changes are made in the front end services, including the SSL modules. This effectively means that we can make hitless backend SPL changes even with one SSL module.

Reason ------ The backend SPL is actually not directly associated to a particular SSL module. The module that is used for decryption is the one that is used for encryption.

When backend SSL configuration is being made, it is simply broadcasted out to all SSL modules. When there are multiple backend SPLs (and remember none of them is attached to any SSL module), any one of them can be used for backend encryption.

Front End Change Procedure

As we has stated earlier that a front end SPL is different from backend SPL in the sense that a frontend SPL is actually associated with a SSL module. For this reason, a hitless change would need more than one SSL module deployed in the chassis.

1. Creat a duplicate front end SPL with a new name, make the changes.

```
ssl-proxy-list frontend2  Note the new name
ssl-server 100
ssl-server 100 rsakey zakrsa
ssl-server 100 rsacert zak-cert-ass
ssl-server 100 vip address 172.18.87.100
~~ssl-server 100 cipher rsa-with-rc4-128-md5 172.18.87.100 9999~~ deleted
ssl-server 100 cipher rsa-with-rc4-128-sha 172.18.87.100 9999 added
active
```

Since this SPL has not been associated with any SSL modules yet, no traffic will hit this SPL.

Note that up to this point, traffic is going through both slots:

```
CSS11503-B#  sh ssl flows

SSL Acceleration Flows for slot 2
      Virtual  Port TCP Proxy Flows  Active SSL Flows  SSL Flows in Handshake
--------------  ---- --------------  ----------------  ----------------------
 172.18.87.100  443          11              0                 0
  10.10.20.111  86            1              1                 0
  10.10.20.222  86           10              9                 0

SSL Acceleration Flows for slot 3
      Virtual  Port TCP Proxy Flows  Active SSL Flows  SSL Flows in Handshake
--------------  ---- --------------  ----------------  ----------------------
 172.18.87.100  443          10              1                 0
  10.10.20.111  86            9              9                 0
  10.10.20.222  86            1              1                 0
```

2. Gracefully suspend one of the SSL module services by adding weight 0. We are doing this because then we can free up the slot and move the new changed SPL to new slot.
3. After suspend the module when no more traffic is going through it, change its SPL association, remove weight 0 and activate the service:

```
#service ssl3
    remove ssl-proxy-list frontend
    add ssl-proxy-list frontend2
    weight 1
    active
```

Now the traffic will be flowing through both slots. Slot 2 is using the old SPL and slot 3 is using new SPL, verified with sniffer trace.

4. Repeat step 2 and 3 above for the other SSL modules.

Now all traffic will be using the new frontend SPL with all new changes. Old SPL can now be suspended and deleted.

Client Authentication

SSL servers sometimes require a client authenticate itself before a data transfer can occur. When a SSL server that is requesting client authentication does not receive the requested client certificate, it may close the connection. In the case of the CSS SSL module implementation, it can either require and then authenticate a client from whom a request to establish a SSL termination tunnel is received (when it is acting as a SSL server to terminate client SSL traffic), or support the client authentication request that is sent from a backend SSL server to whom the SSL module is trying to establish a backend SSL tunnel (when it is acting as a SSL client to initiate or re-encrypt SSL traffic).

While client authentication is disabled by default when the SSL module is acting as a SSL server, and requires several configuration steps, it is enabled when the SSL module is acting as a client, as long as that the client certificates and keys are configured under a back-end server of a SSL proxy list.

The following samples are tested using WebNS 7.5 b4.

