

[OSX][逆向]靜態爆破 + 重簽 Sublime3無限授權 with Hopper

Published by [Adr](#) on [2015-09-15](#)

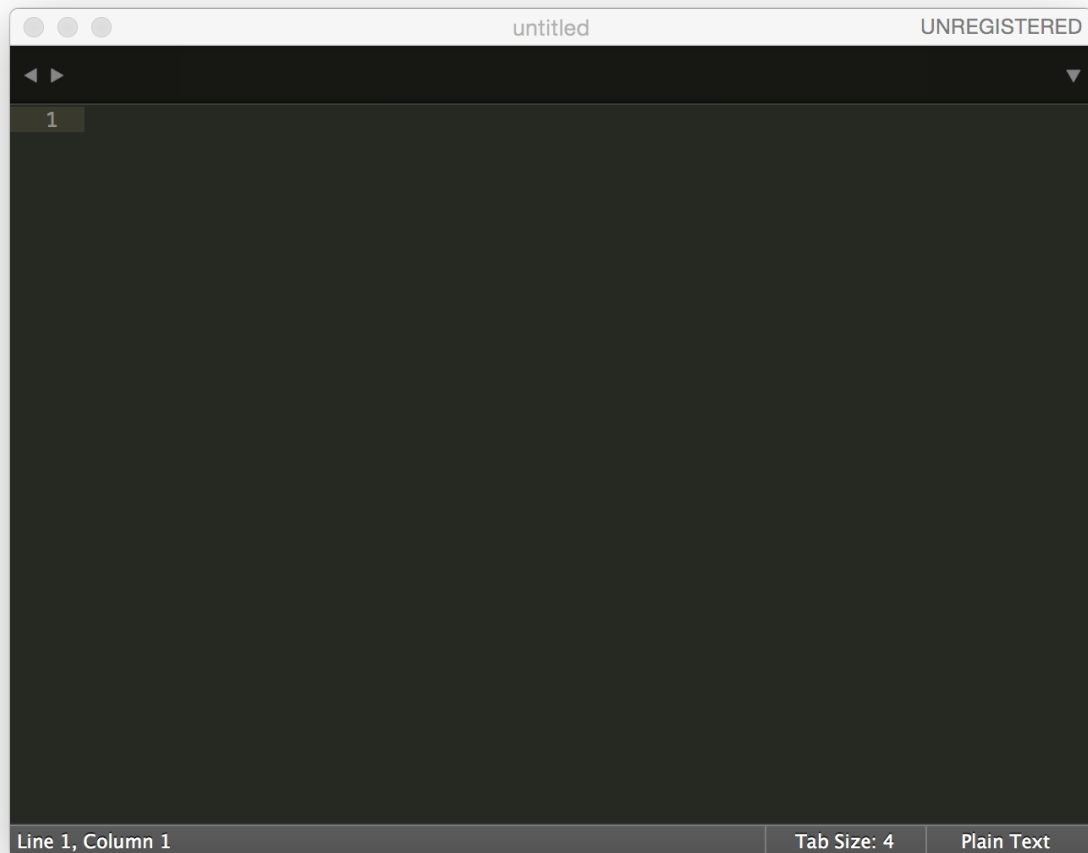
動機



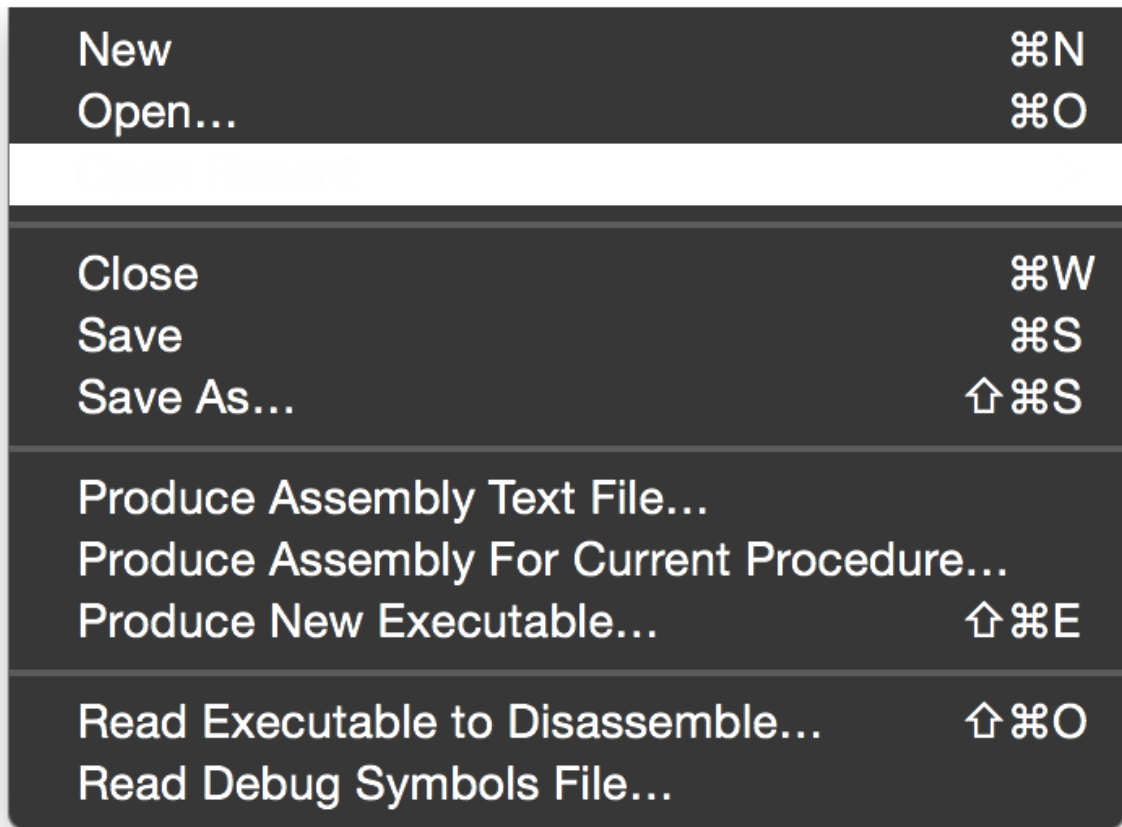
HI，最近轉移到新的部落格Word Press上啦～～來發發廢文洗人氣吧（？）前陣子才上手開始用Hopper練習逆向OSX軟件，最近找些App練手，原本是拿Sublime2練習沒想到練完之後，某OSX大牛表示：「人家Sublime都更新到3啦！」於是我就來摸摸看Sublime3了XD，3的版本內比較多檢測點的樣子，這邊給個通殺檢測點的思路跟破解打補丁後，如何幫破解好的App二次打包簽名的完整過程。（PS：如果用原本Hopper的Patch功能，Patch完是假移除簽名狀態，OSX會偵測簽名損毀導致不顯示在Launchpad上的限制）上圖為完全繞過註冊檢測&簽名完成版本的示意圖。

