SonicWALL
Advanced SSL Offloading
With Content Switches

INTRODUCTION: ........................................................................................................... 2
THE SSL OFFLOADER:................................................................................................. 2
METHODS OF DEPLOYMENT:.......................................................................................... 3

  IN-LINE:..................................................................................................................... 5
  TRANS-WICH:............................................................................................................. 8
  PROXY-MODE:........................................................................................................... 11
  TRANSPARENT-MODE: ............................................................................................... 15
  VLANs, Default Routes, and Traffic Flow ................................................................. 19

CONCLUSION: .............................................................................................................. 20
Introduction:

In recent years, switching technologies have evolved considerably beyond the domain of layer 2. First, layer 3 switching was made popular by offering much of the versatility of routing at the speed of switching. Then layer 4 switching—with such features as QoS and transparent redirection—grew in application among caching and load-balancing installations. Finally, content switches were introduced enabling smart switching decisions based upon layer 7, the payload of the packet. With the advent of content switches the much-needed ability to switch traffic by examining such critical data as cookies and URL’s was offered to sites.

The advanced traffic management capabilities offered by content switches are most likely to be employed by e-commerce sites, web-content providers, and web and application service providers. This same profile group is almost certain to be using HTTPS, or HTTP over Secure Sockets Layer (SSL) to secure some portion of their content. Historically, sites wishing to employ both SSL protected services, and the management capabilities of content switching were faced with a challenging situation: How do we perform content switching based on data content if the payload is encrypted, hiding it from analysis?

The SSL Offloader:

Most high traffic secure sites discovered long ago that the highly CPU intensive SSL handshake process (figure 1) dramatically reduces a server’s capacity to serve pages, and to perform other functions like CGI or other server-side scripting. In an effort to return to
non-encrypted performance levels, many sites began to employ SSL accelerators to handle the public-key cryptographic functions, or the RSA key-exchange. Although this dramatically reduced CPU utilization, it was not a content switch friendly solution because the traffic remained encrypted all the way to the server’s bus. The next generation SSL accelerators, also known as SSL Offloaders, took cryptographic computational assistance one step further by handling not only the RSA key-exchange, but also the bulk-data decryption and encryption, offering even greater performance benefits (figure 2). Generally available in appliance (SSL-IA) or rack-mount (SSL-R) form-factors, SSL Offloaders receive encrypted SSL traffic and transmit decrypted clear-text traffic, enabling them to restore the benefits of content switching to an SSL environment.

Methods Of Deployment:

(Note: The designator SSL-x will be used in the following configuration examples, and is meant to be representative of the functionally interchangeable SSL-IA or SSL-R form-factors.)

The heretofore-conventional method of deploying an SSL accelerator, a method supported by diverse and heterogeneous groupings of equipment, is the In-Line method. The In-Line method places the SSL accelerator at some point on the network between the router and the content switch. This point can be before or after any preliminary switching or firewalling occurs on the network. While the Sonicwall SSL-x products support this method, it is but one of four deployment methods available to the SSL-x family. The full list of deployment methods:

- In-Line
- Transparent Sandwich ("Trans-wich")
- Proxy-Mode
- Transparent-Mode

<table>
<thead>
<tr>
<th>Method</th>
<th>Difficulty</th>
<th>High Availability</th>
<th>Scalability</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-Line</td>
<td>Easy</td>
<td>Low</td>
<td>Poor</td>
<td>All traffic passes through the SSL-x</td>
</tr>
<tr>
<td>Trans-wich</td>
<td>Medium</td>
<td>Medium</td>
<td>Good</td>
<td>Requires min. 2 content switches</td>
</tr>
<tr>
<td>Proxy-Mode</td>
<td>Easy</td>
<td>High</td>
<td>Excellent</td>
<td>Lose client IP tracking capability</td>
</tr>
<tr>
<td>Transparent-Mode</td>
<td>Hard</td>
<td>High</td>
<td>Excellent</td>
<td>Most efficient design overall</td>
</tr>
</tbody>
</table>

The Proxy-Mode and the Transparent-Mode methods take advantage of Sonicwall’s distinctive One-Arm SSL Offloading architecture. Traditional methods of SSL Offloading required the use of two ports: one ingress port for encrypted traffic and one egress port for decrypted traffic. While this remains an acceptable requirement for networks using shared ports (hubs) or inexpensive switched ports (layer 2 switches,) it becomes less endurable in a content switch’s premium-port environment. Sensitive to the value of the content switched-port, Sonicwall engineered a method of SSL Offloading that allows for a single port to be used for both ingress and egress traffic, with no performance degradation.
**In-Line:**

