PLAINBIT

2025 사이버보안전문단 연구 프로젝트

바로가기 악성파일의 구조를 활용한 공격자 프로파일링

2025.11

이준형, 이현진, 안상욱, 현주연

※ 해당 보고서는 사이버보안전문단 연구 프로젝트 결과물로 작성되었습니다.

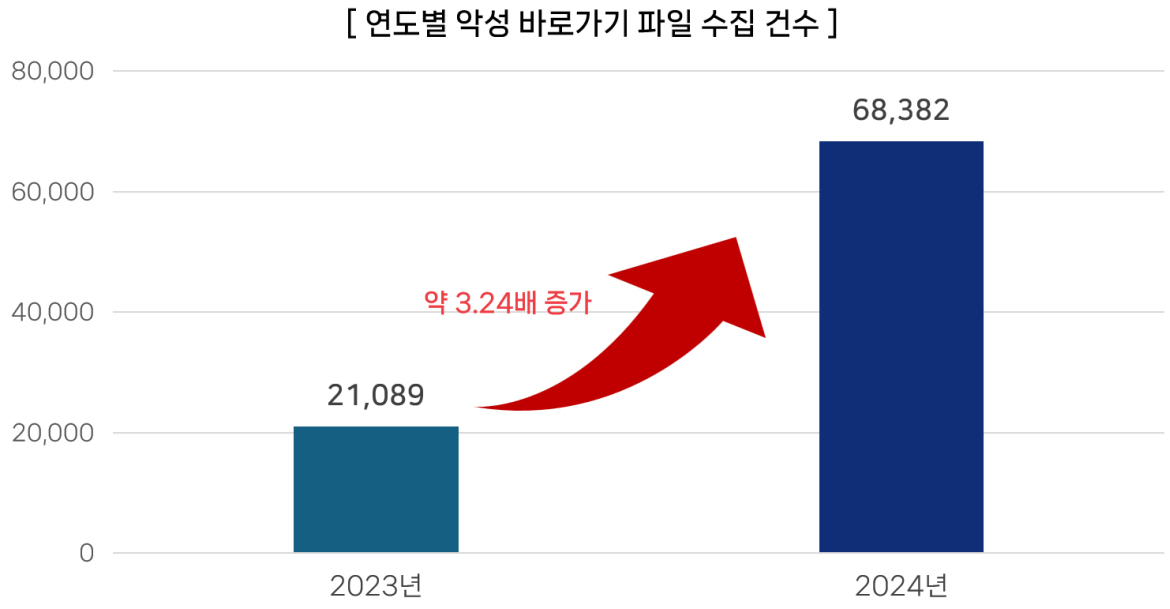
목 차

1. 연구의 개요

1.1. 연구의 배경 및 필요성

최근 사이버 공격은 갈수록 복잡해지고 있으며 공격자는 새로운 유포 경로와 우회 기법을 지속적으로 도입하고 있다. 이러한 변화에 신속하게 대응하기 위해서는 개별 사건 단위의 대응을 넘어 공격의 맥락과 행위 패턴을 체계적으로 분석하는 위협 인텔리전스의 도입이 필수적이다. 공격자가 사용하는 파일 유형, 인프라 구성, 공격 환경 등 세부적인 데이터를 확보하고 비교.분석함으로써 위협 행위자의 전술.기술.절차(TTPs)를 식별해 보다 정교하고 지속성 있는 대응방안을 적용할 수 있기 때문이다.

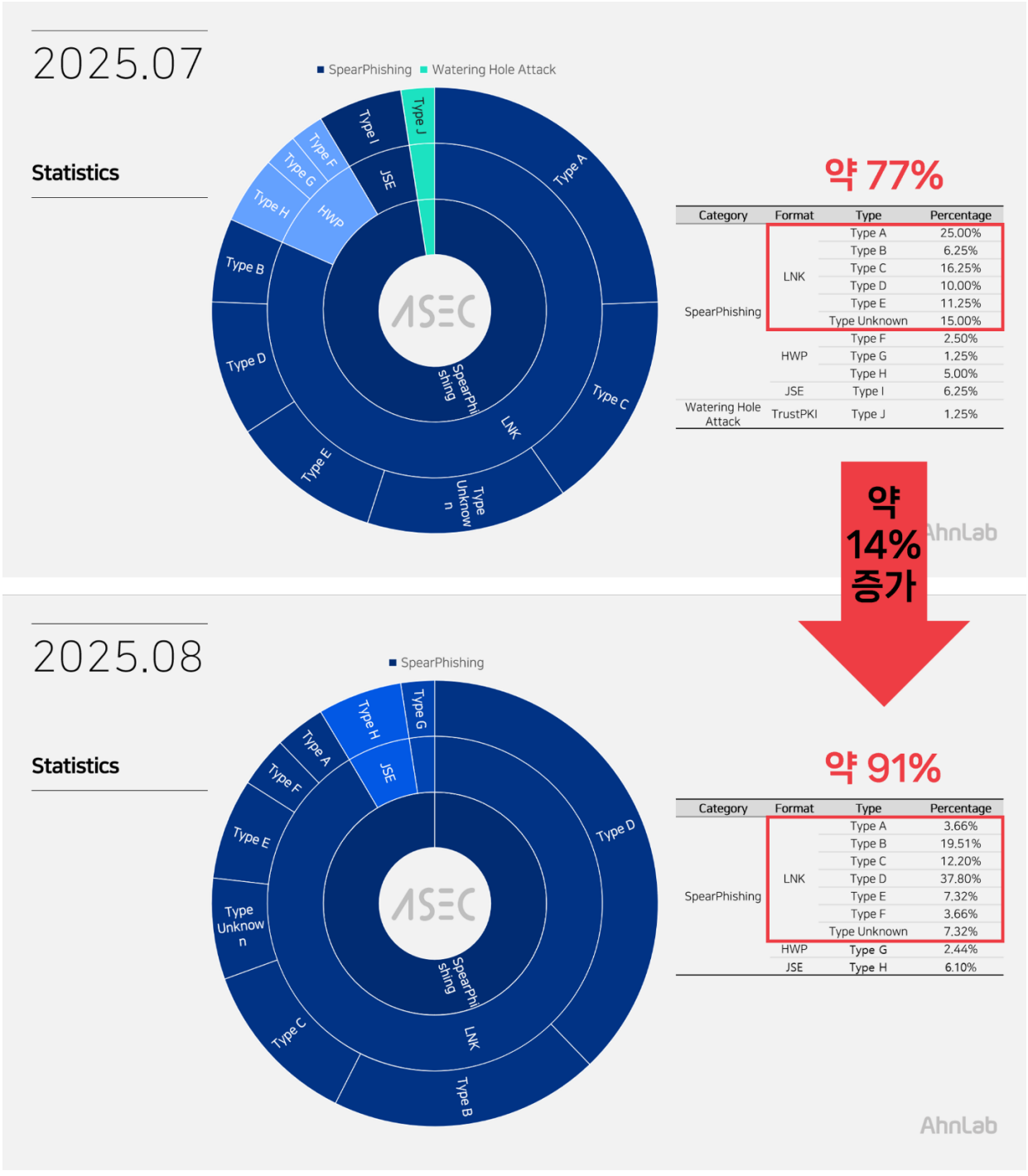
과거에는 실행파일(.exe)이나 문서파일 내 매크로를 활용한 공격이 주로 초기 침투 수단으로 활용되었으나, 최근에는 Windows 바로가기 파일을 악용한 공격이 급격히 증가하고 있다. 팔로알토네트웍스社의 Unit42는 2023년 수집된 악성 바로가기 파일 샘플 수가 21,098건이었던 반면 2024년에는 68,392건으로 약 3.24배 증가했다고 발표했다[1].



