Chương trình KC-01:
Nghiên cứu khoa học phát triển công nghệ thông tin và truyền thông

Đề tài KC-01-01:
Nghiên cứu một số vấn đề bảo mật và an toàn thông tin cho các mạng dùng giao thức liên mạng máy tính IP

Báo cáo kết quả nghiên cứu

PHẦN MỀM CÓ SỬ DỤNG CHỨNG CHỈ SỐ

Quyển 8B: "Bảo mật dịch vụ Web thông qua Proxy Server"

HÀ NỘI-2004

Báo cáo kết quả nghiên cứu

PHẦN MỀM CÓ SỬ DỤNG CHỨNG CHỈ SỐ

Quyển 7B: "Bảo mật dịch vụ Web thông qua Proxy Server"

Chủ trì nhóm thực hiện:
ThS. Đặng Hoà

Mục lục

Chương I
SQUID Proxy Server

1-Giới thiệu về Squid

Squid là ***proxy caching server*** có mã nguồn mở cho các máy khách sử dụng web, hỗ trợ các đối tượng dữ liệu của các giao thức FTP, gopher và HTTP. Squid giữ các dữ liệu và các đối tượng "nóng" (đã được tải qua) trong RAM, lưu trữ tạm cơ sở dữ liệu của các đối tượng trên đĩa (phục vụ việc tìm kiếm đối tượng, DNS, ...).

Squid bao gồm một chương trình chính *squid*, một chương trình tìm kiếm hệ thống tên miền *dnsserver*, một số chương trình tuỳ chọn để viết lại các yêu cầu và thực hiện xác thực, công cụ client. Với khả năng tiết kiệm băng tần, khả năng bảo mật, và tăng tốc độ truy cập Web. Squid là một phần mềm khá tinh vi, miễn phí đã đưa Squid thành một phần mềm được sử dụng khá phổ biến.

Squid được sử dụng ở 2 chế độ: chế độ tăng tốc http (httpd-accelerator) để tăng khả năng cung cấp của Web server, và chế độ proxy-caching server mà ta thường sử dụng.

2-Các thuật ngữ được sử dụng với Squid

Internet Object: là một tệp, tài liệu, ... để truy vấn một dịch vụ internet chẳng hạn FTP, HTTP, hoặc gopher.

Internet object caching: là một cách lưu trữ các các đối tượng được yêu cầu (dữ liệu theo các giao thức HTTP, FTP, và gopher) trên hệ thống, gần hơn so với việc tải đối tượng trực tiếp từ địa chỉ đích. Các trình duyệt Web có thể sử dụng Squid cache cục bộ như một proxy HTTP server, giảm thời gian truy cập cũng như băng tần.

Cache hierarchy:

Là một tập các caching proxy server được tổ chức theo quan hệ cha/con và được sắp xếp theo thứ bậc anh em, và cache nào gần Internet gateway nhất được coi là cache cha để lưu trữ các vị trí từ backbone. Khi cache yêu cầu về một đối tượng từ proxy cha, nếu đối tượng không có trong cache cha, proxy cha sẽ tải đối tượng, lưu trữ nó và chuyển nó về cho proxy con. Điều này giúp giảm tối đa truy cập băng tần liên kết đến backbone, giúp giảm việc nạp thông tin Internet từ mạng bên ngoài.

Thêm vào quan hệ cha/con, squid đưa ra một khái niệm anh em: là các cache cùng mức trong cây lưu trữ (cache hierarchy). Mỗi một cache trong cây lưu trữ quyết định tải các đối tượng một cách độc lập; từ cache của mình, của proxy cha hay từ cache của các sibling. Nó sử dụng thuật toán phân giải cache được mô tả ở dưới.

parent (cache cha):
Trong quan hệ cha, cache con sẽ chuyển các yêu cầu tới cache cha của nó. Nếu cache cha không 'giữ' đối tượng được yêu cầu, nó sẽ chuyển yêu cầu đó thay mặt cho cache con. Các cache cha thường được đặt gần với Internet hơn.

sibling (cache anh em)
Trong quan hệ anh em, một máy peer chỉ chuyển các đối tượng đã nằm trong cache. Quan hệ này được dùng để lấy các thông tin ở cache gần hơn, không là định tuyến tới Internet.

Giao thức ICP (Internet Cache Protocol):
Là một giao thức sử dụng cho việc truyền thông giữa các squid cache. Giao thức ICP được sử dụng trong cây cache để tìm các đối tượng đặc biệt trong các sibling cache. Nếu squid cache không có tài liệu được yêu cầu, nó gửi một tín hiệu ICP query tới các cache 'anh em' (sibling), và sibling sẽ trả lời bằng gói ICP chỉ ra "HIT" hoặc "MISS". Cache khi đó dựa vào các tín hiệu trả lời, chọn cache nào có tín hiệu "MISS" để tải đối tượng về. ICP cũng hỗ trợ đa đường truyền của dòng nhiều đối tượng thông qua một kết nối TCP. Trong phiên bản hiện hành, Squid cũng hỗ trợ ICP qua Multicast.

Giao thức HTCP (Hyper Text Caching Protocol): Nếu như giao thức ICP cho phép truy vấn nội dung của các cache khác, và để tránh việc tải lại các đối tượng từ một máy server ở xa gây lãng phí về thời gian, tài chính. Giao thức HTCP cho phép các yêu cầu có phần header đầy đủ để nhằm mục đích quản lý cache, mở rộng các miền quản lý cache, có khả năng yêu cầu xoá nội dung của cache từ xa và gửi các hint về đối tượng web.

Thuật toán phân giải cache (Squid cache resolution algorithm):
- Gửi gói tin ICP query tới tất cả các sibling được thiết lập.
- Đợi tất cả các tín hiệu trả lời sau một thời gian giới hạn timeout nào đó (ngầm định là 2 giây).
- Bắt đầu tải đối tượng khi nhận các tín hiệu HIT trả lời đầu tiên, hoặc
- Tải đối tượng từ proxy cha đầu tiên trả lời bằng tín hiệu MISS, hoặc
- Tải đối tượng trực tiếp từ nguồn (Web server).

3-Cách làm việc của Squid và một số chương trình đi kèm

Squid được sử dụng trong 2 chế độ: chế độ tăng tốc Web (trong chế độ này Squid được dùng với Webserver) để tăng khả năng của Web server, và chế độ caching proxy server để cho phép tất cả mọi người trong một mạng có thể làm việc với Internet thông qua Squid - nghĩa sẽ tải các đối tượng từ Squid proxy về chứ không tải đối tượng trực tiếp từ Internet.

Khi một máy client (Web browser) yêu cầu một đối tượng Internet; nếu đối tượng chưa có trong cache, proxy sẽ tải đối tượng đó (hoặc từ chính URL đã chỉ, hoặc từ

một cache proxy cha hoặc anh em - ta sẽ nói đến ở phần dưới). Khi sử dụng một cây lưu trữ cache (cache hierarchy) thì hệ thống sẽ sử dụng giao thức ICP để trao đổi thông tin về đối tượng được yêu cầu, và sử dụng thuật toán phân giải cache (đã trình bày ở trên) để tải đối tượng về.

Một số chương trình đi kèm

- **dnsserver**: là một tiến trình được gọi bởi trình squid để phân giải địa chỉ tên miền thành địa chỉ IP.
- **unlinkd**: là một tiến trình ngoài được sử dụng để hủy kết nối các file cache không sử dụng.
- **cachemgr.cgi**: là một chương trình đưa ra những thông tin thống kê về Squid.
- **client**: là một công cụ dùng để kiểm tra kết nối với Webserver.
- Một số chương trình xác thực người dùng được đi kèm với gói source của Squid, đặt trong thư mục auth_modules/.

4-Tệp cấu hình squid.conf

Tệp cấu hình squid.conf định nghĩa các thông tin cấu hình cho Squid, bao gồm số cổng HTTP, cổng HTTPS (làm việc với SSL), số cổng yêu cầu ICP, yêu cầu đến và yêu cầu đi, thông tin về truy cập firewall, và các thông tin timeout khác.

Mỗi một tùy chọn được gọi là một thẻ (tag). Các dấu '#' ở đầu dòng được là dòng giải thích (comment) - trình bày sơ qua về tác dụng của thẻ. Ở đây ta chỉ trình bày một số thẻ quan trọng hoặc cần lưu ý, thông tin về các thẻ khác người sử dụng có thể tham khảo từ phần chú thích trong tệp squid.conf. Tệp squid.conf sử dụng 149 thẻ và được chia thành 14 phần chính như dưới đây. Phần liên quan đến **SSL** được trình bày ở mục **1.4.14**.

4.1-Các tùy chọn liên quan đến mạng (gồm 7 thẻ)

Đoạn này chứa các cấu hình liên quan đến mạng của Squid. Bao gồm các tuỳ chọn đóng vai trò quan trọng trong việc quyết định địa chỉ socket của Squid để truyền thông với các Server từ xa hoặc các Neighbor cache. Các cổng chung mà Squid sử dụng để nghe các yêu cầu và trả lời TCP hoặc ICP và địa chỉ IP mà Squid ràng buộc để tạo các địa chỉ socket.
Dưới đây lần lượt giới thiệu về các thẻ tuỳ chọn liên quan đến mạng:

Thẻ http_port

Thẻ này được dùng để chỉ ra địa chỉ socket mà Squid sẽ nghe các yêu cầu HTTP. Có thể thiết lập nhiều địa chỉ socket theo 3 dạng: số cổng, <địa chỉ IP>:<số cổng> hoặc <tên máy>: <số cổng>. Trong hầu hết các trường hợp ta chỉ cần xác định số cổng, ngầm định là 3128. Nếu chạy trong chế độ tăng tốc http, có thể ta cần sử dụng cổng 80.

Thẻ icp_port

Thẻ này xác định số cổng mà Squid sẽ gửi và nhận các yêu cầu ICP từ các neighbour cache. Giá trị ngầm định là 3130. Nó sẽ bị ghi đè bởi tuỳ chọn từ dòng lệnh /usr/sbin/squid -u icp_port.

Thẻ htcp_port

Được dùng để chỉ ra số cổng mà Squid sẽ gửi và nhận các yêu cầu ICP tới và từ các neighbour cache. Giá trị ngầm định là 4827.

Thẻ mcast_groups

Xác định các nhóm Multicast để Server đăng nhập và nhận các gói truy vấn ICP multicast. Ngầm định Squid không nghe trên bất kỳ một nhóm multicast nào.

Thẻ tcp_outgoing_address

Được dùng cho các kết nối với server từ xa. Thông thường không cần thiết lập cho tuỳ chọn này.

Thẻ udp_incoming_address

Được dùng cho các ICP socket nhận gói từ cache khác. Ngầm định là 0.0.0.0.

Thẻ udp_outgoing_address

Được dùng để gửi các gói ICP tới cache khác. Ngầm định là 255.255.255.255.

4.2-Các tùy chọn liên quan đến cây lưu trữ (gồm 9 thẻ)

Đoạn này chỉ sử dụng khi dùng nhiều Squid cache trong một cây lưu trữ (cache hierarchy). Số máy server cache, kiểu cấu hình, thời gian đợi để truyền thông giữa các server và các đối tượng không được lưu trong cache cục bộ được chỉ định ở đây.

Thẻ cache_peer

Được dùng để chỉ ra các cache khác trong cây lưu trữ. Tuỳ chọn này được chia thành 5 trường: hostname, type, http_port, icp_port, options. Để biết thêm chi tiết xin đọc file /etc/squid/squid.conf.

Thẻ cache_peer_domain

Thẻ này được dùng để giới hạn các miền mà các neighbour cache sẽ query. Nó được dùng để truyền thông với các cache khác, phụ thuộc vào miền được yêu cầu.

Thẻ icp_query_timeout (msec)

Thông thường Squid sẽ tự thiết lập giá trị này, nếu ta muốn thay đổi thời gian giới hạn chờ gói ICP query, thì thay đổi ở đây.

Thẻ maximum_icp_query_timeout (msec)

Xác định giá trị thời gian giới hạn lớn nhất khi chờ gói ICP query. Thông thường Squid sẽ tự quyết định giá trị này.

Thẻ mcast_icp_query_timeout (msec)

Sử dụng khi dùng multicast.

Thẻ dead_peer_timeout (seconds)

Thời gian để Squid khai báo là máy peer cache đã "chết". Nếu sau một khoảng thời gian này không nhận được các gói ICP trả lời, Squid sẽ khai báo máy peer cache này đã chết và sẽ không nhận các gói trả lời từ cache server này nữa.

Thẻ no_cache

Sử dụng tùy chọn này để bắt buộc các đối tượng chắc chắn nào đó không được nằm trong cache.

Ví dụ:

```
acl QUERY urlpath_regex cgi-bin \?
no_cache deny QUERY
```

Tất cả các URL có chứa từ cgi-bin \? đều không được lưu trữ trong cache.

Và một số tùy chọn sau: **neighbor_type_domain, hierarchy_stoplist**

4.3-Những tham số ảnh hưởng đến cache size (gồm 12 thẻ)

Đoạn này cho phép các cấu hình chi tiết việc sử dụng tài nguyên của Squid, tập dữ liệu cache lưu trữ trong đĩa và các chính sách thay thế bộ nhớ.

Thẻ cache_mem (bytes)

Xác định bộ nhớ được sử dụng bởi Squid: các đối tượng đang truyền, các đối tượng 'nóng', các đối tượng không được nằm trong cache. Tham số này không xác định kích cỡ tiến trình lớn nhất. Kích cỡ tiến trình Squid có thể lớn gấp hai hoặc ba lần giá trị thiết lập ở đây.

Thẻ maximum_object_size (bytes)

Các đối tượng lớn hơn kích thước này sẽ không được ghi vào đĩa. Nếu bạn muốn tăng tốc độ hơn là việc tiết kiệm băng tần, thì nên thiết lập tùy chọn này với giá trị nhỏ. Giá trị ngầm định là 4MB.

Thẻ minimum_object_size (bytes)

Các đối tượng có kích thước nhỏ hơn kích thước này sẽ không được ghi vào đĩa. Giá trị ngầm định là 0 (KB).

Thẻ maximum_object_size_in_memory (bytes)

Các đối tượng lớn hơn kích thước này sẽ không được cố giữ lại trong bộ nhớ cache.

Thẻ ipcache_size (number of entries)

Thẻ này chỉ ra kích cỡ của ipcache.

Thẻ cache_replacement_policy

Xác định các tham số chính sách thay thế các đối tượng khi cần không gian đĩa. Hiện tại có các chuẩn sau:

- LRU (Least Recently Used): đây là chính sách gốc của Squid, chính sách này sẽ giữ lại các đối tượng thường xuyên được sử dụng.
- heap GDSF (Greedy-Dual Size Frequency): giữ lại các đối tượng thông dụng trong cache.
- heap LFUDA (Least Frequently Used with Dynamic Aging): giữ lại các đối tượng thông dụng (có kích thước lớn hơn so với GDSF) trong cache
- heap LRU: là chính sách LRU được bổ sung thêm sử dụng heap.

Tuỳ chọn này chỉ được áp dụng cho dòng tham số cache_dir ở bên dưới.

Thẻ memory_replacement_policy

Tham số chính sách thay thế bộ nhớ, quyết định xem những đối tượng nào sẽ bị loại bỏ khỏi bộ nhớ khi cần đến không gian bộ nhớ.

Và một số thẻ sau: **fqdncache_size (number of entries), cache_swap_low (percent, 0-100), cache_swap_high (percent, 0-100), ipcache_low (percent), ipcache_high (percent)**

4.4-Thư mục lưu trữ và các tệp log (gồm 15 thẻ)

Đoạn này cho phép cấu hình các file log (kích cỡ, tên, đường dẫn, kích hoạt), thông tin runtime, lỗi. Dữ liệu này có thể được dùng để gỡ rối khi hệ thống gặp trục trặc và cũng để phân tích mẫu cache.

Thẻ cache_dir

Xác định các thư mục cache để chứa các đối tượng Internet. Thiết lập theo cú pháp như sau:

cache_dir Type Maxobjsize Directory Mbytes Level-1 Level2 [..]

Type xác định kiểu hệ thống lưu trữ để sử dụng. Hầu hết mọi người sẽ muốn sử dụng kiểu 'ufs'. Có thể sử dụng kiểu Async I/O 'asyncufs' (khi biên dịch chạy file configure với tham số--enable async-io).

Maxobjsize tham chiếu tới kích thước đối tượng lớn nhất cho thư mục lưu trữ này. --1 có nghĩa là kích thước bất kỳ.

Directory là thư mục mức cao nhất mà các file cache swap được lưu giữ. Nếu ta muốn sử dụng toàn bộ đĩa (hoặc phân vùng) làm cache, ta có thể chỉ

tới thư mục kết gán. Thư mục phải tồn tại và được quyền ghi bởi tiến trình squid, Squid sẽ không tạo ra bất kỳ một thư mục nào.
***Mbytes*:** là khối lượng không gian đĩa (MB) sử dụng cho thư mục này.
Level-1 là các thư mục con (sử dụng làm cache, lưu trữ các đối tượng) được tạo ở mức đầu tiên.
Level-2 là các thư mục con (sử dụng làm cache, lưu trữ các đối tượng) sẽ được tạo ở dưới mỗi thư mục ở mức 1. Để tạo các thư mục cache swap này, ta sử dụng lệnh /usr/sbin/squid -z.
Ngầm định: cache_dir ufs -1 /var/spool/squid 100 16 256

Thẻ cache_access_log

Thẻ này chỉ ra đường dẫn tới file access.log

Thẻ cache_log

Chỉ ra đường dẫn tới file cache.log

Thẻ cache_store_log

Chỉ đường dẫn tới file store.log, file ghi lại các hoạt động quản lý, ghi lại các đối tượng bị loại bỏ, các đối tượng được ghi vào đĩa.

Thẻ cache_swap_log

Xác định vị trí với file swap.log

Thẻ emulate_httpd_log on|off

Squid có thể mô phỏng định dạng file log, mà chương trình httpd sử dụng.

Thẻ mime_table

Xác định đường dẫn tới file cấu hình mime.conf của Squid. File này chứa các kiểu MIME được Squid hỗ trợ.

Thẻ pid_filename

Xác định vị trí file mà Squid ghi id tiến trình của nó, ngầm định là /var/lock/squid.pid.

Và một số thẻ sau: **log_ip_on_direct, log_mime_hdrs on|off, user agent_log, referer_log, debug_options, log_fqdn on|off, client_netmask**

4.5-Các tùy chọn liên quan đến các chương trình bên ngoài (gồm 18 thẻ)

Đoạn này đưa ra các tuỳ chọn cấu hình cho các chương trình ftpuser, DNS, Redirector, và authenticator được phân phối cùng Squid. Các chương trình bên ngoài được đặt trong thư mục contrib/ trong bản phân phối mã nguồn. Sử dụng đoạn này để thực hiện một số tác vụ đơn giản như định hướng lại URL, xử lý DNS,

các chương trình xác thực. Số tiến trình con của mỗi tiến trình này cũng có thể được chỉ ra ở đây.

Thẻ cache_dns_program

Xác định vị trí của file thi hành lệnh tìm kiếm dns (chính là đường dẫn tới chương trình dnsserver).

Thẻ dns_children

Số tiến trình con có thể sinh ra với dịch vụ tìm kiếm DNS.

Thẻ unlinkd_program

Thẻ này xác định đường dẫn tới chương trình xoá file (liên kết không sử dụng nữa) unlinkd.

Thẻ pinger_program

Đường dẫn tới chương trình pinger. Tuỳ chọn này chỉ được sử dụng khi bạn biên dịch (bằng script configure) Squid với tuỳ chọn '--enable-icmp'.

Thẻ authenticate_program

Xác định đường dẫn tới chương trình thực hiện xác thực. Nếu ta sử dụng chương trình xác thực, ta phải sử dụng một acl kiểu proxy_auth. Squid đã hỗ trợ một số chương trình xác thực trong thư mục /auth_module trong gói mã nguồn của Squid.

Ví dụ để sử dụng chương trình xác thực sử dụng các module xác thực PAM có sẵn trong Linux, ta biên dịch và copy chương trình ***pam_auth*** từ thư mục auth_module/PAM/ vào thư mục /usr/local/squid/. Và trong file cấu hình squid.conf này ta thêm các dòng sau:

```
authenticate_program /usr/local/squid/pam_auth
acl localusers proxy_auth REQUIRED
http_access allow localusers
http_access deny !localusers
```

Khi đó, hệ thống sẽ chỉ cho phép những người dùng được xác thực truy cập cache. Nó kiểm tra username/password dựa theo file /etc/passwd.

Thẻ authenticate_children

Số tiến trình xác thực có thể được gọi (ngầm định là 5).

Thẻ authenticate_ttl

Thẻ này được dùng để xác định thời gian để kiểm tra tổ hợp username/password còn trong cache (tính theo giây, ngầm định là 3600).

Và một số thẻ sau: **authenticate_ip_ttl, ftp_user, ftp_list_width, ftp_passive, dns_retransmit_interval, dns_timeout, dns_defnames on|off,**

dns_nameservers, redirect_program, redirect_children, redirect_rewrites_host_header.

4.6-Các tùy chọn để điều chỉnh cache (gồm 14 thẻ)

Khả năng của squid phụ thuộc nhiều trên đoạn cấu hình này. Đoạn này quyết định các đối tượng được refresh lại bởi một thuật toán phù hợp, kích thước header và phần thân cho cả các yêu cầu và trả lời cho việc quyết định sau đó, hủy kết nối khi máy client đóng kết nối.

Thẻ wais_relay_port

Chuyển tiếp (relay) các yêu cầu WAIS tới máy **wais_relay_host** tại cổng **wais_relay_port**.

Thẻ request_header_max_size (KB)

Xác định kích cỡ lớn nhất với phần đầu của các yêu cầu HTTP. Thông thường kích cỡ phần đầu của các yêu cầu HTTP là nhỏ (512 byte), thay kích cỡ giới hạn cho phần đầu của các yêu cầu HTTP có thể tránh được một số lỗi chắc chắn nào đó (các kết nối dai dẳng, có thể làm tràn bộ nhớ hoặc các kiểu tấn công từ chối dịch vụ).

Thẻ request_body_max_size (KB)

Tuỳ chọn này xác định kích cỡ lớn nhất cho phần thân (body) của các yêu cầu HTTP. Một người dùng nào đó gửi một yêu cầu có kích thước phần thân lớn hơn giới hạn này sẽ nhận được một thông báo lỗi "Invalid request". Nếu thiết lập tham số này bằng 0 thì sẽ là không có giới hạn nào được đặt.

Thẻ reply_body_max_size (KB)

Thẻ này xác định kích cỡ lớn nhất của phần thân gói tin trả lời. Nó được sử dụng để ngăn không cho người dùng download những file quá lớn (chẳng hạn MP3, hoặc phim). Kích thước gói tin trả lời được kiểm tra 2 lần. Trước tiên là kiểm tra giá trị độ dài nội dung trong phần header, nếu lớn hơn giá trị của tham số này thì yêu cầu bị từ chối và người dùng sẽ nhận được một thông báo lỗi "the request or reply too large". Nếu không có độ dài nội dung ở phần này, và kích thước của gói reply lớn hơn giá trị được thiết lập ở đây, thì kết nối của máy client sẽ bị đóng lại.

Thẻ reference_age

Thiết lập thời gian để Squid xoá các đối tượng được lưu trong cache theo thuật toán LRU (Least Recently Used). LRU thực hiện xoá các đối tượng dưa theo lượng không gian đĩa được sử dụng. Tham số này sẽ thiết lập thời gian tối đa cho LRU. Ví dụ: thiết lập là '1 week', thì các đối tượng sẽ bị xóa nếu nó không được truy cập tới sau 1 tuần. Giá trị ngầm định là 1 năm. Chú ý: không sử dụng tham số này khi sử dụng các chính sách thay thế mở rộng GDSH hoặc LFUDA (sử dụng thẻ replacement_policy).

