

## Kulgram Reversing.ID : DLL Injection

Yang perlu disiapkan untuk kulgram hari ini adalah:

- Windows
- Compiler GCC, disarankan TDM GCC
- cheatsheet PE file (<https://github.com/corkami/pics/blob/master/binary/pe101/pe101.pdf>)
- PE Explorer seperti PPEE (<https://www.mzrst.com/>)

Process Explorer dari SysInternals (<https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer>)

kita akan mengeksplorasi tentang DLL, bagaimana format di DLL, sebelum akhirnya kita bisa melakukan injeksi

Dasar DLL dapat dilihat pada repository Reversing.ID di <https://github.com/ReversingID/Materi-Kulgram/tree/master/Others/IDCPLC/20170902%20-%20Everything%20About%20DLL>  
catatan kulgram ini juga akan di-upload ke repository beserta kodenya

Sedikit pengingat bahwa DLL adalah sebuah cara untuk mencapai modularitas, sebuah program yang besar dapat dipecah menjadi beberapa bagian kecil sehingga jika ada perubahan kita dapat mengubah bagian itu saja tanpa harus mengubah keseluruhan program

ada 2 cara penggunaan DLL, yaitu resolve secara statis dan resolve secara dinamis

statis artinya semua fungsi yang dibutuhkan program (dan itu ada di DLL), akan dituliskan di sebuah daftar namanya IAT (Import Address Table).

Jadi, begitu program jalan, windows loader akan menelusuri daftar ini dan meload DLL yang diperlukan serta mengisi alamat menuju fungsi yang diperlukan tersebut.

sementara dinamis berarti kita secara eksplisit memanggil sebuah fungsi / Win32 API untuk meload DLL ke address space program

Nah DLL injection adalah salah satu teknik untuk menambahkan DLL yang kita inginkan di luar kebutuhan program yang sebenarnya.

oke, lalu apa sebenarnya kegunaan kita melakukan DLL injection?

Sebenarnya lebih banyak digunakan untuk melakukan modifikasi binary. Bisa dari sekedar mencari informasi sampai mengubah alur.

kalo kegunaan baiknya sih, buat menambah fitur. Misal ada suatu fitur yang obsolete karena adanya perubahan proses bisnis / prosedur. Kita punya 2 cara untuk melakukan perubahan:

1. ubah kodenya (jika punya kode)
2. ubah binary-nya (patching)

tapi mungkin ada pertimbangan tertentu sehingga kita gak bisa begitu saja melakukan patching. Jadi solusinya, kita gak patch, tapi kita ubah fungsinya pda saat runtime. Ato kita ingin ketika ada pemanggilan fungsi X, menuju ke definisi fungsi yang baru sementara fungsi lainnya menuju ke DLL semula.

Bisa juga karena kita cuman pengen melakukan analisis ketika program itu berjalan. Mengetahui "fungsi ini sebenarnya ngapain?". Atau kita tertarik dengan argumen yang diberikan ke sebuah fungsi? Fungsi kriptografi misalnya, dia butuh kunci dan plaintext yang ingin dienkripsi dan kita ingin dapatkan keduanya saat program itu berjalan.

apapun yang diinginkan, DLL injection hanyalah sebuah teknik untuk memasukkan DLL ke sebuah proses yang sedang berjalan.

Begitu DLL di-load, dia akan menjalankan kode di DLL dengan hak akses yang dimiliki oleh program tersebut. Jadi, apapun kode yang dijalankan itu punya kewenangan yang sama dengan inang.

» Ragam Teknik DLL Injection «

Ada banyak cara untuk menyuntikkan DLL.

1. AppInit\_DLLs key (registry)
2. SetWindowsHookEx() API
3. Image File Execution Options (IFEO) key (registry)
4. Creating Remote Thread

tapi kita batasi pembahasan kita kepada nomer 4, creating remote thread

tapi kita bahas singkat aja ketiga yang lain.

[1] ada suatu registry key di Windows namanya

HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows  
NT\CurrentVersion\Windows\AppInit\_DLL

Ketika aplikasi dijalankan OS loader akan mengimport semua DLL yang dibutuhkan. Sebelumnya OS loader akan melakukan query ke registry ini. Apapun DLL yang ada didaftar sana akan di-load untuk setiap proses yang akan berjalan.

tapi sayangnya untuk Windows 7 ke atas katanya DLL perlu di-signed supaya bisa berjalan

[2] SetWindowsHookEx() adalah sebuah API untuk melakukan hook ke berbagai hook chains (CBT, Journal, Window Message, keyboard message, mouse message, dsb). Kita bisa menggunakan ini untuk menginjeksikan DLL custom yang digunakan untuk hook ke event yang relevant. Tapi yang bisa kita lakukan hanya terbatas ke fitur-fitur tertentu saja yang sudah didefinisikan, misal: membaca pesan / data dari keyboard untuk ke process.

[3] Sebuah registry juga. Gunanya adalah untuk memerintahkan OS menjalankan debugger kemudian melakukan berbagai hal yang kita inginkan

[4] true DLL injection. Kita menggunakan API untuk memaksa process korban melakukan dynamic loading sebuah DLL. Jadi kita memaksa process buat memanggil DLL, pada thread baru.

Kalo kita sederhanakan, apa yang kita lakukan adalah seperti ini:

1. attach ke process korban
2. mengalokasikan ruang memory di korban
3. menuliskan sesuatu ke korban
4. memaksa korban membuat thread baru, thread ini yang akan memanggil fungsi LoadLibrary untuk memanggil DLL.

nah ada 2 variasi skenario yang ada.

1. kita hanya menyuruh korban untuk load DLL. Kita hanya perlu memberikan path yang valid menuju ke DLL tersebut sehingga korban tinggal memanggil saja
2. kita menuliskan keseluruhan DLL ke tubuh korban, korban kemudian kita suruh memanggil fungsi DllMain yang ada di korban tadi.

yang mana yang lebih enak?  
sama aja sih

» Eksplorasi DLL «

[File: kerangka.c]

nyalakan GCC kalian dan compile sesuai instruksi yang ada di file itu

kita break bentar 5 menit buat coba-coba compile dan baca kembali apa yang ada di materi DLL sebelumnya.

kalo ada pertanyaan coba diberikan sekarang

b4mbs y, [19.10.17 20:51]  
apa teknologi dll dari jaman win 3.x sampai sekarang gak berubah?

