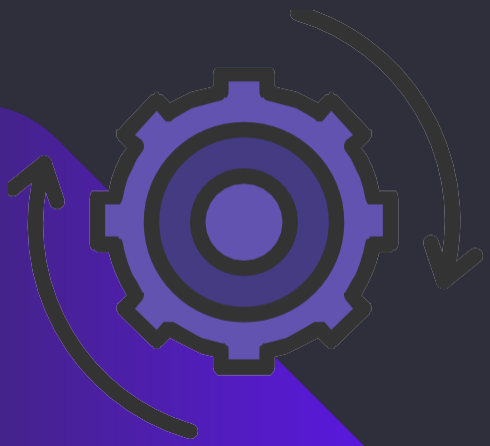


Reverse Engineering

Malware Analysis

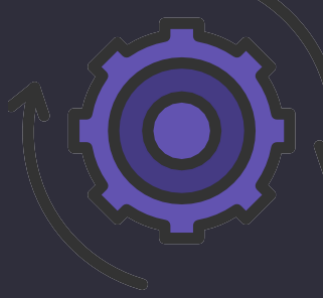
Prepared by

Arya Widyanto Utomo

2024

+6285865492276
utomoa448@gmail.com
www.reallygreatsite.com

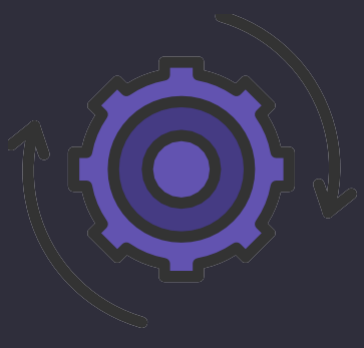
Table of CONTENTS



Apa itu Reverse Engineering ?

Reverse Engineering adalah proses mengambil perangkat lunak atau perangkat keras yang sudah ada dan membongkar struktur dan fungsionalitasnya untuk memahami cara kerjanya. Proses ini melibatkan beberapa langkah teknis seperti : Disassembly,Decompilation,Analisis static dan dinamis

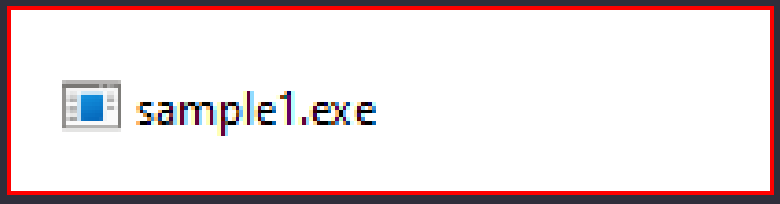
Bergantung pada teknologinya, pengetahuan yang diperoleh selama **reverse engineering** dapat digunakan untuk menggunakan kembali objek yang sudah usang, melakukan analisis keamanan, memperoleh keunggulan kompetitif, atau sekadar mengajari seseorang tentang cara kerja sesuatu. Tidak peduli bagaimana pengetahuan itu digunakan atau apa kaitannya, **reverse engineering** adalah proses memperoleh pengetahuan itu dari objek yang sudah jadi.



Latar Belakang

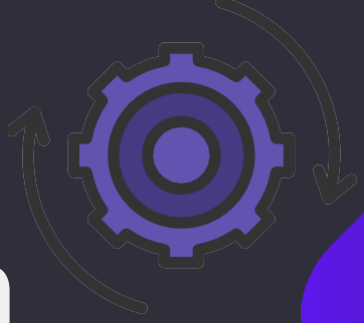
Sample1.exe adalah sebuah executable / program yang berbahaya yang dibuat menggunakan metasploit. Reverse engineering ini bertujuan untuk memahami cara kerja malware dan mengidentifikasi metode infeksinya. Reverse Engineering akan dilakukan dikali linux menggunakan alat seperti IDA Pro,Ghidra,Cutter untuk melakukan disassembly,Discompiler,dan akan dilakukan analisis lebih lanjut.

