# Chapter 3
# The Framework of SSL Proxy Server

## 3.1 General Concept

The design of SSL proxy server aims at being performing SSL traffic decryption and extracting the origin contents of HTTPS traffic. Once the HTTPS traffic has been decrypted, it provides any application layer services with plain-text traffic. For example, it provides the application firewall with plain-text traffic for deep inspection. As the SSL proxy server was proposed in conjunction with application firewall the inspection engine is now able to inspect every packet in an unencrypted format. In addition, the SSL proxy server also provides several application-layer services such as proxy caching technique due to original cipher-text traffic has been decrypted.

The SSL proxy server sits logically intervening between web server and clients. The essential framework was also illustrated in Figure 7. The framework is referred to as a transparent middle-ware service by leveraging available communication methods. And it was built under a common, open protocol for edge-based appliance communication. The SSL proxy server monitors network traffic before the packet reaches the web server and extracts the SSL streams and deals with content processing.

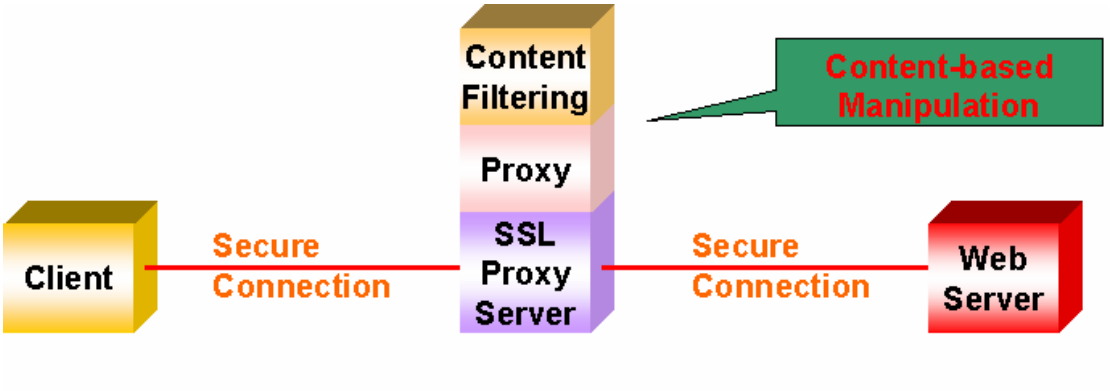



Figure 7. The concept of SSL Proxy Server.

Figure 8 represents the practical network environment. Once the client's request reaches the entry guard firewall on port 80, the firewall always makes a decision to pass the traffic through application firewall or application layer services. After making confirmation of content security, it forwards to origin web server. The novel deployment of SSL proxy server now has a great advantage for keeping present devices unchanged and just adds up another route. Suppose a request attempts a connection to port 443 on the web server, the firewall lets the request to redirect and policy route to SSL proxy server. An open standard protocol, named as ICAP [27], leverages existing equipments available and then chooses the appropriate value-added application provider. The client's request is redirected to application firewall by the intervening SSL proxy server. The application firewall checks the content, ensures it valid, and then sends it back to SSL proxy server. The SSL proxy server forwards it to the origin web server to fulfill client's request. On the opposite direction, the response from origin web server also has the similar processes through SSL proxy server. Now the network administrator performs content-based manipulation as general serving HTTP traffic.