Configuration Points When CSS is Acting as SSL Server

1. Configure “authentication enable” for the ssl-server under the proxy list.
2. Adding the CA certificate to the ssl-server (maximum of 4 “cacert”s are allowed per proxy list). Any clients with a client certificate that is signed by a configured CA certificate will be trusted.
3. Optionally, configure CRL record under the SSL proxy list (maximum of 10 can be configured) and then assign it to a ssl-server (only one can be assigned per ssl-server). Notes – a). The HTTP request to obtain the CRL from the CA publishing site uses the ssl-server VIP as its source IP. So, this VIP needs to be an Internet registered IP. b). A CA certificate must be down loaded and associated before it can be configured as the signing certificate for the CRL record.
4. Three client authentication failure actions (configured under the ssl-server) can be configured: reject (default), ignore, and redirect (for which a “failure-url” needs to be defined).

Importing certificate on the client – Microsoft Internet Explore

1. Request a certificate from a Certificate Authority. (If it is for MS IE, it has to be in PKCS12 format.)
2. Tools -> Internet Options -> Content -> Certificates -> Import
3. Select the certificate and click « Advanced » to make sure the “Client Authentication” box is selected. Note: you need to have both the certificate (which contains your public key) and private key. A certificate of “.pxf” extension has both the public key and private key in it.

Importing and associating CA root certificate on the CSS

(config)# copy ssl ftp <ftp-record> import <cert file name> <file type> “password”

(config)# ssl associate cert <cert name> <cert file name from above>

Note: If the certificate was imported to a wrong file type, then the association will fail.

CSS Configuration

```
!************************** GLOBAL ***************************
 ssl associate rsakey zakrsa zakrsakeypair
 ssl associate cert zak-cert-ass zak-cert
 ssl associate cert thawteca thawteca.cer

 ip route 0.0.0.0 0.0.0.0 172.18.87.97 1

!************************** INTERFACE ************************
interface  1/1
 bridge vlan 100

interface  1/2
 bridge vlan 50

!************************** CIRCUIT **************************
circuit VLAN50

 ip address 172.18.87.98 255.255.255.240

circuit VLAN100

 ip address 10.10.20.1 255.255.255.0

!************************ SSL PROXY LIST *********************
ssl-proxy-list client-auth
 ssl-server 20
 ssl-server 20 rsakey zakrsa
 ssl-server 20 rsacert zak-cert-ass
 ssl-server 20 cipher rsa-with-rc4-128-sha 172.18.87.101 9999
 ssl-server 20 vip address 172.18.87.101
 ssl-server 20 authentication enable
 ssl-server 20 failure ignore
 ssl-server 20 cacert thawteca
 active

!************************** SERVICE **************************
service HTTP-81
 ip address 10.10.20.111
```

```
  port 81
  active

service ssl2
  slot 2
  type ssl-accel
  keepalive type none
  add ssl-proxy-list client-auth
  active

!**************************** OWNER ****************************

owner www.testbackendssl.com

    content client-auth-frontend
    add service ssl2
    protocol tcp
    port 443
    vip address 172.18.87.101
    active

  content client-auth-backend
    vip address 172.18.87.101
    port 9999
    protocol tcp
    add service HTTP-81
    active
```

Configuration Points When Acting as SSL Client

The client certificate and key needs to be associated and added to the proxy list.

Configure Client Authentication on IIS Server

Request client authentication: Under the IIS web site “Properties” -> “Directory Security” -> “Secure Communications” -> “Edit” -> Select “Require Secure Channel” and then select “Require Client Certificates”

Import the Root CA certificate: First import the Root CA onto the Server. Then, under the IIS web site “Properties” -> “Directory Security” -> “Secure Communication” -> Select “Enable Certificate Trust List”. Click “New”, and go through the wizard to add the trusted Root CA certificate.

