# MEMBINA 'PATCHER' UNTUK JAVASCRIPT SCRAMBLER V1.11

hotfloppy
hotfloppy . 6866 @ gmail . com
http://www.hotfloppy.tk/

Tahap Kesukaran : Permulaan

Assalamualaikum.. !!
Ni kali pertama aku buat tutorial berkenaan Reverse Engineering.. Actually, sebelum ni aku pernah buat 1 tutorial untuk pembinaan keygen, tapi softcopy hilang.. Yang ini je yang tinggal.. So, enjoy !

Kali ni, aku nak ajar camana nak patch program lak.. Aku tau dah belambak-lambak tutorial berkenaan patching ni kat Internet, tapi aku tak jumpa lagi yang dalam bahasa ibunda ku yang tercinta ni, Bahasa Malaysia. hehe..
So, takyah buang masa, mari periksa ianya luar (lets check it out).. ;)

## ALAT TUKANG

i. JavaScript Scrambler v1.11
   - Sasaran kita untuk kali ni..!!
   - Kata-kata pembinanya:
     "JavaScript Scrambler is a utility for all JavaScript programmers who are fed up with the fact, that their source code can be stolen and simply modified. Any script source code will be scrambled until it is almost impossible to comprehend for others."

ii. Win32DASM (Win32 Disassembler)
   - Untuk kita menyenarai-mati-kan (deadlisting) program kita.

iii. Pen/Pensil dan Kertas @ Windows Notepad
   - Untuk mencatit maklumat. Terpulang la nak guna ape.. nak catit kat dinding pon takpe. Seniri jawab 'k ;)

iv. Hex Editor (aku gunakan HIEW v6.1)
   - Untuk kita test sama ada offset kita betul atau tidak.

v. Total Commander
   - Ini adalah 'optional', tiada paksaan. Tapi dengan Total Commander, urusan yang berkaitan dengan folder, files, rename (filename atau extension), copy atau moving menjadi lebih mudah.

vi. Bloodshed Dev-Pascal
   - Untuk menulis & compile kod sumber bagi 'patcher' kita.

## PE-NYAH-TUNTUTAN (DIS-CLAIM-ER)

Artikel ini adalah Wakaf kepada scene Pembalikan Kejuruteraan (Reverse Engineering). Oleh itu, artikel ini adalah DIBENARKAN; di salin semula ke dalam apa jua bentuk, disebarkan di mana-mana saja (kecuali tandas), di guna atau di baca oleh sesiapa saja, atau apa-apa saja, selagi ianya dalam bentuk ASAL, tidak diubahsuai walaupun sedikit dan adalah semata-mata untuk TUJUAN PEMBELAJARAN sahaja.

Seandainya ada pihak-pihak tak bertanggung jawab yang melakukan salah guna ke atas artikel ini untuk tujuan jenayah seperti mengebom Parlimen, menghuru-harakan Parti Kerajaan/Pembangkang, merompak bank, memancitkan tayar jiran, menskodeng anak dara Tok Ketua mandi, merosakkan majlis perkahwinan atau jenayah-jenayah lain yang termaktub di dalam perundangan Malaysia, PIHAK PENULIS TIDAK AKAN BERTANGGUNGJAWAB SAMA SEKALI.

Ditekankan disini, bahawasanya, artikel ini ketika ditulis, adalah semata-mata menyasarkan skopnya terhadapat sektor pembelajaran sahaja. Dan diingatkan kepada pengguna/pembaca artikel ini, sila melakukan 'backup' ke atas data-data anda kerana pihak penulis TIDAK AKAN BERTANGGUNGJAWAB seandainya, ketika mengikuti langkah-langkah atau praktikal di dalam artikel ini, komputer anda meletup atau mencair biarpun perkara itu tidak logik untuk berlaku.