Satria Ady Pradana, [19.10.17 20:52]  
secara garis besar gak berubah, terutama yang [4]

Satria Ady Pradana, [19.10.17 20:53]  
karena kita pake API win32 yang memang digunakan untuk debugging. Tapi mungkin bagi AV mungkin akan dianggap malicious

b4mbs y, [19.10.17 20:55]  
potensi wsl (ubuntu for windows) untuk buat 'stealth mode' terbuka lebar dong?

Satria Ady Pradana, [19.10.17 20:55]  
[ Sticker ]

Satria Ady Pradana, [19.10.17 20:56]  
kalo gak ada lagi, kita lanjut

b4mbs y, [19.10.17 20:57]  
lanjuut

kalo berhasil compile (harusnya berhasil), kita akan dapatkan sebuah file kerangka.dll

coba buka source code, ada fungsi namanya DllMain(), fungsi ini semacam fungsi main() kalo di aplikasi biasa. Dia sebagai entryptoint sebuah DLL.

[File: runme.c]  
[File: victim.c]

sekarang kita mulai yang beneran nih. Tadi cuman penyegaran lagi untuk compile file jadi DLL.

kita punya 2 file, runme.c dan victim.c  
keduanya adalah korban kita

runme adalah aplikasi sederhana yang butuh DLL victim.dll  
Dia punya 4 menu, memanggil fungsi f(), list modules, memanggil fungsi dari victim.dll, dan menunjukkan nilai dari global variable

kita compile dulu. Pertama compile si victim.dll kemudian compile si runme

```
gcc -shared -o victim.dll -Wl,--out-implib,libvictim.a victim.c
```

```
gcc runme.c -L. -lvictim -o runme
```

```
C:\WINDOWS\system32\cmd.exe - runme

E:\Learn\Reversing\DLL\Injection
$> gcc -shared -o victim.dll -Wl,--out-implib,libvictim.a victim.c

E:\Learn\Reversing\DLL\Injection
$> gcc runme.c -L. -lvictim -o runme

E:\Learn\Reversing\DLL\Injection
$> runme
My PID is 8404

Menu ----
1. Call f() 10 times
2. List modules
3. Call add,sub,mul (imported from victim.dll)
4. Show value of global
0. Exit
Your choice? (1/2/3/4): _
```

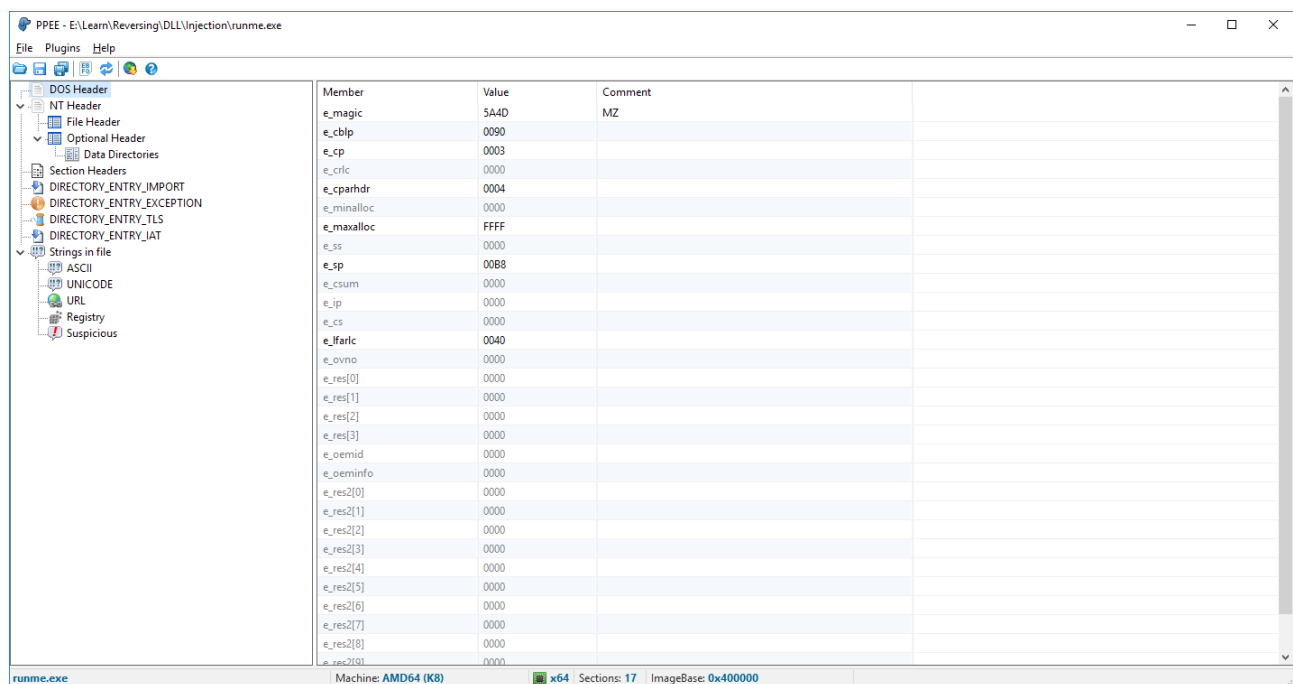
kalo berhasil compile dan berhasil dijalankan, kita akan mendapatkan hasil seperti itu.

akan terlihat PID dari process yang sedang berjalan itu

dan 4 fungsi yang akan kita coba pada kulgram ini (kalo waktunya cukup)

sekarang coba buka PPEE

drag runme.exe ke dalam PPEE sehingga muncul seperti ini



seperti windows explorer. Di bagian kiri akan terlihat ada tulisan-tulisan seperti DOS Header, NT Header, dsb. Di bagian kanan adalah isi dari setiap item yang sedang dipilih

ini apa?