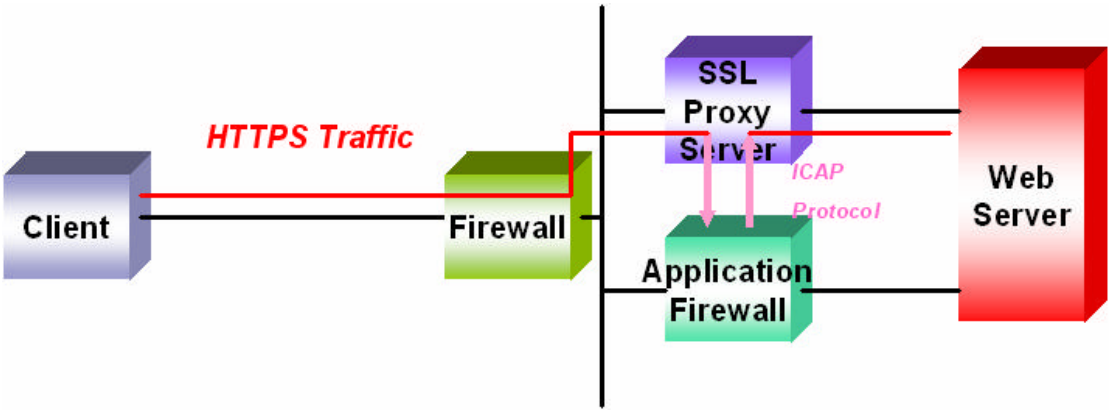



Figure 8. Practical deployment of the SSL proxy server

The potential features of framework of SSL proxy server are required as listed below:

- Security

  Depend upon administrator's control it ensures the certain level of security between end-to-end communications. No one except SSL proxy server eavesdrops the content of the transaction.

- Transparent

  It is a transparent proxy and does not ask for client's configuration. A client thinks it is talking directly to server and on the contrary a server thinks it is talking directly to the client or cache.

- Flexibility

  The framework provides more value-added services and makes scalability and compatibility with existing secure devices such as firewalls. Not only holds the principle of simple is good, but also to be a modular architecture. That is, services should be able to be added and removed and be treated to specific components of the framework.

The primarily design in this thesis was concentrated on the methodology for

extracting HTTPS traffic in SSL proxy server. The SSL proxy server was shown from the essence diagram that acts as a server in facing to incoming clients as well as acts as client in facing to origin web server. The proxy work has both socket and SSL functionalities for both client and server at one time. In later implementation has contained additional codes to present the proxy caching and content filtering. The framework was implemented by the industry-standard OpenSSL library [28] [29]. The library supports SSL 2.0 / 3.0 and TLS.

## 3.2 Essential Framework

The essential framework of SSL proxy server and relationship between client and web server are shown in Figure 7. The primarily effort lies in splitting the secure connection into two connections and then joint them. About the task has separated into two phases. First phase of task is to establish connection and transfer data in the second phase.

The first phase of processing the connection establishment handled by SSL proxy server is shown in Figure 9. While the client initiates request by sending message, SSL proxy server first intercepts the traffic and acts as a server, performs TCP 3-way handshaking and in turn SSL handshaking. The negotiation steps include exchanges certificate, selects cipher suite and generates *pre_master_secret* key. Next, SSL proxy server does not send another request to the web server. Instead, it waits for the first HTTP request that may contain URL within GET, POST, or CONNECT, etc. An apparent methodology may be considered that establishing connection in both sides before content data transmission. However, swapping two steps, connecting to server and receive first HTTP request, it would have more benefits. First, after examining first HTTP request message it then performs both TCP 3-way handshaking and SSL

handshaking to origin web server. Conversely, once any malicious request occurs, SSL proxy server immediately drops the connection rather establishes it to origin web server. After having confirmation that the request is legitimate, the SSL proxy server now joints two segments of connections. The communication between the client and the server walks through SSL proxy server is established.