Akhir sekali, penulis ingin menekankan disini;
PEMBANGUN PERISIAN INI MEMBUTUHKAN IMBUHAN DARI PENAT LELAHNYA. SILA BUAT PEMBAYARAN PENDAFTARAN SEKIRANYA ANDA INGIN TERUS MENGGUNAKAN PERISIAN INI ATAU UNINSTALL SAHAJA SETELAH TAMAT TEMPOH PERCUBAAN.

p/s: Aku dok kat Malaysia, manakala tempat nak register program ni lak kat German.. jauh tu bai even bleh register guna post or credit card. Masalahnya lagi, aku takde kredit kad. Post? Hrmm.. boleh tu mmg la boleh, tapi program ni berharga US$15, kalo convert ke RM.. US$15 * RM3.80 = RM57.00 .. !! Tu tak termasuk lagi dengan wang pos dan lain-lain caj.. ahh.. leceh !! Ala, lagipun aku bukannye nak guna pon program ni, just nak blaja je.

"If seeking a knowledge is a crime, then only God may save me"

# ARTIKEL

***Peringatan 1***

*Korang dah ade semua alat tukangan seperti yang tersenarai kat atas? Bagus. Sebelum kita bermula, aku nak ingatkan bahawa artikel ini ditulis adalah dengan skop pembaca nya adalah para newbies. Jadi, artikel ini mungkin agak panjang sedikit kerana aku akan menerangkan sesuatu perkara dengan se-ter-perinci yang mampu supaya para newbies seperti aku boleh paham dan tidak terkial-kial sepertimana aku belajar dulu. Dan sukacitanya juga aku mengingatkan, artikel ini adalah di dalam kategori 'Reverse Engineering', so, aku takkan sentuh macamana nak unzip file dan nak renama file extension dan sebagainya. Untuk itu, aku menganggap korang suma tahu macamana untuk menggunakan OS korang seniri.*

***Peringatan 2***

*Sebelum kita bermula, suka aku nak ingatkan supaya korang buat sedikit 'backup':*

*i. Copy jsscram.exe kepada jsscram.backup <- semata-mata untuk backup.*

*ii. ii. Copy jsscram.exe kepada jsscram.w32 <- untuk kita guna dengan Win32DASM sebab kadang kala, program*

*takleh run selagi kita tak tutup Win32DASM yang tengah disassemble program tu. Leceh kan? Ini la gunanya.*

***Peringatan 3 (tiada paksaan, terpulang)***

*Pada kertas nota kita, tuliskan 6 pembolehubah (variables):*

*#AddressMula*
*#AddressReg*
*#OffsetMula*
*#OffsetReg*
*#ByteMula*
*#ByteReg*

*Pembolehubah ni adalah semata-mata sebagai 'guides point' untuk kita menganalisa target kita.*

1. Mula-mula sekali, unzip JavaScript Scrambler (aku akan guna singkatan JS bermula dari sekarang) ke mana-mana folder pilihan korang. Run JS untuk seketika dan perhatikan apa maklumat yang programmer nya hidangkan untuk kita.

   Hrm.. ada ruang untuk kita Register eh? Bagus.. bagus.. Cuba register JS dengan nama & serial code tipu dan click Register. *TREETT* "The serial number you entered, was wrong!". Ya, ya.. takyah gitau pon kita tau, ye tak? hehehe.. Okay, tuliskan ayat-ayat tu kat kertas atau hafal je :P

2. Sekarang, run Win32DASM dan disassemble-kan jsscram.w32 tadi. Macamana? Pilih menu Disassembler > Open File to Disassemble.. cari file kita (jsscram.w32) dan klik Open. Tunggu sekejap sampai Win32DASM siap disassemble program tu. Bergantung pada kelajuan PC, kalo RAM lekeh (oppss.. :P), lambat skit ar. Bleh ar nak gi bancuh air ke, masak megi ke.. hehehe.. Dah siap? Bagoss.. Sekarang ni, pegi ke menu Refs > String Data References (SDR). Tetingkap SDR akan terpampang. Cari ayat-ayat yang kita tulis (atau hafal) tadi.. Jumpa? Double click. Pada tetingkap utama, senarai ASM akan memaparkan address di mana ayat tu di panggil/proses/guna (dan lain2). Tutup tetingkap SDR dan lihat pada tetingkap utama. Senarai instruction/arahan ASM seperti berikut akan terpapar:

```
* Referenced by a (U)nconditional or (C)onditional Jump at Addresses:
|:00433F82(C), :00433FA0(C)
|

* Possible StringData Ref from Code Obj ->"The serial number you entered, "
                                        ->"was wrong!"
                                       |
:0043406D       B8A0414300          mov eax, 004341A0
:00434072       E86DB1FFFF          call 0042F1E4
```