Configure Client Authentication on Apache Server

In “httpd.conf”, the following needs to be added:

```
SSLVerifyClient require
SSLVerifyDepth 1
SSLCACertificateFile conf/ssl.crt/ca.crt
```

where the CA certificate should be the one to sign the client certificates. For more details, please visit the Apache site:

http://www.mathcs.bethel.edu/manual/ssl/ssl_howto.html

CSS Configuration

```
!************************** GLOBAL ***************************
 ssl associate rsakey zakrsa zakrsakeypair
 ssl associate cert zak-cert-ass zak-cert
 ssl associate cert TCA tca.pem
 ssl associate rsakey clientrsa clientkey
 ssl associate cert clientcert clientcert.cer

 ip route 0.0.0.0 0.0.0.0 172.18.87.97 1
!************************** INTERFACE ************************
interface  3/1
 phy 100Mbits-FD
 bridge vlan 100

interface  3/2
 phy 100Mbits-FD
 bridge vlan 50

!************************** CIRCUIT **************************
circuit VLAN50

 ip address 172.18.87.98 255.255.255.240

circuit VLAN100

 ip address 10.10.20.1 255.255.255.0

!********************** SSL PROXY LIST ***********************
ssl-proxy-list backend
 backend-server 111
 backend-server 111 ip address 10.10.20.111
 backend-server 111 server-ip 10.10.20.111
 backend-server 111 cipher rsa-with-rc4-128-md5
 backend-server 111 port 85
 backend-server 111 rsakey clientrsa
 backend-server 111 rsacert clientcert
 active
```

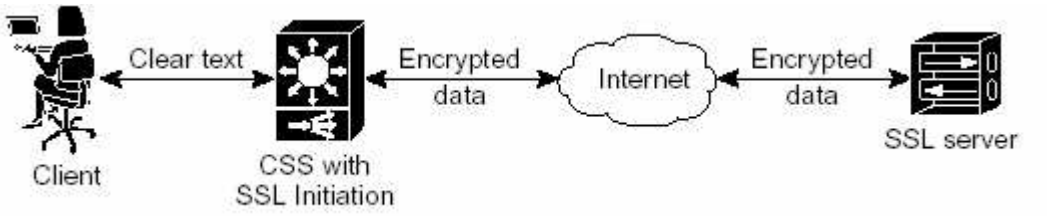
```
ssl-proxy-list client-auth
  ssl-server 20
  ssl-server 20 rsakey zakrsa
  ssl-server 20 rsacert zak-cert-ass
  ssl-server 20 cipher rsa-with-rc4-128-sha 172.18.87.101 9999
  ssl-server 20 vip address 172.18.87.101
  ssl-server 20 authentication enable
  ssl-server 20 failure ignore
  ssl-server 20 cacert TCA
  active

!************************** SERVICE **************************
service backendssl-scv1
  type ssl-accel-backend
  ip address 10.10.20.111
  port 85
  keepalive port 443
  add ssl-proxy-list backend
  keepalive type tcp
  active

service ssl2
  slot 2
  type ssl-accel
  keepalive type none
  add ssl-proxy-list client-auth
  active

!************************** OWNER ***************************
owner www.testbackendssl.com
  content client-auth-frontend
    add service ssl2
    protocol tcp
    port 443
    vip address 172.18.87.101

  content client-auth-backend
    vip address 172.18.87.101
    port 9999
    protocol tcp
    add service backendssl-scv1
```

active

SSL Initiation

SSL initiation, which was implemented in WebNS 7.40, allows the client to send clear text traffic to the CSS, and then originate a SSL session, with either a real backend SSL server, or another SSL module. Return traffic will be decrypted and sent back to the client in clear text. SSL initiation is also used for sending encrypted HTTP keepalives. (See next section for details.)

When is this needed – Customer who has intranet users trying to access a secure server located on the extranet, or a secure server that is located at another location which has to be accessed via an unsecured link would like to implement this feature.