Thẻ quick_abort_pct (percent)

Cache có thể được cấu hình để tiếp tục download các yêu cầu đã bị hủy. Những người dùng không kiên trì có thể bị tắc bởi việc lặp các yêu cầu và ngay lập tức hủy việc download.

Khi người dùng huỷ một yêu cầu, Squid sẽ kiểm tra giá trị quick_abort so với khối lượng dữ liệu đã tải được. Nếu lượng dữ liệu còn lại nhỏ hơn 'quick_abort_min' KB thì nó kết thúc việc tải về. Thiết lập 'quick_abort' bằng -1 để bỏ chức năng này. Và ngược lại nếu lượng dữ liệu còn lại lớn hơn 'quick_abort_max' KB còn lại, nó sẽ hủy việc tải về. Nếu dữ liệu còn lại lớn hơn 'quick_abort_pct' % thì nó sẽ kết thúc việc tải về. Giá trị ngầm định là:

quick_abort_min 16 KB
quick_abort_max 16 KB
quick_abort_pct 95

Và một số thẻ sau: **quick_abort_min (KB), quick_abort_max (KB), refresh_pattern, negative_ttl time-units, positive_dns_ttl time-units, negative_dns_ttl time-units, range_offset_limit (bytes)**.

4.7-Thời gian giới hạn cho các kết nối (gồm 10 thẻ)

Đoạn này thiết lập thời gian giới hạn cho các kết nối. "Timeout" là khoảng thời gian được giới hạn mà squid có thể chờ để hoàn thành một yêu cầu, nếu nó vượt quá thời gian giới hạn này, squid sẽ trả về cho client một thông báo lỗi ngầm định được xác định cho từng timeout riêng.

Thẻ connect_timeout time-units

Thẻ này xác định thời gian đợi kết nối hoàn thành. Ngầm định là 2 phút (120 giây).

Thẻ peer_connect_timeout time-units

Tham số này xác định thời gian đợi trong quá trình kết nối tới cache của máy peer.

Và một số tuỳ chọn sau: **site select_timeout time-units, read_timeout time-units, request_timeout, client_lifetime time-units, half_closed_clients, pconn_timeout, ident_timeout, shutdown_lifetime time-units.**

4.8-Điều khiển truy nhập (gồm 7 thẻ)

Các danh sách điều khiển truy cập (acl - access control list) là phần quan trọng nhất trong việc cấu hình Squid cache. Định nghĩa acl để cho phép, hoặc từ chối các acl được thực hiện bởi thẻ 'access'. Hay nói cách khác, bằng việc sử dụng các acl,

Squid được cấu hình và hoạt động như một firewall (lọc địa chỉ nguồn, đích, số cổng, giao thức, thời gian...).

Thẻ acl

Thẻ này được sử dụng để định nghĩa danh sách truy cập. Cú pháp sử dụng của thẻ acl như sau:

acl name type string1 ...
acl name type "file" ...

Khi sử dụng "file", file phải chứa mỗi đề mục trên một dòng riêng. 'name' là tên danh sách truy cập mà ta cần định nghĩa, 'type' là kiểu acl, bao gồm các kiểu sau:

- **acl name src ip-address/netmask ... :** Địa chỉ của các máy clients
- **acl name dst ip-address/netmask :** Định nghĩa địa chỉ của các máy URL.
- **acl name myip ip-address/netmask :** Địa chỉ IP của socket cục bộ.
- **acl name srcdomain .foo.com ...**

Tên miền để tìm kiếm ngược thành địa chỉ IP của các máy client. Được sử dụng để điều khiển các miền xác định nào đó.

- **acl name dstdomain .foo.com ...:** các server đích từ URL.
- **acl name srcdom_regex [-i] xxx ...** : các máy client có tên máy chứa với xâu chữ này. Ví dụ: acl name srcdom_regex pvkh, thì squid sẽ tìm từ 'pvkh' trong tên miền của máy client.
- **acl name dstdom_regex [-i] xxx...** : tương tự như trên, các máy server có tên chứa xâu chữ này.
- **acl name time [day-abbrevs] [h1:m1-h2:m2]:** định nghĩa danh sách truy cập về thời gian, trong đó day-abbrevs là ngày (thứ) trong tuần, được ký hiệu như sau:
  S - Sunday
  M - Monday
  T - Tuesday
  W - Wednesday
  H - Thursday
  F - Friday
  A - Saturday
  Lưu ý: h1:m1 phải nhỏ hơn h2:m2
- **acl name url_regex [-i] ^http://....** : các địa chỉ URL có tên đúng với tên được chỉ ra.
- **acl name urlpath_regex [-i]**: các xâu phù hợp trong đường dẫn của URL.
- **acl name port ..** : ta có thể điều khiển truy cập thông qua cổng địa chỉ.
- **acl name proto HTTP FTP**... : định nghĩa giao thức truyền.
- **acl name browser <string>**: mẫu biểu thức thông thường xác định thông tin về trình duyệt. Ví dụ:
  acl trinhduyet browser NETSCAPE

http_access deny trinhduyet

Squid sẽ không cho phép các yêu cầu từ trình duyệt Netscape của các máy client được phép truy cập cache.

- **acl name ident username** : xâu phù hợp với tên người dùng. Bạn có thể sử dụng ident để cho phép những người dùng xác định nào đó được truy cập cache của bạn. Điều này yêu cầu một tiến trình ident server chạy trên máy của người dùng. Trong file cấu hình squid.conf bạn viết một số dòng sau:

  ident_lookup on
  acl friends ident thai toannq khoa
  http_access allow friends
  http_access deny all
- **acl name src_as number**
- **acl name dst_as number:** Kiểu này không được sử dụng để điều khiển truy cập, các số AS (Autonomous System) nguồn (máy client) và đích (server) này có thể được sử dụng để định tuyến tới các cache xác định nào đó. Chúng tôi chưa thực hành trên 2 tham số này.
- **acl name proxy_auth username:** Danh sách các người dùng đúng, dùng REQUIRED để chấp nhận tên người dùng đúng. Chú ý: proxy_auth đòi hỏi một chương trình xác thực bên ngoài để kiểm tra tổ hợp username/password (bạn có thể xem thêm phần authenticate_program). Một lưu ý khác là kiểu proxy_auth sẽ không được sử dụng trong chế độ transparent proxy.
- **acl name snmp_community string:** Một xâu truyền thông để giới hạn truy cập cho SNMP Agent.
- **acl name maxconn number:** Giới hạn số kết nối lớn nhất của máy client có nhiều hơn <number> kết nối HTTP được thiết lập.

Định nghĩa ngầm định:

```
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl SSL_ports port 443 563
acl Safe_ports port 80 21 443 563 70 210 1025-65535
acl Safe_ports port 280 # http-mgmt
acl Safe_ports port 488 # gss-http
acl Safe_ports port 591 # filemaker
acl Safe_ports port 777 # multiling http
acl CONNECT method CONNECT
```

Thẻ http_access

Cho phép hoặc từ chối các truy cập tới cổng HTTP dựa trên các danh sách truy cập đã được định nghĩa. Các rule sẽ được đọc từ trên xuống dưới, nếu không có rule nào phù hợp, thì dòng cuối cùng trong danh sách rule được chuyển thành được phép (nếu nó deny) và ngược lại. Vì lý do này, nên thông thường ta sẽ đặt dòng 'deny all' hoặc 'allow all' ở cuối danh sách truy cập để tránh sự lộn xộn. Thẻ http_access được sử dụng như sau:

http_access allow | deny [!]aclname ...

Chú ý: với các danh sách được định nghĩa bằng acl, squid sẽ hiểu biểu thức trong đó là giá trị OR, còn với các thẻ 'access' này, Squid hiểu đó là những phép AND (logic). Ví dụ:

acl Safe_ports port 80 21 443 563 70

Squid hiểu Safe_port là những địa chỉ 80, 21 hoặc 443 hoặc ... Còn trong trường hợp sau:

acl localclient src 130.3.0.0/16

http_access allow localclient Safe_port

Squid sẽ chỉ cho phép các máy cục bộ và với số cổng truy cập có trong danh sách Safe_port mới được truy cập cache.

Thẻ icp_access

Cho phép hoặc từ chối các truy cập tới cổng ICP dựa trên các danh sách truy cập đã được định nghĩa. Thẻ này được sử dụng như sau:

icp_access allow | deny [!]aclname ...

Ví dụ: icp_access allow all sẽ trả lời tất các ICP query mà máy nhận được.

Thẻ miss_access

Được dùng để bắt các máy neighbour sử dụng máy của bạn như là sibling thay cho parent.

Thẻ cache_peer_access

Tương tự như cache_peer_domain, nhưng tuỳ chọn này được sử dụng phức tạp hơn với việc sử dụng các phần tử acl. Sử dụng:

cache_peer_access cache-host allow | deny [!]aclname

Thẻ proxy_auth_realm

Chỉ ra tên thực, được thông báo cho máy client khi xác thực proxy (phần text mà người dùng sẽ thông thấy khi nhắc nhập username và password). Ngầm định là:

proxy_auth_realm Squid proxy-caching web server

Thẻ ident_lookup_access

Tuỳ chọn này được dùng để thiết lập việc tìm kiếm nhiều người dùng trên hệ thống Unix.

4.9-Các tùy chọn liên quan đến việc quản trị hệ thống (gồm 6 thẻ)

Đoạn này báo cho squid biết người dùng và nhóm người dùng nào có quyền chạy squid, tên máy nào sẽ được hiển thị khi thông báo các lỗi và người quản trị cache - là người có thể xem thông tin chi tiết công việc thực hiện của squid.

Thẻ cache_mgr

Địa chỉ e-mail của người quản lý cache cục bộ.

Và các tuỳ chọn sau: **cache_effective_user, cache_effective_group, visible_hostname, unique_hostname, hostname_aliases.**

4.10- Các tùy chọn cho việc đăng ký cache server (gồm 4 thẻ)
Đoạn này được dùng để đăng ký cache server này tại địa chỉ http://ircache.nlanr.net/Cache/Tracker/, đây là một dịch vụ cung cấp thông tin trợ giúp cho người quản trị cache tìm kiếm một cache server khác theo thứ tự để join hoặc tạo ra các cây lưu trữ cache (hierarchy).

Gồm các 4 thẻ sau: **announce_period, announce_host, announce_file, announce_port.**

4.11- Các tùy chọn tăng tốc Web (gồm 5 thẻ)
Squid có thể được dùng như thiết bị nạp trung gian hoặc thiết bị làm giảm thời gian nạp cho Webserver. Nói chung squid không chỉ giúp cho các máy client mà còn giúp các web server giảm thời gian nạp từ phía server. Một số Web cache còn có thể hoạt động như là web server. Squid không được thiết kế để cấu hình theo cách này, do đó Squid chỉ là một Web cache, không phải là một Web server.

Với Transparent caching, Squid có thể được cấu hình một cách "tốt hơn", nó chấp nhận các yêu cầu Web đi ra ngoài và lưu trữ chúng trong cache. Khi các yêu cầu ra ngoài theo định dạng Web-server, nó sẽ chuyển chúng thành các yêu cầu định dạng cache.

Đoạn này cho phép cấu hình khác nhau liên quan tới chế độ tăng tốc và cũng cho chế độ transparent.

Thẻ httpd_accel_host

Xác định tên máy Web server, được dùng trong chế độ httpd_accelerator. Hoặc sử dụng trong chế độ Transparent cache, ta phải dùng từ khoá 'virtual' thay cho tên máy. Bạn có thể tham khảo mục 4.3 (Thiết lập chế độ Transparent kết hợp với ipchains).

Thẻ httpd_accel_port

Các yêu cầu được forward tới một cổng. Chương trình Squid sẽ thiết lập kết nối tới máy server tại cổng được thiết lập trong thẻ này. Thông thường là cổng 80, là cổng mà Web server lắng nghe các kết nối.

Thẻ httpd_accel_with_proxy on|off

Được sử dụng (tùy chọn on) khi ta sử dụng tùy chọn **httpd_accel_host**.

Thẻ httpd_accel_uses_host_header on|off

Các yêu cầu HTTP/1.1 chứa trường Host:header, dựa trên tên máy từ URL. Squid có thể đóng vai trò như bộ tăng tốc cho các HTTP server khác bằng

việc kiểm tra header này. Thông thường chỉ bật tùy chọn này khi cấu hình Squid chạy với Transparent proxy.

Và tuỳ chọn sau là: **httpd_accel_single_host** (chúng tôi chưa kiểm tra tùy chọn này).

4.12- Các tùy chọn khác (gồm 41 thẻ)
Đoạn này bao gồm các cấu hình về giới hạn độ lớn của logfile, hiển thị thông tin đã sửa đổi để gửi cho các máy client khi gặp lỗi hoặc bị từ chối truy cập, định nghĩa khối bộ nhớ cho Squid, quản lý mạng bằng việc sử dụng SNMP, chuyển các yêu cầu trực tiếp tới server gốc hoặc neighbour cache.

Đoạn này bao gồm các thẻ sau: **dns_test names, logfile_rotate, append_domain, tcp_recv_bufsize (bytes), err_html_text, deny_info, memory_pools on|off, memory_pools_limit (bytes), forwarded_for on|off, log_icp_queries on|off, icp_hit_stale on|off, minimum_direct_hops, minimum_direct_rtt, cachemgr_passwd, store_avg_object_size (kbytes), store_objects_per_bucket, client_db on|off, netdb_low, netdb_high, netdb_ping_period, query_icmp on|off, test_reachability on|off, buffered_logs on|off, reload_into_ims on|off, always_direct, never_direct, anonymize_headers, fake_user_agent, icon_directory, error_directory, minimum_retry_timeout (seconds), maximum_single_addr_tries, snmp_port, snmp_access, snmp_incoming_address, snmp_outgoing_address, as_whois_server, wccp_router, wccp_version, wccp_incoming_address, wccp_outgoing_address,**

4.13-Các tùy chọn giới hạn băng tần (gồm 41 thẻ)
Delaypool thực hiện với các ACL. Delay pool cung cấp một cách để giới hạn băng tần của các yêu cầu xác định dựa trên bất kỳ danh sách nào đó. Cách đối xử của Delay được lựa chọn bởi ACL. Trong các delaypool của ISP có thể bổ sung một mạng riêng biệt để cung cấp chất lượng dịch vụ.

Mục này bao gồm các thẻ sau: d**elay_pools, delay_class, delay_access, delay_parameters, delay_initial_bucket_level (percent, 0-100), incoming_icp_average, incoming_http_average, incoming_dns_average, min_icp_poll_cnt, min_dns_poll_cnt, min_http_poll_cnt, max_open_disk_fds, offline_mode, uri_whitespace, broken_posts, mcast_miss_addr, mcast_miss_ttl, mcast_miss_port, mcast_miss_encode_key, nonhierarchical_direct, prefer_direct, strip_query_terms, coredump_dir, redirector_bypass, ignore_unknown_nameservers, digest_generation, digest_bits_per_entry, digest_rebuild_period (seconds), digest_rewrite_period (seconds), digest_swapout_chunk_size (bytes), digest_rebuild_chunk_percentage (percent, 0-100), chroot,**

client_persistent_connections, server_persistent_connections, pipeline_prefetch, extension_methods, high_response_time_warning, high_page_fault_warning, high_memory_warning, store_dir_select_algorithm, ie_refresh.

4.14-Các tuỳ chọn hỗ trợ SSL

Đây là một tuỳ chọn mới, được hỗ trợ từ phiên bản **squid-2.5** trở lên, cho phép cấu hình Squid kết hợp với SSL. Khi sử dụng chức năng này, squid sẽ dùng server certificate để thực hiện xác thực và trao đổi khoá với client. Tuy nhiên, các tuỳ chọn thuộc nhóm này chỉ khả dụng khi ta biên dịch Squid với tùy chọn **--enable-ssl**.

Các kết nối an toàn thông qua dịch vụ Web sử dụng giao thức HTTPS. Với giao thức này thì phía Server thường sử dụng cổng 443 để đợi các yêu cầu kết nối an toàn (https) từ phía Client. Hiện nay, Squid chỉ hỗ trợ 2 tuỳ chọn sau:

Thẻ https_port

Chỉ ra địa chỉ cổng (socket) mà Squid sẽ lắng nghe các yêu cầu kết nối bảo mật HTTPS từ phía Client. Tuỳ chọn này chỉ được sử dụng khi bạn chạy Squid trong chế độ tăng tốc (Accelerator). Bạn có thể chỉ ra nhiều địa chỉ cổng trên nhiều dòng với các chứng chỉ SSL và/hoặc các tuỳ chọn sau:

cert = đường dẫn tới chứng chỉ SSL của squid proxy (theo định dạng PEM).

key = đường dẫn tới file chứa khoá bí mật của squid proxy (theo định dạng PEM).

version = Phiên bản của SSL/TLS được hỗ trợ
- 1 Tự động (ngầm định).
- 3 SSLv3
- 4 TLSv1

cipher = Danh sách các mã pháp được hỗ trợ ngăn cách nhau bởi dấu ':'.
options = Các tuỳ chọn khác, quan trọng nhất là:
- NO_SSLv3 Không sử dụng SSLv3.
- NO_TLSv1 Không sử dụng TLSv1.

Thẻ ssl_unclean_shutdown

Các trình duyệt sẽ đưa ra thông báo lỗi khi SSL shutdown.

Chương II
Tích hợp mật mã cho Proxy

1-Giới thiệu về OpenSSL và MySSL

OpenSSL phiên bản 0.9.7 của Eric A. Young và Tim J. Hudson. MySSL cung cấp giao thức truyền thông an toàn trên Internet là SSLv3 (Secure Sockets Layer) và TLSv1(Transport Layer Security) với các thư viện mã hoá mạnh. Bộ thư viện **OpenSSL** phiên bản 0.9.7 có tổng dung lượng là **14MB** với **1479 file** trong đó có **643 file mã nguồn C** với dung lượng là **5,8MB**.

OpenSSL hỗ trợ rất nhiều mã pháp, hàm băm, các thuật toán trao đổi khoá, chữ ký số và xác thực.

- Mã pháp: DES, RC2, RC4, RC5, IDEA, BLOWFISH, AES, CAST, RIJNDEAL
- Hàm băm: MD2, MD4, MD5, SHA-1, SHA-0, RIPEMD, MDC2
- Trao đổi khoá: RSA, DH, FORTEZZA
- Chữ ký số: DSA, RSA, EC

OpenSSL hỗ trợ ba giao thức truyền thông an toàn trên Internet là:

- SSLv2
- SSLv3
- TLSv1

Với mục đích tích hợp các giải pháp mật mã của ngành vào bộ OpenSSL chúng tôi đã thực hiện những công việc sau đây:

- Loại bỏ những phần mã nguồn không sử dụng, những file và tài liệu không cần thiết.
- Loại bỏ giao thức SSLv2. Giao thức này đã được phát hiện là có một vài kẽ hở.
- Loại bỏ tất cả các thuật toán mã hoá, thêm vào duy nhất một thuật toán mã hoá MK1 của ngành.
- Loại bỏ các thuật toán băm ngoại trừ MD5 và SHA-1.
- Loại bỏ các thuật toán ký, trao đổi khoá và chỉ để lại RSA.
- Loại bỏ chương trình sinh tham số RSA (sinh số nguyên tố theo xác suất)
- Tích hợp thuật toán sinh số nguyên tố mạnh thoả mãn tiêu chuẩn an toàn.
- Tích hợp thuật toán mã hoá MK1 của ngành

Sau khi thu gọn, bộ thư viện **MySSL** có tổng dung lượng là **3MB** với **400 file** trong đó có **300 file mã nguồn .C** với kích thước là **1,7MB**.

2-Cấu trúc file và thư mục của MySSL

2.1-Các file và thư mục gốc

Ở mức ngoài cùng, MySSL bao gồm 7 file và thư mục được mô tả trong bảng dưới đây.

TT	File, Thư mục	Mô tả	Kích cỡ	Ghi chú
1	Thư mục **apps**	Thư mục này bao gồm các chương trình nhỏ thực hiện một số chức năng quan trọng như tạo khoá RSA (genrsa.c), sinh request (req.c), ký và kiểm tra certificate (X509.c, verify.c)…	3,95MB	Trong 4MB này chỉ có 500k file mã nguồn C (26 file), còn lại là hai file thư viện gmp.a và mpfr.a phục vụ cho các phép toán trong genrsa.c
2	Thư mục **crypto**	Gồm 28 thư mục và 10 file .c, thư mục này cung cấp các mã pháp, hàm băm, thuật toán nén, các chuẩn pkcs7, pkcs8, pkcs12 và x509 được sử dụng trong MySSL.	1,85MB	
3	Thư mục **ssl**	Gồm 30 file .c tạo nên thư viện libssl.a, sử dụng cho các chương trình truyền thông viết theo giao thức SSLv3 hoặc TSLv1 (như Web Apache Server, Mozilla client).	350KB	
4	Thư mục **include**	Chứa các file .h	740KB	
5	File **config**, **Configure**, **Makefile.orig**,	Các file cấu hình để thiết lập tuỳ chọn ban đầu trong Makefile.ssl	43KB	
6	File **Makefile.ssl**	Fiel này được tạo ra khi chạy lệnh ./config [tuỳ chọn]	10KB	
7	File **Makefile**			Link đến Makefile.ssl

2.2-Mô tả thư mục apps

Mỗi ứng dụng trong thư mục này được viết riêng trong một file C nhưng khi biên dịch chỉ tạo ra một file chạy là **myssl**, các ứng dụng được kết hợp ở trong **myssl**.

myssl có ba chế độ hoạt động.

1) Chạy lệnh **./myssl** mà không có tham số thì MySSL sẽ chuyển về chế độ dòng lệnh, khi đó xuất hiện dấu nhắc MySSL>_, các ứng dụng được thực thi bằng cách gọi tên và tham số của nó (tên của ứng dụng trùng với tên file .c).
2) Chạy một ứng dụng bằng cách gọi lệnh **./myssl** với tham số đầu tiên là tên ứng dụng, các tham số tiếp theo là tham số của ứng dụng.
3) Đổi tên myssl thành tên của ứng dụng (hoặc tạo liên kết) rồi thực thi ứng dụng.

Chú ý: Nếu gọi sai tên ứng dụng, MySSL sẽ liệt kê tên các ứng dụng của nó. Sử dụng tham số -h sẽ cho những trợ giúp cần thiết.

Chức năng của các ứng dụng được mô tả trong bảng sau.