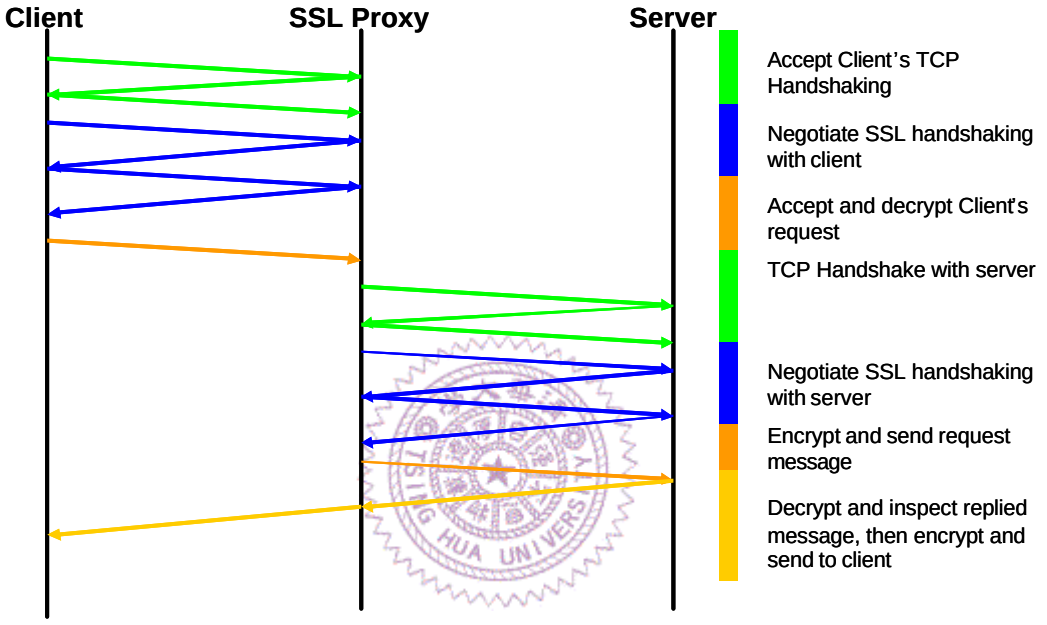



Figure 9. SSL proxy server handles connections.

After two connections are established and jointed, rhe client and server can exchange application data over encrypted, authenticated link that has been established. Meanwhile, the content is extracted by SSL proxy server. It then acquires the plain text in every on-the-fly packet. On the way to another side, the SSL proxy server re-encrypts with another session key and forwards the packet. In fact, the SSL proxy server processes in the transparent mode in the client's standpoint of view.

There is a couple of reasons why receive HTTP first request message before establishing connection to the origin web server. The first reason has been explained

in previous paragraph. The request message could be decrypted and inspected in advance. As long as the incoming request message involved with malicious patterns or anomy signatures, the inspector engine sends an alarm, drops the packet, or takes other protection actions. And it is not necessary to establish another connection between SSL proxy server and the web server. Another reason is due to the system can perform proxy cache function [30] after decrypting client's request. Since the traffic beyond the encryption channel, traditional proxy could not see through the content. However, the system has the cryptographic ability to decrypt and check if it hits in the proxy cache. In the some situation, such as object is cacheable, cache timer is not yet out of expiration, the proxy cache can mitigate the server's loading. The integrity combination of decryption, proxy cache, inspection and encryption was illustrated in the Figure 10 and Figure 11 for request and response respectively.

According to the Figure 10 the detail steps of request procedure are explained in below.

1. First step is to receive incoming traffic, decrypt it using symmetric key and then decode it into plaintext form.
2. Next, the sets of rules are used to determine if the representation exists in the proxy cache or force caches to submit the request to the origin web server for validation again.
3. If the representation is available in the proxy cache, the respondent content is served from cache. Prior to sending back to a client, the representation is required to be encrypted. One critical point we discuss here is, neither request nor response should be inspected result from the content must be not threateningly so that it already cached before.
4. Otherwise, it indicates that the representation misses from the cache or the request explicitly requires up-to-date (no-cache) data.

5. Prior to sending the request to server, delivered it to the inspection engine where the network packets are analyzed against the database of attacks and vulnerabilities.
6. In finally, it encrypts the message with the symmetric key shared with server and then sends the request to origin web server at present.

On the other side, the steps of response procedures are clearly listed as below and briefly shown in Figure 11.

1. At first, once the SSL proxy server receives SSL stream from web server's reply, then it extracts SSL stream.
2. The procedures of examining plain-text content are the same as incoming request procedure mentioned above. Once the malicious patterns or anomy signatures are matched, the SSL proxy server then drops this packet and terminates this connection.
3. Suppose the response content could be cached, the content was been cached as named as its URI in the proxy cache.
4. Regardless of cache or not, finally, the response content has to be encrypted with the key shared with client and passed to client.