Di sini, kod dan kata-kata 'bro_ko_tak_terer_la' dipanggil dan (akan) di papar. hrm.. tak bestkan? Jadi, jom kita buktikan kat programmer ni yg kita ni terer ;)

3. Sekarang, kita perlu jejak balik (trace back) dari mana kata-kata tak sihat ni dipanggil. Nampak tak dua address kat atas tu? Okay, pergi ke menu Goto > Code Location (Shift+F12) dan masukkan mana2 address. Tak kisah yang mana pun sebab kedua-duanya berdekatan. Klik OK. Kod seperti di bawah akan terpapar. Jom kita kaji cebisan kod tersebut:

```
:00433F7F   83F804          cmp eax, 00000004                   a
:00433F82   0F8EE5000000    jle 0043406D                        b
:00433F88   8D4DF4          lea ecx, dword ptr [ebp-0C]         c
:00433F8B   8B55FC          mov edx, dword ptr [ebp-04]         d
:00433F8E   8BC3            mov eax, ebx                        e
:00433F90   E81BFAFFFF      call 004339B0                       f
:00433F95   8B45F4          mov eax, dword ptr [ebp-0C]         g
:00433F98   8B55F8          mov edx, dword ptr [ebp-08]         h
:00433F9B   E8A0FBFCFF      call 00403B40                       i
:00433FA0   0F85C7000000    jne 0043406D                        j
:00433FA6   C6054859430001  mov byte ptr [00435948], 01         k

* Possible StringData Ref from Code Obj ->"JSScrambler.ini"
                                  |
:00433FAD   B9D4404300      mov ecx, 004340D4
:00433FB2   B201            mov dl, 01
:00433FB4   A1D4F34200      mov eax, dword ptr [0042F3D4]
```

Kalo anda perasan pada kod diatas, kedua-dua address tadi ada dalam kod iaitu (b) dan (j). Tugas (b) adalah memastikan input nama kita adalah tidak kurang dari 4 patah huruf manakala (j) pula adalah untuk memastikan serial yang kita masukkan betul atau tidak. Dan ya, (j) adalah address yang kita perlukan :) Jadi sekarang, tuliskan address (j) pada pembolehubah #AddressReg kita.

```
#AddressReg             00433FA0
```

4. Sebelum aku teruskan artikel ini, kat sini aku nak terangkan serba sedikit maklumat yang perlu ada untuk kita patch sesuatu program. Patch disini bererti, kita mengubah secara kekal kod bagi program tersebut. Untuk mengubah kod tersebut, kita perlu mendapatkan offset bagi kod tersebut. Tidak boleh menggunakan address yang diberikan kerana address ini kebiasaannya tidak tetap. Mungkin ada yang confuse kan? hehehe..

   Okay.. Bayangkan program kita adalah sekumpulan kertas-kertas. Manakala kod-kod kita pula adalah nota-nota di atas kertas-kertas tersebut yang ditulis baris demi baris. Bayangkan pula ada sebuah buku yang boleh kita tampal kertas-kertas tadi supaya senang dibaca. Tapi buku tersebut pastinya bukan buku kosong kan? Jadi, kita tak mungkin dapat menampal kertas-kertas tadi bermula dari muka surat 1. Mungkin muka surat 3 atau 45 atau 98 :)
   Sekarang, cuba bayangkan dan kaitkan;

   - **Buku tadi adalah RAM pada PC.**
   - **Manakala muka surat pula adalah address-address dalam memory.**
   - **Kertas-kertas pula adalah lokasi kod-kod kita.**
   - **Baris-baris pula adalah offset bagi kod.**
   - **Dan nota-nota pula adalah instruction-instruction bagi program kita.**

   Jangan gopoh, amik masa untuk hayati dan bayangkan situasi ini. Ini amat penting supaya korang paham apa yang korang buat. Baca berkali-kali sampai paham, baru teruskan ke langkah seterusnya. Tapi terpulang pada korang la nak jujur atau tak. Ilmu tu untuk diri seniri :)

