建置環境與說明:

作業系統 – Ubuntu

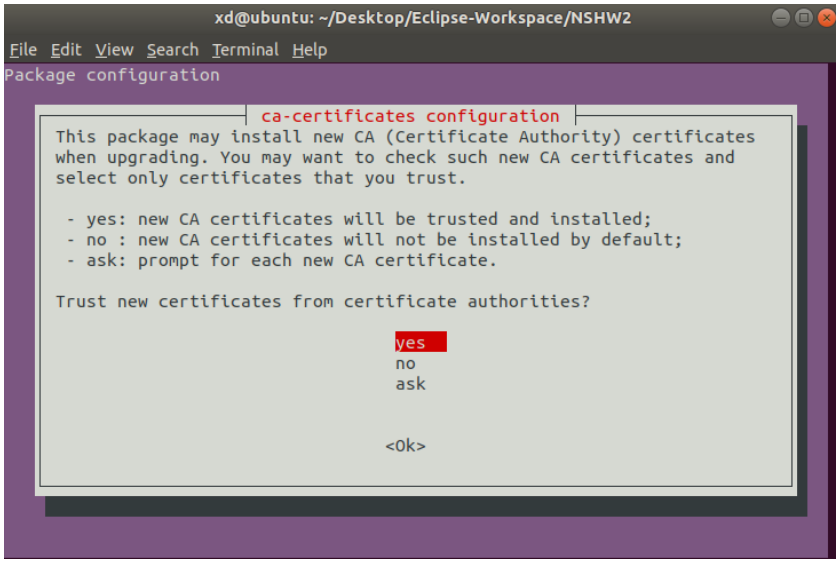
撰寫 IDE – Ellipse

使用 Library – openssl-lib

說明:

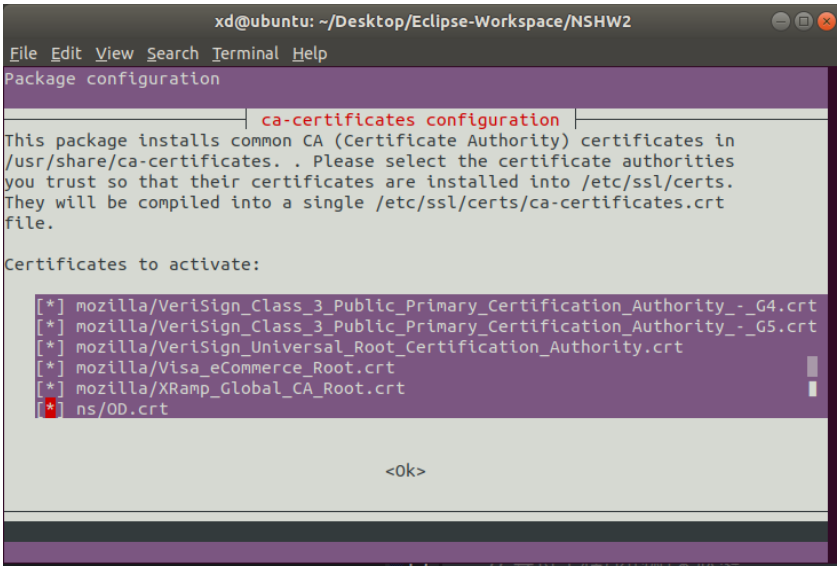
如果要讓 Server 以及 Client 使用預設的憑證建立連線，則在執行 Server/Client 之前，需要先將 ./server/CA-cert.pem 加入 /etc/ssl/certs/ca-certificates 當中。我寫了一個簡單的 shell 來做這一件事:

1. 執行 ./install_ca.sh
2.



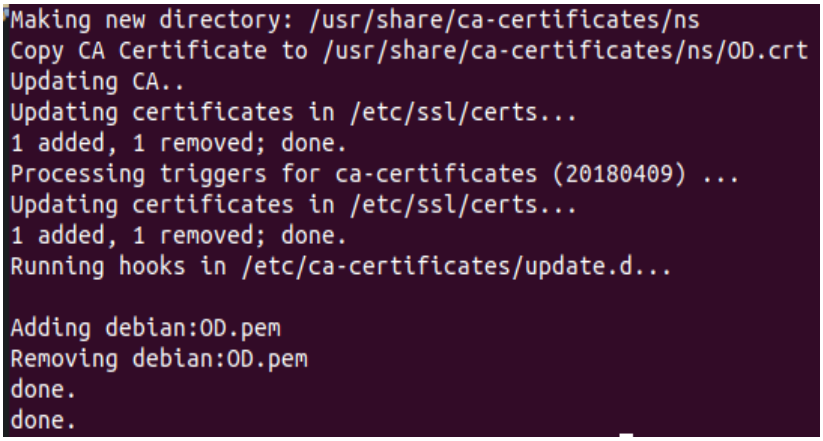
(選擇 yes 後按下 Enter)

3.