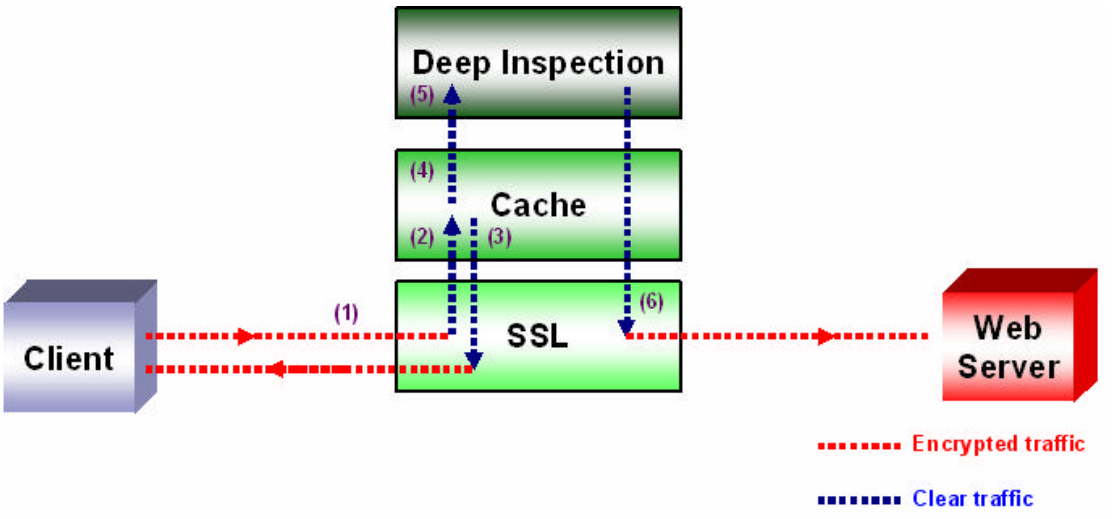



Figure 10. The process of client's request.

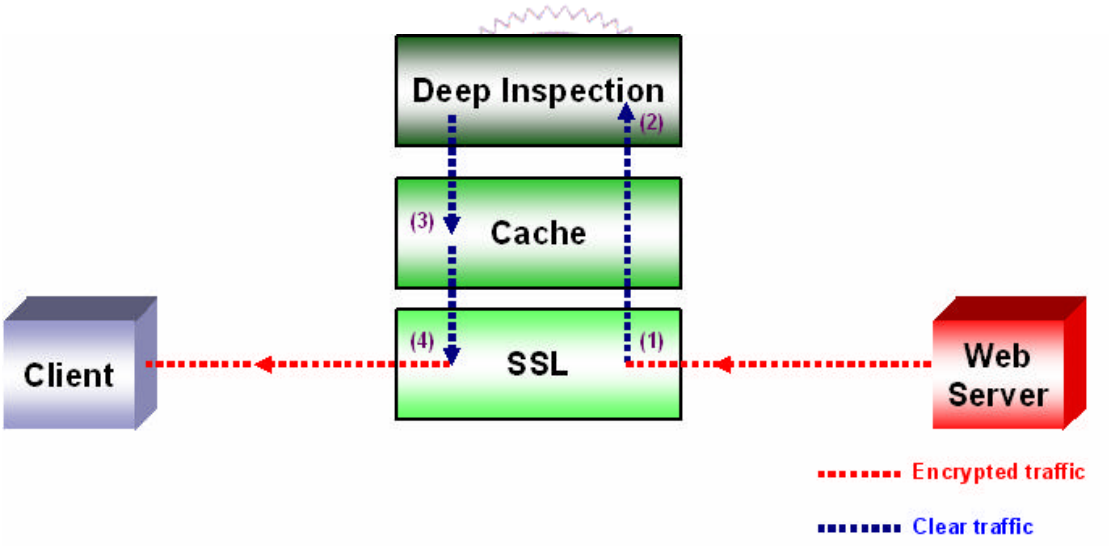
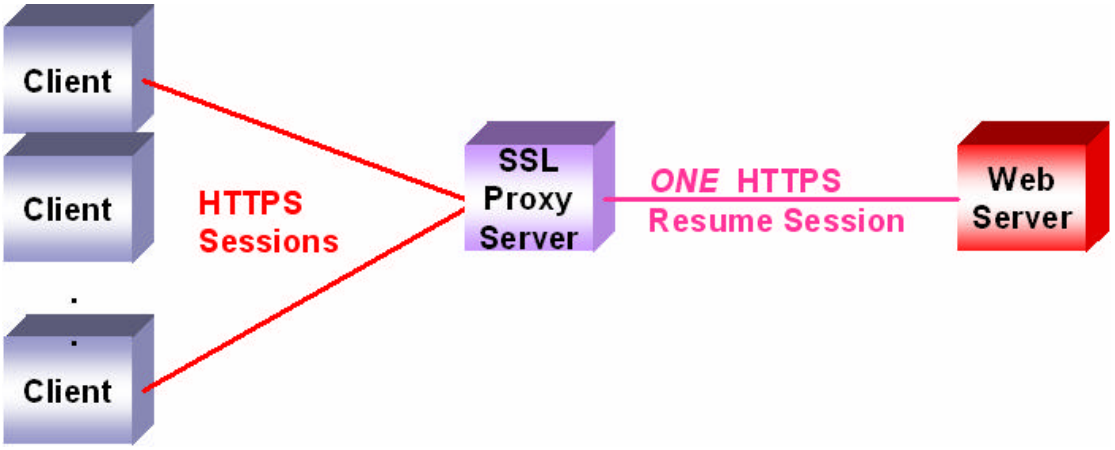