file exe maupun dll yang kita tahu itu bukan sekedar kode dan data aja. Ada formatnya. Kita menyebut mereka sebagai file PE (portable Execution). Format PE memberikan informasi-informasi tentang file. Itulah yang digambarkan di panel bagian kiri.

cukup banyak informasinya, tapi bagian yang perlu kita fokuskan adalah DIRECTORY\_ENTRY\_IMPORT dan DIRECTORY\_ENTRY\_EXPORT untuk file exe dan DLL, di kulgram ini

penjelasan lengkap bagaimana strukturnya bisa diliat pada file <https://github.com/corkami/pics/blob/master/binary/pe101/pe101.pdf>

jadi di runme ini ada entry DIRECTORY\_ENTRY\_IMPORT tapi tidak ada DIRECTORY\_ENTRY\_EXPORT

ini artinya, file exe hanya membutuhkan DLL dari tempat lain.

klik DIRECTORY\_ENTRY\_IMPORT maka kita akan lihat 3 DLL yang dibutuhkan. Klik victim.dll maka kita akan lihat ada 3 fungsi yang dibutuhkan runme dari victim

PEE - E:\Learn\Reversing\DLL\Injection\runme.exe

File Plugins Help

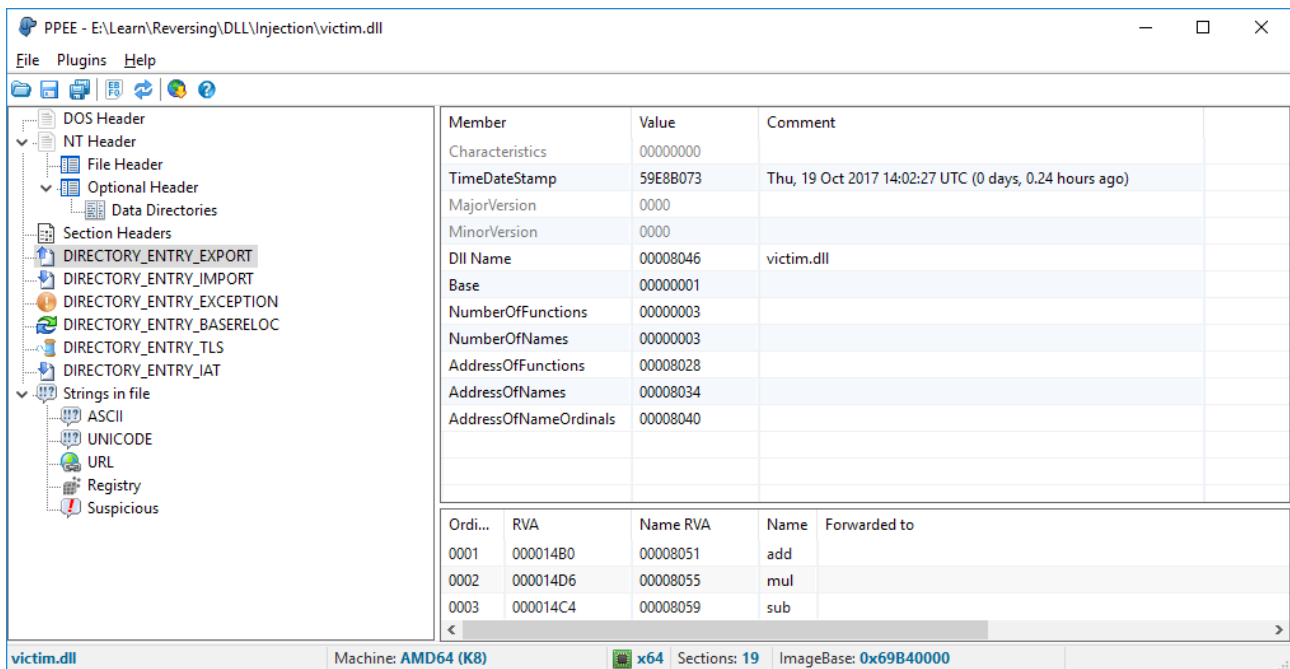
Tree View:

- DOS Header
- NT Header
- File Header
- Optional Header
- Data Directories
- Section Headers
  - DIRECTORY\_ENTRY\_IMPORT** (1)
  - DIRECTORY\_ENTRY\_EXCEPTION
  - DIRECTORY\_ENTRY\_TLS
  - DIRECTORY\_ENTRY\_IAT
- Strings in file
  - ASCII
  - UNICODE
  - URL
  - Registry
  - Suspicious

Name RVA	Name	OriginalFirstThunk	TimeDate Stamp	ForwarderChain	FirstThunk
000097E4	victim.dll	00009050	00000000	00000000	00009248
00009854	KERNEL32.dll	00009070	00000000	00000000	00009268
000098E4	msvcrt.dll	00009140	00000000	00000000	00009338

OFT	FT	Hint	Name	Ordinal
0000000000009440	0000000000009440	0000	add	
0000000000009448	0000000000009448	0001	mul	
0000000000009450	0000000000009450	0002	sub	

runme.exe Machine: AMD64 (K8) x64 Sections: 17 ImageBase: 0x400000



ada bagian DIRECTORY\_ENTRY\_EXPORT dan DIRECTORY\_ENTRY\_IMPORT. Apa yang ada di DIRECTORY\_ENTRY\_EXPORT adalah fungsi-fungsi yang bisa digunakan oleh runme, yaitu add, sub, dan mul

jika dilihat, ada alamat yang disebut dengan RVA, misal fungsi add punya RVA 00014B0 ini adalah alamat relatif dari fungsi. Nantinya ketika runme jalan, kita bisa memanggil fungsi add di alamat ini + alamat dari DLL diletakkan

jadi misal DLL diletakkan di base address 0x69B40000, maka alamat add nanti ada di 0x69B414B0 dari runme

sekarang coba jalankan runme dan masukkan menu 3

kita memanggil fungsi add, sub, dan mul

tujuan akhir kita adalah mengubah fungsi ini, tanpa kita melakukan patching

sampai sini ada pertanyaan mengenai DLL?