Berikut adalah bukti keberadaan program binary/executable tersebut : https://drive.google.com/file/d/1zMFje_PopWRPZAc3PzFW3atTFvuvvQc5/view?usp=sharing



Tools yang akan digunakan :

Tujuan Pemeriksaan



01 Mencari ukuran shellcode-nya?

02 Mencari tahu apa yang dilambangkan oleh nilai *flAllocationType* di VirtualAlloc?

03 Mencari tahu argumen yang dibutuhkan untuk menjalankan shellcode?

04 Mencari tahu payload Metasploit yang digunakan untuk menghasilkan shellcode?

05 Mencari tahu API apa yang digunakan untuk membuat wait objects

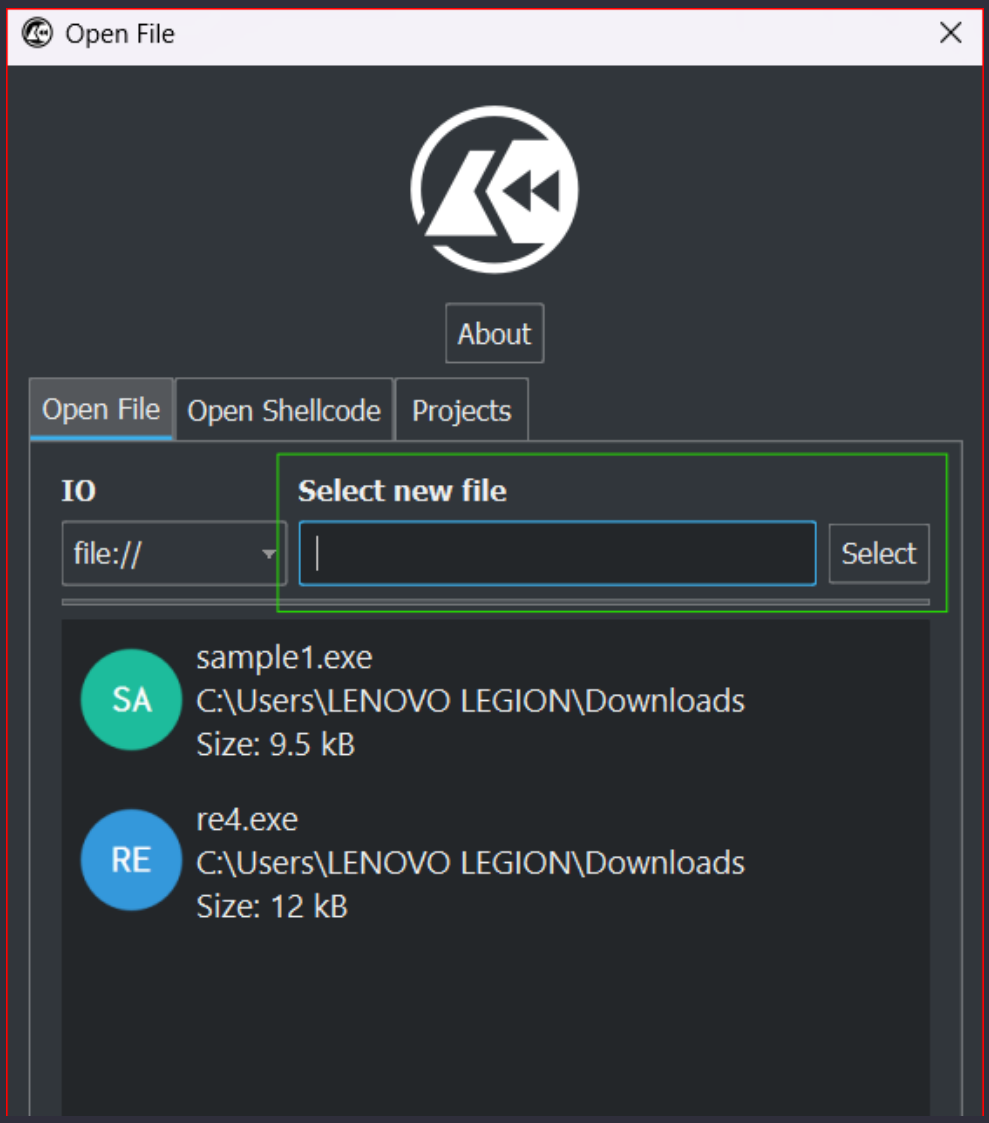
06 Mencari tahu function library yang digunakan untuk menyalin shellcode antar blok memori

07 Mencari tahu daftar nama log seperti dalam urutan dalam skrip

How to use Cutter ?