附註：本文章僅做學術研究，開發者很辛苦，請支持正版軟體付費使用
Sublime3！

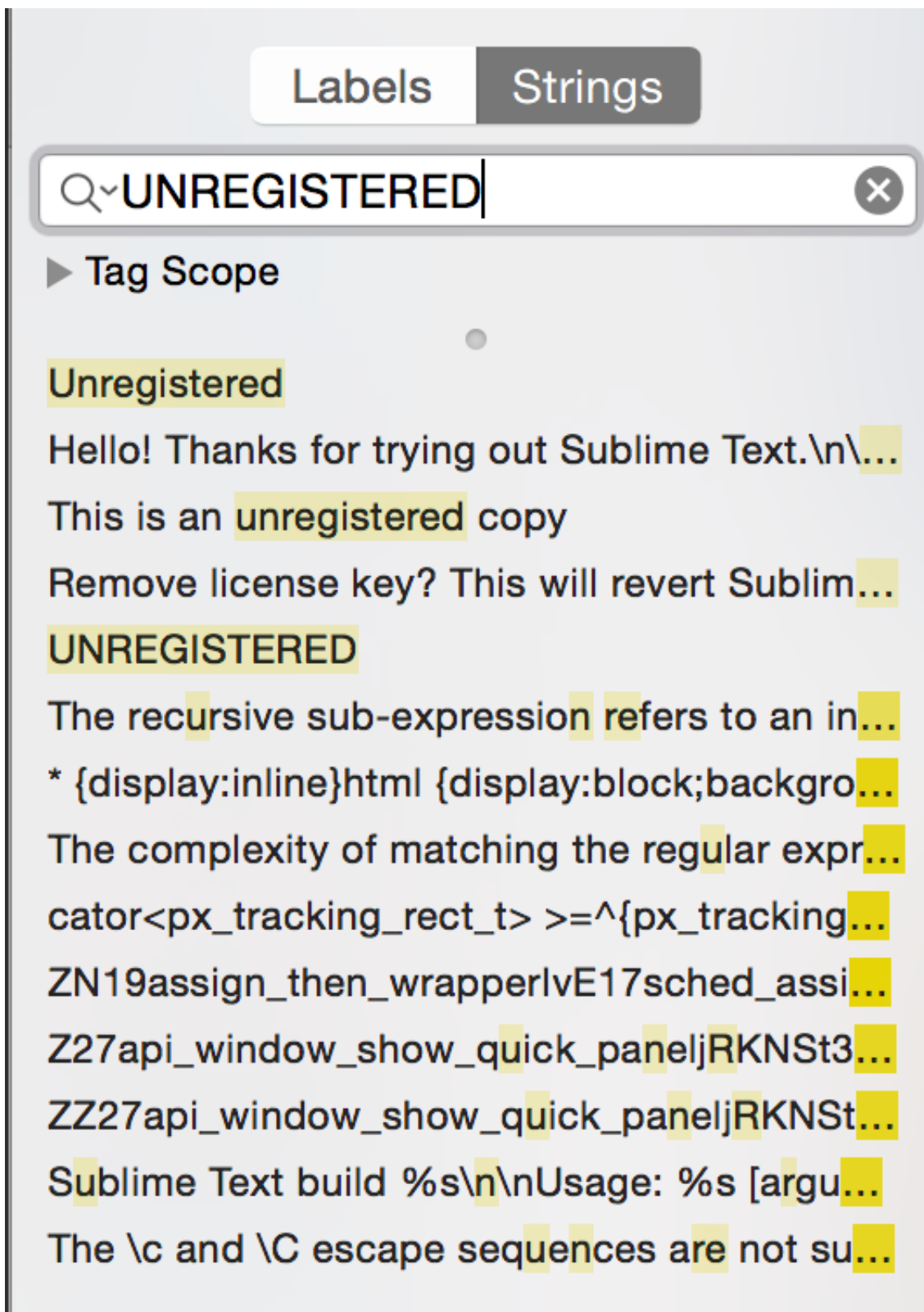
Setup



首先我們需要一個乾淨的Sublime3軟件，我們可以到[Sublime3官網](#)上下載官方正式試用版本（會未完成註冊的字樣與儲存時彈窗提示購買授權信息）安裝完後點開後，如上圖。