Configuration Points

1. The same proxy list that's used for SSL termination or re-encryption may be used for SSL initiation. However, a separate proxy list would offer cleaner and simpler operational support, and will be used as examples in this document.
2. Configure the proxy list with a backend server, of type "initiation". – Note, by default, the type is "backend-ssl", which is for backend SSL re-encryption.
3. The "backend-server port" needs to match that of the service configuration, and the "backend-server server port" needs to match the port that the real server is listening on.
4. A content rule that is listening for the clear text traffic is configured, with services of type "ssl-init" added to the content rule. – Note, the VIP here is not tied into any SSL proxy configuration.
5. Configure the rest of the SSL cipher and session parameters: version, session cache time out, and renegotiation, etc.
6. If client authentication is required by the backend SSL server, then configure the RSA (or DSA) certificates and keys (that were previously imported and associated) under the backend-server.
7. Optionally, a CA certificate can be configured to authenticate the SSL server certificate, with the "backend-server <#> cacert <CA certificate association name>" configuration.

The following sample is tested using WebNS 7.5 b4.

CSS Configuration

```
!**************************** GLOBAL ****************************
  ip route 0.0.0.0 0.0.0.0 172.18.87.97 1

!************************** INTERFACE **************************
        interface  3/1
          phy 100Mbits-FD
          bridge vlan 100

        interface  3/2
          phy 100Mbits-FD
          bridge vlan 50

        !************************** CIRCUIT **************************
        circuit VLAN100

          ip address 10.10.20.1 255.255.255.0

        circuit VLAN50

          ip address 172.18.87.98 255.255.255.240

        !************************ SSL PROXY LIST ************************
        ssl-proxy-list ssl-init
          backend-server 111
          backend-server 111 ip address 10.10.20.111
          backend-server 111 server-ip 10.10.20.111
          backend-server 111 type initiation
          backend-server 111 cipher rsa-with-rc4-128-md5
          active

        !************************** SERVICE **************************
          service ssl-init
          ip address 10.10.20.111
          type ssl-init
          add ssl-proxy-list ssl-init
          slot 2
          active

        service ssl2
          slot 2
```

```
  type ssl-accel
  keepalive type none
  add ssl-proxy-list ssl-init
  active
!************************** OWNER ***************************
owner www.testbackendssl.com

  content ssl-init
    vip address 172.18.87.101
    protocol tcp
    add service ssl-init
    port 80
    active
```

SSL Keepalive

Introduced in WebNS 7.5, the encrypted HTTP keepalive offers verification of a full SSL hand shake with a HTTP GET or HTTP HEAD data returned from the back end secure server. It can be used with backend SSL acceleration (re-encryption), or SSL initiation to the backend secure servers.

SSL keepalive is configured with the backend-server in a ssl-proxy-list. As with all backend-server configurations, it is broadcast to all SSL modules. It picks a module to send the encrypted keepalive, until the keepalive fails, and then it will pick another module, providing there are multiple SSL modules configured.

Cisco documentation provides pretty good SSL keepalive samples for sending encrypted HTTP keepalive to servers that are either configured as backend acceleration servers, or backend SSL initiation servers. The samples provided below, however, are for customers that do not want to terminate the user SSL traffic at the CSS, but would like to provide a full SSL handshake and return data verification on these back end servers.

Configuration with Global Keepalive

In this example, the encrypted keepalive is sent under service “sslkal2” which has a SSL module slot and ssl-proxy-list configured under. The status of this service is tied to the Global named keepalive “sslkal”, which in turn ties to the status of the real backend server “realserver”.