The In-Line configuration (figure 3) is the simplest method to deploy because it requires no specific inter-operation configuration on either the SSL-x or the content switch. To maximize uptime, a high-availability option if offered. High-availability in an In-Line configuration is unique in that it is offered by the SSL-x, rather than by the content switch’s server-health-check mechanisms. The SSL-x offers a hot-standby mode wherein two SSL-x devices—one active, and one standby—are connected via a serial cable for the sake of heartbeat communications. Should the active SSL-x fail, the standby unit will identify the failure, and will assume the role of active SSL-x. When the failed unit resumes operation, it will do so in a standby mode.

Because the SSL-x front-ends the content switch, passing all traffic to the content switch in a decrypted fashion, no additional traffic-handling requirements beyond its normal configuration are necessary on the content switch.

![Diagram of In-Line Configuration with Hot-Standby Fail-over](image)

*Figure 3.*

*(In-Line Configuration with Hot-Standby Fail-over)*

Content switch configuration typically involves a Virtual IP address (VIP) for the virtual server group, and then a number of associated real-servers (back-end servers) that
compose the server farm. The client requesting the content, for example, HTTPS content, makes the request of the VIP address, and then the content switch intelligently distributes the traffic to the real-servers according to a balancing algorithm, or some other higher-level distribution facility.

All traffic in this scenario passes through the SSL-x in a transparent manner, with the exception of **SSL traffic**. The SSL-x family is capable of supporting most any TCP based protocol over SSL. Although the most popular application of SSL by far is HTTPS (TCP port 443) other protocols, such as SPOP3, SSMTP, and SSL-LDAP, to name a few, are also supported. Considering the broad protocol support of the Sonicwall SSL-x family, **SSL-Server** groups must be created through the Configuration Manager in order to designate SSL traffic.

SSL-Server groups are defined on the SSL-x by providing six pieces of information:

1. An SSL-Server name (a moniker for identification purposes)
2. An IP address with netmask
3. An SSL (ingress or listening) port
4. A remote (egress or clear-text) port
5. A key association
6. A security policy

After an SSL-Server group has been defined to listen for a particular IP address and SSL-port combination, the SSL-x will “intercept” only traffic destined for that combination, decrypt it, and forward it on to the IP address and Remoteport combination specified. Non-defined (non-SSL) traffic will be transparently bridged through the SSL-x. Return traffic will follow the reverse path, and will be encrypted prior to returning to the client. The actual packet flow is depicted in figure 4:
The following diagram is that of a simplified In-Line configuration, with sample addressing:

**Figure 5.**
Sample Network – In-Line Configuration
Trans-wich:

This configuration method employs a minimum of two content switches—one on either side of an SSL-x device or devices—to form an easily scalable SSL Offloading solution. This design introduces the concept of the upstream and the downstream content switch (figure 6). The following are the characteristics of the upstream content switch:

- It will be the content switch closest to the perimeter of the network
- It will require that one separate VLAN be configured for each sandwiched SSL-x.
- It will be required to balance and transparently redirect traffic received on a particular TCP Port (e.g. port 443) to the IP address of the complementary VLAN on the downstream content switch. All non-SSL traffic could be directed to the downstream content switch via an alternate or dedicated link.

High-availability is offered in this configuration via the content switch’s health-check mechanism; if one of the SSL-x devices, one of the complementary VLAN ports, or one of the real-server farms fails, the content switch should recognize its unavailability and remove it from its eligible redirection list.
The downstream content switch is configured in much the same fashion as is the single content switch in the In-Line configuration. Its only specific configuration requirement is the need for complementary VLANs. Through its VLAN definitions, it receives decrypted traffic redirected from the upstream content switch through the SSL-x. The actual packet flow is depicted in figure 7. The downstream content switch is also responsible for the Virtual IP address, i.e. the destination IP address to which clients will connect, and for the associated real-servers belonging to that VIP.