Figure 11. The process of response from server.

## 3.3 Aggregative Framework

For the achievement of inspecting secure traffic, the SSL proxy server intercepts the encryption traffic and splits one connection into two connections. It indicates

twice the expensive full handshaking protocol in the interval of client-proxy and that of proxy-server individually. To explore the potentials of further improvements, this framework takes advantage of using resume handshake instead of using full handshake between proxy and server. The burden of establishing a new SSL session is reduced by reusing previously established SSL session ID's. Figure 12 shows the SSL proxy server uses only one SSL session to connect the web server. It looks like aggregation of many pipes into single one. TCP connections remain 1:1, but the relationship between front-end clients and back-end SSL session is N:1. This methodology is then called as *aggregative framework*.




| Server Address | Port | Session ID | Cipher Suite | Cert. Info. | Timer |
|---|---|---|---|---|---|
| 140.114.71.160 | 443 | caf0f930 | RC4-SHA | … | 300 |

Figure 12. Aggregative Framework.

Consider Figure 12 as an example. Once any connection established from outside to the web server belongs IP address 140.114.71.160, the cipher suite RCA-SHA and session id caf0f930 in the segment between SSL proxy server and origin web server are shared. On the other hand, while connecting to the web server 140.114.71.161, the

cipher suite EXP-AES128-MD5 and session id 2d057a4 in the segment between SSL proxy server and origin web server are shared.

According to [31], whereas full handshake needs 172 ms, resume handshake just needs 7 ms in general. The major improved factor is to eliminate the asymmetric cryptographic processing. We also experimented with browsing and requesting a HTTPS web page, observed that 1662 bytes are transferred by the full handshake of requesting a page. However, that for resume handshake of requesting a page is 429 bytes. Even though the precise numbers depend on the selected cipher suite, certificate created by the server, or SSL related settings, it does take advantage of using resume handshake. Not only the response time but also the bandwidth utilization can be improved in the meantime under using resume handshake.

For accomplish the purpose of reusing session from cache, the session id is the key point to use resume handshake. Considerable effort has been invested in sharing the session between several forked processes. Web server program generally has several client requests in progress at the same time. As SSL proxy server forks a process for each connection as it arrives. Therefore, several communications pass through system as pipeline. The share memory mechanism [32] [33] was exploited among these simultaneous running processes so as to access the session data structure. The SSL proxy server is implemented by creating two session tables for reusing handshake information in facing to client's connection and in connecting to server, as shown in Figure 13. Another implementation issue is the web server in general holds a session cache so that the later coming connections could be accepted with resume session ID. However, in the cause of security that if session stays for a long time and the more weak exposed that symmetric cryptographic works. Therefore, the web server will clear its session cache every certain time and force re-perform full handshake. For example, one of the most common web servers, Apache uses the

default value 300 seconds. Consequently, the SSL proxy server has to maintain synchronal session from origin web server so as to having benefit of resume handshaking.