gak ada? Oke

dengan PPEE kita melihat daftar DLL serta fungsi yang diimport. Tapi itu pas aplikasi belum dijalankan. Bagaimana melihat DLL yang di-load oleh aplikasi ketika aplikasi berjalan? kita gunakan Process Explorer

Process Explorer - Sysinternals: www.sysinternals.com [BAHAMUTH\xathrya]

File Options View Process Find DLL Users Help

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
Code.exe		127.940 K	147.168 K	14844	Visual Studio Code	Microsoft Corporation
Code.exe		48.368 K	60.088 K	14936	Visual Studio Code	Microsoft Corporation
Microsoft.VSCode.exe		346.848 K	346.700 K	10488	C/C++ Extension for Visual S...	Microsoft Corporation
conhost.exe		5.332 K	8.976 K	10772	Console Window Host	Microsoft Corporation
platformio.exe		520 K	2.720 K	14000		
conhost.exe		5.336 K	9.116 K	14604	Console Window Host	Microsoft Corporation
python.exe		15.028 K	21.820 K	14856		
pioplus....	< 0.01	20.052 K	28.100 K	4024		
Code.exe		3.812 K	11.556 K	8888	Visual Studio Code	Microsoft Corporation
Code.exe		50.620 K	65.884 K	13732	Visual Studio Code	Microsoft Corporation
cmd.exe		2.104 K	3.460 K	5700	Windows Command Processor	Microsoft Corporation
conhost.exe		5.904 K	15.104 K	11984	Console Window Host	Microsoft Corporation
runme.exe		444 K	2.012 K	8404		

Name	Description	Company Name	Path
kemel32.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\Windows\System32\kemel32.dll
KemelBase.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\Windows\System32\KemelBase.dll
locale.nls			C:\Windows\System32\locale.nls
msvcrt.dll	Windows NT CRT DLL	Microsoft Corporation	C:\Windows\System32\msvcrt.dll
ntdll.dll	NT Layer DLL	Microsoft Corporation	C:\Windows\System32\ntdll.dll
runme.exe			E:\Learn\Reversing\DLL\Injection\runme.exe
victim.dll			E:\Learn\Reversing\DLL\Injection\victim.dll

CPU Usage: 18.29% Commit Charge: 33.79% Processes: 192 Physical Usage: 36.05%

pilih runme.exe  
di bawah ada daftar DLL yang di-load  
jika tak tampil, tekan CTRL+D

ketika kita melakukan injeksi DLL nanti, daftar itu akan berubah alias DLL yang kita inject akan muncul di sana.

oke, lanjut ke buat injector-nya

» Membuat Injector «

[File: injector.cpp]

seperti yang dibahas di awal, kita pake pendekatan ke [4]  
kita menggunakan API untuk memaksa aplikasi korban membuat thread baru dan menjalankan fungsi me-load DLL

proses injeksi terjadi di fungsi inject()  
ada 5 langkah di sana.

di kode ini sudah kukasih banyak sekali komentar penjelasan langkah serta prototype fungsinya

Injeksi artinya kita butuh nulis sesuatu ke process lain. Nah untuk menulis ini kita butuh cara untuk mendapatkan akses ke sana. Jadi langkah paling awal [1] adalah kita dapatkan akses.  
Seperti menuliskan sesuatu ke file, kita harus membuka "descriptor" ato handle. Handle ini adalah pengenalan di OS dan bisa kita gunakan untuk melakukan operasi-operasi.

Handle bisa kita dapatkan dari OpenProcess()

OpenProcess() berarti kita meminta akses untuk memanipulasi process, apakah read/write/atau apapun. Kebetulan di sini kita serakah sehingga kita ingin semua akses. Jika berhasil kita akan dapatkan handle yang kita namai remote.

[2] Setelah dapat handle, kita butuh mengalokasikan sesuatu ke process. Kita butuh alokasi untuk menuliskan data. Data apa?

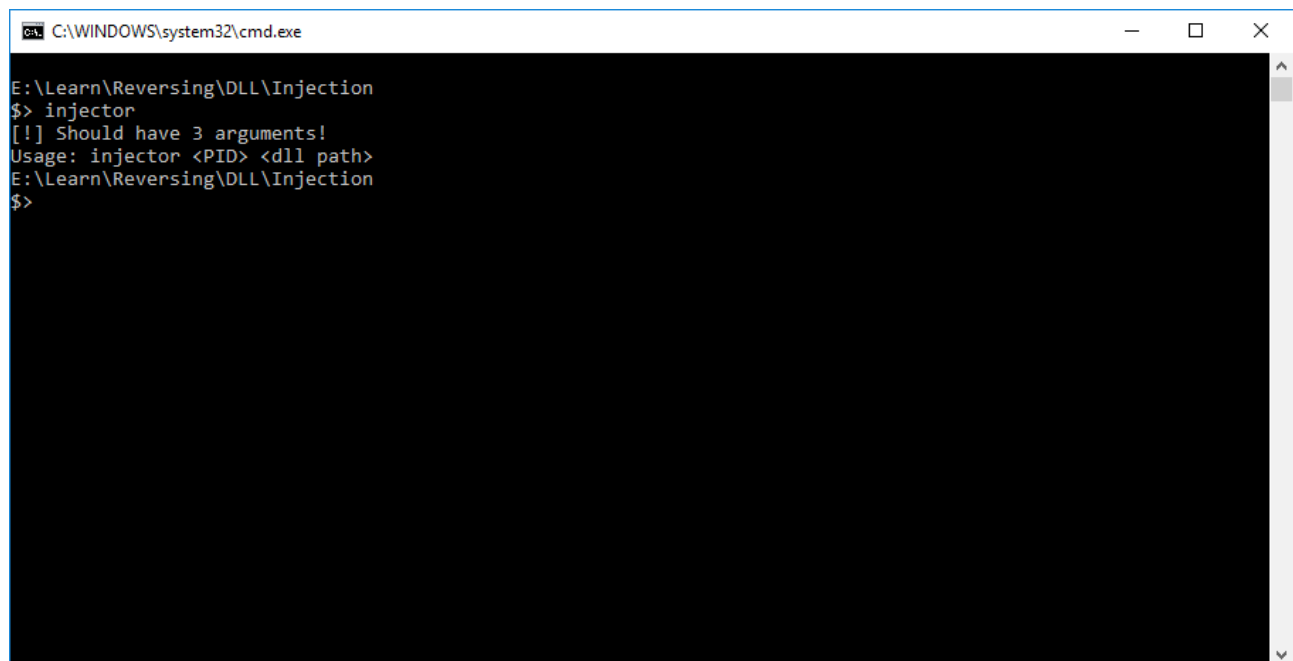
[3] Kita tuliskan full path dari DLL yang akan di-load.