   **Tip: Untuk lebih paham, amik beberapa keping kertas kajang. Pada setiap baris, tuliskan nombor-nombor mengikut turutan. Kemudian, bukak mana-mana senarai 'deadlisting' yang anda buat dan salin balik kod tersebut pada setiap kertas tadi; baris demi baris. Sekarang, anda sepatutnya lebih mudah memahami konsepnya, kan? :)**

5. Okay, harap-harapnya korang dah paham konsep yang aku ceritakan kat atas. Sekarang, lihat pada tetingkap utama Win32DASM yang kita double click tadi. Line sekarang sepatutnya pada kod 'jne 0xaddress'. Lihat pada status bar dan cari perkataan @Offset. Jumpa? Bagos.. Lihat di kanan @Offset itu, ada sebaris nombor kan? Itulah offset yang kita cari. Bagi kes ini, offset kita adalah 333A0h. h di hujung nombor bermaksud hex dan tidak termasuk dalam nombor-nombor tersebut. Catitkan offset tersebut dan tandakan bahawa itu adalah offset untuk pendaftaran. Sekarang, lihat balik pada kod yang kita 'highlight' tadi. Nampak tak opcodes-nya? Ini adalah opcodes untuk instruction tersebut 0F85C7000000. Kita perlu mengubah 'jne' kepada 'je' supaya program akan menerima apa jua serial yang kita masukkan. Opcode bagi 'jne' adalah 0F 85 manakala 'je' adalah 0F 84. Kita akan patch program menggunakan opcode ini. Jadi, opcode yang perlu diubah, cuma 85, kepada 84. Ianya 1 byte lebih dari offset kita jadi kita perlu ubah offset kita kepada 333A1h. Catitkan.

```
#OffsetReg          333A1
#ByteReg            84
```

Fuh.. Setel untuk Register. Sekarang kita nak cari maklumat patching untuk startup check pula sebab program ni akan melakukan check-up setiap kali program bermula untuk memastikan serial yang kita masukkan adalah valid.
Tunggu apa lagi, jom!!

6. JS akan membuat checking setiap kali ia run untuk memastikan sama ada ianya telah di-register atau belum, dan jika sudah, adakah name dan serial tersebut betul? Jadi, kalo ikot logik, JS perlu merujuk kepada sesuatu untuk melakukan checking. Persoalannya, apa yg dirujuk? Hrm.. sekarang ni, tengok balik pada address (j); ya, address di mana routine untuk memastikan serial yang diberikan betol atau tak.

```
:00433F9B    E8A0FBFCFF          call 00403B40                          i
:00433FA0    0F85C7000000        jne 0043406D                           j
:00433FA6    C6054859430001      mov byte ptr [00435948], 01            k

* Possible StringData Ref from Code Obj ->"JSScrambler.ini"
                                  |
:00433FAD    B9D4404300          mov ecx, 004340D4
:00433FB2    B201                mov dl, 01
:00433FB4    A1D4F34200          mov eax, dword ptr [0042F3D4]
```

Nampak tak tu? JSScrambler.ini ..!! Ya, ya, ya.. ini lah tempat yang akan dirujuk oleh JS. Proses nya camni:

1. **User masukkan Name dan Serial.**
2. **JS check samada Name kurang dari 4 atau tidak:**
   **Jika lebih, check Serial (langkah 3).**
   **Jika kurang, 'registration' gagal..!!**
3. **JS akan check Serial:**
   **Jika betul, rekodkan ke dalam JSScrambler.ini (langkah 4).**
   **Jika salah, 'registration' gagal..!!**
4. **Cipta file bernama JSScrambler.ini dan rekodkan Name dan Serial.**
5. **Papar mesej 'registration' berjaya atau gagal.**