TT	File C	Tên ứng dụng	Mô tả	Size	Ghi chú
1	ca.c	ca	Sử dụng để ký một Certificate, tạo ra các CRL, hiển thị Certificate ở dạng text	71	
2	crl.c	crl	Sử lý các file CRL (chẳng hạn như chuyển một CRL từ dạng PEM sang dạng DER, hiển thị CRL ở dạng text, …)	7,5	
3	dgst.c	dgst	Sử dụng hàm Hash (MD5 hoặc SHA1) để biến đổi một file đầu vào thành một file đầu ra là dữ liệu đã được xử lý bởi MD5 hoặc SHA1	7,8	
4	enc.c	enc	Mã hoá một file sử dụng thuật toán MK1	12	
5	genrsa.c	genrsa	Tạo ra khoá bí mật RSA (khoá công khai e=65537)	35	

6	pkcs12.c	pkcs12	Chuyển Certificate thành định dạng PKCS #12 được sử dụng cho một số chương trình như Netscape và MS Outlook. Ngoài ra tiện ích này còn được sử dụng để chuyển đổi, hiển thị một Certificate dạng p12 thành các dạng khác (PEM, text).	24,4	
7	req.c	req	Xử lý và tạo Certificate theo định dạng PKCS #10	30,7	
8	rsa.c	rsa	Tiện ích liên quan đến khóa RSA, bao gồm việc chuyển đổi, kiểm tra khóa RSA, in các tham số liên quan.	6,78	
9	rsautil.c	rsautil	Tiện ích để thực hiện hệ mật, hệ chữ ký số RSA	7	
10	verify.c	verify	Tiện ích này sử dụng để chứng thực Certificate	7, 4	
11	version.c	version	Hiển thị thông tin về phiên bản của MySSL	1,5	
12	x509.c	x509	Ký và hiển thị thông tin về Certificate	29	
13	app_rand.c	Gồm các hàm phục vụ đọc và ghi file dữ liệu giả ngẫu nhiên, được sử dụng bởi các tiện ích ở trên		2,3	
14	apps.c	Các hàm được sử dụng bởi myssl.c		4,5	
15	myssl.c	Gắn kết các tiện ích ở trên để tạo ra một file chạy là myssl		9	
Tổng: 15 file C trong đó có 12 ứng dụng nhỏ (1-12)				280	

2.3-Mô tả thư mục crypto

Thư mục crypto cung cấp thư viện mật mã cho MySSL, bao gồm các thuật toán được sử dụng trong nhiều chuẩn Internet (SHA1, MD5). Ngoài ra còn cung cấp một thư viện mã khối MK1 do Phân viện Khoa học thiết kế. Trong MySSL, thư viện mật mã này được sử dụng bởi các giao thức SSL version 3, TLS version 1 và S/MIME.

Các file trong thư mục này tạo ra thư viện libcrypto.a

TT	Thư mục	Mô tả	Size (KB)	Ghi chú
1	asn1	ASN.1 là một cách để mã hóa dữ liệu kiểu cấu trúc sang dạng nhị phân	286	Có 76 file C
2	bio	Đây là thư viện liên quan đến các kiểu vào ra	135	20 file C
3	bn	Thư viện số lớn Big Number	141	24 file C
4	buffer	Cấp phát bộ nhớ cho các con trỏ	3,5	
5	comp	Cung cấp phương pháp nén trong MySSL	10	
6	conf	Các hàm cho phép các ứng dụng đọc file cấu hình của MySSL	46	
7	dso	Thư viện động	28,4	
8	err	Các hàm thông báo lỗi	24	
9	evp	Khai báo mã pháp, phương pháp ký, hàm băm	100	
10	hmac	Cung cấp hàm băm MD5 hoặc SHA1	3,6	
11	lhash	Thư viện này được dùng để tạo và sử lý các hash table (lưu trữ các kiểu dữ liệu lớn)	13,6	
12	md5	Hàm băm MD5	14	
13	mk1	Mã khối MK1	12,4	
14	objects	Thêm phần định danh cho các đối tượng (cấu trúc, hàm …)	340	
15	ocsp	Giao thức xác định trạng thái thực của Certificate	61	
16	pem	Các hàm trong thư mục này phục vụ việc đọc hoặc ghi các cấu trúc theo định dạng PEM (Dữ liệu được encode theo Base64 và được đóng khung bởi các dòng header và footer)	40,4	
17	pkcs12	PKCS #12 là chuẩn cú pháp để biểu diễn thông tin cá nhân (khóa bí mật, khóa công khai, chứng chỉ số). Chuẩn này được một số ứng dụng hỗ trợ như Netscape, MS OutLook. Các hàm trong thư mục này được sử dụng bởi tiện ích pkcs12.c trong thư mục apps	44,2	

18	pkcs7	PKCS #7 là chuẩn cú pháp của một thông báo được mã hóa	78,3	
19	rand	Thư viện sinh số giả ngẫu nhiên	41,9	
20	rsa	Cung cấp phương pháp ký và mã hóa RSA	49,9	
21	sha	Thuật toán băm SHA-1	15,2	
22	stack	Lưu trữ danh sách các đối tượng đã được sắp xếp	6,89	
23	txt_db	Cơ sở dữ liệu dạng văn bản	7,5	
24	ui	Cho phép các hàm giao tiếp với người sử dụng	31,3	
25	x509	Thư viện này cung cấp các hàm phục vụ nhiều mục đích như hiển thị thông tin về Certificate, ký Certificate Request, sửa đổi những thiết lập trong một Trusted Certificate	120	
26	x509v3	X509 version 3	114	

2.4-Mô tả thư mục ssl

Thư mục ssl cung cấp các hàm để thiết kế nên hai giao thức là SSLv3 và TLSv1. SSLv3 là SSL version 3 (Secure Socket Layer) do Netscape Corporation phát triển từ phiên bản 2. TLSv1 (Transport Layer Security) là giao thức được thiết kế dựa trên SSLv3.

TT	File	Size(KB)	Mô tả
1	s23_clnt.c	7	Các hàm trong các file s23_ được sử dụng để tương thích với phiên bản 2 của SSL (SSLv2)
6	s3_clnt.c	33	Các hàm liên quan đến SSL client (gửi Client hello, nhận Server hello, trao đổi khóa, certificate với SSL server)
7	s3_enc.c	14	Các hàm thực hiện việc tạo khóa và hàm mã hóa kết nối SSL
8	s3_lib.c	16	Các hàm thực hiện kết nối SSL trong quá trình bắt tay của giao thức SSLv3
9	s3_meth.c	0,6	Cung cấp SSL3 method
10	s3_pkt.c	29	Gồm các hàm gửi và nhận dữ liệu, hàm thực hiện nén, giải nén dữ liệu
11	s3_srvr.c	36	Các hàm liên quan đến SSL server (bao gồm các hàm gửi Server hello, nhận Client hello,

			trao đổi khóa, gửi yêu cầu Certificate, kiểm tra Certificate)
12	s3_both.c	12	Các hàm gửi nhận dữ liệu, được xây dựng để sử dụng chung cho cả SSLv3 client và SSLv3 server.
13	ssl_algs.c	0,6	File này chỉ có một hàm khởi tạo các mã pháp và hàm băm được hỗ trợ trong thư mục crypto
14	ssl_asn1.c	8	Lưu trữ và ghi cấu trúc SSL_SESSION theo định dạng ASN1
15	ssl_cert.c	15	Các hàm liên quan đến Certificate trong SSL (thêm, chèn một Cert vào ngăn xếp,…)
16	ssl_ciph.c	23	Tạo một danh sách các mã pháp, lựa chọn phương pháp nén. Trao đổi mã pháp
17	ssl_lib.c	46	Các hàm quan trọng tạo nên thư viện SSL (liên quan đến cấu trúc SSL_CTX, là một cấu trúc chứa thông tin chung cho các kết nối SSL)
18	ssl_rsa.c	15	Các hàm liên quan đến khoá RSA và certificate
19	ssl_sess.c	17	Các hàm tạo và thiết lập phiên liên lạc SSL (SSL_SESSION)
20	ssl_stat.c	14	Trạng thái của phiên liên lạc SSL
21	ssl_txt.c	3	In ra file hoặc BIO các thông tin về phiên liên lạc của SSL

3-Các thuật toán mật mã trong MySSL

MySSL hỗ trợ duy nhất một thuật toán mã khối là MK1 cùng với hai hàm băm thông dụng là MD5, SHA1. Ngoài ra thuật toán mã và ký RSA cũng được cung cấp.

Trong phần này chúng ta sẽ mô tả sơ qua về các mã khối, hàm băm và thuật toán mã, ký RSA.

3.1-Thuật toán mã khối MK1

MK1 là thuật toán mã khối do tác giả Trần Văn Trường thuộc phân viện Khoa học Mật mã thiết kế. Thuật toán này đã được kiểm chứng và sử dụng nhiều trong ngành cơ yếu Việt Nam. Đây là thuật toán mã khối với dữ liệu đầu vào là 128 bit, khoá là 512 bit.

Trong MySSL, thuật toán mã hoá này được xây dựng trong thư mục **crypto/mk1** với hai chế độ sử dụng là CBC và ECB.

Mô tả các file C trong mk1

TT	Tên file	Size	Mô tả
1	mk1_core.c	4	Các hàm thực hiện mã/giải mã theo thuật toán MK1
2	m1_cbc.c	0,8	Mã hoá MK1 theo mode CBC
3	mk1_ecb.c	0,2	Mã hoá MK1 theo mode ECB

- **File mk1_core.c**

```
int MK1_set_encrypt_key(const unsigned char *userKey, const int bits,
            MK1_KEY *key)
```

Hàm này thực hiện việc mở rộng khoá trong 'userKey' thành khoá lưu trong 'key' với độ dài 'bits'. Hàm này được gọi trong quá trình mã hoá.

Cấu trúc MK1_KEY được định nghĩa thông qua mk1_key_st.

```
struct mk1_key_st {
      unsigned long rd_key[32];
      int rounds;
}
```

```
int MK1_set_decrypt_key(const unsigned char *userKey, const int bits,
             MK1_KEY *key) {
```

Tương tự như hàm trên nhưng sử dụng cho quá trình giải mã.

```
void MK1_encrypt(const unsigned char *in, unsigned char *out,
             const MK1_KEY *key) {
```

Mã hoá một khối 16 byte.

```
void MK1_encrypt(const unsigned char *in, unsigned char *out,
             const MK1_KEY *key) {
```

Giải mã một khối 16 byte.

```
void MK1(unsigned short *data,unsigned short *key, int fEnc)
```

Hàm thực hiện việc mã hoá hoặc giảm mã dữ liệu trong 'data' dựa vào cờ mã hoá fEnc. Dữ liệu đầu ra cũng được trả về trong 'data'.

- **File mk1_cbc.c**

```
void MK1_cbc_encrypt(const unsigned char *in, unsigned char *out,
                const unsigned long length, const MK1_KEY *key,
                unsigned char *ivec, const int enc) {
```

Thực hiện việc mã hoá và giải mã theo mode CBC (Cipher Block Chaining mode).

- **File mk1_ecb.c**

```
void MK1_ecb_encrypt(const unsigned char *in, unsigned char *out,
                const MK1_KEY *key, const int enc) {
```

Thực hiện việc mã hoá và giải mã theo mode ECB (Electronic CodeBook mode).

3.2-Thuật toán mã hoá công khai và ký RSA

Thư viện mã hoá công khai và ký được cung cấp trong thư mục **crypto/rsa**. Thư viện này có 4 hàm chính. Hai hàm thực hiện việc ký và kiểm tra. Hai hàm thực hiện việc mã hoá và giải mã. Ký và mã hoá RSA được sử dụng trong giai đoạn bắt tay giữa client và server trong kết nối SSL.

- Phương pháp ký RSA trong MySSL sử dụng padding theo PKCS1 version 1.5
- Phương pháp mã hoá RSA trong MySSL cũng sử dụng padding theo PKCS1 version 1.5 (quá trình mã hoá RSA được sử dụng trong quá trình trao đổi khoá và trong PKCS7)

Mô tả file trong rsa

TT	Tên file	Size(KB)	Mô tả
1	rsa_asn1.c	2	Chuyển đổi ASN1 đối với cấu trúc RSA
2	rsa_chk.c	3	Kiểm tra các tham số RSA
3	rsa_eay.c	14	Thư viện chính
4	rsa_gen.c	3	Sinh tham số cho RSA
5	rsa_lib.c	6	Giao diện
6	rsa_oaep.c	5	Chế độ Padding theo phương pháp OAEP (Optimal Asymmetric Encryption Padding) (có trong PKCS1 version 2.0)
7	rsa_pk1.c	4	RSA padding PKCS1 version 1.5 (mặc định)
8	rsa_sign.c	5	Hệ chữ ký RSA

Thư viện RSA được xây dựng trên cơ sở của thư viện Bignum bn, trong đó có sử dụng các hàm của thư viện X509 để nạp và điều khiển các hàm của thư viện RSA. Thư viện RSA trong MySSL tương thích với chuẩn PKCS1 (chuẩn mã hoá RSA). Vấn đề Padding trong RSA sử dụng theo PKCS1 version 1.5, ngoài ra có hỗ trợ phương pháp padding an toàn RSA OAEP (PKCS1 version 2.0).

Các hàm trong thư viện RSA có thể sử dụng theo bất kỳ độ lớn nào của RSA modulo. Hiện nay người ta thường dùng RSA 1024 bits. RSA 4096 bits là rất rất an toàn nhưng thời gian sử lý sẽ lâu hơn nhiều so với 1024.

Thư viện RSA có một cấu trúc riêng là RSA để lưu trữ dữ liệu, nó bao gồm 8 biến BIGNUM để lưu khoá bí mật và công khai RSA. Cấu trúc này được định nghĩa thông qua rsa_st trong file rsa.h.

```
struct rsa_st
```

```
{
int pad;
long version;
const RSA_METHOD *meth;
BIGNUM *n;
BIGNUM *e;
BIGNUM *d;
BIGNUM *p;
BIGNUM *q;
BIGNUM *dmp1;
BIGNUM *dmq1;
BIGNUM *iqmp;
CRYPTO_EX_DATA ex_data;
int references;
int flags;
BN_MONT_CTX *_method_mod_n;
BN_MONT_CTX *_method_mod_p;
BN_MONT_CTX *_method_mod_q;
char *bignum_data;
BN_BLINDING *blinding;
};
```

Trong đó, các số lớn n, e, d, p ,q là những tham số của hệ mật RSA. ‘dmp1’ là biến để lưu giá trị ‘d modulo p-1’. Tương tự như vậy thì dmq1 là ‘d mod q-1’ và iqmp là nghịch đảo của ‘q mod p’. Cấu trúc RSA_METHOD để xác định các hàm mã, giải mã, ký và kiểm tra của hệ mật RSA.

Cấu trúc RSA_METHOD được xây dựng để khai báo các hàm cho thư viện RSA. File rsa_lib sẽ sử dụng cấu trúc này để tạo ra giao diện cho thư viện RSA. RSA_METHOD được định nghĩa trong file rsa.h như sau.

```
typedef struct rsa_st RSA;

typedef struct rsa_meth_st
      {
      const char *name;
      int (*rsa_pub_enc)(int flen,const unsigned char *from,
            unsigned  char *to,RSA *rsa,int padding);
      int (*rsa_pub_dec)(int flen,const unsigned char *from,
            unsigned char *to, RSA *rsa,int padding);
      int (*rsa_priv_enc)(int flen,const unsigned char *from,
            unsigned char *to,RSA *rsa,int padding);
      int (*rsa_priv_dec)(int flen,const unsigned char *from,
            unsigned char *to,RSA *rsa,int padding);
      int (*rsa_mod_exp)(BIGNUM *r0,const BIGNUM *I,RSA *rsa);
      int (*bn_mod_exp)(BIGNUM *r, const BIGNUM *a, const BIGNUM *p,
            const BIGNUM *m, BN_CTX *ctx,BN_MONT_CTX *m_ctx);
      int (*init)(RSA *rsa);
      int (*finish)(RSA *rsa);
      int flags;
      char *app_data;
      int (*rsa_sign)(int type,const unsigned char *m, unsigned int
            m_length, unsigned char *sigret, unsigned int *siglen,
            const RSA *rsa);
      int (*rsa_verify)(int dtype, const unsigned char *m,
```

```
            unsigned int m_length, unsigned char *sigbuf, unsigned int
            siglen, const RSA *rsa);
      } RSA_METHOD;
```

Các hàm trong cấu trúc này sẽ được gán cho các hàm tương ứng trong rsa_eay.c. Sau đây chúng ta sẽ mô tả các file của thư viện RSA.

- **File rsa_eay.c**

`const RSA_METHOD *RSA_PKCS1_SSLeay(void)`

Hàm này trả về địa chỉ của một hằng cấu trúc rsa_pkcs1_eay_meth kiểu RSA_METHOD. Hằng cấu trúc này được gán như sau.

```
static RSA_METHOD rsa_pkcs1_eay_meth={
      "PKCS#1 RSA",
      RSA_eay_public_encrypt,
      RSA_eay_public_decrypt, /* signature verification */
      RSA_eay_private_encrypt, /* signing */
      RSA_eay_private_decrypt,
      RSA_eay_mod_exp,
      BN_mod_exp_mont,
      RSA_eay_init,
      RSA_eay_finish,
      0, /* flags */
      NULL,
      0, /* rsa_sign */
      0  /* rsa_verify */
            };
```

Các hàm trong cấu trúc rsa_pkcs1_eay_meth được xây dựng với chức năng được mô tả như dưới đây.

```
static int RSA_eay_public_encrypt(int flen, const unsigned char *from,
            unsigned char *to, RSA *rsa, int padding)
```

Hàm này thực hiện phép mã hoá RSA. 'from' là giá trị đầu vào với độ dài là 'flen'. 'from' sẽ được padding mặc định theo PKCS1 version 1.5 (chỉ ra bởi 'padding') bằng việc gọi hàm RSA_padding_add_PKCS1_type2(). Sau khi đã padding, 'from' sẽ được biến đổi thành số lớn f kiểu BIGNUM (hàm BN_bin2bn), gọi hàm lũy thừa modulo để tính ret=f^e mod n. Các số n và e là những tham số khoá công khai của RSA đã được kích hoạt trong 'rsa'. Giá trị trả về ret sẽ được biến đổi thành chuỗi bởi hàm BN_bn2bin() để ghi vào 'to'.

Giá trị trả về của hàm là –1 nếu có lỗi, ngược lại trả về số byte của RSA modulo n.

```
static int RSA_eay_private_encrypt(int flen, const unsigned char *from,
            unsigned char *to, RSA *rsa, int padding)
```

Thực hiện việc ký theo RSA. Tương tự như hàm RSA_eay_public_encrypt() nhưng khác ở chỗ tính ret=$from^d$ mod n. Nếu có lỗi hàm sẽ trả về -1, còn không thì trả về số byte trong 'from'.

```
static int RSA_eay_private_decrypt(int flen, const unsigned char *from,
          unsigned char *to, RSA *rsa, int padding)
```

Hàm giải mã RSA. Tính ret=from^d mod n. Sau đó loại bỏ padding bằng việc gọi hàm RSA_padding_check_PKCS1_type_2 (mặc định).Nếu có lỗi hàm sẽ trả về -1 còn không thì trả về số byte trong 'from'.

```
static int RSA_eay_public_decrypt(int flen, const unsigned char *from,
          unsigned char *to, RSA *rsa, int padding)
```

Hàm này thực hiện việc kiểm tra chữ ký số. Tương tự như hàm trên nhưng luỹ thừa sử dụng là khoá công khai e: ret=from^e mod n. Sau đó loại bỏ padding.

Nếu có lỗi hàm sẽ trả về -1, còn không thì trả về số byte trong 'from'.

```
static int RSA_eay_finish(RSA *rsa)
```

Giải phóng ba biến kiểu BN_MOT_CTX trong 'rsa'.

```
static int RSA_eay_init(RSA *rsa)
```

Gán biến 'flags' trong 'rsa' như sau.

```
rsa->flags|=RSA_FLAG_CACHE_PUBLIC|RSA_FLAG_CACHE_PRIVATE;
```

Sau đó trả về giá trị 1.

- **File rsa_lib.c**

File này sử dụng các hàm của file rsa_eay.c để tạo nên các hàm giao diện chung trong MySSL.

```
RSA *RSA_new_method()
```

Tạo và cấp phát bộ nhớ cho một biến 'rsa' kiểu cấu trúc RSA. Sau đó gán cho rsa->meth bởi hàm RSA_PKCS1_SSLeay() trong file rsa_eay.c. Các biến còn lại trong 'rsa' được gán bằng 0 hoặc NULL.

```
void RSA_free(RSA *r)
```

Giải phóng biến 'r'.

```
int RSA_size(const RSA *r)
```

Trả về số byte của modulo n trong cấu trúc RSA 'r'.

```
void RSA_blinding_off(RSA *rsa)
```

Tắt cờ blinding trong biến flags của 'rsa'. Giải phóng các biến blinding tương ứng.

```
int RSA_blinding_on(RSA *rsa, BN_CTX *p_ctx)
```

Thiết lập blinding trong 'rsa' để chống lại timing attack. Đây là dạng tấn công mà kẻ ở giữa có thể đo thời gian sử lý các phép tính trong RSA để lấy được khoá bí mật d. Để chống lại tấn công này người ta dùng một số ngẫu nhiên r và thay vì thực hiện tiến trình giải mã RSA thông thường m=c^d mod

n, chúng ta thực hiện $m=r^{-1}(c.r^e)^d \bmod n$ với r^{-1} là nghịch đảo của r mod n. Và như vậy việc đo thời gian giải mã của kẻ tấn công sẽ không chính xác do tham số ngẫu nhiên r.

Tấn công này chỉ áp dụng cho thời gian thực. Trong thực hành dạng tấn công này cũng rất khó thực hiện.

Hàm này sẽ thực hiện việc tạo ra số ngẫu nhiên r nhỏ hơn modulo n của RSA, sau đó tính $r^{-1} \bmod n$ và gán các giá trị trả về trong cấu trúc 'p_ctx' (bao gồm cả việc thiết lập cờ BLINDING). Và trong hàm giải mã RSA, nếu cờ BLINDING được thiết lập thì giá trị ngẫu nhiên r sẽ được sử dụng.

- **File rsa_sign.c**

Có hai hàm thực hiện việc ký một thông báo và kiểm tra. Hai hàm này sử dụng hai hàm tương ứng đã trình bày trong file rsa_eay.c

```
int RSA_sign(int type, const unsigned char *m, unsigned int m_len,
           unsigned char *sigret, unsigned int *siglen, RSA *rsa)
```

Thực hiện việc ký chuỗi 'm' với độ dài 'm_len' để tạo thành chuỗi chữ ký trong 'sigret' với độ dài 'siglen' sử dụng hàm RSA_private_encrypt(tương đương với hàm RSA_eay_private_encrypt).

```
int RSA_verify(int dtype, const unsigned char *m, unsigned int m_len,
           unsigned char *sigbuf, unsigned int siglen, RSA *rsa)
```

Thực hiện việc kiểm tra chữ ký số đối với thông báo 'm', giá trị chữ ký số ở trong 'sigbuf'. Giá trị trả về là 1 nếu đúng, ngược lại quá trình kiểm tra chữ ký số lỗi hoặc không đúng chữ ký số.

- **File rsa_gen.c**

```
RSA *my_RSA_generate_key(int bits, void (*callback)(int,int,void *),
                 void *cb_arg, char *kqn, char *kqe, char *kqd,
                 char *kqp, char *kqq)
```

Hàm này được dùng để thiết lập khoá cho RSA. Quá trình sinh khoá được viết trong file genrsa.c của thư mục apps. Trong hàm này chỉ sử dụng những tham số đã có (kqe, kqd, kqp, kqq, kqn) kiểm tra và chuyển đổi thành các số BIGNUM rồi gán cho các biến tương ứng trong cấu trúc RSA (e, d, p, q, n). Sau đó tính một số giá trị còn lại như dmp1, dmq1 và ipmq.