接著打開Hopper Disassembler，選擇File > Read Executable To Disassemble打開Sublime3。



回憶一下未註冊狀態下Sublime會在右上角出現一個“UNREGISTERED”全大寫字樣提示，意味著軟件內應該有一個String字串負責儲存，很快地就可搜索到（如上圖第五個）接著點擊之查閱詳細內容。

```

001004c4772 db "3083", 0 ; XREF= __Z16app_build_numberv+4, __Z11app_versionv+4
001004c4777 db "fork", 0 ; XREF= __ZL13osx_send_dumpPKcS0_Pvb+748
001004c477c db "exec", 0 ; XREF= __ZL13osx_send_dumpPKcS0_Pvb+693, __Z16build_controller5
001004c4781 db "about_window", 0 ; XREF= __Z12about_window15class_get_classEv+52
001004c478e db "Registered to ", 0 ; XREF= __Z12about_window4drawEP17px_render_context4rect+417
001004c479d db "Unregistered", 0 ; XREF= __Z12about_window4drawEP17px_render_context4rect+767
001004c47aa db "Copyright \xC2xA9 2006-2015 Sublime HQ Pty Ltd", 0 ; XREF= __Z12about_window4drawEP17px_render_context4rect+767
001004c47d4 db "Stable Channel", 0 ; XREF= __Z12about_window4drawEP17px_render_context4rect+1163
001004c47e3 db " ", Build ", 0 ; XREF= __Z12about_window4drawEP17px_render_context4rect+1185
001004c47ec db " /_Index", 0 ; XREF= __Z10app_loader7on_initEv+371
001004c47f4 db " /_Update", 0 ; XREF= __Z10app_loader7on_initEv+440
001004c47fd db "Minimap.sublime-settings", 0 ; XREF= __Z10app_loader18on_resources_readyEv+57, __Z10text_sh
001004c4816 db "New Window", 0 ; XREF= __Z10app_loader18on_resources_readyEv+237
001004c4821 db "new window" 0 ; XREF= __Z10app_loader18on_resources_readyEv+251, __Z11window

```

接著可以在該字串右側看到xref = ...，點擊進去可以查看引用該String物件的點

```
001000040aa      addsd      xmm0, xmm1
001000040ae      addsd      xmm0, xmm1
001000040b2      movsd      qword [ss:rbp+var_4C0], xmm0
001000040ba      mov        rbx, qword [ds:r14+0x108]
001000040c1      lea        r15, qword [ds:0x1004c479d] ; "Unregistered"
001000040c8      lea        r13, qword [ds:0x1004c47a9]
001000040cf      mov        rdi, rbx ; argument #1 for method
001000040d2      mov        rsi, r15 ; argument #2 for method
001000040d5      mov        rdx, r13
001000040d8      call       __Z17px_measure_stringP9px_font_t15const_substring ; px_measure_string
001000040dd      call       imp__stubs__ceilf
001000040e2      cvtss2sd   xmm0, xmm0
001000040e6      mulsd      xmm0, qword [ds:0x10045b0c8]
001000040ee      addsd      xmm0, qword [ss:rbp+var_4E0] ; argument #1 for method
001000040f6      call       imp__stubs__floor
001000040fb      mov        rax, qword [ds:r12]
001000040ff      mov        edx, 0xffffffff
00100004104      mov        rdi, r12
00100004107      mov        rsi, rbx
0010000410a      movsd      xmm1, qword [ss:rbp+var_4C0]
00100004112      mov        rcx, r15
00100004115      mov        r8, r13
00100004118      call       qword [ds:rax+0x18]
0010000411b      jmp        0x1000041a4
```

接著很快就可以看到xref追回去的點上，它把文字存放的地址寫入了r15接著底下放到rsi內在call API實作控制UI物件的顯示文字，底下一拖拉庫的組語code...

到這邊我們思路可以很清楚知道，到這邊就是實作了「當未註冊狀態下」該做的事情，意味著往上回推應該可以找到判斷點「判斷為何該執行未註冊事件」，Hopper是個很棒的工具，它有個可以替我們反組譯的功能（回推Obj-C狀態，有點類似IDA Pro的F5，但個人認為Hopper在OSX方面略勝一籌）接著可以看到此段回推Code如下：

```
else {
    xmm0 = intrinsic_movsd(xmm0, var_4C0);
    xmm1 = intrinsic_movsd(xmm1, *0x100463cd8);
    xmm0 = intrinsic_addsd(xmm0, xmm1);
    xmm0 = intrinsic_addsd(xmm0, xmm1);
    var_4C0 = intrinsic_movsd(var_4C0, xmm0);
    rbx = *(r14 + 0x108);
    px_measure_string(rbx, "Unregistered");
    ceilf();
    xmm0 = intrinsic_cvtss2sd(xmm0, xmm0);
    xmm0 = intrinsic_mulsd(xmm0, *0x10045b0c8);
    xmm0 = intrinsic_addsd(xmm0, var_4E0);
    floor();
    rax = *r12;
    xmm1 = intrinsic_movsd(xmm1, var_4C0, 0xffffffff);
    (*(rax + 0x18))();
}
```