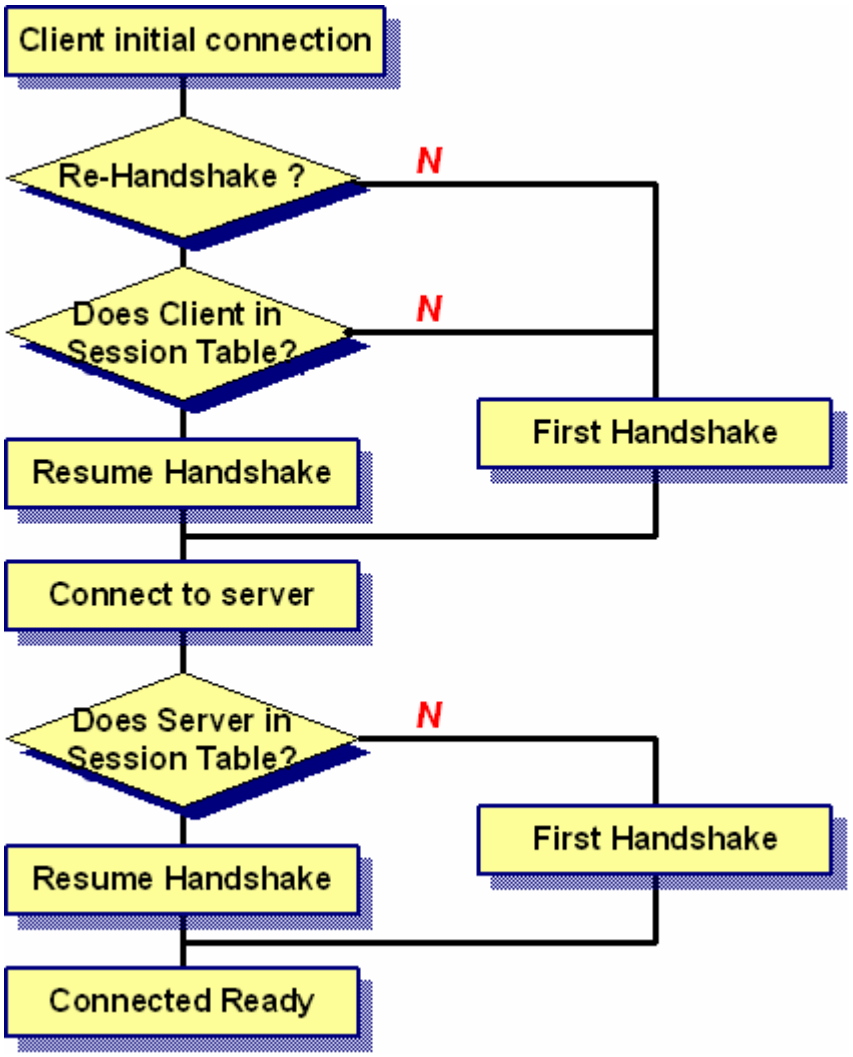



Figure 13. Check session table in order to use resume handshake.

## 3.4 Selectivity Content Framework

The primary advantage of using resume handshake achieved eliminating the computing of asymmetric cryptographic, that resulted in reducing overhead during the

handshaking. In other words, resume handshake technique helps reforming the overhead of establishing connection, however, but not makes any impact on the following request and its response. In order to improve data transferring, this design aims at transferring specific embedded objects with lightweight encryption.

With regard to the browser, browsing a single web page tends to result in numerous connections because the page contains a mix of numerous representations, including HTML text, JPG/GIF pictures, media streaming or embedded Flash objects, etc. When users visit a typical webpage, they download several files. For instance, there are a lot of product's photos in a shopping site. There also are a lot of blank forms in a business site. It has been pointed out that around 81% pages have more than one object and more than 70% pages contain image files [34]. Table 1 shows the similar statistic that more than 77% of the requests contain image types or other non-text types [35].

Table 1. Request object type popularities.

| Object type | Prevalence |
|---|---|
| Images | 77% |
| HTML | 5.2% |
| CGI | 8.6% |
| .doc, .pdf, .ppt, .ps | 0.2% |
| Audio | 0.1% |
| Other | 9.1% |

The hypothesis of these embedded representations could be exposing to inside users. These representations are allowed to transfer without encryption between proxy server and web server since these objects could be identified. For example, if a user visited Amazon's webpage at www.amazon.com, he would download over forty separate files such as advertising banner, sample picture, or a blank form. Therefore, the framework enables that transfer these representations without encryption overhead in the preceding of not to altering the settings of the server.