Tips cara menggunakan

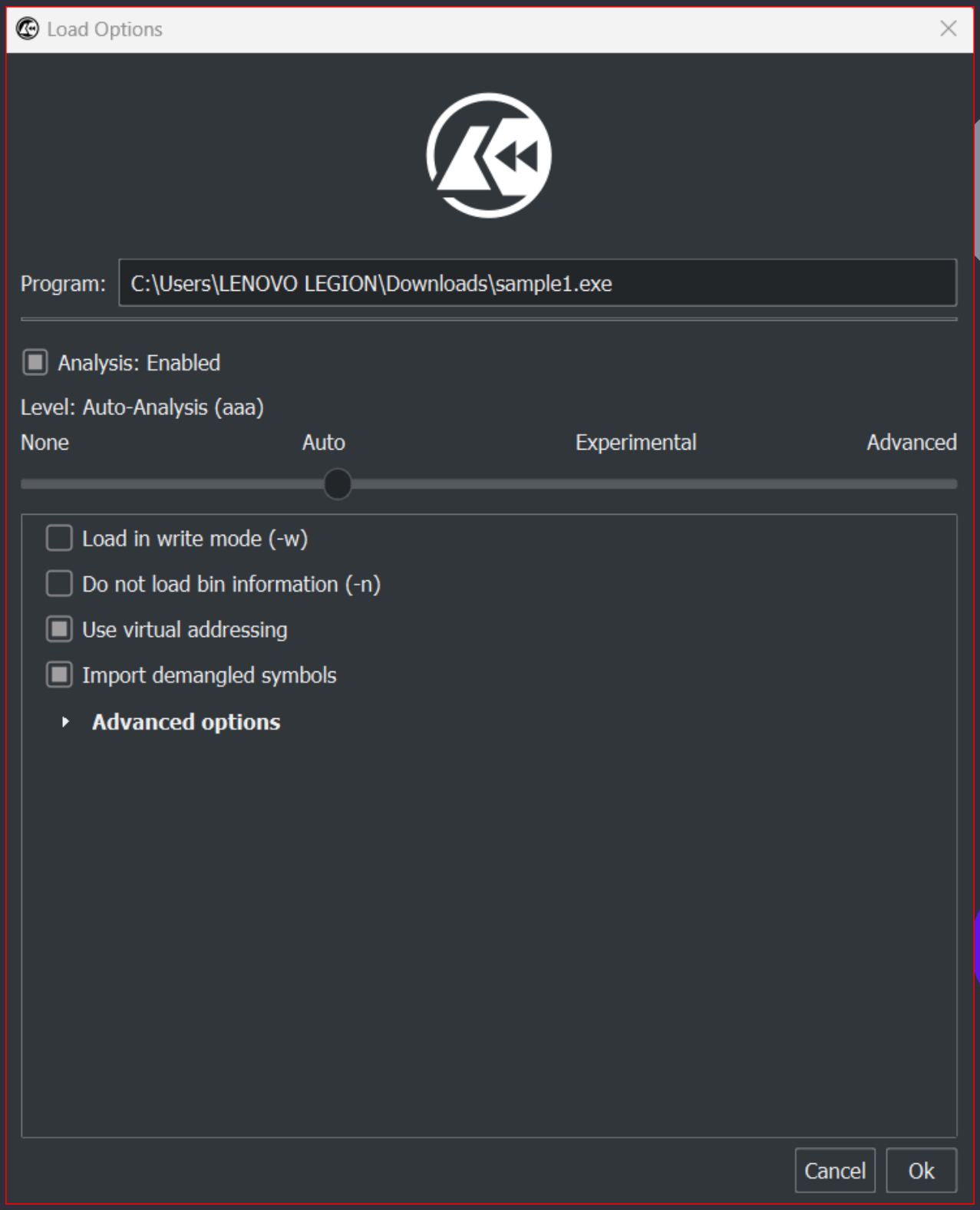
Jadi,pertama tama kita harus melakukan diassembly dan decompiler menggunakan tools Cutter.



yang pertama,kita tinggal ke fitur kotak hijau untuk memilih file binary/executable mana yang ingin kita reverse engineering,setelah itu tinggal oke.

How to use Cutter ?