做了某個if後的else開始，底下做很多事情接著就是取出“UNREGISTERED”出來使用了，往上翻一下可翻到這段if：

```

floor();
rax = *r12;
xmm1 = intrinsic_movsd(xmm1, *0x100463cd8);
(*(rax + 0x18))(r12, rbx, 0xffffffff, "Sublime Text", 0x1004c45a7);
if (*(int8_t *)_g_valid_license != 0x0) {
    string_buffer::string_buffer();
    string_buffer::append(var_240, "Registered to ", 0x1004c479c);
    const_substring::const_substring(var_490);
    string_buffer::append(var_240, var_490, var_488);
    r15 = *(r14 + 0x108);
    string_buffer::add_terminator(var_240);
    rbx = strlen(var_238) + var_238;
    px_measure_string(r15, var_238);
    xmm1 = intrinsic_movsd(xmm1, var_4C0);
    xmm2 = intrinsic_movsd(xmm2, *0x100463cd8);
    xmm1 = intrinsic_addsd(xmm1, xmm2);
    var_4C0 = intrinsic_movsd(var_4C0, xmm1);
    var_4F0 = intrinsic_movsd(var_4F0, intrinsic_addsd(xmm2, xmm1));
    ceilf();
    xmm0 = intrinsic_cvtss2sd(xmm0, xmm0);
    xmm0 = intrinsic_mulsd(xmm0, *0x10045b0c8);
    xmm0 = intrinsic_addsd(xmm0, var_4E0);
    floor();
}

```

可以看到這邊「若*(int8*)_g_valid_license不為空」（_g_valid_license為bool格式），則執行授權訊息的處理事件，否則（else）就跳到我們不希望跳入的「處理未註冊事件」，到這邊我們可以得知「_g_valid_license」是個很關鍵的偵測點！

接著點擊「_g_valid_license」進入查閱該物件詳細引用狀況，花點時間看完xref可以翻到這段很有趣的程式碼：

```

int validate_license_checkv() {
    std::_1::basic_string<char, std::_1::char_traits<char>, std::_1::allocator<char> >::_init(var_60, "Invalid Key\n", 0xc);
    std::_1::basic_string<char, std::_1::char_traits<char>, std::_1::allocator<char> >::_init(var_78, "Unlimited User License\n", 0x17);
    rdx = var_78 & 0xff;
    rax = std::_1::basic_string<char, std::_1::char_traits<char>, std::_1::allocator<char> >::append(var_60);
    rcx = *rax;
    var_48 = rcx;
    *(rax + 0x10) = 0x0;
    *(rax + 0x8) = 0x0;
    *rax = 0x0;
    std::_1::basic_string<char, std::_1::char_traits<char>, std::_1::allocator<char> >::_init(var_90, "EA7E-1000\n", 0xa, rcx);
    rdx = var_90 & 0xff;
    rax = std::_1::basic_string<char, std::_1::char_traits<char>, std::_1::allocator<char> >::append(var_48);
    rcx = *rax;
    var_30 = rcx;
    *(rax + 0x10) = 0x0;
    *(rax + 0x8) = 0x0;
    *rax = 0x0;
    std::_1::basic_string<char, std::_1::char_traits<char>, std::_1::allocator<char> >::~~basic_string(var_90);
    std::_1::basic_string<char, std::_1::char_traits<char>, std::_1::allocator<char> >::~~basic_string(var_48);
    std::_1::basic_string<char, std::_1::char_traits<char>, std::_1::allocator<char> >::~~basic_string(var_78);
    std::_1::basic_string<char, std::_1::char_traits<char>, std::_1::allocator<char> >::~~basic_string(var_60);
    rbx = 0x0;
    r14 = "00000000000000000000000000000000\n";
    r15 = var_30;
    do {
        std::_1::basic_string<char, std::_1::char_traits<char>, std::_1::allocator<char> >::append(r15, r14);
        rbx = rbx + 0x1;
    } while (rbx < 0x9);
    rax = check_license(var_30, 0x0, 0x0, 0x0);
    if (rax == 0x0) {
        *(int8_t *)_g_valid_license = 0x0;
    }
    rax = std::_1::basic_string<char, std::_1::char_traits<char>, std::_1::allocator<char> >::~~basic_string(var_30);
    return rax;
}