[그림 1] 팔로알토네트워크社 Unit42 발표 통계 자료

[1] Haizhou Wang, Ashkan Hosseini, Ashutosh Chitwadgi, “Windows Shortcut (LNK) Malware Strategies”, PaloaltoNetworks Unit42, 2025-07-02, https://unit42.paloaltonetworks.com/lnk-malware/

국내에서도 안랩社의 ASEC은 2025년 8월 발생한 APT 공격의 대부분이 스피어피싱 방식으로 유포되었으며, 그 중 바로가기 파일을 활용한 공격이 전체의 91%를 차지한다고 발표했다[2]. 이는 전월 대비 약 14% 증가한 수치[3]로 최근에도 계속해서 바로가기 파일을 활용한 공격이 빈번하게 발생하고 있음을 시사한다.



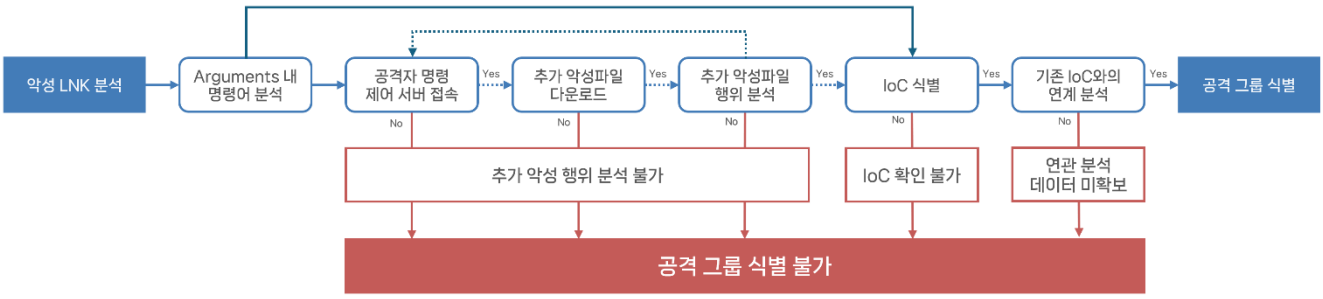
[그림 2] 국내 APT 공격의 분류 및 통계

[2] ASEC, "2025년 8월 APT 공격 동향 보고서(국내)", 2025-09-12, https://asec.ahnlab.com/ko/90103/
[3] ASEC, "2025년 7월 APT 공격 동향 보고서(국내)", 2025-08-14, https://asec.ahnlab.com/ko/89578/

바로가기 파일은 운영체제에서 기본적으로 사용하는 형식이기 때문에 사용자에게 높은 신뢰감을 주며 이메일 첨부나 클라우드 공유 등으로 압축파일을 통해 쉽게 전달할 수 있다. 또한, Windows 환경에서는 기본적으로 확장자가 숨겨져 표시되므로 사용자가 이를 일반 문서파일로 오인해 실행하는 경우가 빈번하다. 공격자는 이러한 특성을 악용해 바로가기 파일 내부에 명령줄 인자를 삽입하고 이를 통해 PowerShell, cmd, mshta 등의 정상 시스템 구성 요소를 호출하도록 설계한다. 사용자가 파일을 클릭하면 공격자의 서버나 인프라에서 추가 페이로드를 다운로드 및 실행하는 다단계 공격이 이루어진다.

이러한 악성 바로가기 파일은 단순한 악성 행위 실행을 위한 파일을 넘어 공격자와 관련된 다양한 정보를 담고 있는 분석 단서로 활용될 수 있다. 예를 들어, 바로가기 내부에 삽입된 명령줄 인자를 분석하면 공격자가 초기 단계에서 접속을 시도하는 서버 주소나 경유지를 파악할 수 있으며 이를 기반으로 해당 서버에서 추가로 다운로드되는 악성 파일을 확보할 수 있다. 확보된 페이로드는 행위 기반 분석을 통해 명령 제어 방식이나 내부 침투 기법 등을 파악하는 데 활용되며, 이를 통해 침해지표(IoC; Indicators of Compromise)를 식별하고 기존 데이터와의 연관성 분석을 통해 공격자를 식별할 수 있다.