selanjutnya kita butuh membuat thread di process korban. Kita akan menjalankan fungsi load library. Tapi dimana fungsi itu berada? Fungsi itu dapat ditemui di dalam kernel32.dll dengan nama LoadLibraryA. Kita butuh mencari alamatnya. Itulah langkah keempat. Kebetulan DLL kernel32.dll termasuk DLL yang pertama kali dipanggil sehingga tidak berubah. Jadi, kernel32.dll di tempat kita memiliki alamat yang sama dengan yang ada di runme. Karena itulah kita bisa mencari alamatnya.

Setelah itu kita panggil API CreateRemoteThread() [5]. Kita membuat LoadLibrary dijalankan di korban

compile

g++ injector -o injector



```
C:\WINDOWS\system32\cmd.exe
E:\Learn\Reversing\DLL\Injection
$> injector
[!] Should have 3 arguments!
Usage: injector <PID> <dll path>
E:\Learn\Reversing\DLL\Injection
$>
```

oke, sekarang kita sudah punya injector. Selangkah lagi untuk menginjeksikan DLL ke korban

» Injeksi 1: Flow DllMain «

sekarang kita kembali ke DllMain

[File: 01.entrypoint.c]

baca, kemudian compile menjadi DLL

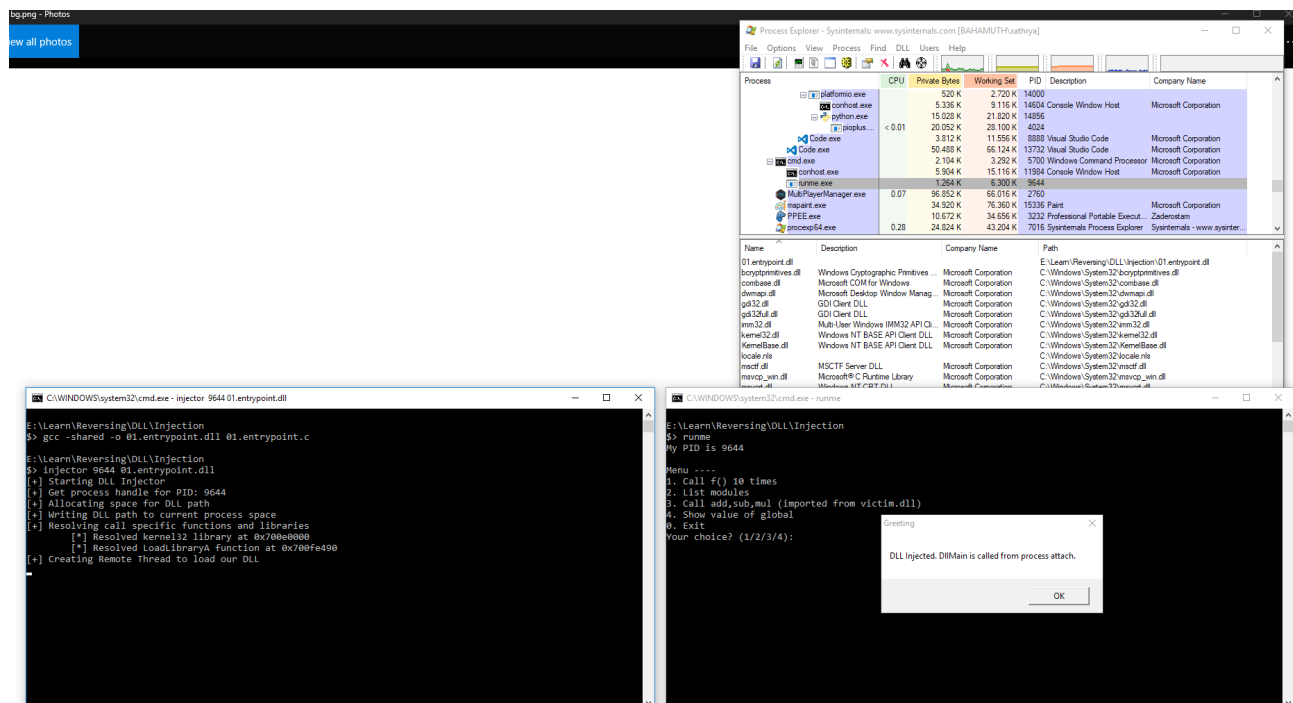


DllMain akan dipanggil untuk 4 kondisi yang berbeda: DLL\_PROCESS\_ATTACH, DLL\_PROCESS\_DETACH, DLL\_THREAD\_ATTACH, DLL\_THREAD\_DETACH

attach artinya DLL dalam proses attach (tempel) / di-load, baik ke process maupun ke thread. Detach artinya DLL dalam proses detach (copot) / unload.