Paham? Bagos. Tak paham? Baca balik sampai paham ;)
Sekarang, kita dah tahu bahawa JS akan rujuk pada JSScrambler.ini setiap kali run.
Jadi, buka tetingkap SDR dan cari 'string' JSScrambler.ini. Double click berkali-kali.
Sebab apa? Sebab kita nak pastikan sama ada file ni dipanggil berapa kali dalam JS. Ada kemungkinan kan untuk JS check 4-5 kali Name dan Serial tersebut.. ;) Masa
double click tu, catitkan address yang di-highlight. Ulang sampai address yang sama saja berulang. Baik, berapa kali? 2 kali kan.. bagos.. Kalo korang cepat 'pickup', korang dah boleh agak sebab ape 2 kali. hehehe.. Sebabnya, JS panggil sekali masa 'register' dan sekali lagi masa 'startup'. Paham? Sekarang ni, dengan merujuk pada #AddressReg,

mana-mana antara 2 address yang baru kita catit tadi yang hampir dengan #AddressReg, tidak kita perlukan sebab itu adalah address ketika proses register. Double click sampai kita dapat address yang 1 lagi. Tutup tetingkap SDR. Kod ASM yang terpapar adalah seperti berikut:

```
:00433B4E    64FF30            push dword ptr fs:[eax]
:00433B51    648920            mov dword ptr fs:[eax], esp

* Possible StringData Ref from Code Obj ->"JSScrambler.ini"
                        |
:00433B54    B9183E4300        mov ecx, 00433E18
:00433B59    B201              mov dl, 01
```

Ini adalah kod di mana JSScrambler.ini di panggil dan Name & Serial yang terkandung di dalamnya di rujuk oleh JS. Kalo kita 'tracing' kod-kod ni, kita akan jumpa beberapa baris kod yang kelihatan 'familiar':

```
:00433BDA    8B45F8            mov eax, dword ptr [ebp-08]
:00433BDD    E85EFFFCFF        call 00403B40
:00433BE2    7566              jne 00433C4A
:00433BE4    C6054859430001    mov byte ptr [00435948], 01
```

Kod ni kan lebih kurang sama dengan kod 'registration' (i) sampai (l).. Kesimpulannya, kita cuma perlu buat apa yang kita buat tadi.. ya, ubah 'jne' kepada 'je' :)

7. Highlight pada kod 'jne' tersebut dan lihat pada offset-nya. Catitkan address dan offset.

```
#AddressMula        00433BE2
#OffsetMula         32FE2
```

Lihat balik pada pembolehubah kita.

```
#AddressMula        00433BE2
#AddressReg         00433FA0
#OffsetMula         32FE2
#OffsetReg          333A1
#ByteMula
#ByteReg            84
```

#ByteMula masih kosong, 'means' masih tak siap lagi la (tapi dah hampir-hampir :). Sekarang ni, kita perlu ubah 'jne' kepada 'je'. Opcode bagi 'jne' adalah 75 manakala 'je' pula adalah 74. Tunggu ape lagi, catit la ;)

```
#ByteMula           74
```

Sekarang segala maklumat kita dah dapat. Yeaa !!~ Teruskan dengan membina patcher.. ops.. tapi sebelum tu, test dulu semua benda yang kita ada ni.

8. Run Hex Editor pilihan korang, aku guna HIEW v6.1. Loadkan jsscram.exe dan pergi ke #OffsetMula dan ubah hex 85 itu kepada 84. Begitu juga untuk #OffsetReg, ubah hex 75 kepada 74. Save. Run JS, register macam biasa dan *POOF*! 'registration' berjaya!. Sekarang, tutup JS pastu bukak balik. JS telah di-register.. !! Tahniah.

Sekarang, mari kita bina patcher kita pula. Tapi sebelum tu, delete dulu jsscramber.ini yang telah JS cipta. Ianya tersembunyi di C:\Windows\JSScrambler.ini. Dah delete? Bagos. Jom teruskan dengan patcher kita.

9. Untuk membina patcher ini, aku akan gunakan Bloodshed Dev-Pascal v1.9.2. Tapi kod sumber (source code) Pascal ini boleh juga di-compile dengan mana-mana compiler Pascal, insyaAllah.

9.1 Seandainya korang ingin menulis 'patcher' ini di dalam bahasa aturcara (programming language) yang lain, di sini aku sertakan kod pseudo (pseudo code) untuk 'patcher' ini.