하지만 이러한 방식은 몇 가지 실질적인 한계를 수반한다. 공격자는 추가 페이로드를 배포하는 서버나 명령 제어 인프라를 빠르게 폐기하거나 변경해 추적을 어렵게 만들며, 최근에는 Google Drive, Dropbox, Github 등과 같은 정상적인 클라우드 기반 서비스를 악용해 악성파일을 유포함으로써 전통적인 탐지 체계를 우회하고 분석의 난이도를 높이고 있다. 이로 인해 추가 악성파일을 확보하지 못하거나 분석 시간이 지연되는 경우가 잦아지고 있으며, 침해지표(IoC) 중심의 사후적 대응만으로는 공격자의 식별이나 전술 파악에 한계가 발생하고 있다.



[그림 3] 악성 바로가기 파일 분석 절차 및 공격자 식별 방안

따라서 외부 인프라나 추가 페이로드 확보 여부에 의존하지 않고 공격자의 정보를 보다 안정적으로 추적할 수 있는 새로운 접근 방식이 요구된다. 이에, 본 연구에서는 그 대안 중 하나로 바로가기 파일 자체에 포함되는 구조적 정보와 제작 환경 정보를 분석하는 방법론을 제안하고자 한다. 바로가기 파일에는 생성 당시의 볼륨 정보, 시스템 식별자 등 다양한 환경 정보가 저장되며 이는 단일 샘플만으로도 공격자의 개발 환경이나 악성 바로가기 파일 제작 방식에 대한 단서를 제공할 수 있다. 더 나아가 다수의 샘플 간 구조적 유사성을 비교함으로써, 동일한 제작 도구의 사용 여부를 식별하고, 이를 통해 특정 공격 그룹의 전술.기술.절차(TTPs)를 사전에 추론하는 위협 인텔리전스 방법을 제안하고자 한다.

1.2. 연구의 방법 및 범위

1) 연구의 방법

본 연구는 바로가기 파일 구조를 기반으로 악성 바로가기 파일을 제작한 공격자를 식별하기 위해 다음과 같은 단계로 진행했다.

먼저, 국내외 보안 업체 블로그, 트위터 등 OSINT를 통해 공개된 악성 바로가기 파일 샘플을 수집했다. 또한, MalwareBazaar, Google 위협 인텔리전스 플랫폼 등에서 악성 바로가기 파일의 특징을 바탕으로 YARA 룰을 기반으로 검색해 추가적으로 악성 바로가기 샘플을 확보했다. 그 결과, 구조가 온전히 보전되어 분석이 가능한 샘플로 한정하여 최종적으로 총 1,135개의 악성 바로가기 파일 샘플을 확보했다. 샘플 수집 과정에서 배후 국가나 공격 그룹이 특정된 경우, 해당 정보를 구조 분석 시 보조 지표로 활용하기 위해 별도로 메타데이터화했다. 이때 출처의 공신력, 분석 공개 여부 등을 고려해 출처별 신뢰도 기준을 설정했으며, 이를 통해 향후 식별된 구조적 특징이 특정 공격 그룹과 연관성이 있는지 해석하는데 참고 지표로 활용했다.

[표 1] 수집 출처별 신뢰도 구분 지표

신뢰도 구분	주요 출처 예시	사유
상	• Google, CrowdStrike, Group-IB 등 위협 인텔리전스 플랫폼에서 제공하는 정보 • 국내외 보안 업체 공식 블로그에서 제공하는 정보	• 악성파일 분석 과정 및 공격자 식별 근거 공개
중	• MAlwareBazaar 등 악성파일 샘플 공유 플랫폼 • Rewterz의 Active IoCs	• 악성파일 분석 과정 및 공격자 식별 근거를 공개하지 않음 • 보안업체에서 운영하는 악성파일 및 침해지표(IoC) 공유 플랫폼
하	• 티스토리 블로그, X 등	• 익명의 분석가가 업로드하는 게시물

둘째, 수집된 악성 바로가기 파일 샘플은 Microsoft에서 공식적으로 공개한 바로가기 파일 포맷 문서[4]를 기반으로 내부 구조를 파싱해 분석했다. 파싱 대상은 ShellLinkHeader, LinkTargetIDList, LinkInfo, StringData, ExtraData 등 주요 구조 필드를 포함하며, 분석 효율성을 높이기 위해 모든 파싱 결과는 정형화된 JSON 형태로 변환해 저장하고 데이터베이스화했다. 특히 LinkTargetIDList 및 ExtraData 영역은 Microsoft 공식 문서에 세부 구조가 일부 항목만 간략히 기술되어 있어, 관련 포맷 문서와 오픈소스 커뮤니티 자료 등을 종합적으로 참고해 구조를 세부적으로 파악하고 직접 파서 로직을 구현했다. 이 과정에서 오픈소스 도구인 LnkParser3[5]을 기반으로 하되, 누락된 필드를 처리하고 확장 블록 해석 기능을 보완해 자동화 된 파싱 및 정제 과정을 수행할 수 있도록 자체적으로 Python 파서를 개발했다.