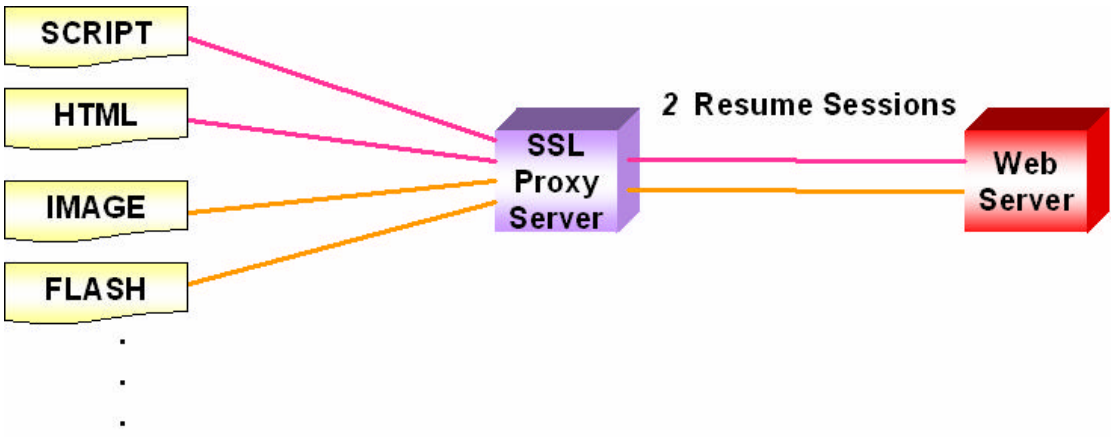
The idea of only applying security process to sensitive elements of a data stream is derived from [36]. Thus, the framework is named as *selectivity content framework*. However, neither motivation nor methodology is the same.

The SSL protocol is designed to support an ample range of selecting specific algorithms used for cryptography, digests, and signatures. Choices are negotiated between client and server at the start of establishing a protocol session. In this regard, the SSL protocol allows the user to choose the available algorithms. For example, the cipher suite SSL_RSA_WITH_RC4_128_MD5 means that using a RSA asymmetric for key exchange and a RC4-128-bits symmetric algorithm for secure transfer and additional MD5 hash function can be used. In particular, take a focus on special cipher suite SSL_NULL_WITH_NULL_NULL, it does not provide any security. It is always set at beginning of the SSL handshake protocol. For the security sake, most web servers and browsers prohibit from setting manually this cipher suite. In the preceding of not to altering the settings of the server, the web server even not supports cipher suite SSL_NULL_WITH_NULL_NULL. Instead, the SSL_RSA_WITH_NULL_MD5 are chosen to apply to the connection between proxy server and web server. Thus, eliminates the effort of symmetric cryptographic computing. As a result, while using both resume handshake and SSL_RSA_WITH_NULL_MD5, the transfer uses neither asymmetric exchange nor symmetric encryption. We call this operation as *Null-Encryption*. The extra gain is to off-load SSL processing from server.

Consider the example shown in Figure 14. Suppose a client attempts to request the general representations such as html type or script type, these text type representations are encrypted with RC4-SHA cipher suite in the connection segment between SSL proxy server and web server. Once it requests other type representations such as image, flash files, or public disclosure files, these objects are transferred

within NULL-MD5 cipher suite. The NULL-MD5 cipher saves the symmetric cryptographic effort than RC4-MD5.

In summary, two virtually connections result from dual sessions between proxy server and web server, one uses resume handshake and symmetric encryption for general private data stream, and the other uses resume handshake but no encryption for certain images, videos, flash objects. The ideas can be applied to those target representations, such as public images, background media streaming, global files, or announcement documents. However, the classification for objects needs to be customized by administrator.




| Server Address | Port | Session ID | Cipher Suite | Cert. Info. | Timer |
|---|---|---|---|---|---|
| 140.114.71.160 | 443 | caf0f930 | RC4-SHA | … | 300 |
| 140.114.71.160 | 443 | 2d057a4 | NULL-MD5 | … | 300 |

Figure 14. Selectivity Content Framework.

## 3.5 Implementation of SSL Proxy Server

This section introduces the functionalities of SSL proxy server. The user interface is designed for monitoring traffic and showing handling results as illustrated

in Figure 15. An amount of events and processes are also stated. Regard of layout of user interface, in top half of Figure 15 all incoming requests are listed in the Connection Info Table. Two text boxes in bottom half of Figure 15 shows out plain-text content include request and response for any given rows in top list table. The Treat column is the most important reaction in the connection segment between SSL proxy server and origin web server.

