

# The Importance of Secure URL Re-Write



## Overview

Because of their significant performance benefits and their ability to enable secure content networking, SSL offloaders have emerged as an integral component to many on-line initiatives such as e-commerce, Customer Relation Management, Enterprise Resource Planning, and Sales Force Automation. Dominant vendors in these spaces, including Siebel, SAP, PeopleSoft, Microsoft, and Oracle all recommend and themselves use SSL offloaders to enhance the performance of their platforms. The evolution of SSL offloaders over the past year has introduced a number of new features, including support for client certificates, back-end encryption, and tight integration with content switches, yet despite these improvements there has remained a marked lack of progress in the area of strengthening interoperability with the aforesaid vendor's products. Because most SSL offloading solutions have remained relatively insensitive to the needs of the very application platforms that they enable and enhance, the ever-increasing burden of integrating the SSL offloader *at the software level* has fallen squarely on the shoulders of the application developers.

The notion that SSL offloaders integrate transparently into a network is, for the most part, true. At the network level, the introduction of an offloader is rarely disruptive, requiring few (if any) changes to the network at either the logical or the physical level. At the software level, however, there are often issues, and the complexity and repercussions of these issues are typically commensurate with the complexity of the application itself.

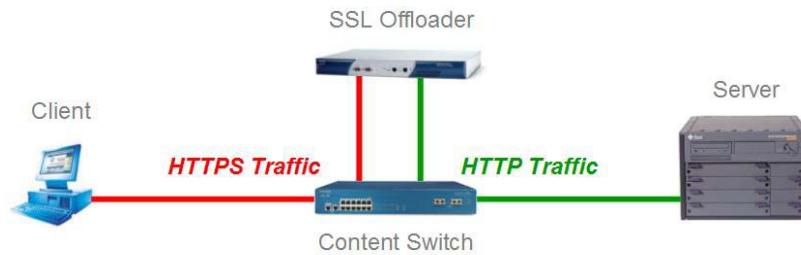


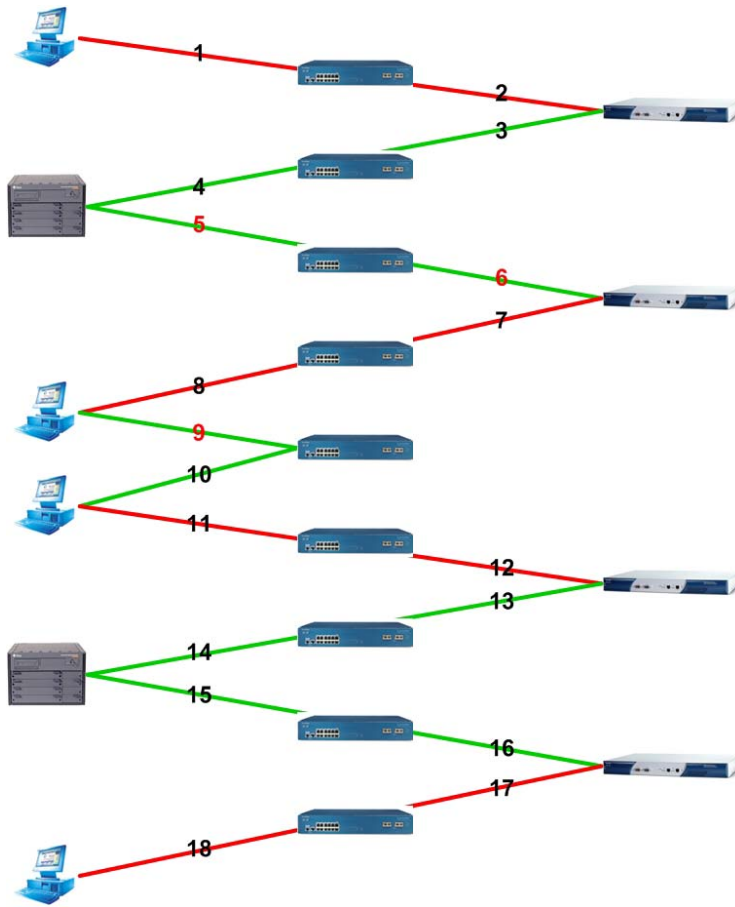
Figure 1. A generic offloaded SSL configuration

The basic offloading model, as depicted in figure 1, consists of HTTPS (encrypted) traffic in, and HTTP (decrypted) traffic out. It is this design that both facilitates content networking in an SSL environment, and relieves the back-end servers of the burden of cryptography. As a result

of this arrangement, the back-end server “talks” HTTP rather than HTTPS.

Since nearly every web-based application today is dynamic by nature, the protocol that the server is “talking” affects the dynamic construction of content.

To illustrate the problem, the following depicts the login sequence commonly used by such platforms as PeopleSoft and Microsoft’s Passport when deployed in conjunction with an offloader:



1. Client requests <https://mysite.com/resource.html> - Request resolves to rule on content-switch
2. Content-switch balances HTTPS hit to SSL offloader.

3. SSL offloader decrypts traffic and sends it to the content-switch for balancing to a back-end server
4. Content-switch balances HTTP hit to server
5. Server checks for authentication credentials (via cookie) and finds none. Since server is talking HTTP it sends a dynamically constructed redirect to the client at <http://mysite.com/login.html>
6. Redirect to non-secure resource is passed to the offloader
7. Offloader re-encrypts the redirect and sends it through the content-switch
8. Content-switch returns encrypted data to the client
9. Client responds to redirect and switches from an encrypted to an unencrypted session, requesting <http://mysite.com/login.html>
10. Content-switch responds to erroneous redirect with another redirect to the secure resource: <https://mysite.com/login.html>
11. Client requests <https://mysite.com/login.html> - Request resolves to rule on content-switch
12. Content-switch balances HTTPS hit to SSL offloader
13. SSL offloader decrypts traffic and sends it to the content-switch for balancing to a back-end server
14. Content-switch balances HTTP hit to server
15. Server sends login page via content-switch
16. Content-switch forwards page to offloader for encryption
17. Offloader encrypts page and send it through content-switch
18. Content-switch returns encrypted page, via HTTPS session to client for secure login.