[4] Microsoft, "[MS-SHLLINK]: Shell Link (.LNK) Binary File Format", 2025-06-10, https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-shllink/16cb4ca1-9339-4d0c-a68d-bf1d6cc0f943

[5] https://github.com/Matmaus/LnkParse3

셋째, 자동화 된 파싱 도구를 통해 산출된 데이터를 분석하고, 실제 바로가기 파일 구조 내에 저장된 데이터와 생성 방식의 원리를 바탕으로 공격자 식별에 활용 가능한 구조 항목을 선별했다. 예를 들어, LinkFlag의 구조, DriveSerialNumber, NTFS File Reference 정보, MAC 주소 등은 악성 바로가기 파일의 특징이나 공격자의 시스템 환경을 유추할 수 있는 주요 단서로 판단되어 분석 대상으로 채택되었다. 선정된 항목들은 구조 필드별로 정규화해 전체 샘플에 대해 비교 분석을 수행했으며 값의 반복 여부나 포맷의 유사성을 기준으로 유형을 분류했다. 특히 각 항목이 공격자 식별에 기여하는 정도에 따라 식별 신뢰도를 평가하고 이를 바탕으로 가중치를 설정해 구조 기반 분석 방법론의 정량적 틀을 수립했다. 또한, 구조 항목 간의 상호 관계와 다수의 샘플 간의 연관성을 시각적으로 확인하기 위해 Obsidian의 그래프 뷰 기능을 활용했다. 이를 통해 악성 바로가기 샘플 간 공유되는 구조적 특징이나 고유 식별자의 연결 형태를 직관적으로 파악할 수 있도록 했다.

마지막으로, 앞서 도출한 구조 기반 분석 방법론의 유효성을 검증하기 위해 북한 배후 공격 그룹이 사용한 것으로 공개된 일부 악성 바로가기 샘플을 표준 검증 데이터셋으로 선정했다. 선정된 샘플에 대해 제안된 분석 절차를 적용해 주요 구조 정보를 추출 및 분류하고, 항목별 가중치 기반 분석을 수행함으로써 공격 그룹별 제작 패턴 및 환경적 특성을 도출할 수 있는지를 확인했다.

2) 연구의 범위

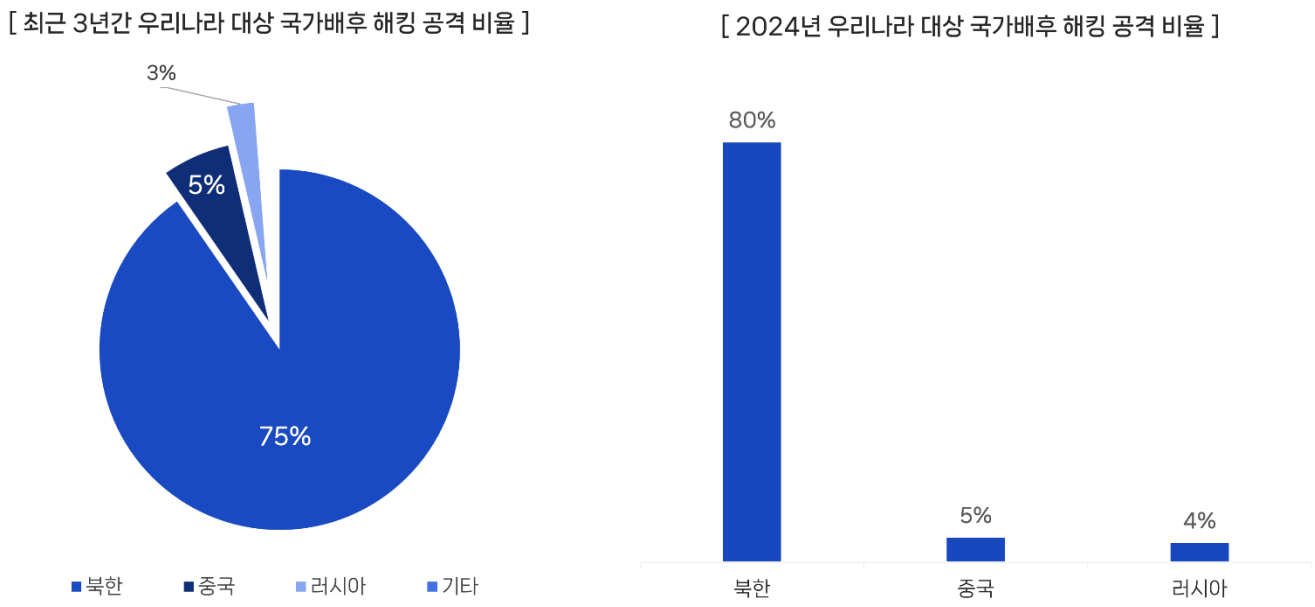
본 연구는 악성 바로가기 파일의 내부 구조에 기반한 공격자 식별 가능성 검증을 목적으로 한다. 우선 분석 대상은 공개 위협 인텔리전스 플랫폼과 보안 업체 블로그 등에 공개된 악성 바로가기 파일로 한정했으며 총 1,135개의 샘플을 수집해 연구에 활용했다. 수집 기준은 샘플에 포함된 파일 구조가 온전히 보존되어 있어 정적 분석이 가능한 경우로 제안하였고 정상 바로가기 파일은 비교 대상으로 포함하지 않았다.

또한, 본 연구는 바로가기 파일 자체에 내재된 구조적 정보만을 기반으로 공격자 특성을 식별하기 위한 연구로써, 명령줄 인자를 분석해 확보할 수 있는 추가 페이로드를 다운로드하는 서버 정보, 추가 악성파일, 외부 명령 제어 인프라 등을 분석하는 것은 범위에서 제외했다. 파일 실행 시 생성되는 행위 기반 정보나 악성코드의 로직 자체는 분석 대상에 포함되지 않는다. 다만, 악성 바로가기 파일 샘플을 수집할 당시 보안 블로그나 위협 인텔리전스 플랫폼에서 악성 페이로드 계열이 언급된 경우 공격 그룹 식별을 위한 보조 지표로 메타데이터화해 저장하여 추후 구조 기반 특징 분석 시 특정 공격 그룹과의 연관성을 해석하는 참고자료로 활용했다.