Untuk lebih mengenalnya, kita coba inject

injector <PID> <DLL path>



terlihat sebuah messagebox keluar dan di process explorer kita melihat ada tambahan DLL baru

messagebox ini memiliki pesan "DLL Injected. DllMain is called from process attach." yang artinya kita sedang menjalankan DllMain() untuk event DLL\_PROCESS\_ATTACH, yang artinya lagi kita sedang dalam proses load DLL

jika kita klik OK, akan ada message lagi: "DllMain is called from thread detach."

artinya thread yang digunakan untuk LoadLibrary() tadi alias hasil dari pemanggilan CreateRemoteThread() di injector telah berjalan dengan lancar. Kini DLL sudah di-load dengan sempurna di tempat runme

sekarang kita exit dari runme.

bisa masukkan perintah 0, atau CTRL+C

ketika program berhenti, semua resource harus dimusnahkan. Untuk itulah DllMain akan dipanggil kembali dengan DLL\_PROCESS\_DETACH dalam hal ini

ketika DllMain dijalankan, kita ingin melakukan sesuatu

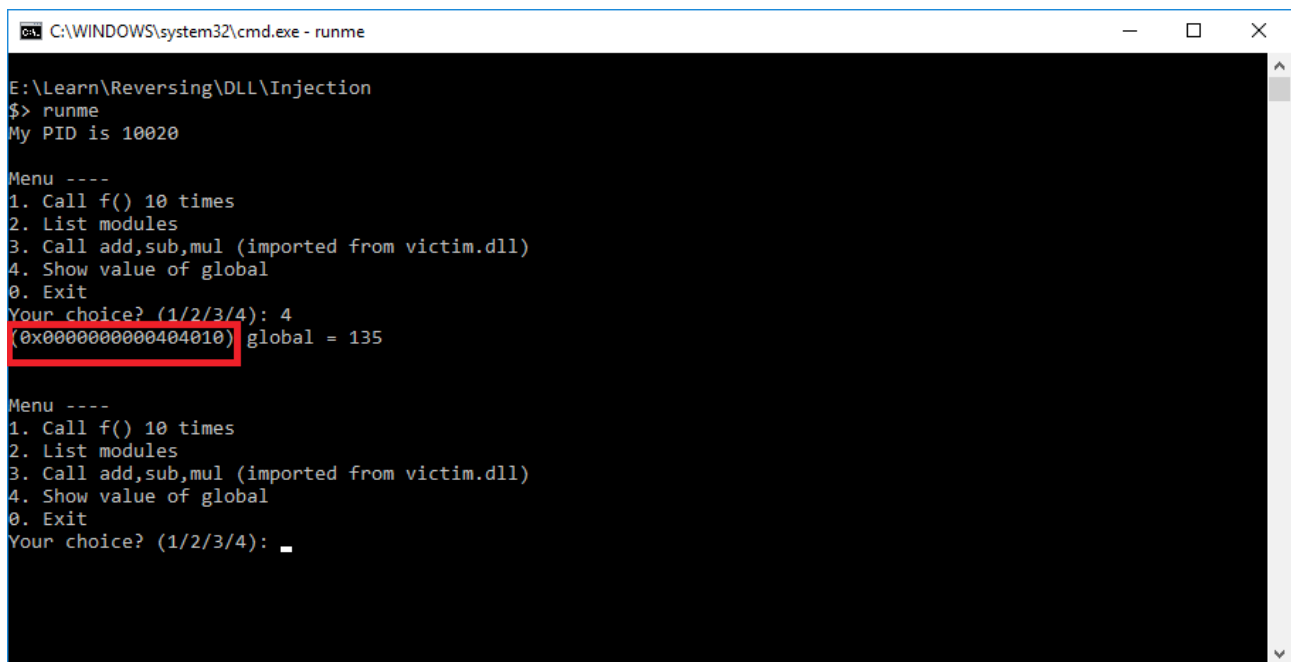
sampai sini ada yang bingung?

## » Injeksi 2: Mengubah Global Variable «

Ketika DLL di-load, aku ingin mengubah nilai si global variable

jalankan runme kemudian masukkan menu 4

aku sudah menampilkan alamat dari si global variable, contohnya seperti ini



```
C:\WINDOWS\system32\cmd.exe - runme

E:\Learn\Reversing\DLL\Injection
$> runme
My PID is 10020

Menu ----
1. Call f() 10 times
2. List modules
3. Call add,sub,mul (imported from victim.dll)
4. Show value of global
0. Exit
Your choice? (1/2/3/4): 4
(0x0000000000404010) global = 135

Menu ----
1. Call f() 10 times
2. List modules
3. Call add,sub,mul (imported from victim.dll)
4. Show value of global
0. Exit
Your choice? (1/2/3/4):
```

di kasus ini, alamatnya ada di 0x0000000000404010

kalo di program C biasa, kita bisa aja mengubahnya dengan pointer. Tapi ini kita ada di proses korban. Bagaimana caranya?

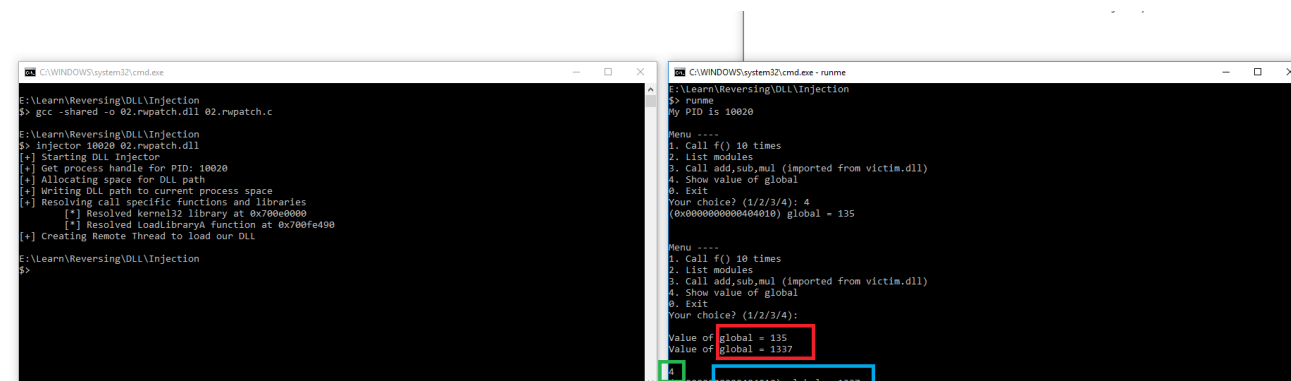
sama aja. Yang penting kita tau alamatnya