Note: for each backend server that needs to have encrypted keepalive sent to, one Global Keepalive and one type ssl-init service need be configured, in addition to the real server, and a ssl-proxy-list. Also, the http method and uri can also be configured under the Global Keepalive “sslkal”, instead of the service “sslkal2”.

```
!************************** GLOBAL ***************************
  ip route 0.0.0.0 0.0.0.0 172.18.87.97 1

!************************* INTERFACE *************************
interface  3/1
phy 100Mbits-FD
bridge vlan 100

interface  3/2
phy 100Mbits-FD
bridge vlan 50

!************************* CIRCUIT **************************
circuit VLAN50

 ip address 172.18.87.98 255.255.255.240

circuit VLAN100
```

```
 ip address 10.10.20.1 255.255.255.0

!************************* KEEPALIVE *************************
! The global keepalive that will be shared by the SSL keepalive service, and also the service that's under the content rule.
keepalive sslkal
ip address 10.10.20.111
type http encrypt
active

!********************** SSL PROXY LIST **********************

! The ssl proxy list that used for the SSL keepalive. Multiple backend servers can be configured here.
ssl-proxy-list sslkal
backend-server 10
backend-server 10 ip address 10.10.20.111
backend-server 10 server-ip 10.10.20.111
backend-server 10 type initiation
active


!************************** SERVICE **************************

! It is under this service configuration that the SSL keepalive is sent. "ssl-init" tells the !SCM that the CSS needs to initiate SSL

! and needs to specify which proxy list and SSL module to use for the SSL keepalive !and which URI to send the HTTPS GET.
service sslkal2

 ip address 10.10.20.111
 type ssl-init
 keepalive type named sslkal
 slot 2
 add ssl-proxy-list kal
 keepalive method get
 keepalive uri "/"
 active

! This is the SSL module that will be used for sending the SSL keepalive
service ssl2
slot 2
type ssl-accel
keepalive type none
add ssl-proxy-list sslkal
active


! This is the service that will be used under the content rule.
service realserver
ip address 10.10.20.111
port 443
keepalive type named sslkal
active

!************************** OWNER ***************************
owner www.testbackendssl.com

 content ssltraffic
  protocol tcp
  vip address 172.18.87.101
```

```
add service realserver
port 443
active
```

Configuration with Scripted Keepalive

In the following sample, a scripted keepalive is configured to bring down the real server under the content rule, when the type ssl-init service (under which the encrypted keepalive is sent) is down. The customized script is also attached.

Note: customized script is not supported by Cisco TAC. The script needs to be ftp'ed to the CSS, under the /script directory.

```
!**************************** GLOBAL ****************************
   ip route 0.0.0.0 0.0.0.0 172.18.87.97 1

!************************** INTERFACE **************************
interface  3/1
 phy 100Mbits-FD
 bridge vlan 100

interface  3/2
 phy 100Mbits-FD
 bridge vlan 50

!************************** CIRCUIT **************************
circuit VLAN50

 ip address 172.18.87.98 255.255.255.240

circuit VLAN100

 ip address 10.10.20.1 255.255.255.0


!************************ SSL PROXY LIST ************************

ssl-proxy-list kal
 backend-server 111
 backend-server 111 ip address 10.10.20.111
 backend-server 111 server-ip 10.10.20.111
 backend-server 111 type initiation
 active

!************************** SERVICE **************************
```

```
service ssl2
  slot 2
  type ssl-accel
  keepalive type none
  add ssl-proxy-list kal
  active

service sslkal
  ip address 10.10.20.111
  protocol tcp
  keepalive method get
  keepalive type http encrypt
  add ssl-proxy-list kal
  keepalive uri "/"
  keepalive frequency 10
  keepalive retryperiod 30
  port 80
  type ssl-init
  slot 2
  active

service sslserver
  ip address 10.10.20.111
  port 443
  keepalive frequency 10
  keepalive retryperiod 30
  keepalive type script kal-service-check "sslkal" use-output
  active

!**************************** OWNER ****************************

owner www.testbackendssl.com

  content ssl
    vip address 172.18.87.101
    protocol tcp
    port 443
    add service sslserver
    active
```

kal-service-check script:

```
!no echo
```

```
!
!Script : ap-kal-service-check
!Author : yixue@cisco.com
!
!Version 1.0    April 27th 2005
!
!Checks the status of a particular service.
!
!Usage: ap-kal-service-check "service" use-output
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

set service "${ARGS}[1]"



show service ${service} | grep Alive

if STATUS "==" "1"
echo "Alive"
endbranch

show service ${service} | grep Alive
if STATUS "==" "0"
echo "Down"
exit script 1
endbranch

echo "Exit"
exit script 0
```