제안한 분석 방법론의 유효성을 검증하기 위해서는 북한 배후 공격 그룹이 사용한 것으로 보고된 약 200개의 악성 바로가기 샘플을 검증 데이터셋으로 선정했다. 북한 공격 그룹을 검증 데이터셋으로 선정한 이유는 해당 국가가 우리나라를 주요 공격 대상으로 삼고 있으며 실제 공격 비중이 압도적으로 높기 때문이다. 국가정보원이 2025년 6월 발표한 통계에 따르면, 최근 3년간 우리나라를 대상으로 수행된 국가 배후 사이버 공격 중 약 75%가 북한 소행으로 확인되었고 2024년 국내 해킹 피해 건수의 약 80% 또한 북한 배후로 지목될 정도로 그 활동 빈도가 높은 것으로 보고[6]되었다.

---

[6] 한국무역협회, ""사방이 적" 북·중·러發 해킹 82%…새 정부, 사이버안보 컨트롤타워 세워야", 2025-06-04, https://www.kita.net/board/totalTradeNews/totalTradeNewsDetail.do?no=92368&siteId=1



[그림 4] 우리나라 대상 국가 배후 해킹 공격 비율

또한, 북한 공격 그룹은 국내외 보안업체 및 사이버 위협 인텔리전스 업체들에 의해 지속적으로 분석되어 온 대상이며 축적된 연구 자료와 공개된 샘플이 비교적 풍부하여 구조적 특징을 공격 그룹별로 비교 및 도출하기 용이하다. 따라서, 본 연구는 제안한 분석 절차를 적용해 검증 데이터셋에서 구조적 특징을 추출 및 분류하고 이를 통해 제안한 방법론이 실제 공격 그룹 식별에 유효하게 작동하는지를 평가했다.

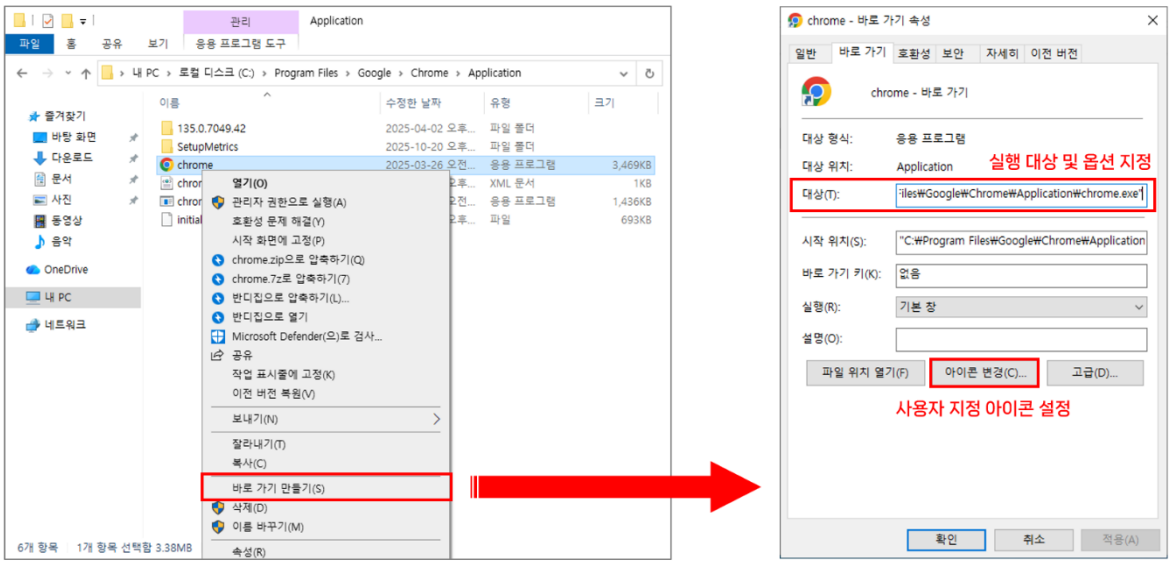
2. 배경지식

2.1. 바로가기 파일이란?

바로가기 파일은 Windows 운영체제에서 사용자가 자주 사용하는 파일, 폴더, 응용 프로그램, 네트워크 경로, 웹 링크 등에 빠르게 접근할 수 있도록 설계된 기능으로, 해당 파일은 '.lnk' 확장자를 가진다. 일반적으로 사용자는 바탕화면, 시작 메뉴, 작업 표시줄 등에서 바로가기 아이콘을 통해 원하는 리소스에 접근하며 이는 실제 파일이나 폴더를 직접 복사하거나 이동하지 않고도 실행을 가능하게 한다.

바로가기 파일은 실행 대상의 경로, 아이콘, 실행 방식, 작업 디렉터리, 설명 텍스트 등 다양한 정보를 포함할 수 있으며, 운영체제는 이를 참조해 대상 리소스를 자동으로 호출한다. 이처럼 바로가기 파일은 원본 파일에 대한 간접적인 참조의 역할을 수행하는 시스템 구성 요소로서, 사용자의 편의성을 높이기 위해 도입된 기능이다.

Windows에서는 이러한 바로가기 파일을 생성할 수 있도록 '마우스 우클릭' → '바로가기 파일 만들기' 기능을 제공하며, 생성된 바로가기 파일은 사용자 지정 아이콘, 실행 옵션 등을 자유롭게 설정할 수 있다.



[그림 5] Windows에서 바로가기 파일 생성 방법

이와 같이 바로가기 파일은 운영체제에서 매우 일반적으로 사용되는 구성요소이며, 파일 자체에는 원본 리소스를 직접 포함하지 않으면서도 실행 대상에 대한 세부 정보를 구조적으로 저장하고 있는 것이 특징이다. 이러한 구조적 특성은 시스템 내부 경로, 드라이브 정보, 생성 환경 등의 정황 정보가 파일 자체에 포함될 수 있도록 하며, 이는 이후의 분석 과정에서 유용한 단서로 활용될 수 있다.