The sequence illustrated above occurs anytime an SSL offloader is used with a site that employs redirects, regardless of whether the offloader is an external appliance or integrated directly into the content-switch or load-balancer.

In addition to affecting sites dependent upon redirects, this “contextual-inconsistency” or unintentional switching from HTTPS to HTTP occurs whenever an application performs the perfectly common task of dynamically generating content relative to the protocol that it is talking.

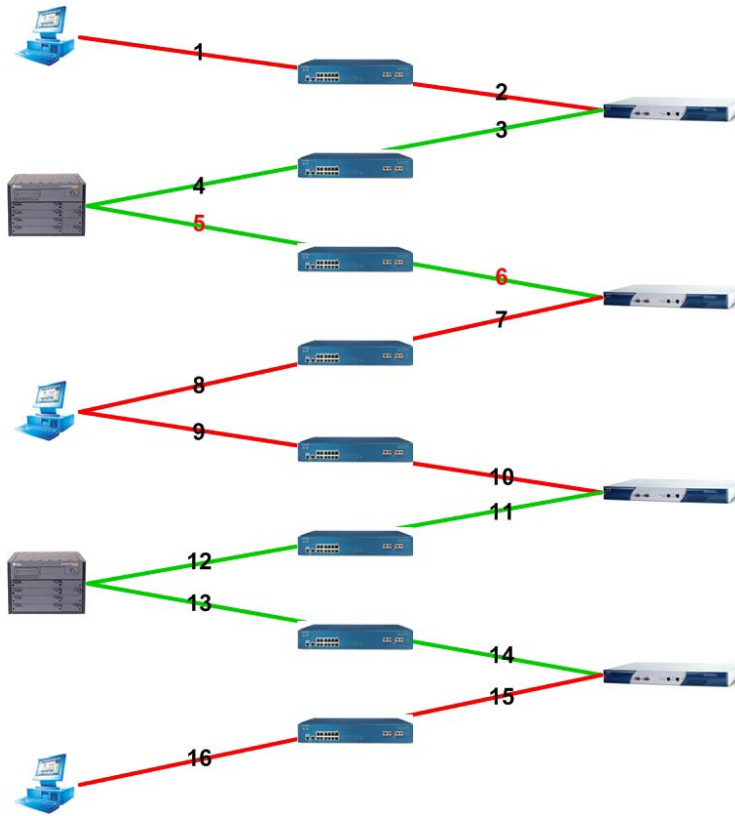
In the past, the only secure way to handle this situation was to painstakingly redesign the application itself to work in conjunction with an offloader by forcing all protocol references to HTTPS. This solution

ranged from impractical to virtually impossible, typically relegating the problem to insecure handling by the content-switch.

To understand the full security implications of the situation illustrated above, consider the sorts of data that could be sent by the client during that erroneously unencrypted leg of what should be a fully encrypted and secure session: during that one non-secure traversal, the client could and often does send via cookies such sensitive data as single-sign-on information or username/password and other account information, potentially compromising the security of the entire session, or even the entire site.

### **SonicWALL Secure URL Re-Write**

Secure URL Re-Write is the only complete and secure solution to this problem, and it requires no modifications to the application. It works as follows:



1. Client requests <https://mysite.com/resource.html> - Request resolves to rule on content-switch
2. Content-switch balances HTTPS hit to SSL offloader.
3. SSL offloader decrypts traffic and sends it to the content-switch for balancing to a back-end server
4. Content-switch balances HTTP hit to server
5. Server checks for authentication credentials (via cookie) and finds none. Since server is talking HTTP it sends a dynamically constructed redirect to the client at <http://mysite.com/login.html>
6. Redirect to non-secure resource is passed to the offloader. Secure URL Re-Write recognizes the contextual error and changes the redirect from HTTP to HTTPS, ensuring that a fully secure session is maintained
7. Offloader re-encrypts the corrected redirect and sends it through the content-switch
8. Content-switch returns encrypted data to the client
9. Client requests <https://mysite.com/login.html> - Request resolves to rule on content-switch
10. Content-switch balances HTTPS hit to SSL offloader.
11. SSL offloader decrypts traffic and sends it to the content-switch for balancing to a back-end server
12. Content-switch balances HTTP hit to server
13. Server sends login page via content-switch
14. Content-switch forwards page to offloader for encryption
15. Offloader encrypts page and send it through content-switch
16. Content-switch returns encrypted page, via HTTPS session to client for secure login.

In addition to improving efficiency by removing two hops, including a redundant redirect, **Secure URL Re-Write guarantees that no sensitive data is ever exposed.** No other solution offers this level of consistent protection. Whether the contextual-inconsistency is a simple 302 series redirect, or a full web-page comprising erroneous references to dynamically linked HTTP resources, Secure URL Re-Write can completely, efficiently, and securely rewrite the data before the site's security is potentially compromised. Only with SonicWALL's exclusive Secure URL-Rewrite feature does SSL offloading finally deliver on its promise of fully transparent *and secure* integration.

To learn how SonicWALL's comprehensive SSL offloading solutions can meet your specific business needs, contact SonicWALL at (888) 557-6642 or visit us online at [www.sonicwall.com](http://www.sonicwall.com).