SSL Public / Private Key Pair and Digital Certificate Generation

There will be one SSL certificate/key pair on the CSS for the front end communication, which is for the website's domain name in the content rule, which will be returned to the client for each connection to the HTTPS content rule. When the CSS connects to the backend HTTPS server, it will then receive a certificate from the server, for the back end SSL communication. So, there will be one certificate installed on the CSS per ssl-server, and one certificate received per backend server, for all clients. Note: the backend servers could all share the same certificate, or each has its own certificate.

Client: SSL ClientHello → CSS

CSS: ssl-server SSL Cert → Client

CSS: SSL ClientHello → Server

Server: SSL Cert → CSS

Encryption keys and certificates can be either imported to the CSS, or generated on the CSS and be exported. The import and export uses sftp. Details are documented at:

http://www.cisco.com/univercd/cc/td/doc/product/webscale/css/css_720/advcggd/ssl.htm

Note: the only way to retrieve / store the keys and certificates is through the **copy ssl sftp** … **import** and **export** CSS commands. Any other method will corrupt the CertStore library and cause existing keys / certificates to be lost.

This section will only cover the steps needed to generate the public/private key pair and digital certificate on the CSS.

Here are the steps involved:

1. Generate the keys (RSA, DSA, and DH) - ssl genrsa/gendsa/gendh **mykeyfile** *keylength "password"*. The generated keys will be output in a file called **mykeyfile**.
2. Generate the CSR -ssl gencsr **mykeyfile**. The output will be displayed on the CSS. This step is only needed if you are requesting a Certificate Authority (CA) for issuing the certificate.
3. Associate private key name with the key file - ssl associate rsakey/dsakey/dhparam **mykey mykeyfile**.
4. Generate a self-signed certificate (skip this step if you are using a CA) - ssl gencert certkey **mykey** signkey **mysignkey mycertfile** *"password"*. The generated certificate will be stored in a file called **mycertfile**. If you are generating a RSA certificate, you may choose to use either the RSA key or a DH key for the key exchange (the "certkey"), but you have to use a RSA as the signkey. For DSA certificates, you have to use the DSA key for signkey and DH key for certkey.

Associate the certificate name with the certificate file - ssl associate cert **mycert mycertfile**

Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

European Headquarters
Cisco Systems Europe
11 Rue Camille Desmoulins
92782 Issy-Les-Moulineaux
Cedex 9
France
www-europe.cisco.com
Tel: 33 1 58 04 60 00
Fax: 33 1 58 04 61 00

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-7660
Fax: 408 527-0883

Asia Pacific Headquarters
Cisco Systems Australia, Pty., Ltd
Level 9, 80 Pacific Highway
P.O. Box 469
North Sydney
NSW 2060 Australia
www.cisco.com
Tel: +61 2 8448 7100
Fax: +61 2 9957 4350

Cisco Systems has more than 200 offices in the following countries and regions. Addresses, phone numbers, and fax numbers are listed on the

Cisco Web site at www.cisco.com/go/offices.

Argentina • Australia • Austria • Belgium • Brazil • Bulgaria • Canada • Chile • China • Colombia • Costa Rica • Croatia • Czech Republic Denmark • Dubai, UAE
Finland • France • Germany • Greece • Hong Kong SAR • Hungary • India • Indonesia • Ireland • Israel • Italy • Japan • Korea • Luxembourg • Malaysia • Mexico
The Netherlands • New Zealand • Norway • Peru • Philippines • Poland • Portugal • Puerto Rico • Romania • Russia • Saudi Arabia • Singapore • Slovakia • Slovenia
South Africa • Spain • Sweden • Switzerland • Taiwan • Thailand • Turkey • Ukraine • United Kingdom • United States • Venezuela • Vietnam • Zimbabwe