2.2. 바로가기 파일의 구조

바로가기 파일의 구조인 Shell Link 이진 형식(Shell Link Binary File Format)은 “ShellLinkHeader”, “LinkTargetIDList”, “LinkInfo”, “StringData”, “ExtraData”로 5개의 영역이 순서대로 구성되며, 각 영역은 블록 단위로 구조화 되어있다. 첫 번째 영역인 “ShellLinkHeader”는 바로가기 파일 구조에서 반드시 존재해야 하는 영역이며, 다른 영역은 “ShellLinkHeader”의 “LinkFlags” 값에 의해 선택적으로 존재한다.

바로가기 파일 내 영역별 주요 데이터의 의미는 다음과 같다.

[표 2] 바로가기 파일 내 구조별 주요 데이터 의미

영역	필수 여부	설명
ShellLinkHeader	필수	• 바로가기 파일의 크기, 생성·수정·접근 시각과 대상 파일의 속성 등 바로가기 파일의 전반적인 내용을 저장
LinkTargetIDList	선택	• 바로가기 파일이 참조하는 실제 대상 파일 및 폴더의 경로 정보를 리스트 형식으로 저장
LinkInfo	선택	• 바로가기 대상이 위치한 드라이브, 로컬 혹은 네트워크 경로 등 바로가기 파일이 생성된 환경 정보를 저장
StringData	선택	• 바로가기 파일의 작업 수행 디렉터리 및 아이콘 경로 정보와 실행 시 수행되는 명령 인자, 바로가기 파일 설명 등의 문자열 정보를 저장
ExtraData	선택	• 바로가기 파일이 생성된 시스템 환경 및 사용자 관련 데이터 등 확장 정보를 저장

1) ShellLinkHeader

ShellLinkHeader는 바로가기 파일의 가장 처음에 위치하는 필수 영역으로서 바로가기 파일 전체의 기본 정보를 저장한다. 바로가기 파일의 크기, 식별자, LinkFlags 등 바로가기 파일의 정보와 바로가기 대상의 속성 값, 생성·수정·접근 시각 등의 기본 메타데이터도 함께 포함하고 있다.

Windows 운영체제는 ShellLinkHeader의 데이터를 해석해 바로가기 파일의 전체 구조를 식별하고 바로가기 대상에 대한 기본적인 정보를 확인한다. ShellLinkHeader의 상세 구조는 다음과 같다. ShellLinkHeader의 영역 중 추후 구조별 데이터 분석 및 특징 도출 단계에서 식별 데이터로 활용되는 필드는 별도의 색상으로 구분하여 표시하였다.

[표 3] ShellLinkHeader의 필드별 의미

범위	크기	필드	설명
0 – 3	4 bytes	HeaderSize	• 헤더 크기
4 – 19	16 bytes	LinkCLSID	• 클래스 식별자
20 – 23	4 bytes	LinkFlags	• 바로가기 파일 구조 정의

24 – 27	4 bytes	FileAttributes	• 바로가기 대상의 속성 정보
28 - 35	8 bytes	CreationTime	• 바로가기 대상의 생성 시간(UTC)
36 – 43	8 bytes	AccessTime	• 바로가기 대상의 접근 시간(UTC)
44 – 51	8 bytes	WriteTime	• 바로가기 대상의 수정 시간(UTC)
52 – 56	4 bytes	FileSize	• 바로가기 대상 파일의 크기
56 – 59	4 bytes	IconIndex	• 바로가기 파일의 아이콘 Index
60 – 63	4 bytes	ShowCommand	• 바로가기 대상의 실행 형태
64 - 65	2 bytes	HotKey	• 바로가기 대상의 실행 단축키
66 - 75	10 bytes	Reserved	• 예약된 영역(항상0)

HeaderSize

HeaderSize 값은 ShellLinkHeader의 크기를 나타내며, 항상 0x0000004C 값을 갖는다.

LinkCLSID

LinkCLSID는 클래스 식별자로 해당 파일이 Shell Link 이진 파일 형식의 바로가기 파일임을 명시적으로 식별하는 값이다. 해당 값은 항상 00021401-0000-0000-C000-000000000046 값을 가진다.

LinkFlags

LinkFlags는 바로가기 파일의 구조를 정의하는 부호 없는 정수 필드 값으로, ShellLinkHeader 영역 뒤에 이어지는 선택적 데이터 영역들의 존재 여부와 바로가기 파일의 특정 동작 방식을 지정한다. LinkFlags는 총 32개의 비트 값으로 구성되어 있으며, 이 중 7개의 비트는 사용되지 않는 비트 값으로 25개의 비트 값만 사용된다. LinkFlags의 각 비트별 상세 의미는 다음과 같다.

[표 4] LinkFlags별 상세 의미

비트	값	설명
1	HasLinkTargetIDList	• LinkTargetIDList 영역이 존재함
2	HasLinkInfo	• LinkInfo 영역이 존재함
3	HasName	• StringData 영역에 NameString 필드가 존재함
4	HasRelativePath	• StringData 영역에 RelativePath 필드가 존재함
5	HasWorkingDir	• StringData 영역에 WorkingDir 필드가 존재함
6	HasArguments	• StringData 영역에 CommandLineArguments 필드가 존재함
7	HasIconLocation	• StringData 영역에 IconLocation 필드가 존재함