Thuật toán sinh tham số cho hệ mật RSA tham khảo [Khoa]

- **File rsa_pk1.c**

Thực hiện padding trong RSA theo PKCS1 version 1.5. Ký thì dùng type 1 còn mã thì dùng type 2.

```
int RSA_padding_add_PKCS1_type_1(unsigned char *to, int tlen,
           const unsigned char *from, int flen)
```

Padding ‘flen’ byte trong ‘from’ vào ‘to’ (theo PKCS1 type 1).
Kết quả được một chuỗi dạng:
to=[01][0xff với độ dài tlen-flen-3][from]

```
int RSA_padding_check_PKCS1_type_1(unsigned char *to, int tlen,
          const unsigned char *from, int flen, int num)
```

Loại bỏ padding trong ‘from’ để trả về dữ liệu nguyên bản trong ‘to’ (PKCS1 type 1)

```
int RSA_padding_add_PKCS1_type_2(unsigned char *to, int tlen,
          const unsigned char *from, int flen)
```

Padding ‘flen’ byte từ ‘from’ thành ‘to’ (theo PKCS1 type 2).
Kết quả được một chuỗi dạng:
to=[02][các số ngẫu nhiên với độ dài tlen-flen-3][from]

- **File rsa_chk.c**

```
int RSA_check_key(const RSA *key)
```

Kiểm tra các tham số RSA trong ‘key’: p, q có là số nguyên tố không, n có bằng p.q không,…

- **File rsa_asn1.c**

Khai báo một cấu trúc ASN1_METHOD trong đó có 4 hàm như sau.

```
static ASN1_METHOD method={
        (int (*)())  i2d_RSAPrivateKey,
        (char *(*)())d2i_RSAPrivateKey,
        (char *(*)())RSA_new,
        (void (*)()) RSA_free};

ASN1_METHOD *RSAPrivateKey_asn1_meth(void)
      {
      return(&method);
      }
```

Sau đó các khai báo các Macro ASN1 để tạo các hàm i2d_RSAPrivateKey() và d2i_RSAPriveteKey().

3.3-Thuật toán băm MD5

Đây là thuật toán băm mà dữ liệu đầu vào được ‘hasing’ để thành đầu ra có độ dài là 16 byte.

Trong MySSL thuật toán MD5 được cung cấp trong thư mục **crypto/md5**. Các hàm trong thư viện này sử dụng một cấu trúc chung là MD5_CTX được định nghĩa trong file md5.h.

```
typedef struct MD5state_st
      {
```

```
MD5_LONG A,B,C,D;
MD5_LONG Nl,Nh;
MD5_LONG data[MD5_LBLOCK];
int num;
} MD5_CTX;
```

Trong đó MD5_LBLOCK là 16, MD5_LONG được định nghĩa là kiểu unsigned long.

`int MD5_Init(MD5_CTX *c)`

Khởi tạo và gán các giá trị ban đầu của 'c' theo thuật toán MD5.

`int MD5_Update(MD5_CTX *c, const void *data, unsigned long len)`

Hàm này được định nghĩa thông qua hàm HASH_UPDATE trong file md32_common.c ở thư mục crypto. Hàm này thực hiện việc băm dữ liệu trong 'data' theo thuật toán MD5 (sử dụng hàm MD5_Transform).

`int MD5_Final(unsigned char *md, MD5_CTX *c)`

Tính đầu ra 'md' từ cấu trúc 'c' theo MD5.

`void md5_block_host_order (MD5_CTX *c, const void *data, int num)`

Đây là hàm thực hiện việc biến đổi dữ liệu theo công thức của thuật toán MD5. Hàm này sẽ được định nghĩa là MD5_Transform()

3.4-Thuật toán băm SHA1

Đây là thuật toán băm mà dữ liệu đầu vào được 'hashing' để thành đầu ra có độ dài là 20 byte.

Thuật toán này được cung cấp trong thư mục **crypto/sha**. Các hàm trong thư viện này sử dụng một cấu trúc chung là SHA_CTX được định nghĩa trong file sha.h.

```
typedef struct SHAstate_st
      {
      SHA_LONG h0,h1,h2,h3,h4;
      SHA_LONG Nl,Nh;
      SHA_LONG data[SHA_LBLOCK];
      int num;
      } SHA_CTX;
```

Trong đó SHA_LBLOCK là 20, SHA_LONG được định nghĩa là kiểu unsigned long.

`int HASH_INIT (SHA_CTX *c)`

Khởi tạo cấu trúc SHA_CTX 'c' với các giá trị ban đầu theo thuật toán hashing SHA1.

`void HASH_BLOCK_HOST_ORDER (SHA_CTX *c, const void *d, int num)`

Đây là hàm thực hiện việc biến đổi dữ liệu theo công thức của thuật toán SHA1. Hàm này sẽ được định nghĩa là SHA1_Transform()

Các hàm SHA1_Update(), SHA1_Final() và SHA1_Transform cũng được định xác định tương tự như trong md5, sử dụng các hàm HASH_UPDATE(), HASH_FINAL() và HASH_TRANSFORM() trong file md32_common.c ở thư mục crypto.

3.5-Thư viện HMAC

Các hàm HMAC được tạo ra thực chất là sử dụng các hàm băm MD5 và SHA1. Nó được sử dụng trong quá trình xác thực thông tin.

Thư viện này được cung cấp trong **crypto/hmac**. Cấu trúc chung mà các hàm HMAC sử dụng là HMAC_CTX được định nghĩa trong file hmac.h

```
typedef struct hmac_ctx_st
      {
      const EVP_MD *md;
      EVP_MD_CTX md_ctx;
      EVP_MD_CTX i_ctx;
      EVP_MD_CTX o_ctx;
      unsigned int key_length;
      unsigned char key[HMAC_MAX_MD_CBLOCK];
      } HMAC_CTX;
```

```
void HMAC_Init_ex(HMAC_CTX *ctx, const void *key, int len,
             const EVP_MD *md)
```

Sử dụng hàm băm ‘md’ để thiết lập giá trị cho ‘md_ctx’ và ‘key’ trong ‘ctx’ từ ‘key’ bằng cách sử dụng các hàm EVP_DigestInit(), EVP_DigestUpdate() và EVP_DigestFinal(). Sau đó, lần lượt 64 (HMAC_MAX_MD_CBLOCK) phần tử trong ‘key’ sẽ được XOR với 0x36 để làm dữ liệu cho hàm băm ‘md’ và tạo giá trị cho ‘i_ctx’ trong cấu trúc ‘ctx’. Tương tự như vậy, 64 phần tử trong ‘key’ sẽ được XOR với 0x5c để làm dữ liệu đầu vào cho hàm băm ‘md’ và tạo giá trị cho ‘o_ctx’ trong ‘ctx’. Các giá trị trong ‘i_ctx’ và ‘o_ctx’ sau này sẽ được sử dụng để trộn với dữ liệu trong ‘md_ctx’.

```
void HMAC_Update(HMAC_CTX *ctx, const unsigned char *data, int len)
```

Gọi hàm EVP_DigestUpdate() để băm dữ liệu trong ‘data’.

```
void HMAC_Final(HMAC_CTX *ctx, unsigned char *md, unsigned int *len)
```

Trả về đầu ra của hàm băm trong ‘md’.

4-Biên dịch, cài đặt MySSL

Quá trình biên dịch và cài đặt được hướng dẫn trên hệ điều hành Linux (Redhat Linux 7.2).

4.1-Biên dịch

Để biên dịch được chương trình yêu cầu trên hệ thống phải cài đầy đủ các công cụ phát triển như gcc, make, perl, …

Bộ chương trình MySSL được đóng gói dưới dạng file tgz (cụ thể là tệp proxy\myssl-mk1.tgz trong CD ROM). Copy file này vào thư mục **/tmp** và tải nén bằng lệnh:

 # tar –xzvf myssl-mk1.tgz

Sau lệnh này chúng ta sẽ có thư mục lưu bộ chương trình nguồn MySSL ở **/tmp**. Chuyển vào thư mục này, thực hiện lệnh make để biên dịch chương trình (tạo ra file chạy **myssl** và các thư viện tĩnh .a).

 # make

MySSL cũng có hai file thư viện là **libssl.a** và **libcrypto.a** nằm ở thư mục hiện tại (myssl-1.0-beta1).

Chú ý: Nếu không muốn dùng các tuỳ chọn mặc định, hãy dùng lệnh sau đây để xem hướng dẫn về việc thiết lập cấu hình cho quá trình biên dịch:

 # ./config -h

Lệnh này sẽ đưa ra một danh sách các tham số cho chúng ta lựa chọn để cấu hình biên dịch. Sau khi đã tìm hiểu kỹ, chúng ta dùng lệnh sau đây:

 # ./config <options>

Các tuỳ chọn <options> có thể là:

--prefix:	xác định thư mục để cài đặt MySSL. Mặc định là /usr/local
no-threads:	Tạo thư viện MySSL không hỗ trợ lập trình đa tuyến.
threads:	Tạo thư viện MySSL hỗ trợ lập trình đa tuyến.
no-shared:	Không tạo thư viện chia xẻ (thư viện tĩnh .a, mặc định)
shared:	Tạo thư viện chia xẻ (thư viện động .so)
no-<cipher>:	cipher ở đây duy nhất là mk1, tuỳ chọn này sẽ không hổ trợ mã pháp <cipher> khi biên dịch chương trình. Nó có lợi ích khi cung cấp nhiều thuật toán mã hoá.
386:	Tạo mã máy 80386
-<xxx>:	Các tuỳ chọn cho trình biên dịch (chẳng hạn -DMYSSL_NO_RSA, -fPIC, …)

4.2-Cài đặt

Sau khi đã biên dịch xong, chúng ta thực hiện việc cài đặt chương trình bằng lệnh:

 # make install

Theo mặc định, chương trình sẽ được cài vào thư mục **/usr/local/ssl**. Trong thư mục **ssl** này bao gồm các thư mục con sau đây.

TT	Tên thư mục/file	Mô tả
1	bin	Chứa file chạy **myssl**
2	certs	Chứa các file Certificate (dùng để test)
3	include	Trong này có thư mục con **myssl** chứa các file inculude của MySSL
4	lib	Chứa hai thư viện của MySSL là libcrypto.a và libssl.a
5	misc	Chứa các file liên quan khác
6	private	Dùng để chứa khoá bí mật RSA
7	file **myssl.conf**	Đây là file cấu hình làm ví dụ của MySSL

5-Biên dịch, cài đặt SQUID có trợ giúp dịch vụ mật mã từ MySSL

Chúng tôi sử dụng Squid phiên bản 2.5.Pre11 đã được sửa đổi để chạy với MySSL. Bộ chương trình sau khi sửa đổi được lưu dưới dạng file nén tgz (tệp proxy\squid-mk1.tgz trên CDROM). Thực hiện việc tải nén chương trình nguồn như sau:

tar -xzvf squid-mk1.tgz

Chuyển vào thư mục squid-myssl-2.5.Pre11 thực hiện lệnh thiết lập thuộc tính cho SQUID sử dụng các thuộc tính mật mã từ myssl:

./configure --enable-ssl --with-myssl=/usr/local/ssl

Thực hiện lệnh biên dịch:

make

Thực hiện lệnh cài đặt:

make install

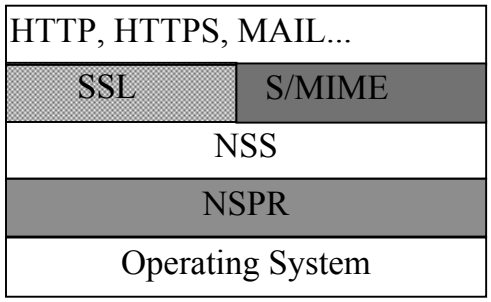
Khi đó Squid sẽ được cài đặt mặc định vào thư mục **/usr/local/squid**.

Chương III
Trình duyệt Mybrowser và tích hợp mật mã cho trình duyệt Mybrowser

1-Mozilla 1.0 và trình duyệt Mybrowser

1.1-Giới thiệu chung về bộ chương trình Mozilla 1.0

Mozilla là một trình duyệt Web mã nguồn mở được thiết kế bởi tổ chức phần mềm Mozilla (mozilla.org). Phiên bản mozilla 1.0 chạy ổn định, tin cậy, và đặc biệt hoàn toàn thoát khỏi sự phụ thuộc vào thư viện mật mã của hệ điều hành (IE phải phụ thuộc vào CRYPT32.dll của Microsoft), trình duyệt Mozilla xây dựng riêng cho nó một thư viện mật mã NSS (Network Security Service). Trên nền mật mã đó, Mozilla cung cấp đầy đủ các công cụ, dịch vụ duyệt Web an toàn dựa trên giao thức SSL, và gửi/nhận thư an toàn dựa trên giao thức S/MIME,... Cấu trúc chung của bộ chương trình Mozilla có thể được minh hoạ trong hình dưới đây:



Cấu trúc bộ chương trình Mozilla được xây dựng dựa trên tập hợp rất nhiều công nghệ và gồm rất các thành phần. Kiến trúc này đã được sử dụng để thiết kế cho Mozilla browser và Netscape Communicator 6.0, tất nhiên trên thực tế nó cũng có thể đã được sử dụng cho các kiểu ứng dụng tương tự khác.

Sau khi phát hành bản Mozilla đầu tiên vào 31/03/1998, xây dựng dựa trên mã nguồn cơ sở (được gọi là classic code) theo trình tự từ dưới lên, với đầy đủ tính năng CSS (Cascadding Style Sheets) và mức 1 của DOM (Document Object Model). Dựa trên classic code để tạo ra các tác vụ cho ứng dụng là rất khó và dễ sinh lỗi. Do vậy, thay vì tiếp tục thực hiện trên nguyên bản mã cơ sở, thì kiến trúc hiện tại được tạo ra với sự kết hợp chặt chẽ khi phát triển các thành phần xung quanh cross-platform. Mã nguồn cơ sở mới được thiết kế cung cấp cấu trúc các thành phần theo trình tự từ trên xuống và lấy trình duyệt Web của Mozilla làm cơ sở nền tảng. Có thể dễ dàng tạo một ứng dụng cross-platform mới, thông qua việc sử dụng, C++, JavaScript hoặc XUL. Dưới đây chúng tôi sẽ trình bày một số công nghệ mới được sử dụng khi xây dựng lên cấu trúc của bộ chương trình nguồn mozilla 1.0.

1.2-Một số công nghệ chính trong bộ chương trình Mozilla 1.0

1.2.1-XPCOM

XPCOM viết tắt của cụm từ Cross-Platform Component Object Module, là một mô hình được sử dụng trong việc xây dựng các phần mềm cross-platform. Cũng như một ứng dụng, XPCOM sử dụng tập hợp các thư viện nhân XPCOM để lựa chọn nạp và quản lý các thành phần XPCOM. Các thành phần XPCOM có thể được viết bằng ngôn ngữ C, C++, JavaScript và chúng cũng có thể được sử dụng bởi C, C++, JavaScript với phần mở rộng cho Perl hoặc Python.
Dưới đây là các platform XPCOM trợ giúp:

- Microsoft Windows
- Linux
- HP-UX
- AIX
- Solaris
- OpenVMS
- MacOS
- BSD

Các công cụ cần để người sử dụng xây dựng phần mềm XPCOM gồm có: bộ biên dịch C++, Perl, và một số công cụ GNU.

Microsoft COM và XPCOM

Nếu so sánh XPCOM với Microsoft COM, thì giữa chúng có nhữn điểm giống nhau và khác nhau nhất định. Trợ giúp các thành phần uỷ quyền là một trong những điểm khác nhau giữa chúng.

Microsoft COM trợ giúp cơ chế uỷ quyền rất tinh vi trong việc thu xếp cách các kiểu ứng dụng khác nhau giao tiếp với các thành phần, áp dụng cho các tiến trình độc lập thậm chí trên cả các máy khác nhau. MSCOM cho phép người sử dụng tạo các thành phần với các mô hình định tuyến khác nhau hoặc hạn chế một thành phần đối với một mô hình định tuyến cụ thể. Các thành phần có thể được tạo như các tiến trình bên trong (chạy bên trong các ứng dụng) hoặc các tiến trình bên ngoài (chạy như một ứng dụng độc lập). Một thành phần định tuyến đơn phải có tuyến riêng của bản thân thành phần đó, và các tuyến khác phải sử dụng cơ chế uỷ quyền để truy cập nó. một tập hợp các thành phần định tuyến có thể chia sẻ một tuyến nhưng cần có một sự uỷ quyền. Các thành phần định tuyến tự do không cần phải có các cơ chế uỷ quyền bên trong như các tiến trình tương tự khác.

XPCOM được thiết kế cho mục đích cung cấp sự trợ giúp thiết lập module thành phần ở mức ứng dụng. Nếu gnười sử dụng muốn truy cập đến một thành phần từ xa, người sử dụng cần viết một cơ chế uỷ quyền riêng phụ trách dữ liệu tiền và hậu xử lý cho đối tượng ở xa. Có một số thành phần có sẵn có thể giúp người sử dụng thực hiện điều này, bởi vậy nó không thực sự khó như những gì đã nghe.

Xét ở mức cụ thể, XPCOM và MSCOM có vẻ giống nhau: cả hai đều đặt nền tảng cho giao diện và yêu cầu tất cả các giao diện khác phải xuất phát từ giao diện cơ sở như nhau. Giao diện cơ sở này định nghĩa ba phương thức QueryInterface, AddRef và Release. Mặc dù có sự thừa kế lẫn nhau, nhưng các thành phần MSCOM và XPCOM cũng không có sự tương thích hoặc có thể thay đổi cho nhau, bởi vậy để chúng có thể làm việc với nhau cần có thêm phần mã gắn đính, liên kết. Một ví dụ rất cụ thể cho điều này chính là phần nguồn đóng gói nhúng cho Mozilla. Nó cho phép trình duyệt xuất hiện như một MSCOM ActiveX control trong khi trình duyệt hoạt động bên trong một trên các thành phần XPCOM.

Sự khác nhau lớn nhất giữa XPCOM và MSCOM là ở chỗ: XPCOM là công nghệ nguồn mở. Nếu trong quá trình người sử dụng phát triển phần mềm MSCOM, để hiểu được sự khác nhau trong phương pháp các thư viện MSCOM nạp và tạo các thnàh phần của ngươời sử dụng thì người sử dụng chỉ có cách phải dựa hoàn toàn toàn vào các tài liệu đã được Microsoft công bố. Tuy nhiên, nếu Microsoft thay đối các thư viện hệ thống nó sẽ có ảnh hưởng đến sự tác động các thành phần mà người sử dụng xây dựng và các ứng dụng sử dụng các thành phần đó.

Trái lại, chương trình nguồn xây dựng nên các thư viện sẽ làm cho cấu trúc của XPCOM thật sự có giá trị cho người sử dụng kiểm tra, theo dõi, gỡ rối trong suốt quá trình xây dựng ứng dụng. Người sử thậm chí có thể thay đổi cả phần nguồn cơ sở cho phù hợp với cấu trúc của bản thân họ, điều này hoàn toàn không thể thực hiện được đối với MSCOM.

Nếu người sử dụng chưa thật sự sẵn sàng với các phần mềm có nguồn mở, có thể làm quen với các dự án nguồn mởt có tiếng như: OPENBSD, Linux, OpenLDAP, OpenSSL, ...

XPCOM, các tuyến, CORBA và một số thông tin khác

Chúng ta cũng cần biết cách mà XPCOM quản lý các đối tượng ở xa, các tuyến và các scripting. Một số công nghệ XPCOM được thừa kế từ OMG CORBA. CORBA là một công nghệ thành phần ngôn ngữ trung lập platform được tiêu chuẩn hoá bởi Object Management Group (OMG). Nó bao hàm hầu hết các cấu trúc cho các thủ tục gọi các đối tượng từ xa, một ngôn ngữ định nghĩa giao diện (IDL),...

XPCOM không trợ giúp các đói tương từ xa như CORBA, nhưng cũng không hạn chế người sử dụng trong việc viết các thành phần sử dụng socket hoặc các công cụ khác để giao tiếp với các thành phần khác. Rất nhiều các ứng dụng XPCOM làm cho việc sử dụng HTTP như một công cụ đưa ra các thủ tục từ xa gọi tới các thành phần được khai báo trên Web server. Mối liên hệ giữa XPCOM và CORBA chính là bộ biên dịch IDL của XPCOM, được gọi là XPIDL, nó được lấy từ bộ biên dịch IDL nguồn mở của CORBA.

XPCOM trợ giúp scripting thông qua một tầng bổ sung được gọi là XPConnect, bao gồm một JavaScript engine và một cơ chế thư viện kiểu. Điều này cho phép mã

nguồn JavaScript có thể quản ký các thành phần của XPCOM thậm chí cho phép thiết lập các thành phần XPCOM từ ngôn ngữ JavaScript và có thể truy nhập bởi mã nguồn C++.

1.2.2- Giao diện người dùng: XPToolkit

Ban đầu, để phát triển giao diện cần viết bằng ngôn ngữ C++cho mỗi platform. Mozilla cung cấp một tập hợp các công cụ gọi là XPToolkit (Cross-Platform Toolkit), các công cụ này có thể được sử dụng để định nghĩa sự xuất hiện của giao diện người dùng và tác động lên ứng dụng có sử dụng các chuẩn W3C.

XPToolkit giảm thiểu số lượng mã nguồn định danh platform cần thiết khi xây dựng ứng dụng kiểu như các browser và các mail client. XPToolkit nâng cao hiệu lực cho một tập chuẩn các dịch vụ widget và graphic để biểu diễn các phần tử UI, cho phép điều khiển mức điểm (pixel level) của UI khi cần thiết.

- ***XUL***

XUL (XML-based User Interface Language) cung cấp một phương thức xuất hiện của UI và tính logic của ứng dụng khi sử dụng XML, CSS, và JavaScript (hoặc C++).

XUL là XML đơn giản với các định nghĩa cho một số kiểu phần tử, và trong đó có lúc JavaScript được nhúng vào mã nguồn. Các phần tử được biết như là XUL widget và bao gồm các điều khiển UI như là các menu, tree, drop-down boxes. XUL giúp cho các người phát triển sẽ dễ dàng thao tác hơn khi biết về SGML - Standard General Markup Language (như HTML) và JavaScript để thiết kế giao diện người dùng. Ứng dụng JavaScript có thể truy nhập tới XPCOM (và hầu hết các thư viện) sử dụng XPConnect, như vậy có thể dùng nó để xây dựng các ứng dụng hoàn chỉnh.

Sự kết hợp giữa giao diện và khả năng tác động được gọi là "chrome". Chrome được nạp từ các tệp có đuôi .xul và được đi kèm với tệp JavaScript và CSS. Mã script trong các tệp đó có đủ hiệu lực thay đổi và mở rộng tác động của trình duyệt, do vậy điều này chỉ được phép với những script đáng tin cậy. Ngược lại, các "skins" lại là sự kết hợp của các giao diện đồ hoạ và CSS, trong đó chỉ thay đổi sự xuất hiện của trình duyệt mà không thay đổi khả năng tác động.

Hầu hết UI của trình duyệt được viết bởi mã nguồn XUL. Để thêm một nút lệnh trên toolbar thì chỉ việc bổ sung và định vị nút lệnh vào cấu trúc cây biểu diễn các nút lệnh và xây dựng mã nguồn điều khiển nút lệnh này bằng XML và JavaScript.

XUL đã làm cho bản thân nó có thể phát triển UI bằng cách sử dụng các công nghệ web quen thuộc. Một trong những mục đích của XUL là người khai thác không chính quy có thể tạo các tính năng mới một cách dễ dàng và dùng chung

chúng với các đặc tính khác. Điều này sẽ cho phép UI có được sự phát triển song song một cách mạnh mẽ, đem lại lợi ích cho các nhóm mã nguồn mở khác.

Hiện tại ngôn ngữ XBL (eXtensible Bingdings Language) được sử dụng rộng rãi mặc dù rất phức tạp. Ví dụ, XBL cho phép việc tạo các widget mới mà có thể truy nhập thông qua XUL, tự động lấy các phương thức và các tính chất từ các thành phần XPCOM, và mở rộng các điều khiển sự kiện bao gồm cả việc bắt phím.

- ***XBL***

XUL cung cấp một số lớn các widget đang tồn tại như là các text boxe và scroll bar. XBL làm cho có thể thay đổi các phần tử XUL đang tồn tại và tạo các phần tử mới.