[File: 02.rwpatch.c]

bisa dilihat, aku melakukan sesuatu ketika DLL\_PROCESS\_ATTACH terjadi

aku membuat sebuah pointer bernama global yang menunjuk ke alamat 0X404010 kemudian aku isikan nilai ke \*global dengan 1337

jika sukses dieksekusi maka nilai akan berubah



```
C:\WINDOWS\system32\cmd.exe

E:\Learn\Reversing\DLL\Injection
$> gcc -shared -o 02.rwpatch.dll 02.rwpatch.c

E:\Learn\Reversing\DLL\Injection
$> injector 10020 02.rwpatch.dll
[*] Starting DLL Injector
[*] Get process handle for PID: 10020
[*] Allocating space for DLL path
[*] Writing DLL path to current process space
[*] Resolving call specific functions and libraries
[*] Resolved kernel32 library at 0x780e0000
[*] Resolved loadlibrary function at 0x780e4000
[*] Creating Remote Thread to load our DLL

E:\Learn\Reversing\DLL\Injection
$>

C:\WINDOWS\system32\cmd.exe - runme

E:\Learn\Reversing\DLL\Injection
$> runme
My PID is 10020

Menu ----
1. Call f() 10 times
2. List modules
3. Call add,sub,mul (imported from victim.dll)
4. Show value of global
0. Exit
Your choice? (1/2/3/4): 4
(0x0000000000404010) global = 135

Menu ----
1. Call f() 10 times
2. List modules
3. Call add,sub,mul (imported from victim.dll)
4. Show value of global
0. Exit
Your choice? (1/2/3/4):
Value of global = 135
Value of global = 1337
(0x0000000000404010) global = 1337
```

terlihat awalnya variabel global bernilai 135  
tetapi ketika selesai diinject, kita mengubahnya menjadi 1337, seperti terlihat di kotak biru.  
apa yang ada di kotak merah adalah printf yang dijalankan oleh DllMain()

pertanyaannya, apakah cuman begini doang gunanya? tidak.

Kita bisa melakukan yang lebih kompleks. Misalnya kita mengimplementasikan cheat seperti yang dipake di CheatEngine. Tapi kali ini kita tuliskan alamat alamat apa saja yang perlu diubah di DLL kemudian kita inject

sekarang kita menuju yang lebih badass lagi

### » Injeksi 3: Mengubah IAT «

tadi sudah disinggung bahwa add, sub, dan mul adalah fungsi yang ada di DLL victim. Runme mengakses fungsi tersebut dari IAT. Di IAT nanti akan berisi alamat yang dapat dipakai oleh runme untuk memanggil fungsi di DLL. Apa jadinya jika alamat di IAT ini diubah?

sekarang jalankan runme dan masukkan menu 2  
lihat apa yang ada di sana kemudian liat source code.  
silahkan eksplorasi dulu 5 menit untuk memahami kodenya.

```
printf("add ( 0x%p ) == pointers[0] ( 0x%p )\n", add, pointers[0]);
```

fungsi add adalah fungsi yang didapat dari IAT,  
sementara fungsi pointers[0] adalah alamat add didapatkan dengan cara mengambil langsung dari alamat yang ada di victim.dll  
alamat ini didapatkan dari fungsi

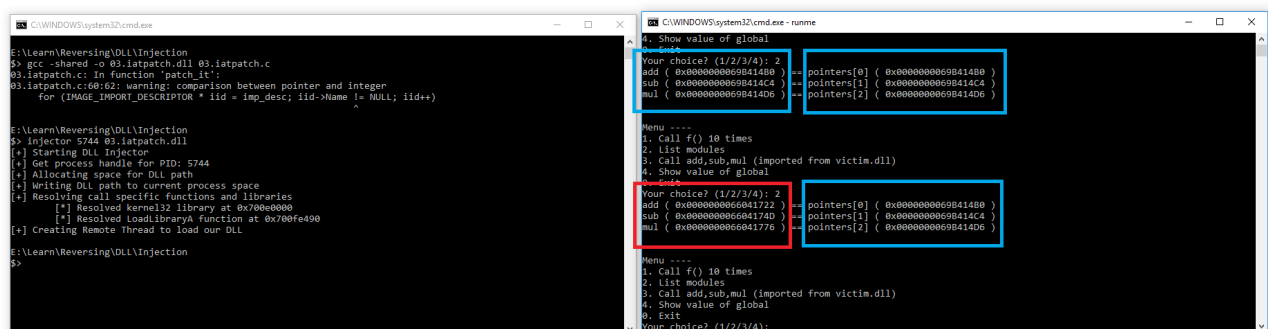
```
pointers[0] = (func_ptr) GetProcAddress(handle, "add");
```

harusnya, keduanya (add dan pointers[0]) akan bernilai sama  
sekarang tujuan kita adalah:

ketika add() dipanggil, yang dipanggil sebenarnya adalah fungsi yang sudah kita bikin, bukan fungsi add. begitu juga untuk sub dan mul

[File: 03.iatpatch.c]

inject



The image shows two side-by-side Windows command prompt windows. The left window shows the compilation and execution of a DLL injector. The right window shows a menu-driven program where the user selects option 2 to show global values. In the right window, two boxes highlight specific memory addresses: a red box highlights the address 0x0000000000001776 (which is 1337) and a blue box highlights the address 0x0000000000001400 (which is 135).

```
C:\WINDOWS\system32\cmd.exe
E:\Learn\Reversing\DLL\Injection
$> gcc -shared -o 03.iatpatch.dll 03.iatpatch.c
03.iatpatch.c: In function 'patch_it':
03.iatpatch.c:60:62: warning: comparison between pointer and integer
    for (IMAGE_IMPORT_DESCRIPTOR * iid = imp_desc; iid->Name != NULL; iid++)
                                                             ^
E:\Learn\Reversing\DLL\Injection
$> injector 5744 03.iatpatch.dll
[*] Starting Dll Injector
[*] Get process handle for PID: 5744
[*] Allocating space for DLL path
[*] Writing DLL path to current process space
[*] Resolving call specific functions and libraries
    [*] Resolved kernel32 library at 0x700e0000
    [*] Resolved LoadLibraryA function at 0x700fe490
[*] Creating Remote Thread to load our DLL
E:\Learn\Reversing\DLL\Injection
$>
```