Kod pseudo untuk 'patcher':

```
//========== Mula kod pseudo =================

MULA

Buka program untuk di patch:
Jika berjaya, pergi ke @Patch
Jika tidak, pergi ke @Keluar

@Patch:
Pergi ke #OffsetMula dan ubah nilai asal (75h) kepada #ByteMula (74h),
Pergi ke #OffsetReg  dan ubah nilai asal (85h) kepada #ByteReg  (84h),
Papar ke skrin mesej memberitahu 'patching' berjaya,
Terus ke TAMAT.

@Keluar
Papar ke skrin mesej memberitahu program tidak dijumpai / tidak dapat dibuka,
Terus ke TAMAT.

TAMAT

//========== Tamat kod pseudo =================
```

9.2 Aku harap korang dapat memahami konsep untuk menulis aturcara bagi 'patcher' ini. Sekarang, mari kita bina 'patcher' sebenar kita. Aku akan menganggap yang korang dah tahu bagaimana untuk menulis aturcara Pascal.

Kod sumber untuk patcher ini adalah seperti berikut:

```
//========== Mula aturcara JS_Patcher =================

program JS_Patcher;

uses crt;

const
patch: array [1..2] of record                  // ====================================
      offset : longint;                        // constant patch ini gunanya untuk kita
      data   : byte;                           // memberitahu patcher di offset mana
end = ((offset : $32FE2; byte : $74),          // dan byte apa yang perlu di 'patch'
      (offset : $333A1; byte : $84));          // ====================================

var
fail  : file;
pos   : byte;
tulis : char;

begin
   assign (fail, 'jsscram.exe');                                        // ========================
   {+I} reset (fail); {-I}                                              //
   if IOResult <> 0 then                                                //
   begin                                                                // proses membuka program
      writeln ('Fail JavaScript Scrambler (jsscram.exe) tidak dijumpai.'); //
      halt (1);                                                         //
   end;                                                                 // ========================

   for pos := 1 to 2 do                                                 // ========================
   begin                                                                // proses 'patching' pada
      seek (fail, patch[pos]);                                          // kedua-dua offset;
      tulis := char (fail, patch[pos].data);                            // #OffsetMula dan juga
      blockwrite (fail, tulis, 1);                                      // #OffsetReg
   end;                                                                 // ========================

   writeln ('Proses "patching" berjaya..!!');                           // mesej patching berjaya :)
end.

//========== Tamat aturcara JS_Patcher =================
```

Siap !!
Dalam kod sumber di atas, aku letakkan 'comment' supaya memudahkan korang membaca dan memahami kod sumber tersebut. Korang boleh 'copy' dan 'paste' saja kod sumber ni dan 'compile'. Kod sumber untuk patcher ini adalah seringkas mungkin; sekadar melakukan proses 'patching'. Korang boleh kembangkan lagi idea; mungkin dengan meletakkan beberapa lagi 'error-checking' selain dari yang sedia ada. Gunakan otak dan kreativiti korang :)

10. Panjangnya artikel ni.. hahaha.. tapi peduli apa kan? Semua ni ilmu. Sebenarnya, artikel ni tidak perlu sepanjang ini, tapi aku saja buat bagi jadi panjang supaya aku mempunyai lebih ruang untuk menerangkan secara terperinci yang mungkin segala teknik yang kita gunakan. Apa guna artikel ni ditulis jika bukan untuk memberi ilmu kan? Dan apa guna ilmu tersebut kalo korang tak paham :) Sekiranya, setelah membaca dan mengikuti setiap baris langkah yang tergarap dalam artikel ini, korang masih tak berjaya atau ada bahagian yang tak paham, jangan susah hati. Korang boleh hubungi aku melalui email. Tapi pastikan korang tuliskan 'JS Patch' sebagai subjek/tajuk email korang.

Sekian :)

Penghargaan:

**Gamehacking Scene**
Razali Rambli a.k.a Bie - Miss u a lot bro !! Dia ni guru aku yang pertama dalam Gamehacking.
Razak a.k.a Ajaxx - Also u bro..!! Teman seperjuangan.. :)
Extalia & Devious - Forum yang banyak memberi ilmu Gamehacking
& All Gamehackers around the World..

**Reverse Engineering Scene**
tKC (Founder of PC / CiA) - Aku banyak belajar dari tutorial dia ni..
& All Crackers around the World..

Dan yang terakhir tapi tetap pertama di hati..
ANDA !! :)