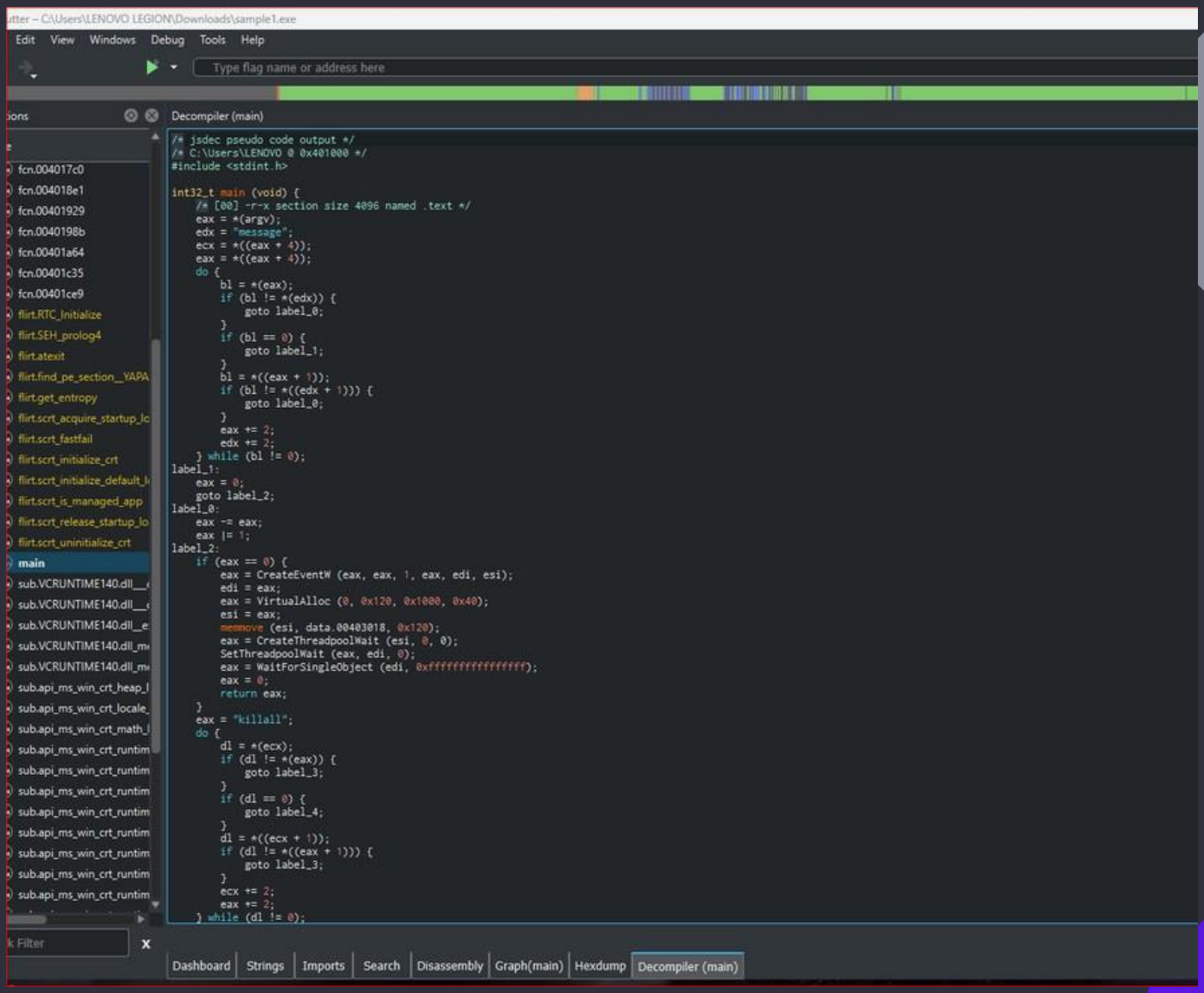
Tips cara menggunakan

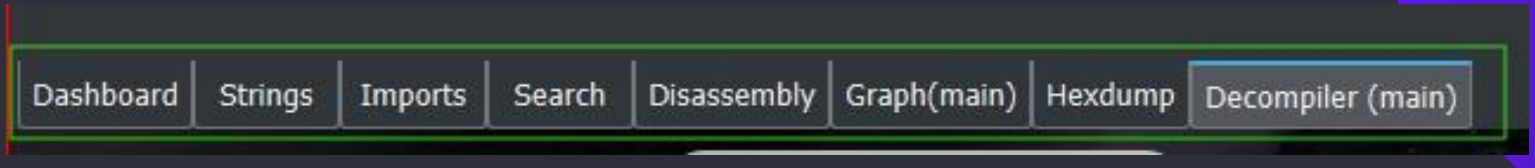


Ketika dibagian ini,kita tinggal oke saja,biarkan saja default.

How to use Cutter ?

Tips cara menggunakan





Berikutlah hasilnya,kita tinggal pilih saja hasil yang kita mau dipilihan plg bawah,dari hasil dashboard,strings,sampai hasil decompiler.