The SSL-x sits inline between the two content switches. When the upstream content switch receives TCP port 443 traffic, it redirects it to the IP address of the downstream content switch’s VLAN. During this redirection, the SSL-x intercepts traffic designated within its SSL-Server definition, decrypts it, and forwards the unencrypted traffic to the downstream content switch where it is balanced to the real-server farm. The return traffic flow is the effective reverse of this.
To illustrate the addressing of the Trans-wich configuration, consider the network depicted in figure 8. Six VLANs have been created on the downstream content switch, one for each of the SSL-x devices, one for an unencrypted traffic path (supporting standard HTTP over TCP port 80, for example), and one for server-farm’s real-servers. The upstream content switch will not require a VLAN for the real-servers, but would instead use a VLAN for its upstream router.

Each SSL-x should be configured with a default route of its VLAN’s upstream content switch (e.g. the SSL-x on VLAN2 would have a default route of 10.2.0.1). Additionally, each SSL-x should have a static route to the server-farm’s VIP via the corresponding VLAN IP of the downstream content switch (e.g. SSL-x on VLAN2 would have a route to 10.100.0.0/24 via 10.2.0.2, VLAN3 would have a route to 10.100.0.0/24 via 10.3.0.2, etc.).

The downstream content switch should be configured with multiple default routes pointing to the corresponding upstream content switch’s VLAN IP address. IP Load-Sharing, or Equal Cost Multi Path (ECMP) capabilities on the content switch will allow it to return the traffic along the same path through which it arrived. It achieves this by maintaining an IP forwarding cache, and preferring that cache information for traffic paths in the event of multiple available routes. It is this mechanism that ensures that traffic returning to an SSL client be sent back through the SSL-x through which it came so that it may be encrypted.
**Proxy-Mode:**

Proxy-Mode is relatively simple to configure and scales well, but has one caveat, namely, it prevents client IP recording by real-server access logs. The reason this occurs is because of the path the traffic flows in a Proxy-Mode configuration (figure 9), and because of the unique role the SSL-x plays in this design.

![Figure 9. (Proxy-Mode Configuration)](image)

Unlike the other methods of deployment, when used in Proxy-Mode the SSL-x does not act transparently. Instead, it acts—as the name implies—as a proxy between the requesting client and the back-end real-server. Because of this behavior, the real-server sees the access as coming from the IP address of the SSL-x, rather than from the originating client. While this effect is acceptable in many installations, there will be instances where client IP accounting accuracy will be required. In such cases, an alternate design method is encouraged.

Proxy-Mode is the first of two modes that employs Sonicwall’s One-Armed SSL Offloading. This design difference, combined with proxy rather than transparent operation results in a flow of traffic (figure 10) that is notably different from the others. Because only a single port will be used on the SSL-x, the introduction of bridge loops by the SSL-x will not be a concern. Additionally, because the redirection method will be
non-transparent, Equal Cost routing will not be employed when supporting multiple SSL devices, and separate VLANs for each attached SSL-x device will not be required.

Another unique property of Proxy-Mode is the need to configure separate IP address/TCP port combinations on the SSL-x devices for each VIP in use. This need arises because the redirection from the Content Switch is not transparent in Proxy-Mode, meaning the destination address is re-written by the Content Switch to that of the SSL-x. In the event of a single VIP configuration (e.g. Group1 listening on 10.2.2.100:443 and the complementary Real1 listening on 10.2.2.100:81 on the Content Switch) there are no unique requirements. If, however, multiple VIPs are desired, the SSL-x needs some way to distinguish between the different VIPs with some facility other than the re-written, non-unique IP address. We therefore configure the content switch to perform unique Port Address Translation on all VIPs subsequent to the first, and we also configure the
different SSL-Server Groups on the SSL-x to listen for SSL traffic on those unique ports. A sample addressing scheme, and a numbered traversal path is illustrated in figure 11.

1. Client (200.1.1.1) requests https://10.2.2.100:443
2. Content switch receives request on VIP belonging to Group1
3. Non-transparently forwards client request to SSL-x at 10.1.1.10 port 443
4. SSL-x decrypts SSL traffic, and forwards to 10.2.2.100 port 81
5. Content switch receives 10.2.2.100 port 81 traffic on VIP belonging to Real1, and forwards to group member 10.100.1.10 on port 81
6. Real server 10.100.1.10 sends reply back through the content switch. It forwarded back to the originator, namely the SSL-x, where it is encrypted and sent back out
7. The content switch forwards the packet to its upstream router which sends it to 200.1.1.1
If the client requested the second VIP, https://10.2.2.101:443, Port Address Translation would be employed, and the path would be as follows:

1. Client (200.1.1.1) requests https://10.2.2.101:443
2. Content switch receives request on VIP belonging to Group1
3. The Content Switch Port Address Translates the packet to 444. Non-transparently forwards client request to SSL-x at 10.1.1.10 port 444
4. SSL-x decrypts SSL traffic, and forwards to 10.2.2.101 port 81
5. Content switch receives 10.2.2.101 port 81 traffic on VIP belonging to Real1, and forwards to group member 10.100.1.12 on port 81
6. Real server 10.100.1.12 sends reply back through the content switch. It’s forwarded back to the originator, namely the SSL-x, where it is encrypted and sent back out
7. The content switch forwards the packet to its upstream router which sends it to 200.1.1.1
**Transparent-Mode:**

Offering a scalable, single content switch solution without the IP accounting limitations of the Proxy-Mode, Transparent-Mode is both the most complex, and the most feature-rich of all the configurations. Transparent-mode’s intricate packet flow requires a certain set of capabilities from the content switch. Specifically, the content switch must offer two features: flow switching (figure 12) and cache-device ACL support.

---

**The Routing Switch**
Routing calculations take place at layer 3 in hardware or software, while the actual packet processing takes place at layer 2. Similar in many respects to routers, the speed gain is accomplished by reducing feature overhead and moving as much logic as possible into hardware.

**The Flow switch**
Route calculation and packet processing takes place at layer 3 until a flow is detected. The flow is then switched at layer 2 through the network. Flow switching caches connection characteristics, including such information as source and destination IP address, source and destination TCP port, and ingress port for reference and route path consistency. It is this mechanism that enables reliable transparent mode operation with the SSL-x.

**The Switched router**
Route calculation and packet processing take place at layer 3. Tag switching relies on two principal components: forwarding and control. The forwarding component uses the tag information (tags) carried by packets and the tag-forwarding information maintained by a tag switch to perform packet forwarding. The control component is responsible for maintaining correct tag-forwarding information among a group of interconnected tag switches.

---

Figure 12.
(An Overview of Layer 3+ Switching Models)

Flow switching support is necessary in order to ensure that traffic passing transparently from the content switch to the SSL-x follows the following path:

Client → content switch → SSL-x → content switch → real-server → content switch → SSL-x → Client

While this path is comparatively easy to ensure in a non-transparent proxy implementation, it is somewhat more difficult to guarantee in a transparent One-Armed SSL Offloading model. This model can easily breakdown after traffic passes from the real-server back to the content switch. The default behavior of most switching devices at this point would be to pass the traffic directly back to the client rather than to the SSL-x. The reason it is absolutely essential to pass it back to the SSL-x is so that the traffic be encrypted prior to passing back to the client. More on how proper routing is guaranteed later.
The second requirement, cache-device ACL support, is also worthy of specific mention because of its unconventionality. Cache-device ACL support is a generic term for the handling of traffic in a transparent redirection scenario, and does not, in our application, refer to a cache device, but rather to an SSL-x device. This feature enables the content switch to apply access-controls to traffic returning from a transparent cache-device. These access-controls will be used to govern routing of traffic on the upstream router VLAN, and on the SSL-x device VLANs, as well as to control configuration manager access (TCP and UDP port 2932).

As figure 13 suggests, the physical layout of Transparent-Mode is very much like that of Proxy-Mode. Unlike Proxy-Mode, which does not require separate VLANs be defined for each SSL-x device, Transparent-Mode does require that each SSL-x be on its own VLAN. This requirement is part of the mechanism designed to ensure that real-server traffic be routed back through the correct SSL-x device before returning to the client.

The following diagram (figure 14) of packet-flow shows further similarities between the two modes, with the only distinguishable difference being the method of initial redirection to the SSL-x (redirection to IP address of SSL-x in Proxy-Mode versus transparent-redirection in Transparent-Mode).
A sample addressing scheme and enumerated traffic flow is provided in figure 15.
1. Client (200.1.1.1) requests https://10.20.20.100:443
2. Content switch receives request on VIP belonging to Group1
3. Content switch transparently forwards client request to 10.20.20.100 port 443. It is processed en-route by the SSL-x and output as port 81 clear-text
4. Content switch receives packet for 10.20.20.100 port 81 on VIP for group Real1
5. Forwards packet to group member 10.100.1.10 on port 81
6. Real server 10.100.1.10 sends reply back through the content switch. Although destined for 200.1.1.1, it is forced through the flow-involved SSL-x, where it is encrypted and sent back out
7. The content switch passes the data to its upstream router which sends it to 200.1.1.1
VLANs, Default Routes, and Traffic Flow

Referring again to figure 15, we can examine how proper traffic flow, specifically the encryption of return traffic, is achieved. Another common question this examination will answer is “why do we require separate VLANs for each attached SSL-x?”