Các tệp XBL bao gồm tập hợp các ràng buộc, mỗi ràng buộc đó mô tả khả năng tác động của một phần tử XUL. Các ràng buộc này có thể xác định rõ như sau: nội dung con (child content), như là các method và các propertie của một phần tử XUL, nơi mà tiến trình và dữ liệu có thể hoặc là JavaScript nội tuyến hoặc là trỏ tới các sự kiện giao diện XPCOM mà phần tử này đang thực hiện tác vụ, như là các sự kiện phím và chuột tương thích với các kiểu tính chất khác nhau.

Ví dụ, một widget mới của ảnh đã được gán nhãn có thể được tạo khi mở rộng box widget đang tồn tại bằng cách thêm một ảnh và gán một nhãn như sau:

```
<binding name="captionbinding" extends="xul:box">
<content>
        <xul:box xmlns="there.is.only.xul">
                <xul:image inherits="image:src"/>
        <xul:text inherits="caption:value"/>
        <xul:box/>
</content>
</binding>
```

Sau đó chúng ta gắn ràng buộc bằng cách sử dụng đoạn mã dưới đây vào trong tệp CSS tương ứng.

```
captionimg
        behavior: url("mybindings.xbl#captionbinding");
}
```

Đoạn mã trên được bắt đầu bằng tên của phần tử mới được định nghĩa (ở đây là captionimg). URL chỉ ra tên tệp sau từ khoá binding (ở đây là captionbinding) mà đã định nghĩa trước đó.

Tổng hợp kết quả trong XUL có dạng như sau:

```
<captionimg
        caption="My image caption" image="myimage.png"/>
```

1.2.3- XPFE

XPFE được viết tắt của Cross-Platform Front End, là tên nhận được từ Mozilla browser front end (chrome). Định nghĩa và sử dụng thuật ngữ này là không rõ ràng, nhưng xu hướng hiện nay là sử dụng XPFE khi nói đến giao diện người dùng trình duyệt Mozilla được xây dựng dựa trên XPToolkit.

1.3-Tối thiểu hoá bộ chương trình Mozilla 1.0

Trên cơ sở tìm hiểu các công nghệ, kiến trúc bộ chương trình nguồn của bản Mozilla 1.0 chúng tôi tối thiểu với tiêu chí: xây dựng trình duyệt web Mybrowser. Sau khi đã tối thiểu trình duyệt này sẽ được tích hợp thuật toán mã hoá MK1 của ngành, sử dụng chứng chỉ số để xác thực,... Trong mục này chúng tôi chỉ giới thiệu về công việc tối thiểu hoá, phần tích hợp mật mã sẽ được đề cập ở trong phần tiếp theo.

Công việc tối thiểu hoá có thể được chia ra làm các bước sau:

- Tìm hiểu tổng quan và phân tích chức năng của từng module chương trình nguồn.
- Thực hiện loại bỏ phần chương trình nguồn thực hiện các chức năng không thật quan trọng đối với các tính năng của một trình duyệt Web, và cả các module chương trình với vai trò là các module plug-in khác (gồm các thư mục như: accessible, extensions, calendar, ...).
- Loại bỏ phần chương trình trợ giúp các platform không sử dụng: với tiêu chí chỉ sử dụng trên hệ điều hành Windows (Win95/98/SE/2000/XP/ME) nên chúng ta loại bỏ các thư mục, file và đoạn mã nguồn hỗ trợ các platform như: mac, beos, unix, os2, qt, photon, gtl.
- Thay đổi giao diện, tên sản phẩm

Sau khi thực hiện tối thiểu bộ chương trình nguồn Mybrowser gồm các thư mục chính dưới đây:

Thư mục Build: Chứa các script (thường viết bằng Perl) và các chương trình được sử dụng để xây dựng và quản lý mã nguồn Mybrowser. Những chương trình này khi chạy tạo ra các tệp Makefile và thiết lập thư mục dist.

Thư mục caps: chứa các giao diện C++ và chương trình nguồn thực hiện việc xác định khả năng của việc thiết lập cơ chế an toàn và sử dụng chứng chỉ số cơ sở.

Thư mục config: Chứa các script và chương trình được sử dụng để thiết lập các biến môi trường, thiết lập các lựa chọn trong makefile.

Thư mục Docshell: Chứa các giao diện và mã nguồn C++ thực hiện việc nạp và hiển thị một trang web đơn. Phần nguồn trong thư mục embedding sẽ tích hợp phần chương trình nguồn này với các thuộc tính trình duyệt ở mức cao hơn như forword, back và history.

Thư mục dom: Chứa các giao diện và mã nguồn C++ để thực thi và lưu vết các đối tượng DOM (Document Object Model) trong Javascript. Nó thiết lập cấu trúc con C++ để tạo, huỷ bỏ và thao tác với các đối tượng được tích hợp sẵn (built-in) và được định nghĩa bởi người dùng (user-defined) trên cơ sở ngôn ngữ Javascript.

Thư mục editor: Chứa các giao diện C++, mã nguồn C++ và XUL thực hiện trình soạn thảo có nhúng có thể soạn thảo cả plaintext và HTML. Nó được sử dụng cho trình soạn thảo HTML cho trình duyệt.

Thư mục embedding: Chứa các giao diện và mã C++ cho các chức năng của trình duyệt ở mức cao (như forward, back, history). Mã nguồn trong webshell sẽ tích hợp các giao diện này theo nền và cách mà platform đó được sẽ trợ giúp (như ActiveX).

Thư mục expat: Chứa mã nguồn C++ để phân tích các kiểu định dạng phổ thông như XML và DTD. Phần nguồn này xuất phát từ ứng dụng expat của James Clark.

Thư mục gfx: Chứa các giao diện C++ và mã độc lập platform xử lý đồ hoạ và hình ảnh. Nó có thể được sử dụng để vẽ hình chữ nhật, đường thẳng, ảnh... Nó không xử lý widget hoặc các thủ tục vẽ cụ thể; nó chỉ cung cấp thao tác cơ bản để vẽ.

Thư mục htmlparser: Chứa các giao diện và mã C++ cho việc phân tích, biểu hiện HTML, XUL và các nội dung khác dưới định dạng cấu trúc hình cây trong bộ nhớ. Nó không thực hiện việc bố trí hoặc hiển thị nội dung; nó chỉ biến đổi từ một dòng (stream) thành một cấu trúc dữ liệu.

Thư mục intl : Chứa các giao diện C++ và mã cho việc phát triển trên từng vùng (localization). Nó chứa mã của các tập ký tự, các định dạng địa phương, các định dạng về ngày tháng thời gian cho từng vùng khác nhau.

Thư mục jpeg: Phần chương trình xử lý ảnh JPEG.

Thư mục js: Phần chương trình biểu thị, phân tích, tích hợp và thực thi các JavaScript scripts.

Thư mục layout: Chứa các giao diện C++ và mã cho layout engine. Nó thay đổi kích cỡ và căn lề các thành phần của nội dung theo CSS1 và CSS2 (kiểu trang nhiều tầng). Mã này cũng được biết là "NGLayout" và "Gecko".

Thư mục modules: Chứa mã C cho rất nhiều các đặc tính khác nhau được xây dựng và tích hợp cho trong Mybrowser. Nó chứa các mã để quản lý các dạng ảnh khác nhau (như .PNG, .GIF), cho phép các máy ảo drop-in Java (được gọi là OJI, cho "Giao diện Java mở"), hỗ trợ plug-ins và đọc các dạng nén khác nhau (như .JAR, .ZIP, .ZLIB).

Thư mục netwerk: Chứa các giao diện và mã C++ thực hiện truy nhập mạng ở mức thấp (sử dụng socket, file và memory cache) cũng như truy ở nhập mức cao hơn (sử dụng các giao thức khác nhau như http, ftp, gopher, castanet). Phần mã nguồn này cũng được biết đến với các tên "netlib" và "Necko".

Thư mục nsprpub: Chứa mã nguồn C cho "C" Runtime Library nền chéo (cross platform). Thư viện "C" Runtime định nghĩa lại các hàm C cơ bản như các hàm cấp phát và thu hồi bộ nhớ, các hàm lấy thời gian và ngày tháng, đọc và ghi các file, quản lý tuyến (thread), so sánh các chuỗi theo tất cả các nền. Mã này cũng được biết đến với tên gọi là "nspr" và "Netscape Portable Runtime".

Thư mục profile: Chứa các giao diện C++, mã C++, XUL và Javascript để tạo hồ sơ người dùng mới, quản lý các hồ sơ người dùng đã tồn tại, thay đổi các profile từ Mozilla Classic và sử dụng các profile mặc định của các ISP phổ thông.

Thư mục rdf: Chứa các giao diện và mã nguồn C++ để truy nhập tới nhiều dữ liệu khác nhau và tổ chức mối liên hệ của chúng theo RDF. RDF là từ viết tắt của "Resource Description Framework", đây là một chuẩn mở. Phần mã nguồn này đọc và ghi dữ liệu từ hệ thống file cục bộ, cơ sở dữ liệu, Internet và từ các nguồn khác sử dụng cú pháp như URL.

Thư mục security: lưu các tệp chương trình nguồn cung cấp các dịch vụ mật mã như các hàm mã hoá dữ liệu, các hàm băm, các thuật toán chữ ký số, trao đổi khoá, các giao thức bảo mật SSL, TLS và các chuẩn mật mã khác như PKCS5, PKCS7, PKCS12, ...

Thư mục sun-java: Chứa mã C để Mozilla có thể truyền thông với Sun JVM (Java Virtual Machine). Nó không gồm mã nguồn cho bản thân máy ảo.

Thư mục uriloader: Chứa các giao diện và mã nguồn C++ để gọi chính xác trình hiển thị nội dung được chứa trong một URL nhất định. Ví dụ, nếu mã này xác định rằng nội dung là một thông điệp mail, thì nó tìm kiếm trình lắng nghe thích hợp (có thể là Netscape Messager) và chuyển thông báo mail này tới nó để hiển thị.

Thư mục view: Chứa giao diện và mã nguồn C++ cho các kiểu hiển thị khác nhau (như scrolling view). Một kiểu hiển thị bao hàm nội dung nhưng không chứa thanh tiêu đề, các đường viền hoặc sự trang trí khác.

Thư mục webshell: Chứa các giao diện C++ và mã C++, script shell của Linux và các tệp khác để thực hiện nhúng Mozilla trong các chương trình khác trên nhiều platform với các cách khác nhau (như plug-in, thành phần ActiveX, các lớp XPCOM).

Thư mục widget: Chứa các giao diện và mã nguồn C++ thực hiện sự điều khiển không phụ thuộc vào nền (widgets), như các thanh cuộn, nút bấm radio, các hộp liệt kê.

Thư mục xpcom: Chứa giao diện C++ mức thấp, mã C++, mã C, một ít mã assembly và các tiện ích dòng lệnh để cài đặt công cụ cơ bản của thành phần XPCOM (chuẩn này cho "Cross Platform Component Object Model"). XPCOM là một cơ chế mà cho phép Mybrowser xuất các giao diện và làm cho chúng có thuộc tính tự động và sẵn sàng đối với các Javascript script, Microsoft COM và mã Mybrowser C++. Một vài các lớp XPCOM mức thấp và các giao diện được định nghĩa ở đây (như sự kiện lặp cho tất cả các nền). XPCOM thì tương thích và thân thiện với Microsoft COM (tất nhiên XPCOM là cross-platform).

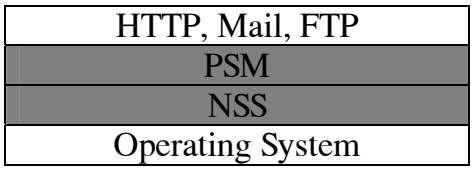
Thư mục xpfe: Chứa các giao diện C++, mã C++ và XUL cho việc cài đặt "Cross Platform Front End". Về cơ bản, đây là nơi mà chương trình Mybrowser khởi động và quản lý các thành phần khác để thực hiện các tác vụ. Mã này chứa một ít mã phụ thuộc nền;

Thư mục xpinstall: Chứa các giao diện và mã nguồn C++ cho việc thực thi đặc điểm SmartUpdate từ Mozilla Classic. XPInstall cung cấp mã cho các tệp được tải, giải nén chúng và cài đặt chúng theo cách không phụ thuộc vào platform.

Kết luận: Với bộ chương trình nguồn Mozilla có dung lượng trên 300MB, chúng tôi tối thiểu để chỉ sử dụng chức năng chủ yếu là một trình duyệt Web với dung lượng khoảng 80MB, sau khi thực hiện biên dịch sản phẩm được đóng gói thành một bộ cài có dung lương xấp xỉ 7 MB. Trình duyệt Web này được đặt tên là Mybrowser với tất cả những tính năng thông dụng nhất của một trình duyệt Web.

2-Tích hợp mật mã cho Mybrowser

Module chương trình cung cấp các dịch vụ mật mã trong bộ chương trình nguồn Mybrowser, được gói gọn hoàn toàn trong một thư mục riêng có tên là "security". Mối quan hệ giữa module này với các module thực hiện các chức năng khác có thể được mô tả như mô hình dưới đây:



Trong đó: -HTTP, Mail, FTP: Các thành phần ứng dụng của Mybrowser
-PSM (Personal Sercurity Manager): Quản lý dịch vụ bảo mật riêng sử dụng cho Mybrowser.
-NSS (Network Sercurity Services): Các dịch vụ bảo mật mạng.
-NSPR (Netscape Portable Runtime): Nền tảng công nghệ cho Mybrowser.

Như vậy phần lõi của các chương trình thực hiện các thuật toán mật mã nằm trong module NSS, việc thực hiện giải pháp thay thế các thuật toán chỉ được thực hiện trong module này. Còn đối với module PSM chỉ là một module trung gian, nếu có cần thì cũng chỉ là sự thay đổi một số giao diện cho phù hợp với quá trình đã được thực hiện ở NSS.

NSS là bộ chương trình nguồn cung cấp một thư viện độc lập thực hiện các dịch vụ bảo mật phục vụ cho việc phát triển các ứng dụng cross-platform. Khi xây dựng một ứng dụng sử dụng NSS, ứng dụng đó có thể được cung cấp các giao thức SSL v1, SSL v2, TLS, các chuẩn mật mã khoá công khai PKCS#5, PKCS#7, PKCS#11, PKCS#12, S/MIME, chứng chỉ số theo chuẩn X.509 v3 và rất nhiều các chuẩn mật mã khác.

Dưới đây là các chuẩn mật mã được xây dựng trong NSS:

- Giao thức bảo mật tầng socket phiên bản 2 và 3 (SSL v2 và v3): Đây là giao thức cho phép thiết lập một kết nối tin tưởng và thực hiện mã hoá dữ liệu trao đổi qua lại trong mỗi phiên liên lạc giữa server/client. Giao thức này được thiết kế và công bố bởi Netscape.
- Giao thức bảo mật tầng vận tải phiên bản 1 (Transport Layer Security): Là một giao thức được thiết kế và công bố bởi IETF dựa trên cơ sở SSL, do đó nó hoàn toàn tương thích với SSL.
- PKCS#1 (Chuẩn mật mã khoá công khai - Public-Key Cryptography Standard): Chuẩn RSA cho việc thực hiện mật mã khoá công khai trên cơ sở thuật toán RSA.
- PKCS#3: Chuẩn RSA cho việc thực hiện trao đổi khoá Diffie-Hellman.
- PKCS#5: Chuẩn RSA cho mật mã trên cơ sở mật khẩu, chẳng hạn mã pháp mã mật khẩu dùng làm khoá để mã khoá bí mật khi lưu trữ.
- PKCS#7:Chuẩn RSA cho việc định dạng dữ liệu được sử dụng bởi các ứng mật mã, chẳng hạn định dạng dữ liệu chữ ký số.
- PKCS#8: Chuẩn RSA cho việc lưu trữ và mã hoá khoá bí mật.
- PKCS#9: Chuẩn RSA cho việc lựa chọn các kiểu thuộc tính, được sử dụng cho PKCS#7, PKCS#8, và PKCS#10.

- PKCS#10: Chuẩn RSA cho cú pháp yêu cầu chứng chỉ số (khi một người sử dụng muốn được cấp chứng chỉ số theo chuẩn RSA, yêu cầu của người sử dụng được định dạng theo qui định của chuẩn PKCS#10).
- PKCS#11: Chuẩn RSA cho việc truyền thông với các module và các thẻ mật mã (như smart card).
- PKCS#12: Chuẩn RSA cho định dạng được sử dụng trong việc lưu trữ và trao đổi khoá bí mật, chứng chỉ số và các dữ liệu bí mật khác.
- S/MIME: Định dạng thông báo IETF trên cơ sở chuẩn Internet MIME (Multipurposse Internet Mail Extensions) phổ thông, cung cấp một phương pháp phù hợp cho việc gửi/nhận dữ liệu MIME đã được ký và mã hoá.
- X509 v3: Chuẩn ITU cho định dạng của các chứng chỉ số được sử dụng cho xác thực trong mật mã khoá công khai.
- OCSP (giao thức xác định trạng thái của chứng chỉ số thời gian thực): Giao thức xác định tính hợp lệ của một chứng chỉ số thời gian thực.
- Mô tả danh sách các chứng chỉ số bị huỷ bỏ (Certificates Revoked List) và chứng chỉ số PKIX: Các chuẩn cho việc phát triển cơ sở hạ tầng khoá công khai theo chuẩn X509.
- Các thuật toán mã hoá dữ liệu: chuẩn mã hoá Mỹ AES (American Encryption Standard), chuẩn mã hoá dữ liệu DES (Data Encryption Standard), Triple DES, RSA, RC2, RC4.
- Các thuật toán hàm băm: SHA-1 (Security Hash Algorithm), MD2, MD5.
- Các thuật toán chữ ký số và trao đổi khoá: RSA, DSA, Diffie-Hellman.

Loai bỏ môt số thuât toán hàm băm, trao đổi khoá, chữ ký số:

Với tiêu chí chỉ giữ lại phần nguồn thực hiện thuật toán mã hoá và chữ ký số RSA, thuật toán thực hiện hàm băm SHA-1, chúng tôi thực hiện loại bỏ phần chương trình nguồn thực hiện các thuật toán DSA, DH, MD2, MD5 theo các bước dưới đây:

- Loại bỏ các tệp chương trình nguồn thực hiện các thuật toán trên trong module NSS bao gồm (cụ thể là trong thư mục security\nss\lib\freebl): dsa.c, dh.c, dh_bsf.c, md2.c, md5.c
- Loại bỏ các ID định danh cho các thuật toán trên:

-Loại bỏ định danh ban đầu cho các thuật toán trong tệp security\nss\lib\util\secoidt.h. Ví dụ đối với thuật toán DSA chúng ta cần bỏ phần định nghĩa:

```
SEC_OID_ANSIX9_DSA_SIGNATURE = 124,
SEC_OID_ANSIX9_DSA_SIGNATURE_WITH_SHA1_DIGEST = 125,
SEC_OID_BOGUS_DSA_SIGNATURE_WITH_SHA1_DIGEST = 126,
```

-Loại bỏ định nghĩa việc sử dụng các thuật toán trên cho ứng dụng Web, chẳng hạn đối với thuật toán hàm băm MD5 trong tệp chương trình nguồn secdig.c bỏ phần nguồn:

```
case SEC_OID_MD2:
case SEC_OID_MD5:
```

Tích hợp thuật toán mã hoá MK1:

Trong phần trên chúng tôi đã liệt kê các thuật toán đã được NSS cung cấp cho Mybrowser, như vậy để tích hợp một thuật toán mã hoá mới cho Mybrowser có hai giải pháp để thực hiện:

- Để nguyên các thuật toán có sẵn, bổ sung thuật toán mới và người sử dụng cần chọn thuật toán này khi thực hiện kích hoạt một ứng dụng của Mybrowser. Giải pháp này đơn giản hơn trong thực hiện việc bổ sung, tuy nhiên về mặt nghiệp vụ bảo mật là không đảm bảo (có thể gây nhầm lẫn trong quá trình sử dụng), hơn nữa sẽ gây phức tạp trong qui trình sử dụng ứng dụng.
- Loại bỏ hoàn toàn các thuật toán có sẵn, tích hợp duy nhất một thuật toán mã hoá theo yêu cầu đặt ra, thiết lập các thuộc tính để mọi ứng dụng của Mybrowser đều sử dụng thuật toán này khi sử dụng dịch vụ mật mã. Rõ ràng với giải pháp này chúng ta hoàn toàn yên tâm về tính "được sử dụng" của thuật toán, và đơn giản trong qui trình sử dụng.

Trong hai giải pháp trên chúng tôi chọn giải pháp thứ hai và thuật toán được chúng tôi lựa chọn là thuật toán mã khối với 512 bít khoá, 128 bít IV có tên là thuật toán MK1, một kết quả nghiên cứu của ngành. Dưới đây là qui trình thực hiện việc tích hợp thuật toán MK1 cho Mybrowser thực tế thuật toán này được sử dụng như thế nào.

Bước 1: Thực hiện loại bỏ các thuật toán có sẵn trong NSS bao gồm:

- Loại bỏ các tệp chương trình thực hiện các thuật toán trên, bao gồm các tệp sau đây trong thư mục \security\nss\freebl: arcfive.c, arcfour.c, des.c, des.h, desblapi.c, rijndael.c, rijndael.h, ...
- Loại bỏ các ID định danh cho các thuật toán trên

  -Loại bỏ ID định danh thuật toán, chẳng hạn đối với thuật toán DES trong tệp \security\nss\lib\freebl\secoidt.h loại bỏ các định nghĩa:

  SEC_OID_CMS_3DES_KEY_WRAP = 180,
  SEC_OID_DES_EDE3_CBC = 7,
  SEC_OID_RC5_CBC_PAD = 8,
  SEC_OID_DES_ECB = 9,
  SEC_OID_DES_CBC = 10,
  SEC_OID_DES_OFB = 11,
  SEC_OID_DES_CFB = 12,
  SEC_OID_DES_MAC = 13,
  SEC_OID_DES_EDE = 14,

  -Loại bỏ các định nghĩa sử dụng các thuật toán mã hoá trên cho các ứng dụng mã mật khẩu, mã khoá bí mật, bảo mật Web, bảo mật Mail tương tự như đã tiến hành với các thuật toán MD2, MD5, DSA.

Sau khi bỏ phần chương trình nguồn thực hiện các thuật toán trên, việc tiếp theo bỏ phần giao diện trong PSM. Chẳng hạn đối với SSL v3, loại bỏ phần nguồn cho phép người sử dụng lựa chọn thuật toán mã hoá:

{"security.ssl3.fortezza_fortezza_sha",
SSL_FORTEZZA_DMS_WITH_FORTEZZA_CBC_SHA},
{"security.ssl3.fortezza_rc4_sha", SSL_FORTEZZA_DMS_WITH_RC4_128_SHA},

```
{"security.ssl3.rsa_rc4_128_md5", SSL_RSA_WITH_RC4_128_MD5},
{"security.ssl3.rsa_fips_des_ede3_sha",
SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA},
{"security.ssl3.rsa_des_ede3_sha", SSL_RSA_WITH_3DES_EDE_CBC_SHA},
{"security.ssl3.rsa_fips_des_sha", SSL_RSA_FIPS_WITH_DES_CBC_SHA},
{"security.ssl3.rsa_des_sha", SSL_RSA_WITH_DES_CBC_SHA},
{"security.ssl3.rsa_1024_rc4_56_sha",
TLS_RSA_EXPORT1024_WITH_RC4_56_SHA},
{"security.ssl3.rsa_1024_des_cbc_sha",
TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA},
{"security.ssl3.rsa_rc4_40_md5", SSL_RSA_EXPORT_WITH_RC4_40_MD5},
{"security.ssl3.rsa_rc2_40_md5", SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5},
{"security.ssl3.fortezza_null_sha", SSL_FORTEZZA_DMS_WITH_NULL_SHA},
{"security.ssl3.rsa_null_md5", SSL_RSA_WITH_NULL_MD5},
```

Bước 2: Tích hợp thuật toán mã hoá và giải mã dữ liệu MK1.