```

「validate_license_checkv」函數你花點時間看一下Code順序（或者理論上推理一下）基本上可以得知它會是所有「判定授權確認」都做完檢查後，最後才會透過「validate_license_checkv」這個函數來替我們處理UI應該顯示當前Sublime3是授權給誰、並且二度檢查授權顯示的文字應該要沒問題的，如果被檢查有問題，又會將「_g_valid_license」清空（終止授權）


```

-----
                Z22validate_license_checkv:      // validate_license_check()
00000000100006c1b    jmp      0x100006d6e          ; XREF=

```

看到這邊我想到一個很有趣的想法，既然它會做清空，那如果我從 validate_license_checkv 函數頭跳到清空的位置上

```

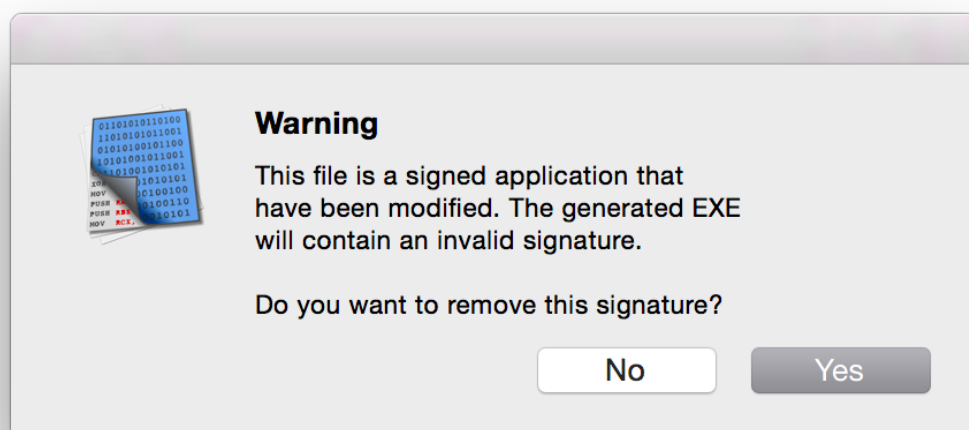
00000000100006d6e    lea      rax, qword [ds:_g_valid_license]
00000000100006d75    mov      byte [ds:rax], 0x1
00000000100006d79    ret

```

再將設為0（清空）改為設為1（true）做回傳，那麼不就繞過驗證了嗎XD（不過就不會特別處理顯示文字授權部分了）至於你會問我幹嘛這麼麻煩，跳到清除點、在修改清除點為true；不直接在函數頭修改「_g_valid_license」為1就好？我有試過，但是Hopper對這個Import到全域變數做修改會直接Crash，所以只能這樣做二段跳XD，知道怎麼處理的麻煩教我一下QQ。

最後透過Hopper幫我們打包Patch回去（File > Produce New Executable」覆蓋回去原本exec，這樣就算本地端破解成功囉！為何我會說本地端？**如果你這樣Patch完的結果會是無法分享的狀態、並且丟進「應用程式」資料夾是無法顯示在Launchpad內的！**該怎麼解決呢？主要這個驗證是透過簽章來驗證的，所以我們底下要開始探討如何幫OSX軟件重簽章（或稱二次打包）。