Hasil Pemeriksaan

1.1

Masalah Pertama

Untuk menjadikan shellcode berfungsi dalam program, aspek yang krusial adalah pengaturan 'ukuran' dalam sebuah memori. function yang digunakan untuk mendapatkan alokasi memori yang tepat untuk menyimpan shellcode adalah melalui API VirtualAlloc.

```
bel_2:
  if (eax == 0) {
      eax = CreateEventW (eax, eax, 1, eax, edi, esi);
      edi = eax;
      eax = VirtualAlloc (0, 0x120, 0x1000, 0x40);
      esi = eax;
      memmove (esi, data.00403018, 0x120);
      eax = CreateThreadpoolWait (esi, 0, 0);
      SetThreadpoolWait (eax, edi, 0);
      eax = WaitForSingleObject (edi, 0xffffffffffffffff);
      eax = 0;
      return eax;
  }
```

Setelah melakukan Decompiler pada file executable menggunakan Cutter,disini saya mendapatkan API VirtualAlloc yang digunakan untuk alokasi memori,diatas terlihat juga nilai dari setiap argumennya dalam bentuk hexadesimal,tapi kita belum mengetahui argumen apa saja itu.

Hasil Pemeriksaan

1.2

Masalah Pertama

Jadi,disini saya coba mencari tahu dengan membuka lewat referensi microsoft docs : https://learn.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualalloc#syntax

Syntax

C++

```
LPVOID VirtualAlloc(
  [in, optional] LPVOID lpAddress,
  [in]           SIZE_T dwSize,
  [in]           DWORD  flAllocationType,
  [in]           DWORD  flProtect
);
```

Pada argumen kedua dari VirtualAlloc diatas menunjukan nilai tentang ukuran. jika dilihat dari isi argumen kedua dari hasil decompiler tadi bernilai 0x120,nilai masih dalam bentuk hexadesimal,jadi saya akan mengubahnya ke bentuk desimal terlebih dahulu,dan menghasilkan nilai desimal : ***288***

Hasil Pemeriksaan

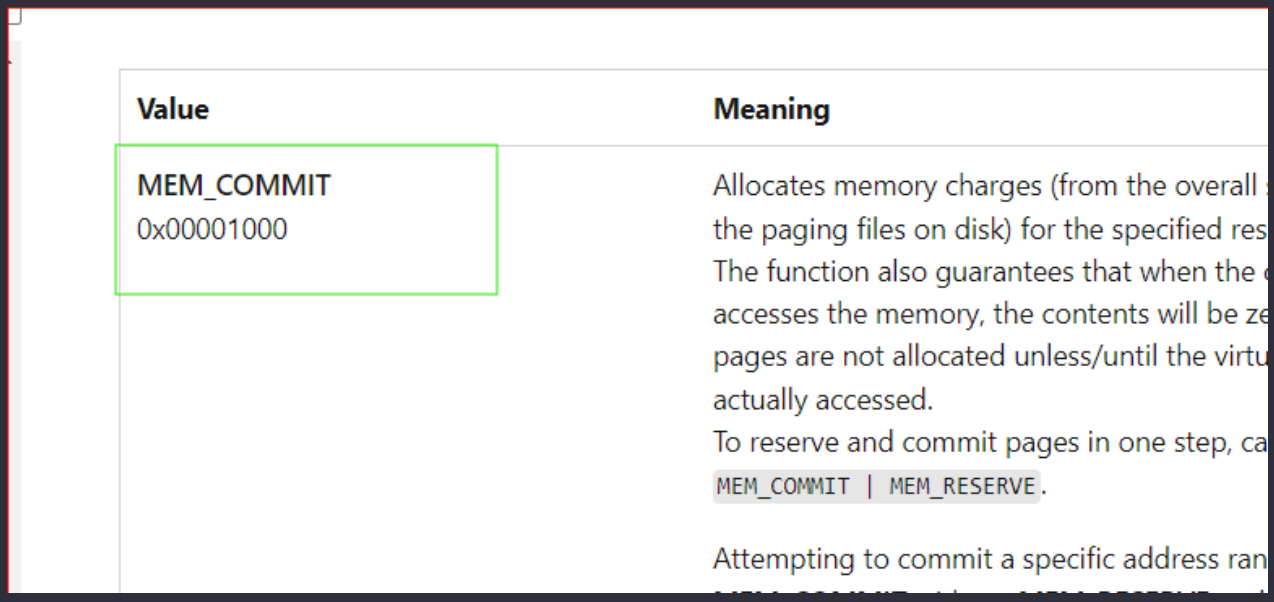
2

Masalah Kedua

Masih berkaitan dengan documentasi microsoft dan function VirtualAlloc tadi,kali Ini kita akan mencari arti dari argumen ke 3,yaitu fiAllocationType.Pada hasil decompiler tersebut,argumen tersebut memiliki nilai 0x1000.

```
VirtualAlloc (0, 0x120, 0x1000, 0x40);
```

Jadi kita akan mencari arti value dari nilai 0x1000 diatas dengan referensi microsoft docs.