- Bổ sung các tệp chương trình (tệp mk1.c, mk1.h) thực hiện thuật toán mã khối MK1 vào thư mục "security\nss\lib\freebl", sửa đổi makefile trong thư mục này để chương trình được biên dịch.
- Định nghĩa ID định danh thuật toán MK1, sửa đổi độ dài của khoá và IV cho phù hợp với thuật toán. ID định danh cho thuật toán MK1 được định nghĩa như sau:

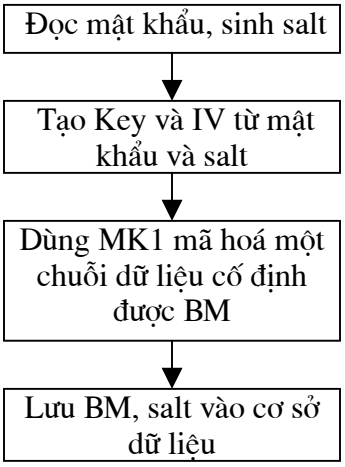
```
SEC_OID_MK1_512_ECB      = 142,
SEC_OID_MK1_512_CBC      = 143,
```

- Thiết lập thuộc tính sử dụng duy nhất MK1 cho các nhóm hàm được PSM sử dụng. Chẳng hạn đối thông tin về các thuật toán mã hoá được trợ giúp cho giao thức SSL:

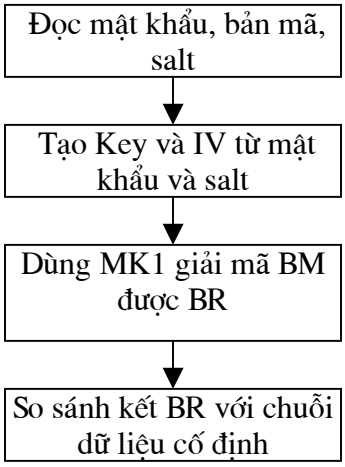
```
static const SSLCipherSuiteInfo suiteInfo[] = {
{0,CS(TLS_RSA_WITH_MK1_512_CBC_SHA), S_RSA, K_RSA, C_MK1, B_512, M_SHA,
0, 0, 0, },};
```

Sau khi quá trình thay thế các thuật toán mã hoá bởi thuật toán MK1, cơ chế sử dụng MK1 cho các ứng dụng của Mybrowser được lần lượt mô tả dưới đây.

- MK1 với qui trình thiết lập và kiểm tra mật khẩu đăng nhập cơ sở dữ liệu lưu khoá và chứng chỉ:



Thiết lập mật khẩu

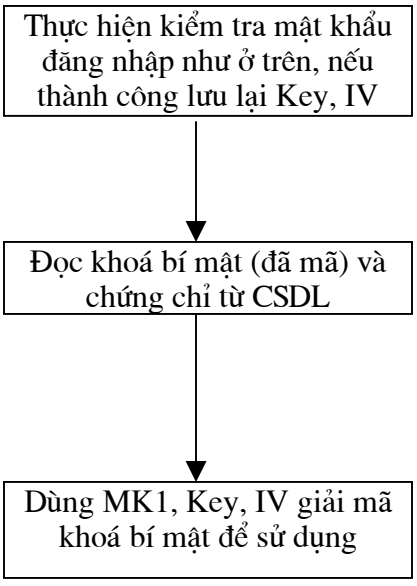


Kiểm tra mật khẩu

Trong đó salt là một giá trị ngẫu nhiên được sinh ra mỗi lần thiết lập mật khẩu. Thuật toán sinh khoá và IV từ mật khẩu và được biết đến là các thuật toán PBDKF1 (theo PKCS#5 phiên bản 1.0) và PBDKF2 (theo PKCS#5 phiên bản 2.0).

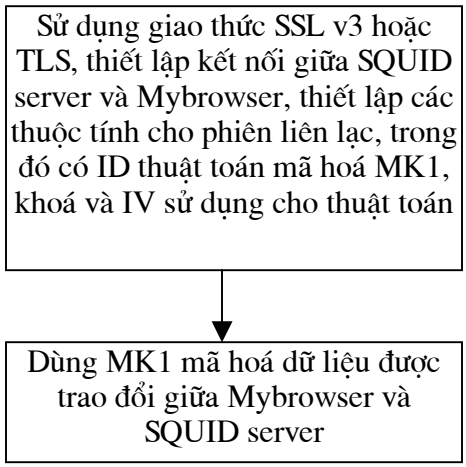
- MK1 sử dụng với PKCS#12:

Qui trình nạp chứng chỉ số và khoá bí mât



Qui trình đọc chứng chỉ số và khoá bí mật

- MK1 Với việc bảo mật web thông qua các giao thức SSL v3, TLS



Sử dụng SSL/TLS bảo mật Web

3-Biên dịch Mybrowser

Bộ chương trình nguồn Mybrowser là một bộ chương trình rất đồ sộ, có thể biên dịch trên nhiều platform (Microsoft, Linux, Macintosh ...), kết quả biên dịch là một phần mềm hoàn chỉnh với chức năng chính là trình duyệt Web, ngoài ra phần mềm còn bao gồm rất nhiều các chức năng khác theo được tích hợp từ các module chương trình.

Có hai cách với các yêu cầu về cấu hình phần cứng và phần mềm khác nhau để biên dịch bộ chương trình nguồn: dùng môi trường giả Linux trên platform Windows thông qua tiện ích Cygwin hoặc dùng tiện ích nmake của Visual C++ với sự trợ giúp của một số module khác. Trong tài liệu này chúng tôi chỉ trình bày việc sử dụng tiện ích nmake của Visual C++ để biên dịch.

Yêu cầu về phần cứng

Máy tính dùng để biên dịch cần có cấu hình tối thiểu như sau:

- Pentium 133 MHz trở lên (tuy nhiên theo nhận xét của chúng tôi khi thực hiện biên dịch trên thực tế với các máy tính có tốc độ bộ vi xử lý bé hơn 500 MHz thì để dịch được bộ chương trình mất tương đối nhiều thời gian)
- 64 MB RAM, 128 MB thì tốt hơn.
- Đĩa cứng cần trống ít nhất là 1 GB đối với NTFS hoặc 2 GB đối với FAT (theo tài liệu hướng dẫn biên dịch) .
- Được cài đặt Win95, Win98, hoặc Windows 2000.

Yêu cầu về phần mềm

Trên máy tính dùng để biên dịch phần mềm, cần cài đặt các tiện ích sau:

- Microsoft Visual C++ 6.0 hoặc cao hơn.
- MSVC++ Service Pack 5.
- MSVC++ Professor Pack
- Netscape Wintools.

- Perl5 for Win32.
- Zip for Win32
- GNU Tools for Microsoft Windows gồm các thành phần như: ash, bzip2, cvs, cygwin, ...

Thiết lập các biến môi trường và biên dịch:

Thiết lập các biến môi trường

Trước khi thực hiện biên dịch cần thiết lập các biến môi trường bằng các lệnh sau từ DOS commandline:

set MOZ_BITS=32

set MOZ_DEBUG=1 nếu người sử dụng debug build, ngược lại để trống

set MOZ_SOURCE=đường dẫn tới nơi lưu bộ chương trình nguồn.

set MOZ_TOOLS=đường dẫn tới nơi cài đặt Netscape wintools.

set OS_TARGET=WIN95 nếu dịch trên win95, win98, và WINNT nếu dịch trên windows NT.

set WINOS=%OS_TARGET%;

set PATH=%PATH%;c:\cygwin\bin;%MOZ_TOOLS%\bin;

set _MSC_VER=1200 nếu dùng Visual C++ 6.0

set DISABLE_TESTS=1 để loại bỏ việc dịch trong các thư mục test.

set MOZ_DISABLE_JAR_PACKEGING=

Thiết lập môi trường cho Visual C++ bằng cách chạy tệp c:\Program files\Microsoft Visual Studio\VC98\Bin\vcvars32.bat.

Thực hiên biên dịch

Chuyển thư mục hiện hành thành \mycagroup và chạy lệnh:

```
nmake /f client.mak build_all
```

Để tạo bộ cài đặt cho phần mềm cần thực hiện biên dịch bộ cài như sau:

Chuyển thư mục hiện hành thành \mycagroup\xpinstall\wizard\windows\builder.

Chạy lệnh perl build.pl

Kết quả bộ cài phần mềm Mybrowser được sinh trong thư mục \mycagroup\dist\install.

Chương IV
Bảo mật dịch vụ web thông qua Proxy

1-Cấu hình và cài đặt hệ thống
1.1-Web Server
1.1.1-Thiết lập cấu hình cho Apache Web Server
Chúng tôi sử dụng Web server là Apache trên phiên bản Linux 7.2. Khi thực hiện cài đặt Linux 7.2 Apache đồng thời cũng đã được cài đặt. Dưới đây chúng tôi trình bày việc thiết lập một trang Web trên Apache.
Mở tệp config cho Apache (tệp /etc/httpd/httpd.conf), tìm chuỗi "ServerName" và bỏ dấu comment ở đầu, thêm vào tên máy hoặc địa chỉ IP của máy chạy Apache. Cụ thể để cấu hình cho một virtual host có tên là rootra thì trong file **httpd.conf** cấu hình như sau:

```
ServerName 200.1.1.1
<VirtualHost 200.1.1.1>
    DocumentRoot "/home/httpd/htdocs-ra/"
    ServerName rootra
    Errorlog logs/ra/error_log
    CustomLog logs/ra/access_log common
    ScriptAlias /cgi-bin/ "/home/httpd/cgi-ra/"
    <Directory "/home/httpd/cgi-ra">
      AllowOverride None
      Options ExecCGI
      Order allow,deny
      Allow from all
    </Directory>
</VirtualHost>
```

Sau đó thực hiện lệnh khởi động Apache:

/etc/init.d/httpd start

Muốn khởi động lại Apache thì thực hiện lệnh:

/etc/init.d/httpd restart

1.1.2-Chạy Window và cài đặt Internet Information Server
Khi cài Windows 2000 Server hoặc WinNT, IIS đã được cài đặt sẵn và cấu hình mặc định. Chúng ta chỉ cần kiểm tra xem dịch vụ Web của IIS đã chạy hay chưa. Nếu chưa chạy (stop) thì phải khởi động dịch vụ này (running) trước khi sử dụng Squid.

1.2-Thiết lập cấu hình Proxy Server

Để Squid chạy được với MySSL chúng ta phải cấu hình squid chạy ở chế độ tăng tốc Web (xem chương về Squid). Sau đó sẽ thêm các tuỳ chọn để hỗ trợ MySSL. Phiên bản 2.5 của Squid chỉ hỗ trợ 2 tuỳ chọn sau:

- ✓ **https_port**
  Chỉ ra địa chỉ cổng (socket) mà Squid sẽ lắng nghe các yêu cầu kết nối bảo mật HTTPS từ phía Client. Tuỳ chọn này được sử dụng khi bạn chạy Squid trong chế độ tăng tốc (Accelerator). Bạn có thể chỉ ra nhiều địa chỉ cổng trên nhiều dòng với các chứng chỉ SSL và/hoặc các tuỳ chọn sau:

  cert = đường dẫn tới chứng chỉ SSL của squid proxy (theo định dạng PEM).

  key = tên thiết bị nghiệp vụ lưu khoá bí mật (cụ thể là thiết bị SD)

  version = Phiên bản của SSL/TLS được hỗ trợ
  1 Tự động (ngầm định).
  3 SSLv3
  4 TLSv1
  cipher = Danh sách các mã pháp được hỗ trợ ngăn cách nhau bởi dấu ':'.
  options = Các tuỳ chọn khác, quan trọng nhất là:
  NO_SSLv3 Không sử dụng SSLv3.
  NO_TLSv1 Không sử dụng TLSv1.

- ✓ **ssl_unclean_shutdown:** Các trình duyệt sẽ đưa ra thông báo lỗi khi SSL shutdown.

File cấu hình cho **squid.conf** sử dụng cho chế độ Accelerator chỉ cần thêm một dòng như sau:

https_port 443 cert=dir/server.crt key=SD

Trong đó **dir/** là thư mục chứa file server.crt.

Với sơ đồ trên, file cấu hình của chúng ta như sau;

```
## squid.conf support authentication Squid Server (SSL)

## map to Real Server
dns_nameservers 200.1.1.1

## Visible
visible_hostname 128.1.1.2
```

SSL - Chỉ ra tên Certificate File và Key File cho Squid Server
https_port 443 cert=/tmp/srv.crt key=SD

virtual host
Phải thiết lập là on khi trên Web Server có nhiều Virtual Host
httpd_accel_uses_host_header on

accelerator option
httpd_accel_port 80
httpd_accel_host 200.1.1.1

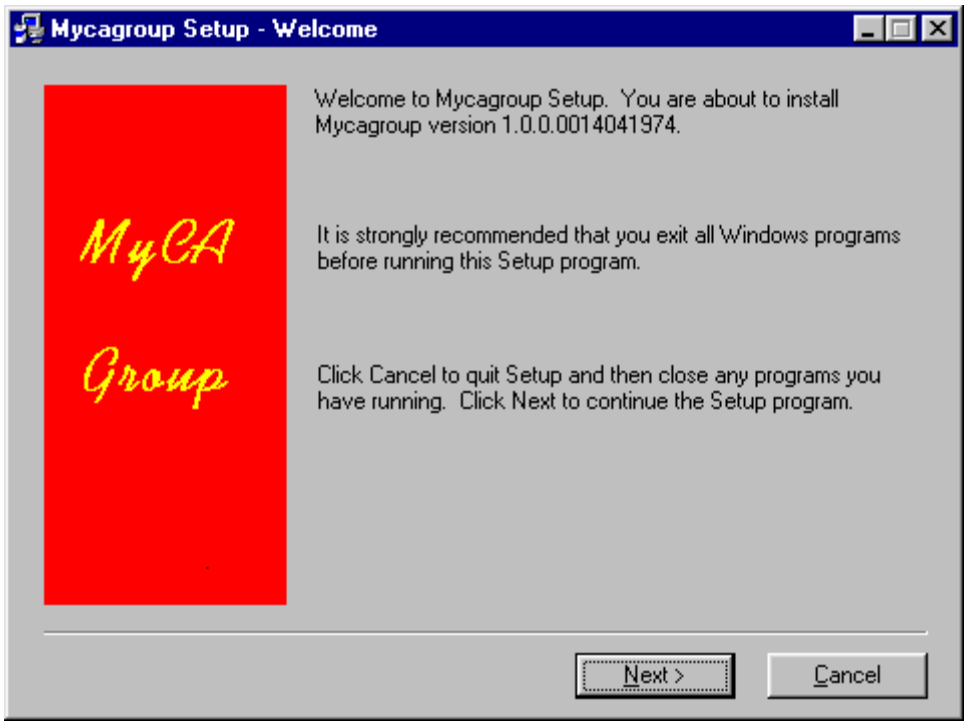
ACCESS CONTROLS
http_access allow all

1.3-Cài đặt và thiết lập cấu hình Mybrowser

1.3.1-Cài đặt trình duyệt Mybrowser

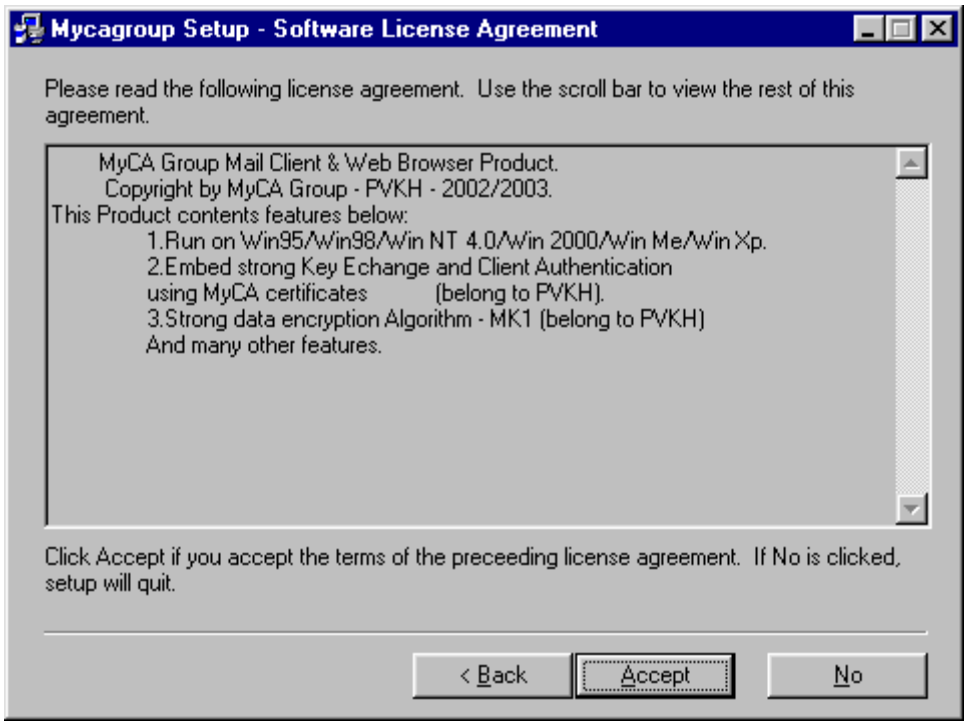
Bộ cài đặt phần mềm Mybrowser được lưu trên đĩa CD-ROM, người sử dụng có thể cài đặt phần mềm trên các platform: Windows 95/Windows 98/Windows NT/Windows 2000/Win Me hoặc WinXP. Quá trình cài đặt trên các hệ điều hành này hoàn toàn giống nhau, do đó tất cả các bước thực hiện qui trình cài đặt trong tài liệu này có thể áp dụng cho tất các các platform trên.

Để cài đặt phần mềm, người sử dụng đặt đĩa CD-ROM vào ổ đĩa, chương trình Autorun sẽ tự động kích hoạt tiện ích thực hiện việc cài đặt phần mềm (trong trường hợp tiện ích không được tự động kích hoạt người sử dụng có thể chạy trình setup.exe). Khi đó trên màn hình xuất hiện hộp hội thoại "Wellcome" như hình 1.



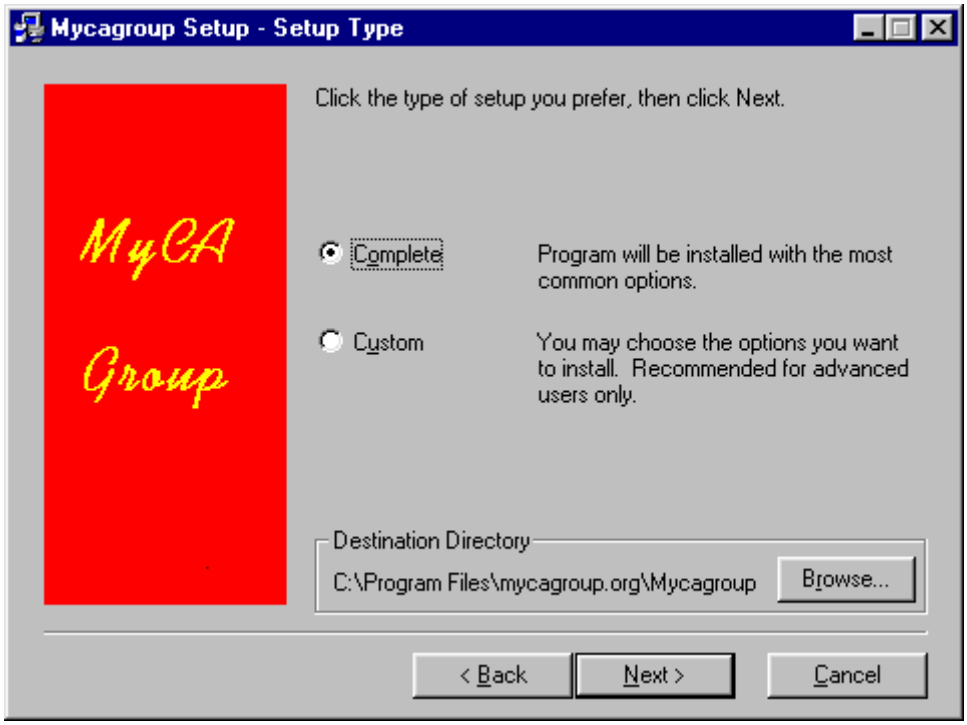
Hình 1

Người sử dụng chọn "Next", trên màn hình xuất hiện hộp hội thoại "License" như hình 2.



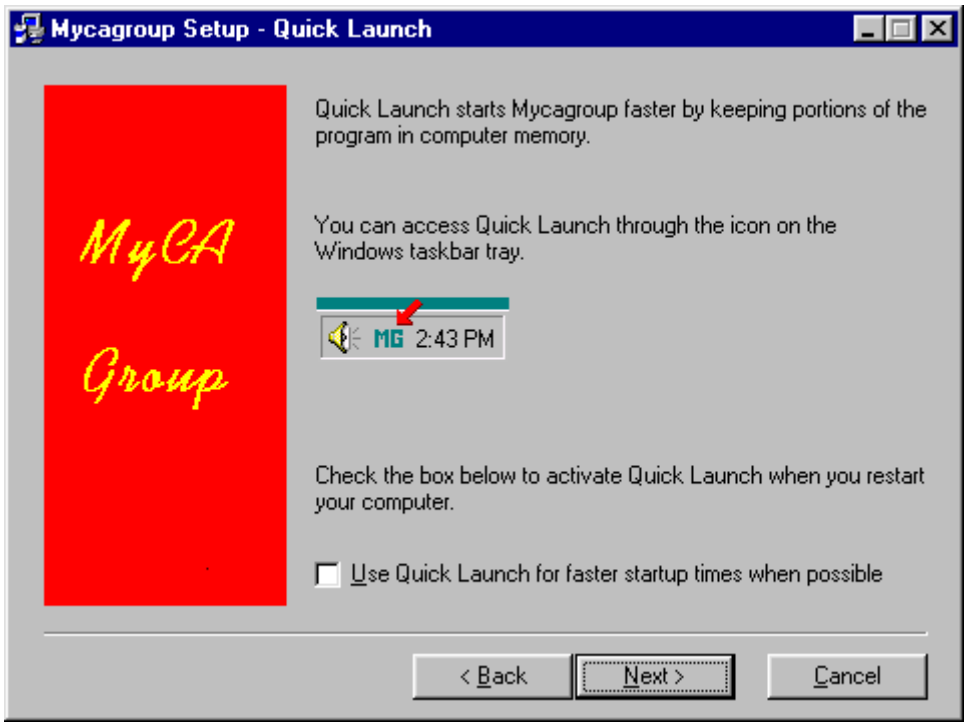
Hình 2

Người sử dụng chọn nút lệnh "Accept", trên màn hình xuất hiện hộp hội thoại "Setup type" như hình 3



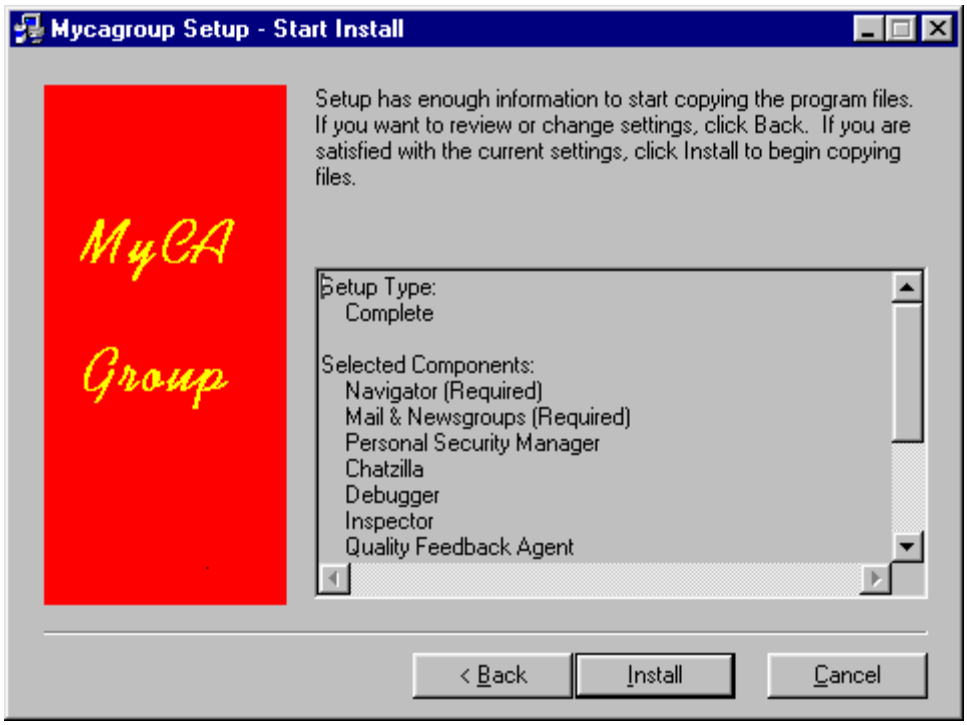
Hình 3

Người sử dụng có thể chọn một trong hai cách cài đặt: cài đặt toàn bộ phần mềm "Complete", hoặc cài đặt các thành phần theo yêu cầu của người sử dụng. Người sử dụng cũng có thể đặt đường dẫn đến nơi phần mềm sẽ được cài bằng cách sử dụng nút lệnh "Browse". Sau khi đã thực hiện chọn kiểu cài đặt cũng như nơi cài đặt, người sử dụng chọn nút lệnh "Next", trên màn hình xuất hiện hộp hội thoại "Quick Launch" như hình 4.