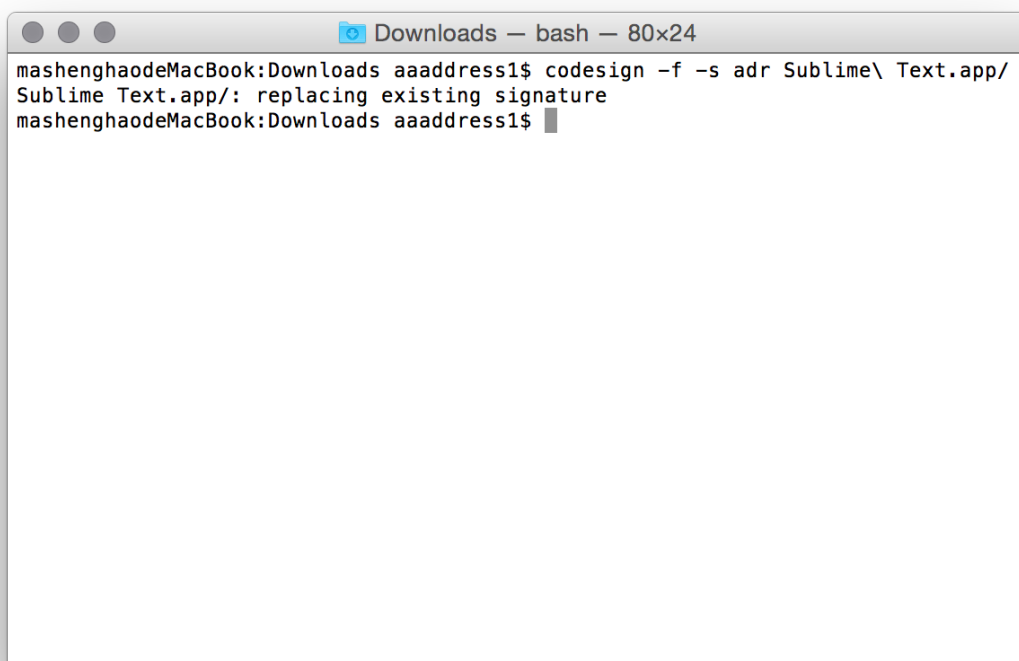
重簽章



首先特別注意的是，當初你在用Hopper覆蓋的時候這一步驟記得要按「No」不要

讓Hopper替你做虛假拔簽章（雖然文字上寫為拔簽章，但是根據某OSX大牛表示沒辦法這樣拔，Hopper實作方式只是詐欺系統沒有簽章而已）

接著覆蓋完原本的exec做完本地端破解後就可以來替軟件做簽章了，首先你得先用鑰匙圈製造一個自根簽憑證（我這邊自根簽憑證名稱為adr）



```
Downloads — bash — 80x24
mashenghaodeMacBook:Downloads aaaddress1$ codesign -f -s adr Sublime\ Text.app/
Sublime Text.app/: replacing existing signature
mashenghaodeMacBook:Downloads aaaddress1$
```

接著下：

`codesign -f -s` 自根簽名 二次簽章的App完整檔名

附註：`codesign`為XCode安裝之後的命令，如果沒有請記得安裝XCode

最後，跑起來就是可以分享給朋友的已簽章並且可以分享的破解版Sublime3啦！

開發者如何防禦？

雖然說像這樣靜態分析很好做繞過驗證並授權，但是一般開發者若是依賴開發這種小型軟體維生，那麼並不樂見這種破解行為，這邊特別提Sublime3的問題，它只在開頭處對授權驗證做檢測並把檢測結果放入一個布林值內，之後便只檢測該布林值，這並不是一個安全的做法；安全、有效的做法例如：授權的Key會對程式本身加解密、Demo版程式跟付費版程式分開兩支撰寫，不要放一起、或者實際功能放於後端處理，…等都是可行的做法！這邊做一個簡單的結論 😊

Published in [Crack](#) [OSX](#)

Previous Post

[AIS3 Final-Exam Binary1](#)

No Newer Posts

[Return to Blog](#)

Be First to Comment



[OOEE](#)

Your comment is awaiting moderation.

好物收藏先!!

2015-09-15 | [Reply](#)

發表迴響

你的電子郵件位址並不會被公開。必要欄位標記為 *

Name*

OOEE

Email*

aa@gmail.com

Website

http://無

Comment

張貼迴響

[Author WordPress Theme](#) by Compete Themes