8	IsUnicode	• 바로가기 파일에 유니코드 인코딩 문자열이 포함되었음[7]
9	ForceNoLinkInfo	• LinkInfo 영역이 있더라도 무시함
10	HasExpString	• ExtraData 영역에 EnvironmentVariableDataBlock 필드 값이 존재함
11	RunInSeparateProcess	• 바로가기 대상이 16bit 애플리케이션인 경우 별도의 가상머신에서 실행
12	UnUsed	• 사용되지 않는 비트 값
13	HasDarwinID	• ExtraData 영역에 DarwinDataBlock 필드가 존재함
14	RunAsUser	• 바로가기 대상 파일 실행 시 다른 사용자로 실행
15	HasExpIcon	• ExtraData 영역에 IconEnvironmentDataBlock 필드가 존재함
16	NoPidlAlias	• 경로가 IDList로 변환되어 파싱될 때 파일 시스템 위치는 Shell Namespace 내 경로로 표현
17	UnUsed	• 사용되지 않는 비트 값
18	RunWithShimLayer	• ExtraData 영역에 ShimDataBlock 필드가 존재함
19	ForceNoLinkTrack	• ExtraData 영역 내 TrackerDataBlock 필드 무시
20	EnableTargetMetadata	• 바로가기 대상의 메타데이터를 ExtraData 영역 중 PropertyStoreDataBlock 필드에 저장
21	DisableLinkPathTracking	• ExtraData 영역 중 EnvironmentVariableDataBlock 무시
22	DisableKnownFolderTracking	• ExtraData 영역 중 KnownFolderDataBlock 필드 혹은 SpecialFolderDataBlock 필드 무시
23	DisableKnownFolderAlias	• ExtraData 영역 중 KnownFolderDataBlock 필드가 있는 경우 알려진 폴더의 IDList는 별칭이 없는 형식으로 사용
24	AllowLinkToLink	• 다른 바로가기 대상을 참조하는 링크 생성이 활성화됨
25	UnaliasOnSave	• 링크 저장 시 대상 IDList가 알려진 폴더에 있는 경우, 별칭이 없는 폴더 경로나 원본 IDList 중 하나를 사용
26	PreferEnvironmentPath	• 대상 IDList를 저장하지 않고 ExtraData 영역 중 EnvironmentVariableDataBlock 필드에 지정된 환경 변수 경로를 사용해 바로가기 대상 참조
27	KeepLocalIDListForUNCTarget	• 바로가기 대상이 UNC 경로인 경우 ExtraData 영역 중 PropertyStoreDataBlock의 IDList 경로를 사용함
28	UnUsed	• 사용되지 않는 비트 값
29	UnUsed	• 사용되지 않는 비트 값
30	UnUsed	• 사용되지 않는 비트 값
31	UnUsed	• 사용되지 않는 비트 값
32	UnUsed	• 사용되지 않는 비트 값

[7] 해당 비트는 항상 활성화되어 있음

FileAttributes

FileAttributes는 바로가기 대상이 파일 시스템 항목일 경우, 해당 대상의 속성 정보를 저장한다. 이 필드는 바로가기 대상을 직접 사용할 수 없거나 대상 접근이 효율적이지 못한 경우 대체 정보로 활용될 수 있다. 단, 해당 값은 실제 바로가기 대상의 속성과 항상 일치하지 않을 수 있다. FileAttributes는 총 32개의 비트로 구성되며, 이 중 13개 비트를 제외한 나머지 비트는 사용되지 않는다. FileAttributes의 각 비트별 상세 의미는 다음과 같다.

[표 5] FileAttributes 비트별 상세 의미

비트	값	설명
1	FILE_ATTRIBUTE_READONLY	• 파일 또는 디렉터리가 읽기 전용임
2	FILE_ATTRIBUTE_HIDDEN	• 파일 또는 디렉터리가 숨겨져 있음
3	FILE_ATTRIBUTE_SYSTEM	• 파일 또는 디렉터리가 운영체제와 관련된 것임
4	Reserved	• 예약된 비트 값으로 사용되지 않는 값
5	FILE_ATTRIBUTE_DIRECTORY	• 바로가기 대상이 파일이 아닌 디렉터리
6	FILE_ATTRIBUTE_ARCHIVE	• 파일 또는 디렉터리가 아카이브 파일
7	Reserved	• 예약된 비트 값으로 사용되지 않는 값
8	FILE_ATTRIBUTE_NORMAL	• 파일 또는 디렉터리에 다른 플래그가 설정되지 않았음[8]
9	FILE_ATTRIBUTE_TEMPORARY	• 파일이 임시 저장소로 사용되고 있음
10	FILE_ATTRIBUTE_SPARSE_FILE	• 파일이 내용이 0으로 채워진 Sparse File임
11	FILE_ATTRIBUTE_REPARSE_POINT	• 파일 또는 디렉터리에 Reparse Point가 연결되어 있음
12	FILE_ATTRIBUTE_COMPRESSED	• 파일 또는 디렉터리가 압축되었음
13	FILE_ATTRIBUTE_OFFLINE	• 파일의 데이터가 즉시 사용 가능하지 않음
14	FILE_ATTRIBUTE_NOT_CONTENT_INDEXED	• 파일 내용을 검색 인덱싱 대상에서 제외
15	FILE_ATTRIBUTE_ENCRYPTED	• 파일 또는 디렉터리가 암호화되었음
16	UnUsed	• 사용되지 않는 비트 값
17	UnUsed	• 사용되지 않는 비트 값
18	UnUsed	• 사용되지 않는 비트 값
19	UnUsed	• 사용되지 않는 비트 값
20	UnUsed	• 사용되지 않는 비트 값
21	UnUsed	• 사용되지 않는 비트 값
22	UnUsed	• 사용되지 않는 비트 값
23	UnUsed	• 사용되지 않는 비트 값
24	UnUsed	• 사용되지 않는 비트 값

