Contents

- 1 Goal
- 2 How SSL Stickiness Works
- 3 Limitations of SSL stickiness
- 4 Better Alternatives to SSL stickiness
- 5 Sample SSL stickiness Configuration
- 6 Comments
- 7 show running-config
- 8 Related Information

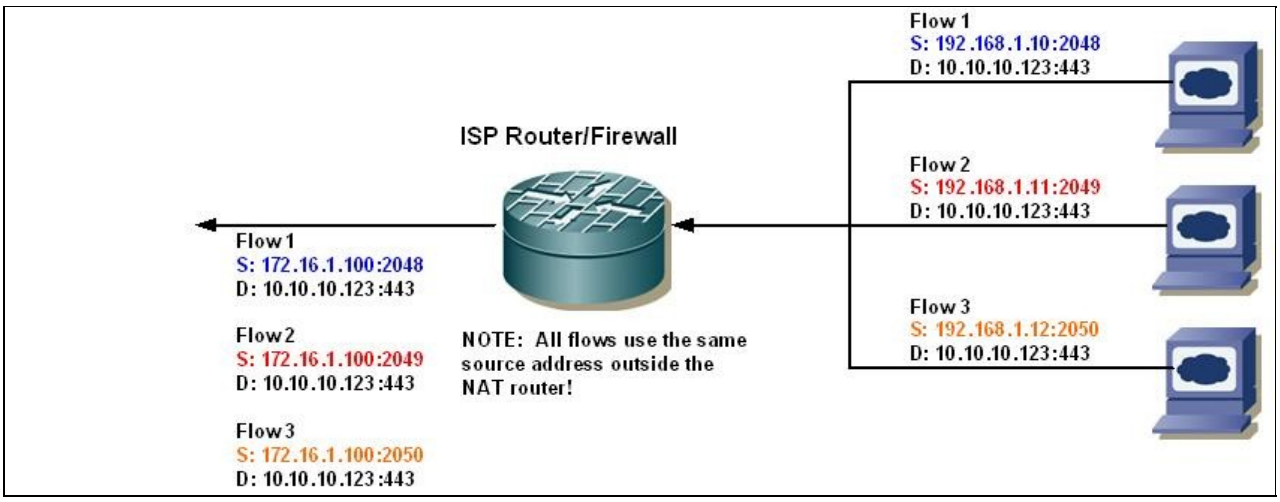
Goal

This document explains why Secure Sockets Layer (SSL) persistence (stickiness) has historically been widely used, why it is no longer a recommended configuration, and how to configure it on the Cisco® ACE Application Control Engine Module if SSL stickiness is absolutely required in spite of its limitations.

How SSL Stickiness Works

SSL was developed to provide a mechanism to encrypt data sent between a client and a server. It is often used to encrypt HTTP traffic, providing a secure mechanism for users to submit credit card information to e-commerce sites and to allow for secure transmission of other sensitive information. When a client initiates an SSL (HTTPS) connection to a web server, the server provides an SSL session ID to the client (a new ID or a resumption of an existing ID). This session ID is passed to the client in clear text, making it visible to network devices and easily readable in packet captures. When the same client initiates subsequent connections to the web server, it supplies the SSL session ID to identify itself. SSL stickiness on any load balancer (such as a Cisco Content Services Switch [CSS] or Cisco Services Module) functions by inspecting the SSL session ID as it is being set by the server. The load balancer creates a sticky entry tying a specific SSL session ID to the specific real server that was chosen by the load-balancing algorithm. By doing this, the load balancer is able to recognize a known SSL session ID in future connections and direct the flow to the correct real server.

Why Many Customers Use SSL Stickiness Historically, when deploying a secure (HTTPS) web application using a load balancer, SSL stickiness has been a popular method for providing client persistence. Sticking based on client IP does not accurately identify a specific user, since Network Address Translation (NAT) is commonly used to reduce the usage of IPv4 addresses on the Internet. NAT causes multiple clients to use the same source IP address, leaving the load balancer with no way of distinguishing them from one another when sticking based on client IP. This behavior is commonly known as the mega-proxy problem, as it creates the possibility that thousands of users will be identified by the load balancer as a single user. Sticking them all to a single real server could quickly oversubscribe its resources. The following figure illustrates the mega-proxy problem.



Traditionally, load balancers have not provided the ability to decrypt (terminate) SSL flows, which means there was no way for a load balancer to inspect the encrypted HTTP data as it passed through. This eliminated the possibility of using the load balancer to stick based on an HTTP cookie, since the cookie is hidden within the encrypted data. These limitations left SSL session ID stickiness as the only suitable choice for many end users. Even after migrating to newer equipment, many end users have not changed their sticky configuration and may not be aware of the limitations of SSL stickiness.