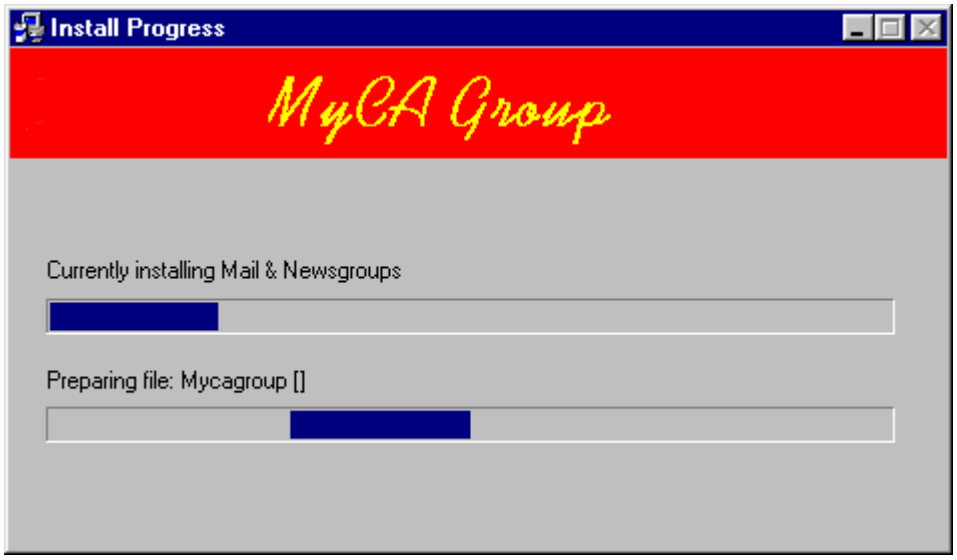
Hình 4

Khi người sử dụng sử dụng chức năng "Quick Launch" chương trình cài đặt sẽ tạo một biểu tượng của phần mềm Mybrowser trên thanh TaskBar, để người sử dụng có thể kích hoạt nhanh ứng dụng. Chọn "Next", trên màn hình xuất hiện hộp hội thoại như hình 5.



Hình 5.

Để bắt đầu quá trình cài đặt người sử dụng chọn "Install", quá trình cài đặt được bắt đầu.

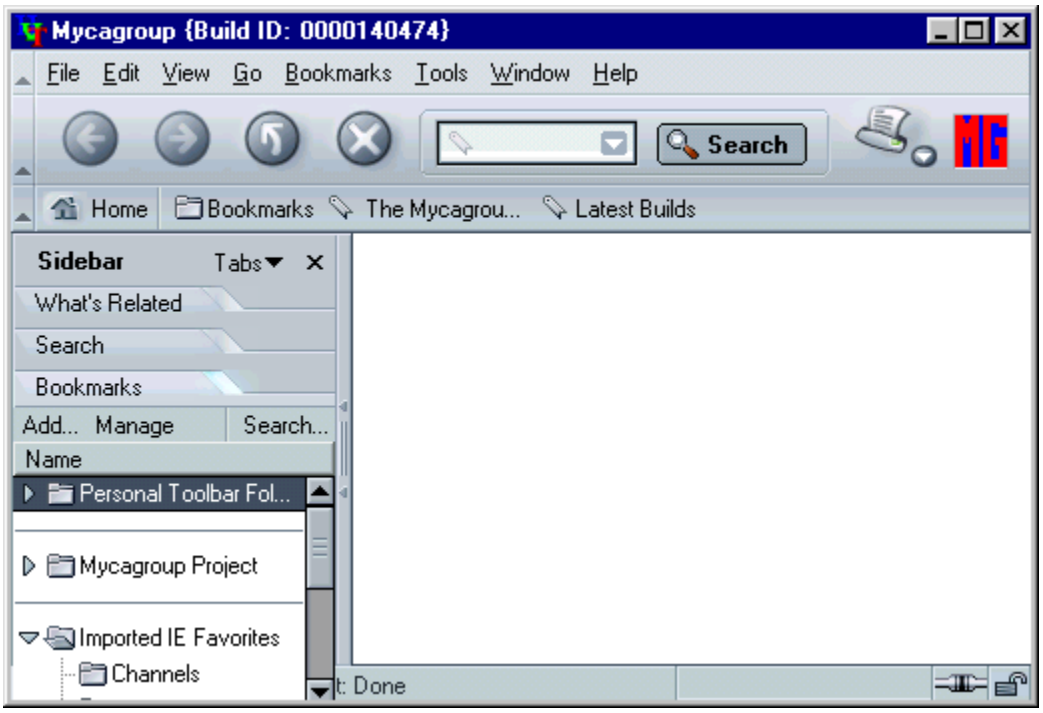


Hình 6

Quá trình cài đặt kết thúc khi trên màn hình xuất hiện logo của của phần mềm như hình 7.

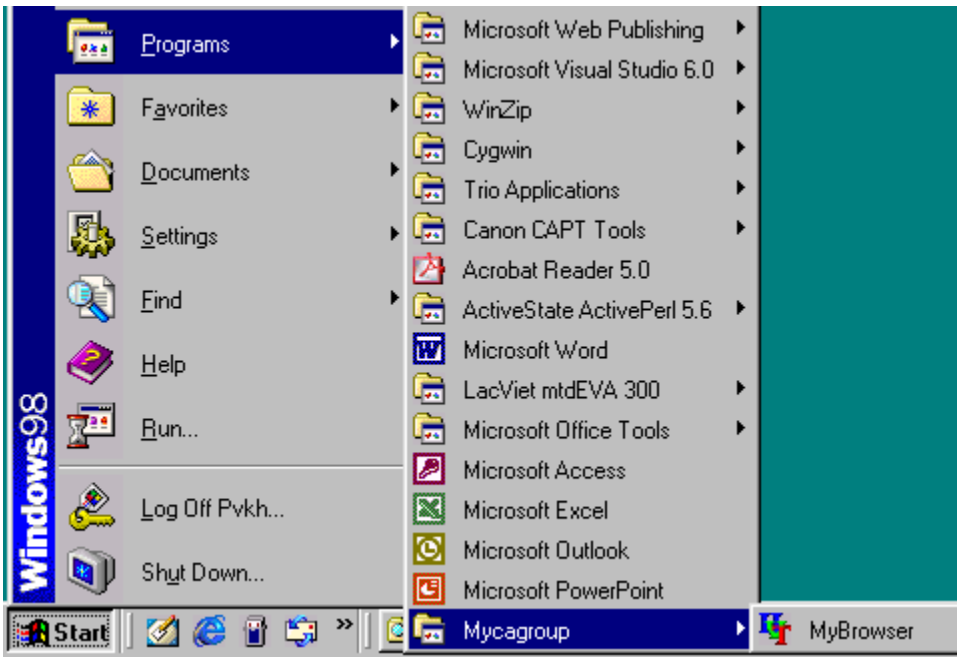
Hình 7

Sau khi logo biến mất trên màn hình xuất hiện giao diện chính của phần mềm Mybrowser như hình 8.



Hình 8

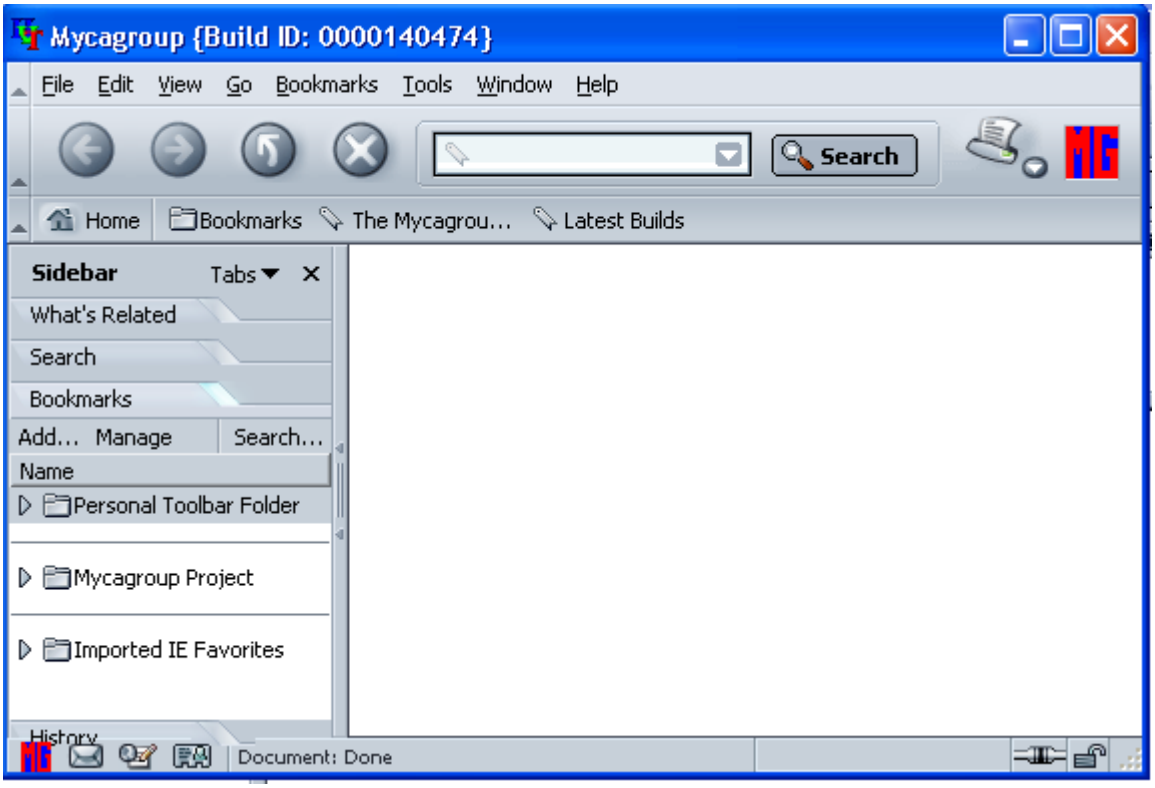
Sau khi cài đặt xong nếu người sử dụng vào mở menu Start/Programs/ sẽ thấy có mục Mycagroup/MyBrowser như hình 9.



Hình 9

1.3.2-Cài đặt CA Certificate

Certificate của nơi mà đã cấp Certificate cho Squid Server để xác thực Squid Server. Mục đích chính của việc này để xác thực Proxy Server. Mặt khác nếu bạn đã cài đặt đúng CA certificate, nhưng có lỗi không xác thực được Proxy Server thì tức là Proxy Server đó là giả mạo. Để chạy chương trình Web Client bạn chọn Start/Programs/Mycagroup/Mybrowser, xuất hiện giao diện chính của trình duyệt như hình 10 dưới đây.

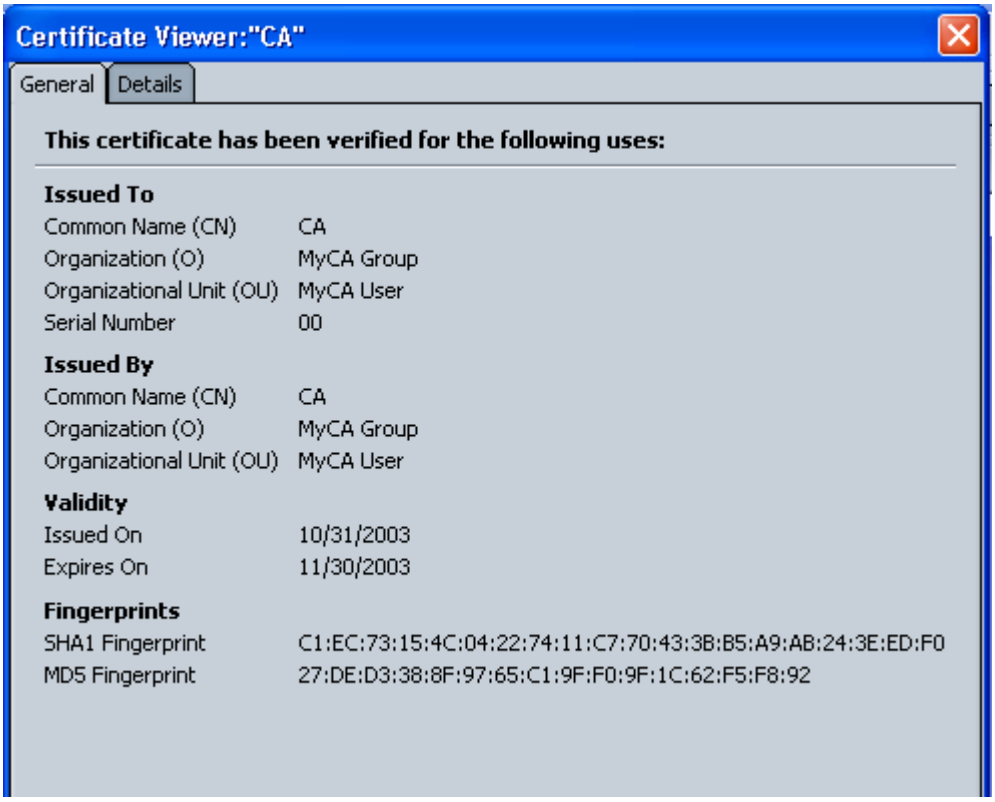


Hình 10

Để cài đặt CA certificate bạn chọn File/Open file... sau đó chọn tệp chứng chỉ của CA (có thể là RootCA hoặc None-RootCA). Chọn Open, xuất hiện hộp thoại như hình 11 dưới đây.

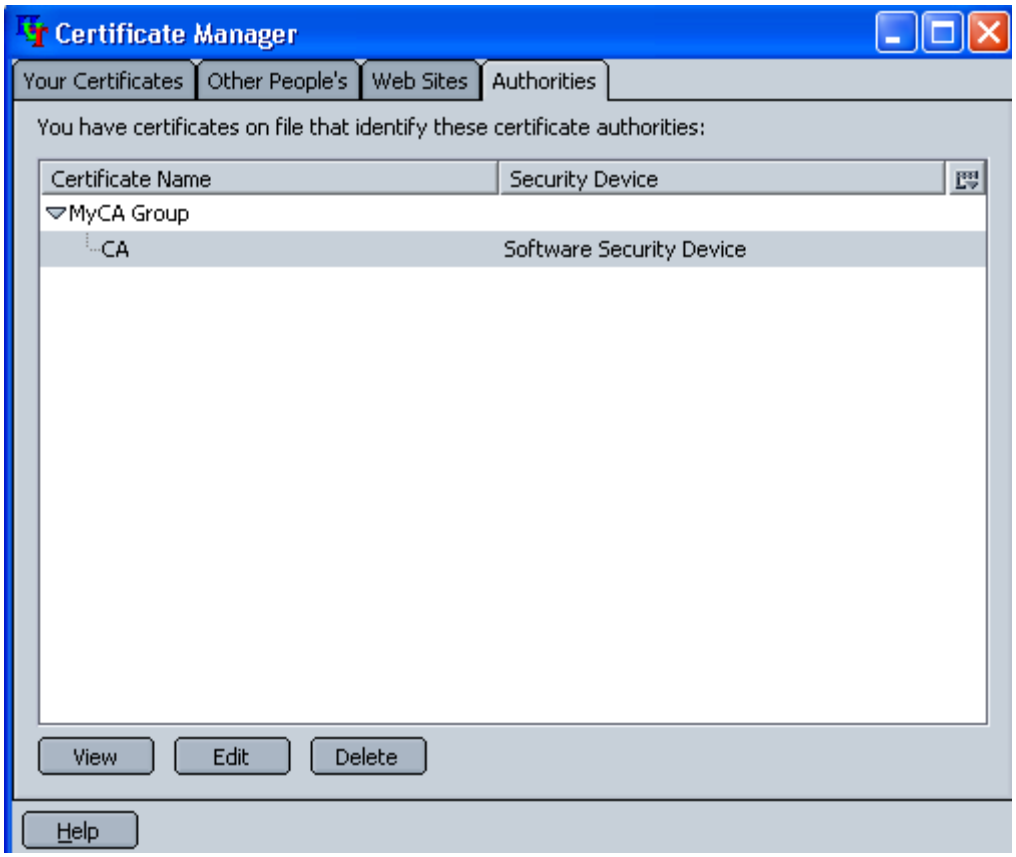
Hình 11

Bạn check vào mục "Trust this CA to indentify web sites", sau đó chọn OK. Để hiển thị thông tin về CA bạn chọn View, xuất hiện hộp thoại như hình 12 dưới đây.



Hình 12

Để kiểm tra xem chứng chỉ của CA đã được cài vào Web browser hay chưa bạn chọn Edit/Preferences.../Privacy & Security/Certificates/Manage Certificates.../Authorities, khi đó xuất hiện hộp thoại như hình 13 dưới đây.



Hình 13

Trong ví dụ của chúng tôi thì chứng chỉ để xác thực Proxy Server có tên là CA.

Chú ý: Trong file hosts (nếu Web client chạy trên nền Linux thì ở trong thư mục /etc, nếu Web client chạy trên nền Windows thì ở trong thư mục \windows), ta thực hiện ánh xạ địa chỉ thành tên virtual host như sau:

128.1.1.2 rao1
128.1.1.2 rao2

2-Các mô hình thực hiện bảo mật dịch vụ Web

Để đơn giản chúng tôi thể đưa ra một mô hình hệ thống như sau: một máy Web Client thực hiện truy cập trang web đặt trên máy chủ Web Server, đặt giữa Web Client và Web Server là một máy Squid Proxy Server (có chạy MySSL).

- **Máy1:**Có thể sử dụng hệ điều hành Linux hoặc Windows, cài đặt Web Server như MS IIS hoặc Apache Server.
- **Máy2:** Sử dụng hệ điều hành Linux, cài đặt Squid kết hợp với MySSL
- **Máy3:**Có thể sử dụng hệ điều hành Linux hoặc Windows, cài đặt trình duyệt Mybrowser.

Chú ý:

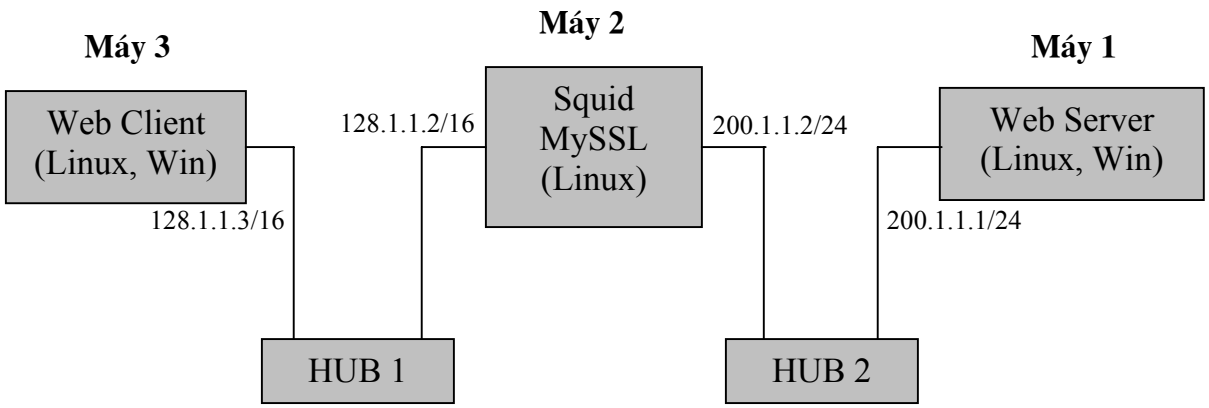
- Trên máy2 (Squid) thực hiện hai lệnh sau:
  /usr/local/squid/sbin/squid -z
  /usr/local/squid/sbin/squid -D -N -X

- Chạy trình duyệt Mybrowser, sử dụng giao thức https để kết nối tới Squid trên **máy2**. Ví dụ với Apache được cài trên **máy1**, tại **máy3** ta gõ địa chỉ: https://128.1.1.2 sẽ hiện trang chủ mặc định của **Apache**. Với IIS được cài trên **máy1**, tại **máy3** ta gõ https://128.1.1.2/CertSr sẽ hiện trang chủ của **IIS**.
- Việc hỗ trợ SSL chỉ được thực hiện khi cấu hình Squid chạy trong chế độ tăng tốc Web (Accelerator).

- Web Server hoạt động ở chế độ bình thường (không hỗ trợ SSL). Chúng ta không động gì vào Web Server.