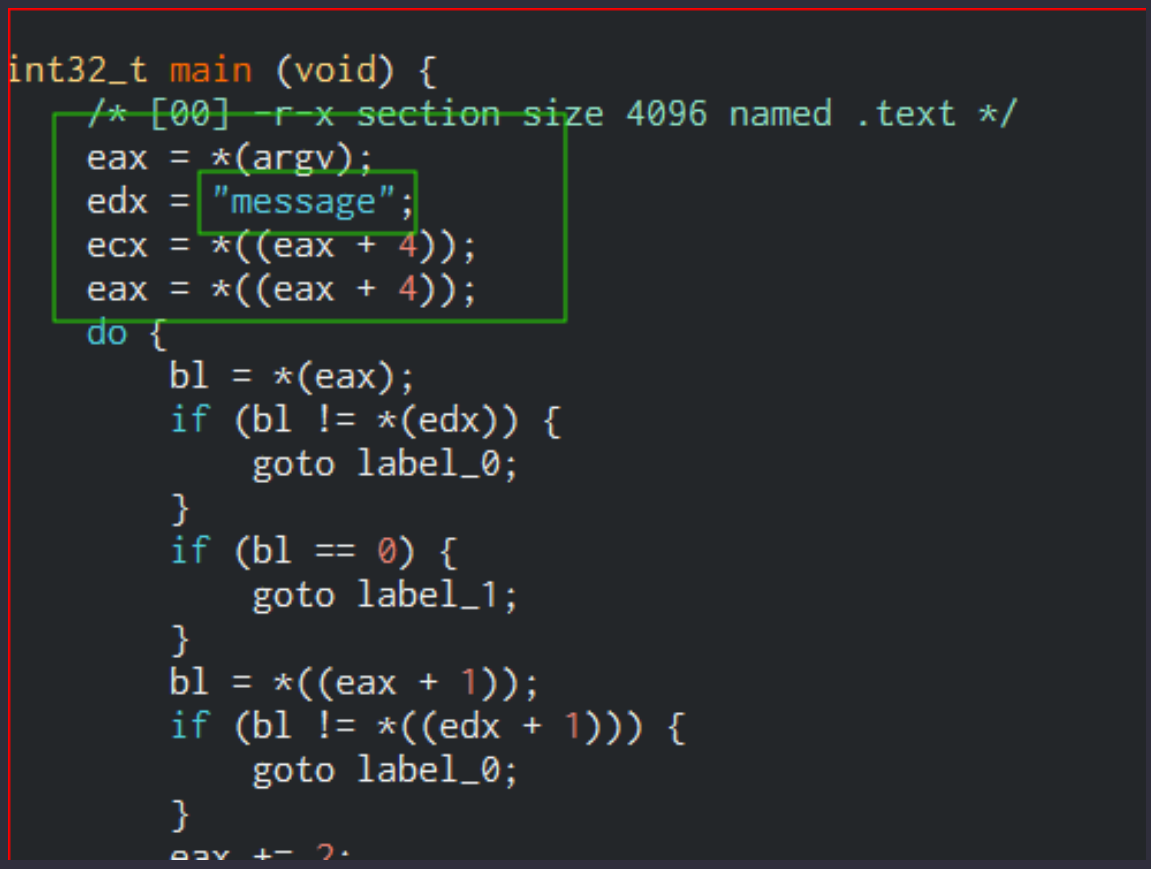
Value	Meaning
MEM_COMMIT 0x00001000	Allocates memory charges (from the overall the paging files on disk) for the specified res The function also guarantees that when the accesses the memory, the contents will be ze pages are not allocated unless/until the virtu actually accessed. To reserve and commit pages in one step, ca `MEM_COMMIT | MEM_RESERVE`. Attempting to commit a specific address ran

Menurut referensi microsoft docs,nilai tersebut memiliki value ***MEM_COMMIT***

Hasil Pemeriksaan 3

Masalah Ketiga

Kita akan memeriksa lagi hasil decompilernya untuk mencari argumen yang diperlukan untuk menjalankan shellcode.



```
int32_t main (void) {
    /* [00] -r-x section size 4096 named .text */
    eax = *(argv);
    edx = "message";
    ecx = *((eax + 4));
    eax = *((eax + 4));
    do {
        bl = *(eax);
        if (bl != *(edx)) {
            goto label_0;
        }
        if (bl == 0) {
            goto label_1;
        }
        bl = *((eax + 1));
        if (bl != *((edx + 1))) {
            goto label_0;
        }
        eax += 2;
```