Limitations of SSL stickiness

Although web browsers implement their own suite of SSL capabilities, the behavior differs slightly from browser to browser. The way a browser handles SSL session IDs can have a detrimental effect on the usefulness of SSL stickiness. For example, Internet Explorer version 5.x will force a renegotiation of the SSL session ID every two minutes. Each time this occurs, the client reconnects using a blank SSL session ID, leaving the load balancer with no ability to identify the client and potentially causing the flow to be directed to a different real server. This means that, in effect, users of Internet Explorer 5.x can expect to be reliably ?stuck? to a server for only two minutes. This behavior is documented in the Microsoft knowledgebase.

- Internet Explorer Renegotiates Secure Sockets Layer Connection Every Two Minutes:

http://support.microsoft.com/kb/265369

SSL session handling in Internet Explorer 6 and 7 has been improved so that they retain the SSL session ID for at least an hour using their default configurations. There are, however, other actions that can cause Internet Explorer 6 and 7 to force a renegotiation. Clicking on embedded links that generate pop-up windows before the parent page is fully loaded can force a renegotiation. Users have also reported similar behavior when clicking on links that open new tabs within the browser. This behavior is also documented in the Microsoft knowledgebase.

- You experience connection problems when you use Internet Explorer to browse through pages on a secure Web site:

http://support.microsoft.com/default.aspx/kb/937480

Internet Explorer?s inconsistent behavior serves as a warning that SSL stickiness relies on obedient behavior by the client browser to function properly. Since application developers and network administrators cannot guarantee that all clients are running a specific browser or version, and since servers can also become the culprits in forcing session ID renegotiations, SSL stickiness is generally deemed unreliable.

Better Alternatives to SSL stickiness

Clearly the developers of SSL implemented the session ID with good intentions, but due to browser behavior it has proven not to be a reliable means of identification. Since all other information that could be used to identify the client is encrypted within the SSL flow, it would be beneficial for a load balancer to decrypt the flow and inspect the underlying HTTP data. In the past few years nearly all load balancers have evolved to include either a built-in feature or an add-on option to decrypt SSL flows. This ability is typically referred to as SSL termination, as the load balancer becomes the final destination of the encrypted data.

By decrypting the SSL flow, the load balancer gains the ability to inspect the HTTP data as if it were a normal, nonencrypted flow. Most load balancers are able to identify a client either by looking for a cookie that was set by an application (such as JSESSIONID) or by inserting their own cookie into the response being sent to the client. By using a cookie insert, the Cisco ACE devices can inject a cookie into server responses, to help ensure client persistence to an application instance.

Sample SSL stickiness Configuration

Some customers may be reluctant to change to a different method due to a lack of awareness of the limitations of SSL stickiness or possibly due to a security policy prohibiting the decryption of SSL traffic on any device that is not the endpoint for communication. In such cases, the Cisco ACE can provide SSL stickiness capabilities, as the following steps explain.

Many products require a custom function to provide SSL stickiness for a virtual server. The Cisco ACE devices use generic protocol parsing (GPP) to search through the TCP connection, locate the SSL session ID, and provide client persistence.

First the Layer 3 class needs to be configured to accept connections on port 443, and the rservers need to be added to the server farm such that the Cisco ACE sends traffic to them on port 443 as well. Since the rservers will inherit the port configured on the service policy if they are configured without a port, this method will be used.

```
ACE-1/routed(config)# class-map match-all slb-vip
ACE-1/routed(config-cmap)# match virtual-address 172.16.1.105 tcp eq https

ACE-1/routed(config)# serverfarm host web
ACE-1/routed(config-sfarm-host-rs)# rserver lnx1
ACE-1/routed(config-sfarm-host-rs)# inservice
ACE-1/routed(config-sfarm-host-rs)# rserver lnx2
ACE-1/routed(config-sfarm-host-rs)# inservice
ACE-1/routed(config-sfarm-host-rs)# rserver lnx3
ACE-1/routed(config-sfarm-host-rs)# inservice
ACE-1/routed(config-sfarm-host-rs)# rserver lnx4
ACE-1/routed(config-sfarm-host-rs)# inservice
ACE-1/routed(config-sfarm-host-rs)# rserver lnx5
ACE-1/routed(config-sfarm-host-rs)# inservice
```

Next a sticky group is configured to parse for the SSL session ID. This allows the Cisco ACE to create a unique entry in its sticky table for each session ID it encounters. Within the TCP payload of a new SSL connection, the 44th byte is a 1-byte length field indicating how long the session ID will be. The length is typically 32 bytes, in which case the 32 bytes of data following the length field would be the session ID. The parser needs to be configured to skip the first 44 bytes and then read in the following 32 bytes as the session ID to be used in the sticky entry.