(按住鍵盤的下鍵到最底端後，把紅方塊移到 ns/OD.crt，
按下空白鍵選擇，確定方塊內有*圖示後按下 Enter)

4.



(出現 Adding debian:OD.pem 就表示成功加入)

重要程式碼說明:

```
//Setup Openssl Prerequisites
init_openssl();
ctx = create_context();
configure_context(ctx, "./server/cert.pem", "./server/key.pem");
```

(SSL 環境的建立)

```
void configure_context(SSL_CTX *ctx, const char cert_loc[],
                const char key_loc[]) {
        //      SSL_CTX_set_options(ctx, SSL_OP_NO_SSLv2);
        //      SSL_CTX_set_ecdh_auto(ctx, 1);

        //enable verification
        SSL_CTX_set_verify(ctx, SSL_VERIFY_PEER, verify_callback);

        //set ca location to /etc/ssl/certs
        if (SSL_CTX_load_verify_locations(ctx, NULL, "/etc/ssl/certs") == 0) {
                ERR_print_errors_fp(stderr);
                exit(EXIT_FAILURE);
        }

        //set the key and cert
        if (SSL_CTX_use_certificate_file(ctx, cert_loc, SSL_FILETYPE_PEM) <= 0) {
                ERR_print_errors_fp(stderr);
                exit(EXIT_FAILURE);
        }

        if (SSL_CTX_use_PrivateKey_file(ctx, key_loc, SSL_FILETYPE_PEM) <= 0) {
                ERR_print_errors_fp(stderr);
                exit(EXIT_FAILURE);
        }

        PINFO("SSL Context Configured.");
}
```

(ssl_helper.cpp:42 :SSL 開啟夥伴驗證機制,

SSL_CTX_load_verify_locations 設定去哪裡找 Issuer CA。

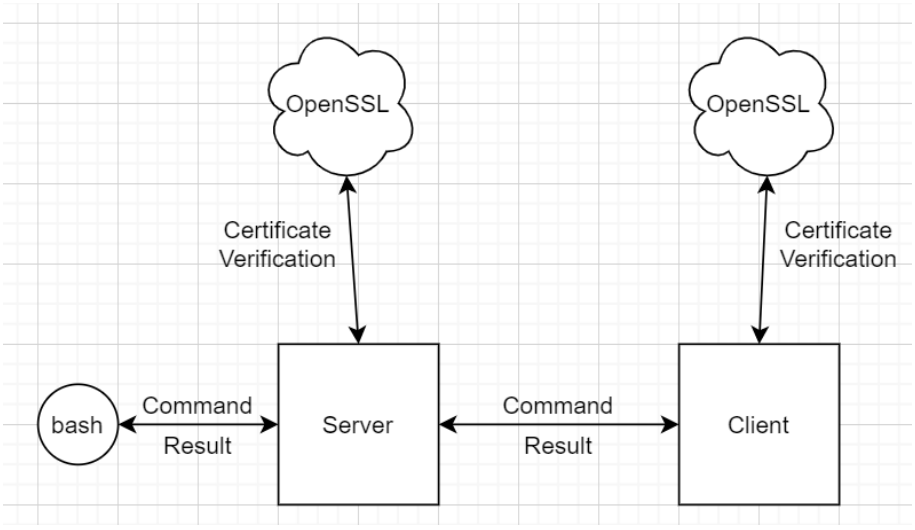
ca-certificates 在/etc/ssl/certs 裏頭,

而 OD.crt 的內容在 ca-certificates 裏頭)

```
if ((code & EPOLLIN) > 0 && strlen(char_s2b_array) < BUF_LEN) {
        ssize_t initial_size = strlen(char_s2b_array);
        remain_size = BUF_LEN - strlen(char_s2b_array);
        read_size = SSL_read(ssl, &char_s2b_array[initial_size],
                        remain_size);
        PINFO(
                        "Client command received: [" << &char_s2
        super weird condition.
        epoll told me there were data in file descriptor, but re
        suppose this descriptor is broken?
        even the event codes above didn't detect the error.
        if (read_size == 0) {
                goto EPOLL_END;
        }
}
```