In the beginning, the first row properly indicates the first incoming request. Therefore, full handshake is required to establish in the connection segment between SSL proxy server and web server. Afterward, all incoming requests need only resume handshake until the 8th row. The 8th row exploits full handshake again result from session time is expired for 300 seconds. In reality, another specialized process is implemented to maintain synchronized session information periodically and provide up-to-date session id. These two connections (i.e. 1st and 8th) are demonstrated to perform full handshake and session timer settings.

The following second and third incoming connections attempt to request individual files, they have benefit of using resume handshake. The 4th request acquires the same object that first request does, hence it gets the response, a copy of representation from cache. As to 5th and 6th requests, they ask for image type objects, the connections between SSL proxy server and web server use Null-Encryption mechanism. Finally, last two rows are explained as shown in the Figure. Actually, they represent malicious traffic: the former has predefined suspicious patterns in the request packet, and the latter has been found in the response message. The action will be taken for dropping them. We observed that, in the case proxy caching or attack activity in request (ex. 11th row), these requests need no another connections to be established from SSL proxy server to web server. Therefore, the P-S Session and P-S Cipher left a blank cell.

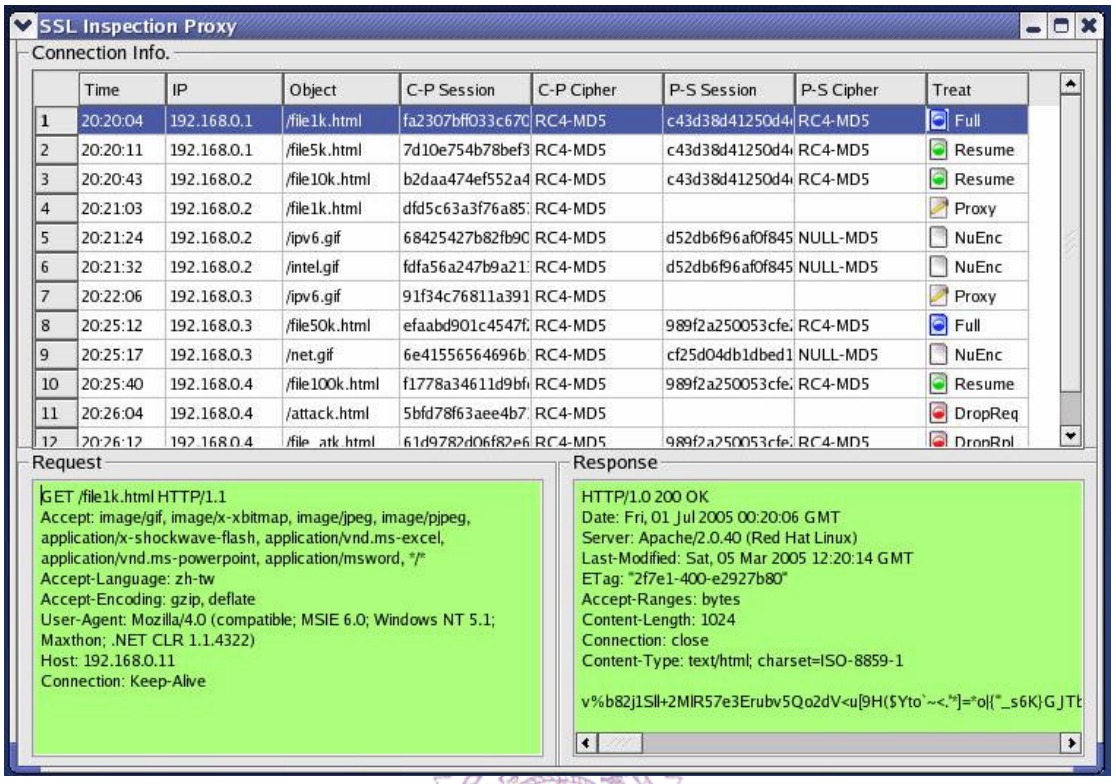



Figure 15. User interface for monitoring the incoming requests.