2.1-Mô hình thử nghiệm qua Ethernet

Mô hình thử nghiệm qua Ethernet được chúng tôi thiết lập đơn giản như sau: Máy 1 là máy Web Server, có thể chạy MS IIS (Windows) hoặc Apache Server (Linux), với địa chỉ Ethernet là 200.1.1.1/24. Máy này được nối với Máy 2 là Proxy Server thông qua HUB 2. Máy Proxy Server là máy cài hệ điều hành Linux RedHat 7.2 và chạy chương trình Squid+SSL, có địa chỉ Ethernet thứ nhất eth0 là 200.1.1.2/24 và địa chỉ Ethernet thứ 2 eth1 là 128.1.1.2/16. Máy 3 là Web Client chạy trên hệ điều hành Windows hoặc Linux, có địa chỉ Ethernet là 128.1.1.3/16 và Web Client truy cập vào Web Server sử dụng trình duyệt Mybrowser. Mô hình cụ thể như sau:

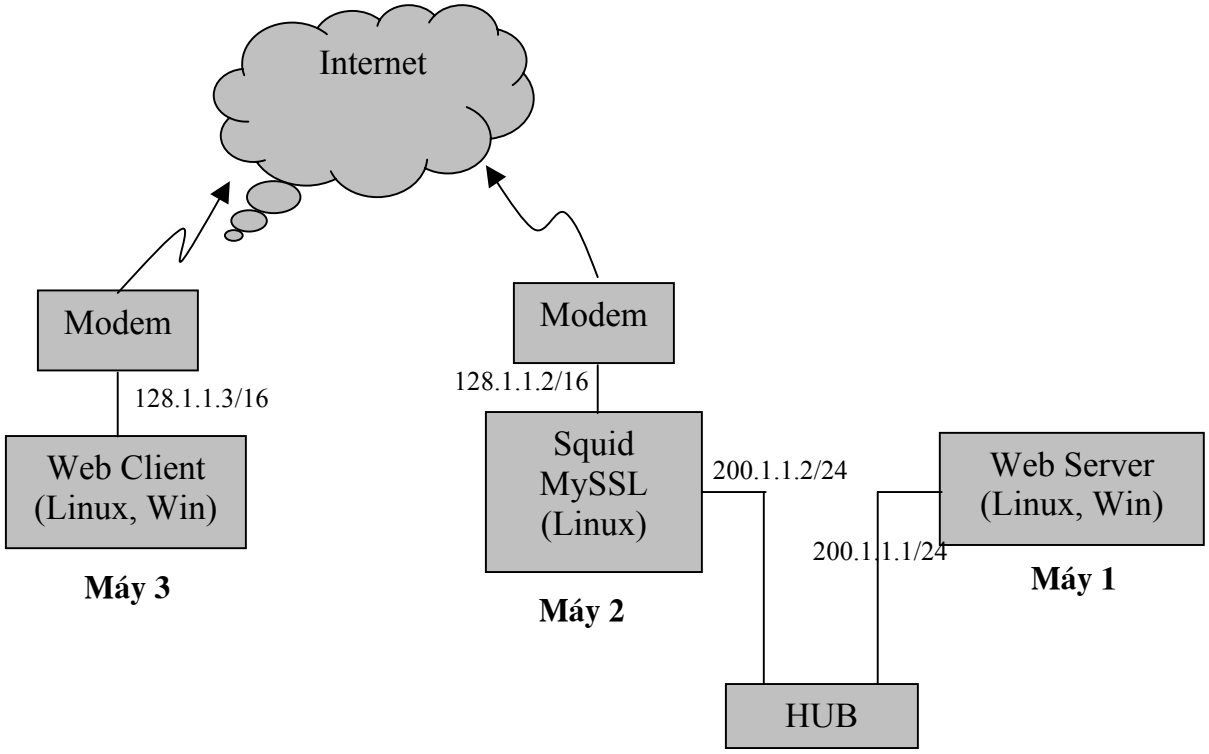


Mô hình thử nghiệm qua Ethernet

2.2-Mô hình thử nghiệm qua Dial-up

Mô hình thử nghiệm qua Dial-up được chúng tôi thiết lập đơn giản như sau: Máy 1 là máy Web Server, có thể chạy MS IIS (Windows) hoặc Apache Server (Linux), với địa chỉ Ethernet là 200.1.1.1/24. Máy này được nối với Máy 2 là Proxy Server

thông qua HUB. Máy Proxy Server là máy cài hệ điều hành Linux RedHat 7.2 và chạy chương trình Squid+SSL, có địa chỉ Ethernet là 200.1.1.2/24 và địa chỉ Internet là 128.1.1.2/16. Máy 3 là Web Client chạy trên hệ điều hành Windows hoặc Linux, có địa chỉ Internet là 128.1.1.3/16 và Web Client truy cập vào Web Server sử dụng trình duyệt Mybrowser. Mô hình cụ thể như sau:



Mô hình thử nghiệm qua Dial-up

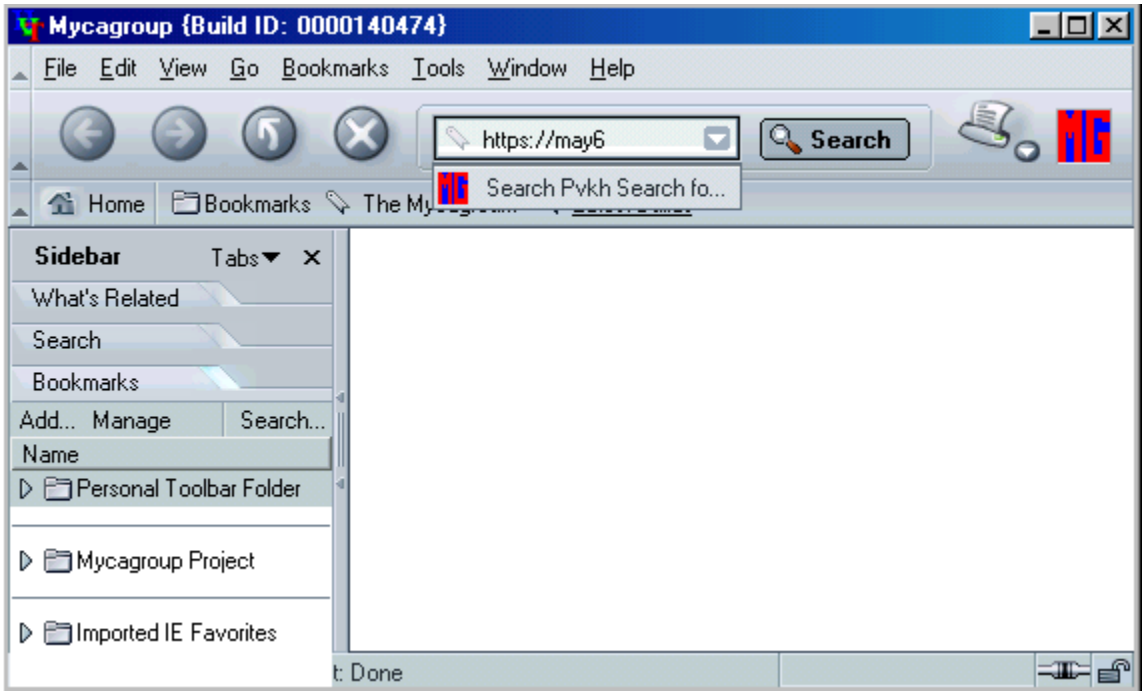
3-Thực hiện bảo mật dịch vụ Web

Các dịch vụ web được truy cập bảo mật từ trình duyệt Mybrowser thông qua Proxy (sử dụng Squid+MySSL) hoàn toàn trong suốt đối với người sử dụng, hoạt động như thông thường. Nhưng thông tin trang web trên đường truyền sẽ được bảo mật theo chuẩn SSLv3/TLS1. Trong phiên bản này thì chưa hỗ trợ đầy đủ mô hình xác thực SSLv3/TLS1: Web client gửi client-hello-message sang Proxy Server, khi nhận được hello message từ phía Client thì Proxy Server gửi server-hello-message, sau bước này thì 2 bên đã thoả thuận xong các thuật toán mã hoá, xác thực (MK1, RSA) và khoá của phiên liên lạc. Sau đó, Proxy Server tiếp tục gửi certificate sang client. Bên client sử dụng CA certificate đã được cài trong Web Client để xác thực Proxy Server. Web Client gửi Client-done tới Proxy Server và ngược lại Proxy Server gửi lại trả lại Server-done. Đến đây, mọi yêu cầu của Web Client tới Proxy Server và ngược lại đều được bảo mật, và yêu cầu này sẽ được Proxy dịch và gửi tới Web Server như thông thường. Như vậy, so với mô hình xác thực đầy đủ thì Web

Client cũng phải gửi certificate sang để Proxy Server tiến hành việc xác thực Client. Mới đây đã ra bản Squid-3.0 hỗ trợ đầy đủ mô hình xác thực SSLv3/TLS1, nhưng chưa được hoàn thiện, trong tương lai chúng ta sẽ sử dụng bản Squid này. Dưới đây chúng tôi sẽ giới thiệu việc thiết lập, thực hiện dịch vụ bảo mật Web vơi Web server là IIS trên Windows 2000. Đối vơi Web server khác hoàn tương tự, vì chúng ta không hề can thiệp đến Web server trong quá trình thực hiện bài toán bảo mật

3.1-Truy nhập trang Web

Để sử dụng dịch vụ truy cập Web bảo mật, bạn chạy chương trình Mybrowser, chọn Start/Programs/Mycagroup/Mybrowser, và gõ trang web cần truy cập tới như hình 14 dưới đây.

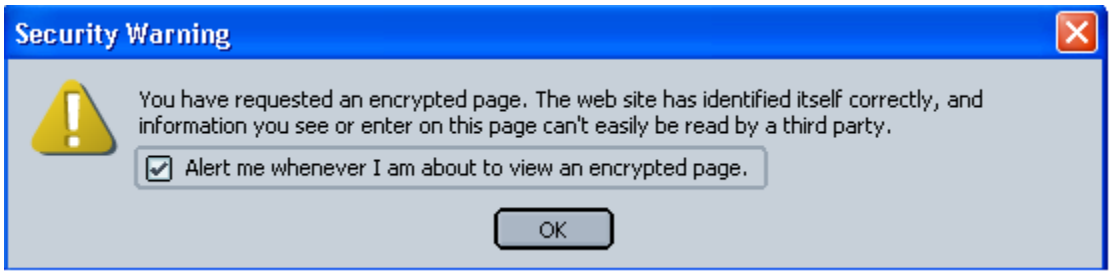


Hình 14

Nếu bạn không thực hiện cài đặt CA certificate (đã cấp chứng chỉ cho Proxy Server) hoặc việc xác thực Proxy Server sai, thì Mybrowser sẽ đưa ra hộp thoại cảnh báo rằng không biết CA đã cấp chứng chỉ cho Proxy Server, do đó, việc xác thực Proxy Server coi như lỗi. Hộp thoại thông báo việc xác thực Proxy Server như hình 15 dưới đây.

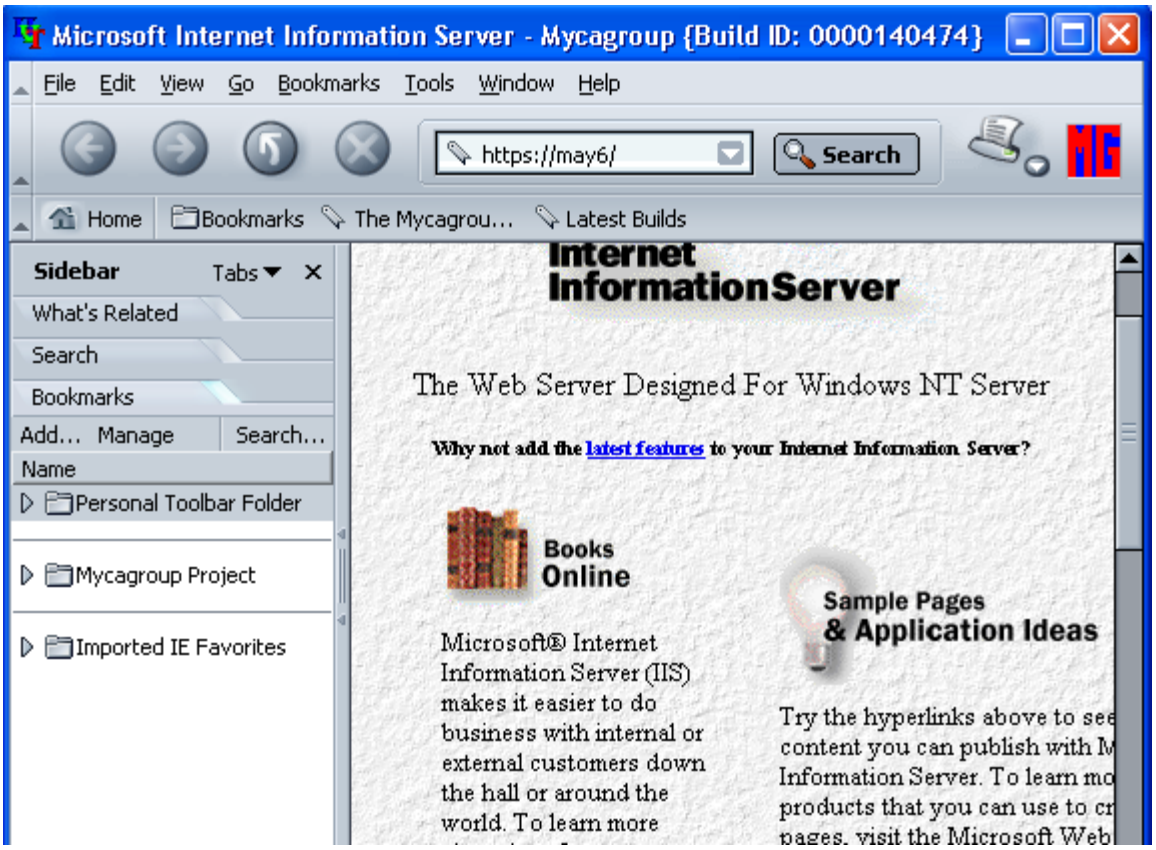
Hình 15

Nếu công việc xác thực Proxy Server đúng thì xuất hiện hộp thoại cảnh báo như hình 16 dưới đây.



Hình 16

Chọn OK để tiếp tục, khi đó trang web bảo mật như hình 17 dưới đây.



Hình 17

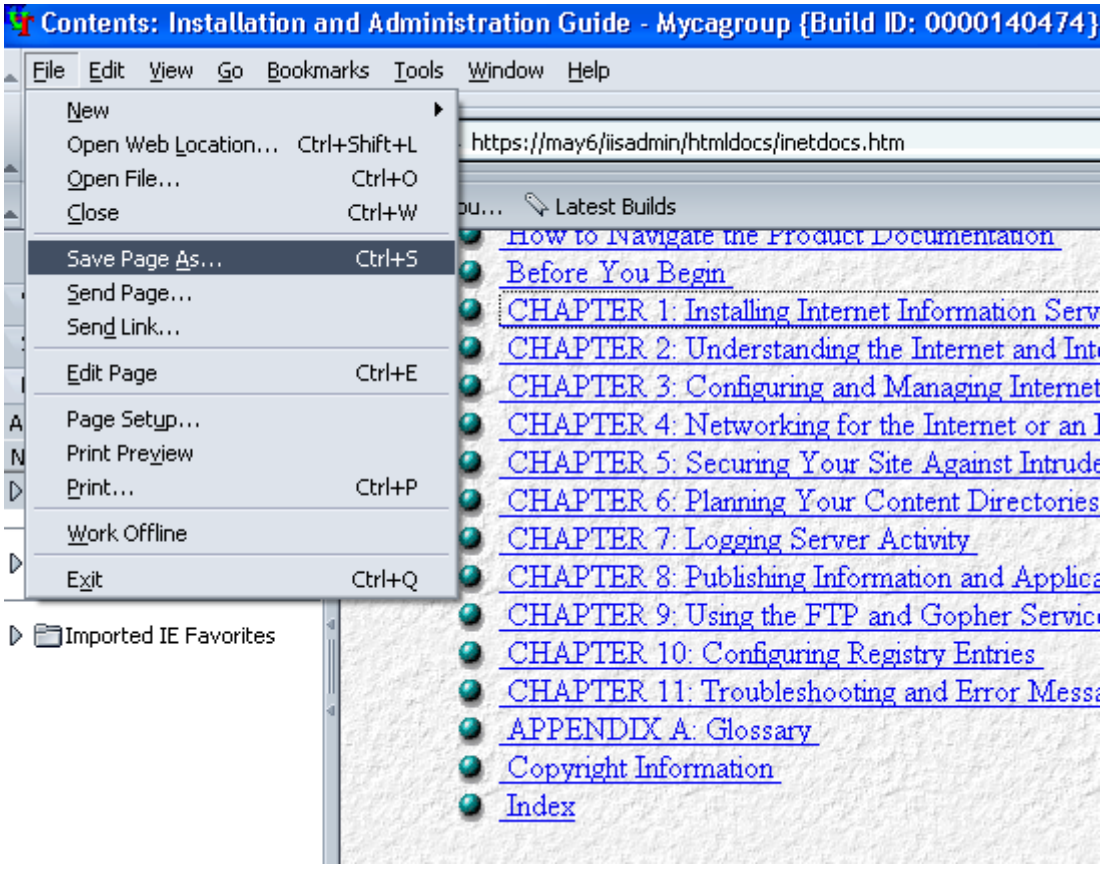
Đến đây dịch vụ truy nhập trang web từ trên Web Server được thao tác hoàn toàn như các trình duyệt web khác.

3.2-Tải tệp

Các thao tác đối với dịch vụ tải tệp từ trang Web của Web Server cũng như đối với các trình duyệt khác. Cụ thể, bạn có thể chia ra làm 2 trường hợp: ghi trang web vào hệ thống, và download tệp.

3.2.1-Ghi trang web

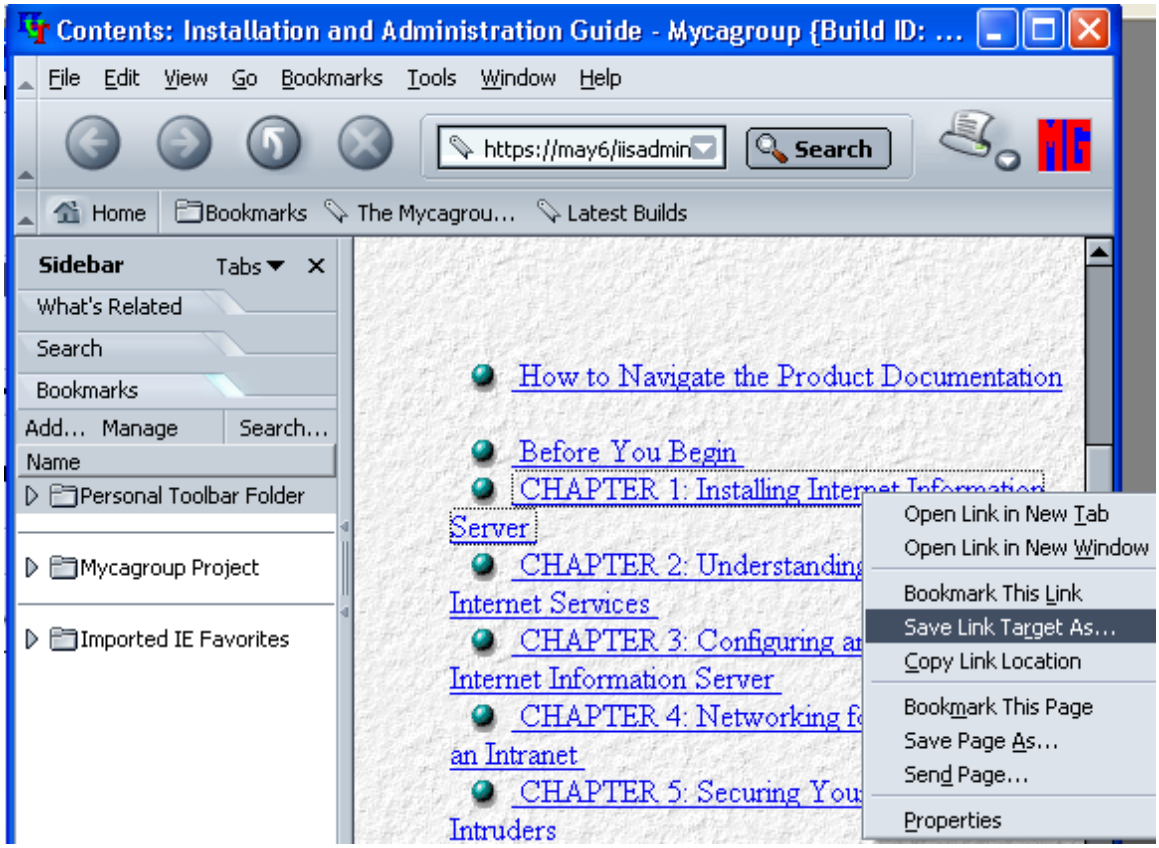
Ví dụ bạn đã truy cập vào trang web của Web Server có tên là may6 như hình 18 ở trên, muốn ghi trang web vào hệ thống của mình bạn chọn File/Save Page As... như hình 18 dưới đây.



Hình 18

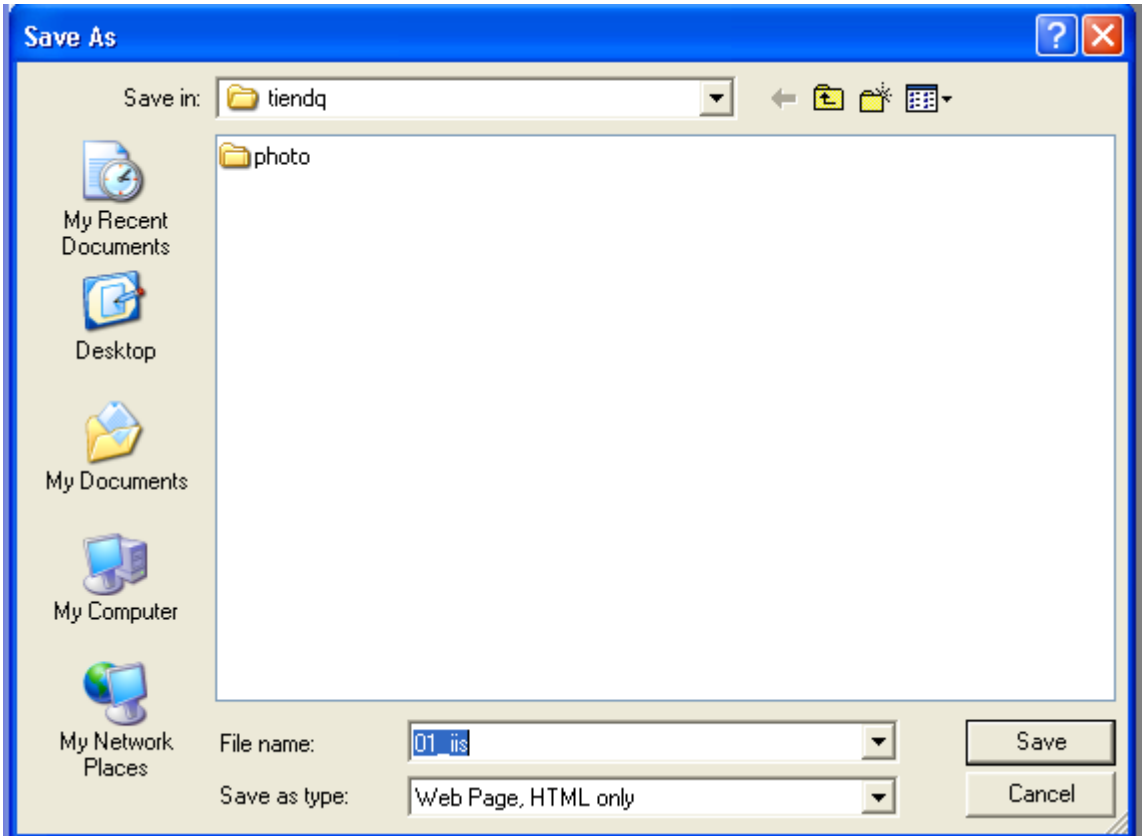
3.2.2-Download tệp

Giả sử bạn đã truy cập vào trang web của Web Server có tên là may6 như hình 18 ở trên. Muốn download tệp, bạn đặt con trỏ chuột vào tệp cần download và bấm phải chuột, sau đó chọn Save Link Target As... như hình 19 dưới đây.



Hình 19

Với thao tác đó cho phép bạn tải tệp và ghi vào hệ thống của bạn, như hình 20 dưới đây.



Hình 20

Bạn chọn nơi cần lưu tệp tải về và sau đó chọn "Save" để hệ thống tải tệp, ở ví dụ trên chúng tôi tải tệp 01_iis ghi vào thư mục c:/tiendq.