By default, the Cisco ACE parses only the TCP payload of data being sent from the client to the server

(request payload). Since the session ID is being set by the server, the ACE needs to be configured to parse both the client request and the server response. This is done by adding the ?response sticky? command. The resulting sticky configuration will look like the following.

```
ACE-1/routed(config)# sticky layer4-payload SSL_GROUP
ACE-1/routed(config-sticky-l4payloa)# serverfarm web
ACE-1/routed(config-sticky-l4payloa)# response sticky
ACE-1/routed(config-sticky-l4payloa)# layer4-payload offset 43 length 32 begin-pattern "(\x20|\x00
```

Note: The default timeout for sticky entries is 24 hours (1440 minutes). This can be changed by using the ?timeout? command within the sticky group created above. In order for the payload inspection to function, you need to set an appropriate max-parse-length. This is necessary to tell the parser how much data to inspect in the payload, and by default it is set to 4096 bytes. For this configuration it should be set to 70 bytes, both for performance and to ensure that sufficient TCP data is parsed to provide the desired session persistence. Setting this value larger than 70 bytes may result in page loads that appear to hang.

Note 2: we use "(\x20|\x00\xST)" as pattern instead of just "\x20" because short "client hello" packets can otherwise lead to the pattern not to match. This will result in a connection that will time out. The \xST pattern was introduced in versions were CSCsh04655/CSCte81287 was fixed.

```
ACE-1/routed(config)# parameter-map type generic sslidparam
ACE-1/routed(config-parammap-generi)# set max-parse-length 70
```

Next a Layer 7 policy must be defined that will send traffic to the new sticky group. The matching policy type for the layer4-payload sticky is ?loadbalance generic,? which indicates a policy configured to perform GPP.

```
ACE-1/routed(config)# policy-map type loadbalance generic first-match gppmatch
ACE-1/routed(config-pmap-lb-generic)# class class-default
ACE-1/routed(config-pmap-lb-generic-c)# sticky-serverfarm SSL_GROUP
```

The policy and parameter maps are then applied to the class reference within the multimatch policy.

```
ACE-1/routed(config)# policy-map multi-match client-vips
ACE-1/routed(config-pmap)# class slb-vip
ACE-1/routed(config-pmap-c)# loadbalance policy gppmatch
ACE-1/routed(config-pmap-c)# appl-parameter generic advanced-options sslidparam
ACE-1/routed(config-pmap-c)# loadbalance vip inservice
```

At this point the configuration is complete, and the Cisco ACE should be ready to inspect the SSL traffic.

Related show Commands

It is important to remember that the session ID itself is not inserted into the sticky table; a hash value is generated from it and is inserted into the sticky table. To test this, initiate an HTTPS connection to the VIP and use the following command to look for a new sticky table entry.

```
ACE-1/routed# sho sticky database group SSL_GROUP
sticky group : SSL_GROUP
type         : LAYER4-PAYLOAD
timeout      : 600           timeout-activeconns : FALSE
  sticky-entry             rserver-instance              time-to-expire flags
  ------------------------+----------------------------+--------------+-------+
  8340144415558738875   lnx1:0                     86394        -
```

Use the ?show service-policy? and ?show serverfarm detail? commands to verify that the ACE is maintaining persistence between the client and server application.

```
ACE-1/routed#  show service-policy detail

Policy-map : client-vips
Status     : ACTIVE
Description: -
----------------------------------------
Interface: vlan 20
  service-policy: client-vips
    class: slb-vip
     VIP Address:    Protocol:  Port:
     172.16.1.105    tcp        eq    443
      loadbalance:
        L7 loadbalance policy: gppmatch
        VIP Route Metric     : 77
        VIP Route Advertise  : DISABLED
        VIP ICMP Reply       : DISABLED
        VIP State: INSERVICE
        curr conns       : 0         , hit count        : 20
        dropped conns    : 0
        client pkt count : 1106      , client byte count: 1324188
        server pkt count : 343       , server byte count: 31885
        conn-rate-limit      : 0         , drop-count : 0
        bandwidth-rate-limit : 0         , drop-count : 0
        L7 Loadbalance policy : gppmatch
          class/match : class-default
            LB action :
               sticky group: SSL_GROUP
                  primary serverfarm: web
                    state: UP
                  backup serverfarm : -
            hit count        : 20
            dropped conns    : 0


ACE-1/routed# show serverfarm web detail
 serverfarm     : web, type: HOST
 total rservers : 5
 active rservers: 5
 description    : -
 state          : ACTIVE
 predictor      : ROUNDROBIN
 failaction     : -
 back-inservice    : 0
 partial-threshold : 0
 num times failover       : 0
 num times back inservice : 0
 total conn-dropcount : 0

 ------------------------------
                                          --------connections--------
       real              weight state       current   total    failures
   ---+----------------+------+-----------+---------+--------+--------
   rserver: lnx1
       192.168.1.11:0    8      OPERATIONAL 0         20       0
         max-conns            : -         , out-of-rotation count : -
         min-conns            : -
         conn-rate-limit      : -         , out-of-rotation count : -
         bandwidth-rate-limit : -         , out-of-rotation count : -
         retcode out-of-rotation count : -
         load value           : 0
```

```
   rserver: lnx2
       192.168.1.12:0    8      OPERATIONAL 0         0        0
         max-conns            : -         , out-of-rotation count : -
         min-conns            : -
         conn-rate-limit      : -         , out-of-rotation count : -
         bandwidth-rate-limit : -         , out-of-rotation count : -
         retcode out-of-rotation count : -
...skipping
```