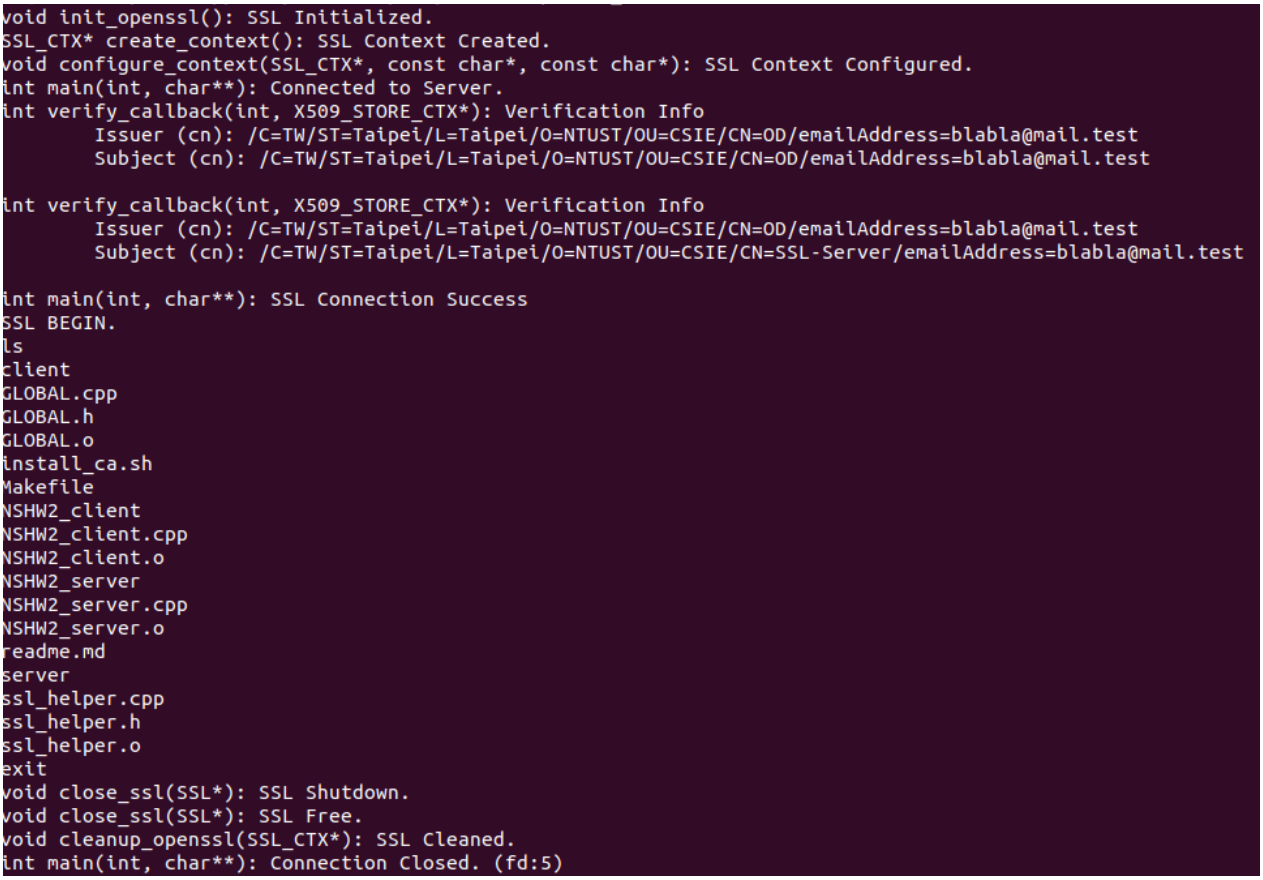
(NSHW2_server:168 : if(read_size==0)這邊是特別處理情況，
當 ssl 連線突然關閉時 epoll 會通知 socket 可以進行 read，
可是不管怎麼 read 都會是 0，詭異的是 socket 的狀態顯示連線正常，
因此在這裡強制結束 socket 連線以及 client 的 service。)

設計架構與功能說明:



Server 會在 localhost:5000 接聽連線，提供 bash shell 給 Client，建立在 TLS/SSL 連線上。Server 以及 Client 會驗證憑證，驗證 CA 是否在/etc/ssl/certs/這個資料夾底下。

成果截圖:



(執行 ls 後，server 回傳執行結果。)

困難與心得:

整個過程中最難的點有兩個 1.)SSL Validation　2.)Epoll 的使用，SSL Validation 在重要程式碼的第二張圖片中列出了解法，當初一直遇到 Issuer Certificate Not Found 相關的 Error Code，找個好久才解出來。Epoll 因為是新東西不太熟悉，再搭配 multiprocessing 不是很好 debug，IDE 只能 trace 一個 process 造成 debug 的效率很低。還有 ssl 沒有辦法斷乾淨的問題也卡一段時間。

不過寫這個還蠻好玩的，讚讚。