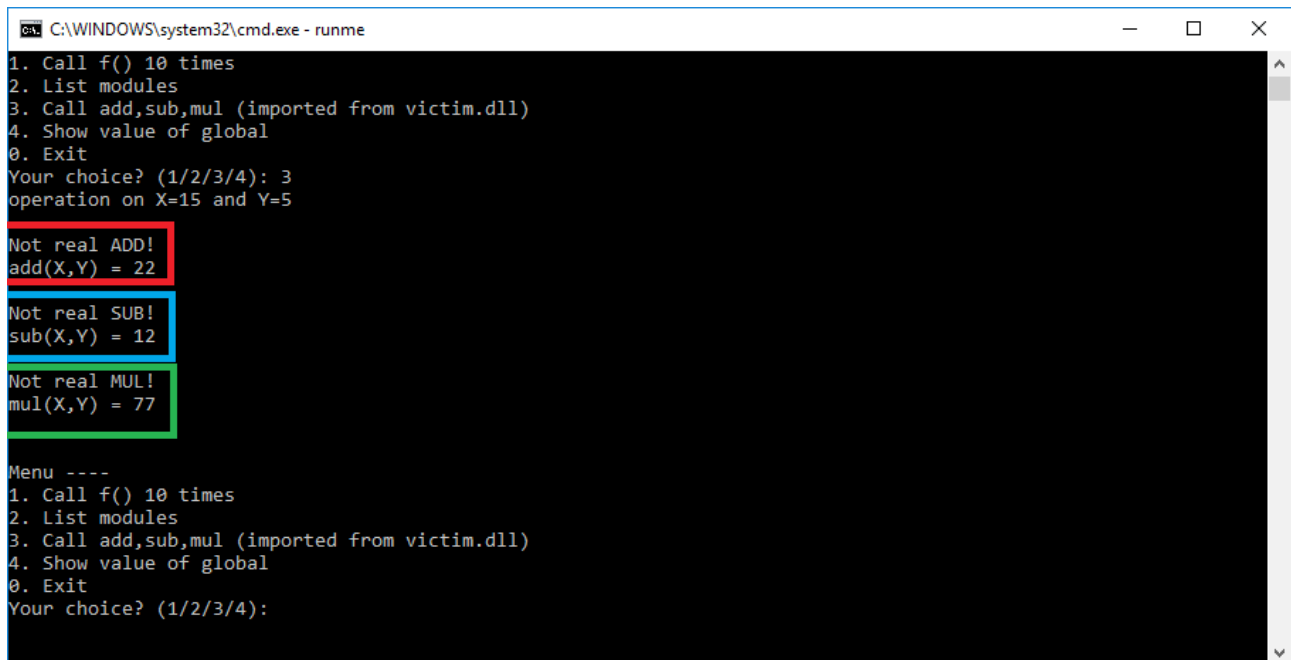
```
C:\WINDOWS\system32\cmd.exe - runme
4. Show value of global
Your choice? (1/2/3/4): 2
add ( 0x0000000000001400 ) == pointers[0] ( 0x0000000000001400 )
sub ( 0x00000000000014c4 ) == pointers[1] ( 0x00000000000014c4 )
mul ( 0x00000000000014d6 ) == pointers[2] ( 0x00000000000014d6 )

Menu ----
1. Call f() 10 times
2. List modules
3. Call add,sub,mul (imported from victim.dll)
4. Show value of global
Your choice? (1/2/3/4): 2
add ( 0x0000000000001772 ) == pointers[0] ( 0x0000000000001400 )
sub ( 0x0000000000001740 ) == pointers[1] ( 0x00000000000014c4 )
mul ( 0x0000000000001776 ) == pointers[2] ( 0x00000000000014d6 )

Menu ----
1. Call f() 10 times
2. List modules
3. Call add,sub,mul (imported from victim.dll)
4. Show value of global
0. Exit
Your choice? (1/2/3/4):
```

sekarang bandingkan dengan memasukkan menu 2 lagi

alamat IAT untuk add, sub, dan mul telah berubah



```
C:\WINDOWS\system32\cmd.exe - runme
1. Call f() 10 times
2. List modules
3. Call add,sub,mul (imported from victim.dll)
4. Show value of global
0. Exit
Your choice? (1/2/3/4): 3
operation on X=15 and Y=5
Not real ADD!
add(X,Y) = 22
Not real SUB!
sub(X,Y) = 12
Not real MUL!
mul(X,Y) = 77
Menu ----
1. Call f() 10 times
2. List modules
3. Call add,sub,mul (imported from victim.dll)
4. Show value of global
0. Exit
Your choice? (1/2/3/4):
```

sehingga kalo kita panggil ketiga fungsi itu (dengan menu 3), kita malah memanggil fungsi yang ada di 03.iatpatch.dll bukan di victim.dll  
alamat di victim.dll tetap sama, yang berbeda hanya alamat yang dikenali di runme saja

sekarang bayangkan kalo kita bisa membuat fungsi penting, menjadi fungsi yang kita definisikan.  
sampai kita ke baddas terakhir di kulgram ini

» Injeksi 4: Manipulasi «

[File: 04.iatinstrument.c]

kita tetap target IAT, tapi kini kita fokus ke bagaimana modifikasi fungsi tersebut.  
Kita sekarang nggak pengen mengganti fungsi begitu aja. Ada 3 hal yang bisa kita lakukan.

1. melihat argumen yang ketika fungsi dipanggil
2. memodifikasi argumen itu
3. melihat return value hasil eksekusi fungsi

kita akan tetap memanggil fungsi asli.

jalankan runme dan inject 04.iatinstrument.dll

kita bisa liat kemampuan dari DLL injection ini bisa kita manfaatkan untuk memanipulasi kelakuan dari sebuah binary