When the client at 200.1.1.1 [1] requests a secure web-page from 10.20.20.100 [2], the flow switching content switch records the source IP address, source port, destination IP address, destination port and protocol, and associates this flow information with the physical port on the switch on which the initiating packet was received. Assuming the router on our sample network was attached to physical port 1 on the content switch, the flow information for the initial connection would like the following:

<table>
<thead>
<tr>
<th>Physical Port</th>
<th>Source IP</th>
<th>Source Port</th>
<th>Dest. IP</th>
<th>Dest. Port</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>200.1.1.1</td>
<td>63001</td>
<td>10.20.20.100</td>
<td>443</td>
<td>TCP</td>
</tr>
</tbody>
</table>

This traffic, in Transparent Mode, would then be redirected to the MAC address of the first SSL-x, for example [3], while maintaining the IP address 10.20.20.100. This is achieved by putting the packet back on the wire with the MAC address of the SSL-x, and the IP address 10.20.20.100. The SSL-x recognizes this packet, determines that it has a service defined for it, and engages in the TCP and SSL handshaking routines with the client. Following the completion of the SSL handshake, the SSL-x initiates a TCP session on port 81 back to the content switch [4]. This begins the second of the two flows that will be established during Transparent Mode operation. Assuming the SSL-x is on physical port 2 of the content switch, the flow table will now look as follows:

<table>
<thead>
<tr>
<th>Physical Port</th>
<th>Source IP</th>
<th>Source Port</th>
<th>Dest. IP</th>
<th>Dest. Port</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>200.1.1.1</td>
<td>63001</td>
<td>10.20.20.100</td>
<td>443</td>
<td>TCP</td>
</tr>
<tr>
<td>2</td>
<td>200.1.1.1</td>
<td>63001</td>
<td>10.20.20.100</td>
<td>81</td>
<td>TCP</td>
</tr>
</tbody>
</table>

As a function of transparent redirection, the source IP and source port remain unchanged from the initial connection. What does change on this entry, and what enables differentiation between the two flows is the destination port.

After the TCP session on port 81 is established, the content switch establishes a TCP session to real-server1 at 10.100.1.10 [5]. Real-server1 replies to 200.1.1.1 via its default gateway, 10.100.1.1, the IP address of VLAN 100 on the content switch [6a].

As we mentioned earlier, the content switch is configured with default routes pointing to each of the SSL-x’s connected to it, and also to its upstream router. Without ECMP routing, it would not be possibly to reliably or predictably use multiple default gateways. However, with a combination of ECMP and flow capabilities, the content switch will examine its flow table for characteristics matching those of the packet returned from real-server1. It will find a match with our second entry, and determine that the traffic must be sent out physical port 2, VLAN1, to the equal cost gateway 10.1.1.10 [6b]. The packet is
sent back to the SSL-x, and is encrypted for return. The SSL-x begins the return of the encrypted packet to 200.1.1.1 via its default gateway, 10.1.1.1, the IP address of VLAN1 of the content switch. The content switch again examines its flow table. This time it will find a packet characteristic match with the first entry, and will send the packet out physical port 1, VLAN10, to the equal cost gateway 10.10.1.1, its upstream router [7] which then returns it to the client.

**Conclusion:**

Content switching and SSL Offloading both stand on their own as powerful technologies enabling content providers, service providers, and e-commerce sites to deliver secure, high-performance, and reliable services to their clients. SSL Offloading allows web sites to offer privacy and security to their clients without compromising the performance of their site. Additionally, SSL Offloading offers a convenient method for Certificate and Key Management; a task that quickly becomes increasingly unwieldy as a company’s Internet presence expands. Most importantly, SSL Offloading allows content switches to work in an SSL environment, an environment that had previously rendered content switching utterly ineffectual. With the ability to again act on layer 7 data, secure content switching offers enhanced site performance through manageable persistence models, content location flexibility, differentiated service capabilities, and replication services. Clearly, no matter how great the benefit that these advances offer separately, it pales when compared to the potent versatility provided by the marrying of the two perfectly suited technologies.