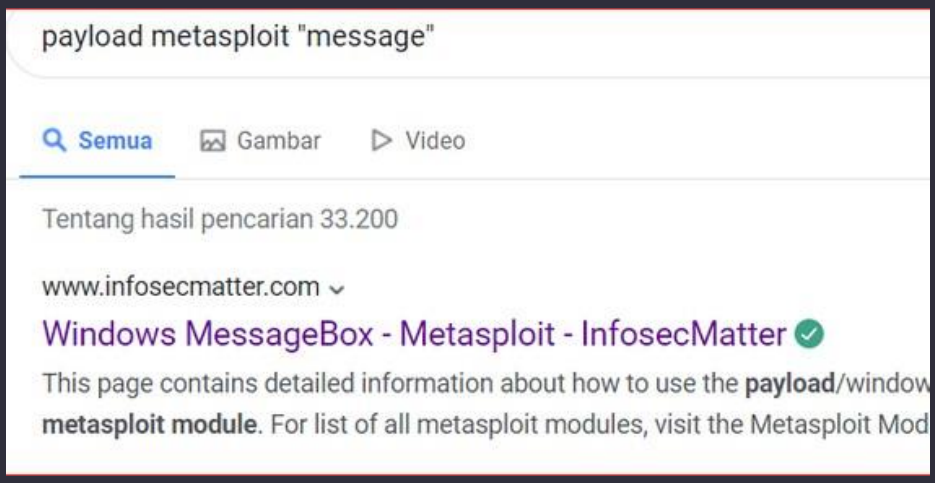
pada bagian register edx,menyimpan variabel "***message***" untuk mencari dan membandingkan string dalam konteks program yang diberikan. Jika string yang dicari ("message") cocok dengan string yang ditemukan dalam proses pemindaian, program akan menjalankan shellcode yang tersembunyi di dalamnya menggunakan PowerShell dengan parameter tertentu.

Hasil Pemeriksaan

4

Masalah Keempat

Untuk mencari payload yang sesuai untuk menghasilkan shellcode berdasarkan hasil decompiler tersebut,saya akan menggunakan search engine.Saya akan memakai kata message untuk mencari sebuah payload metasploitnya.



Berikut adalah sumbernya yang saya temukan : https://www.infosecmatter.com/metasploit-module-library/?mm=payload/windows/messagebox.Berdasarkan documentasi tersebut payload metasploitnya adalah ***windows/messagebox***

Hasil Pemeriksaan

5

Masalah Kelima

Kita akan memeriksa lagi hasil decompilernya untuk mencari API yang membuat atau menghasilkan wait object.

```
if (eax == 0) {
    eax = CreateEventW (eax, eax, 1, eax, edi, esi);
    edi = eax;
    eax = VirtualAlloc (0, 0x120, 0x1000, 0x40);
    esi = eax;
    memmove (esi, data.00403018, 0x120);
    eax = CreateThreadpoolWait (esi, 0, 0);
    SetThreadpoolWait (eax, edi, 0);
    eax = WaitForSingleObject (edi, 0xffffffffffffffff);
    eax = 0;
    return eax;
}
```

Jadi,API yang digunakan untuk menghasilkan wait object dari decompiler diatas adalah ***CreateThreadpoolWait.*** Fungsi ini memungkinkan aplikasi untuk menentukan suatu fungsi yang akan dipanggil ketika kondisi tunggu tertentu tercapai. Objek tunggu yang dibuat dengan CreateThreadpoolWait dapat digunakan bersama dengan fungsi lain dalam thread pool API, seperti SetThreadpoolWait untuk mengatur objek tunggu yang telah dibuat. Berbeda dengan API CreateEventW yang digunakan ntuk membuat objek event (HANDLE) yang digunakan untuk sinkronisasi antar thread atau proses.

Hasil Pemeriksaan

6

Masalah Keenam

```
if (eax == 0) {
    eax = CreateEventW (eax, eax, 1, eax, edi, esi);
    edi = eax;
    eax = VirtualAlloc (0, 0x120, 0x1000, 0x40);
    esi = eax;
    memmove (esi, data.00403018, 0x120);
    eax = CreateThreadpoolWait (esi, 0, 0);
    SetThreadpoolWait (eax, edi, 0);
    eax = WaitForSingleObject (edi, 0xffffffffffffffff);
    eax = 0;
    return eax;
}
```

Library untuk menyalin blok memory umumnya adalah memcpy dan memmove,tapi dalam decompiler diatas menggunakan ***memmove***. dalam kode decompiler diatas memmove digunakan dengan benar untuk menyalin shellcode dari data.00403018 ke lokasi yang dialokasikan dengan VirtualAlloc.