While new connections are being inspected by GPP and looked up in the sticky database, they will appear as one-sided connections in the connection table, until a real server has been determined. These can be seen in a ?show conn? as dashed entry pairs.

```
ACE-1/routed# show conn

total current connections : 5
conn-id  np dir proto vlan source               destination        state
--------+--+---+-----+----+--------------------+------------------+------+
21       1  in  TCP   20   209.165.201.10:8431 172.16.1.105:443    ESTAB
7        1  out TCP   40   192.168.1.12:443    209.165.201.10:8431 ESTAB
15       2  in  TCP   20   209.165.201.23:5829 172.16.1.105:443    ESTAB
--       -  -   --    --   --                   --                 --
21       2  in  TCP   20   209.165.201.11:1786 172.16.1.105:443    ESTAB
7        2  out TCP   40   192.168.1.11:443    209.165.201.11:1786 ESTAB
```

Comments

- Although the Cisco ACE Module can provide effective SSL persistence based on session ID , SSL stickiness is not recommended due to the numerous limitations that can break client persistence. The recommended approach is to use SSL termination in conjunction with HTTP cookie stickiness to help ensure application persistence per unique user session.
- SSL stickiness relies on the use of servers that support session ID reuse. Some servers or appliances will send the client a blank session ID during the server hello, or will set a new session ID at the start of each TCP connection. These situations will cause SSL stickiness not to function. Additionally, when a server does not support session ID reuse, SSL performance will be degraded. Each SSL handshake is very CPU intensive on the server. By supporting session ID reuse, the server and client establish subsequent SSL connections without having to perform another SSL handshake.
- SSL stickiness is supported only on SSLv3 and TLSv1 sessions. The reason for this is that SSLv2 places the session ID within the encrypted data, making it impossible for the ACE or any other device to inspect it.

show running-config

```
logging enable

access-list everyone line 8 extended permit ip any any
access-list everyone line 16 extended permit icmp any any

parameter-map type generic sslidparam
  set max-parse-length 70

rserver host lnx1
  ip address 192.168.1.11
```

```
  inservice
rserver host lnx2
  ip address 192.168.1.12
  inservice
rserver host lnx3
  ip address 192.168.1.13
  inservice
rserver host lnx4
  ip address 192.168.1.14
  inservice
rserver host lnx5
  ip address 192.168.1.15
  inservice

serverfarm host web
  rserver lnx1
    inservice
  rserver lnx2
    inservice
  rserver lnx3
    inservice
  rserver lnx4
    inservice
  rserver lnx5
    inservice

sticky layer4-payload SSL_GROUP
  serverfarm web
  response sticky
  layer4-payload offset 43 length 32 begin-pattern  "(\x20|\x00\xST)"

class-map match-all slb-vip
  3 match virtual-address 172.16.1.105 tcp eq https

policy-map type management first-match remote-access
  class class-default
    permit

policy-map type loadbalance generic first-match gppmatch
  class class-default
    sticky-serverfarm SSL_GROUP

policy-map multi-match client-vips
  class slb-vip
    loadbalance vip inservice
    loadbalance policy gppmatch
    appl-parameter generic advanced-options sslidparam

interface vlan 20
  description Client Side
  ip address 172.16.1.5 255.255.255.0
  access-group input everyone
  service-policy input client-vips
  no shutdown
interface vlan 40
  description Default gateway of real servers
  ip address 192.168.1.1 255.255.255.0
  service-policy input remote-access
  no shutdown

ip route 0.0.0.0 0.0.0.0 172.16.1.1
```

Related Information

Technical Support & Documentation - Cisco Systems