Adapun perbedaan antara memcpy dan memmove,yaitu memcpy: Ideal untuk menyalin data di mana tidak ada tumpang tindih antara src dan dest. Tidak menjamin hasil yang benar jika src dan dest tumpang tindih sedangkan memmove aman untuk menyalin data bahkan jika src dan dest tumpang tindih. Lebih lambat daripada memcpy dalam kasus tidak ada tumpang tindih karena perlu menangani kasus-kasus tumpang tindih.

Hasil Pemeriksaan

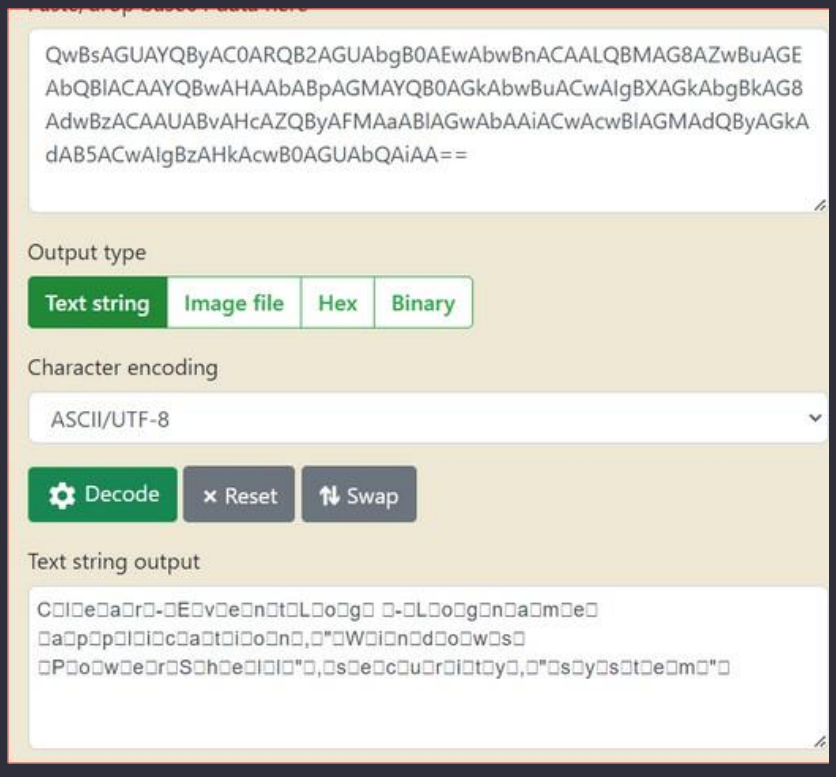
Masalah Ketujuh

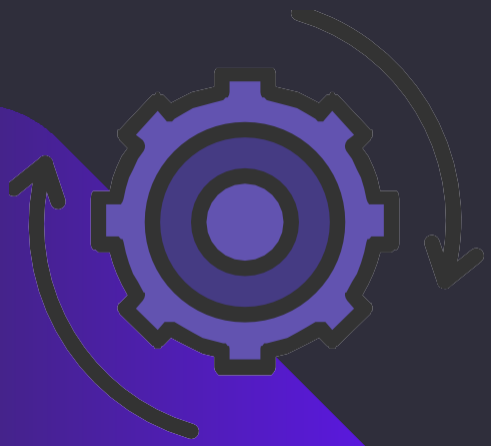
```
eax |= 1;
l_5:
if (eax == 0) {
    eax = system ("powershell -ep bypass -enc QwBsAGUAYQByA(
}
eax = 0;
return eax;
```

```
AAUABvAHcAZQByAFMAaABlAGwAbAAiACwAcwBlAGMAdQByAGkAdAB5ACwAIgBzAHkAcwB0AGUAbQAiAA==");
```

Pada potongan kode decompiler diatas,terlihat powershell mengeksekusi sebuah string yang diencode. Diihat dari karakteristiknya dengan padding "==" dibelakangnya,sepertinya merupakan encoding base64. Dan setelah didecode hasilnya adalah

Clear-EventLog -Logname ***application,"Windows PowerShell",security,"system***"





TERIMA KASIH

2024

+6285865492276
utomoa448@gmail.com
www.reallygreatsite.com