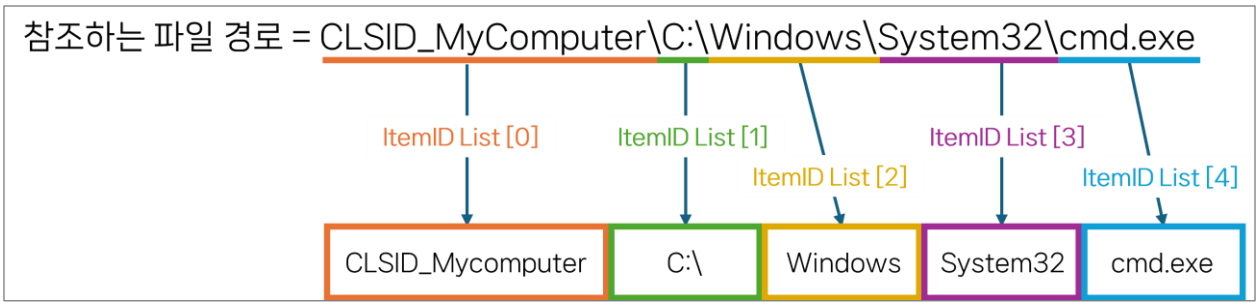
[8] 해당 비트가 1인 경우 다른 비트는 반드시 0이어야 함

2) LinkTargetIDList

LinkTargetIDList 영역은 바로가기 대상의 경로 정보를 리스트 형식으로 저장하며, LinkFlags 중 'HasLinkTargetIDList' 비트 값이 설정되어 있어야만 존재한다. LinkTargetIDList 영역은 IDList 구조체의 크기를 지정하는 2bytes 크기의 IDListSize와 ItemID List를 저장하는 가변 크기의 IDList로 구성되어 있다.

[표 6] LinkTargetIDList의 필드별 의미

범위	크기	필드	설명
0 – 1	2 bytes	IDListSize	• IDList 크기
2 -	가변	IDList	• ItemID List 저장



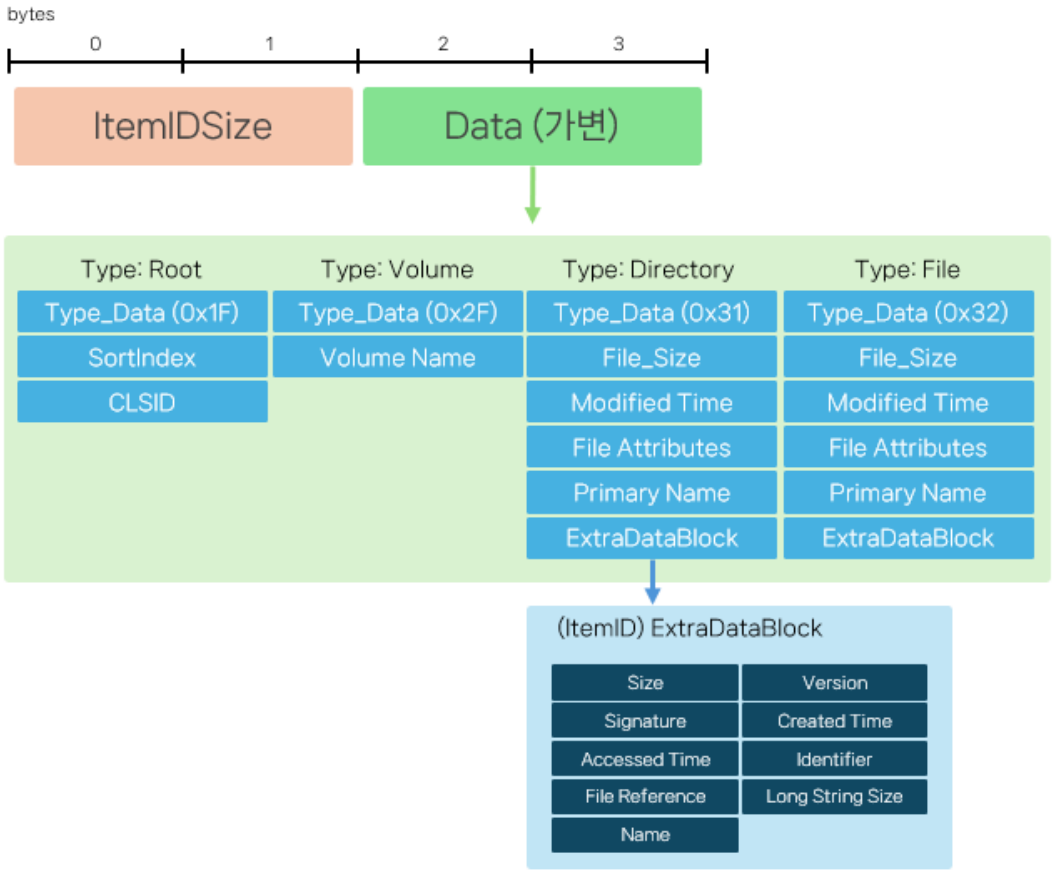
[그림 6] LinkTargetIDList에 저장되는 IDList 예시

각 IDList는 바로가기 대상의 경로를 단계별로 저장하며, 그 구조는 위 그림과 같이 여러 개의 ItemID 구조가 연속적으로 이어지는 리스트 형태로 구성된다. IDList는 가변 크기의 ItemIDList와 리스트의 끝을 나타내는 TerminalID로 구성된다. TerminalID는 항상 값이 0으로 설정되어 ItemID의 끝을 의미한다.

[표 7] LinkTargetIDList의 IDList 구조

범위	크기	필드	설명
0 -	가변	ItemIDList	• 리스트 형태의 ItemID
…	2 bytes	TerminalID	• ItemID의 끝

각 ItemID는 Windows Shell에서 사용하는 Shell Item 구조체와 동일한 형식으로 저장되어 있으며 하나의 ItemID는 드라이브, 폴더, 파일 등과 같이 일반적으로 경로를 구성하는 한 단계의 요소를 나타낸다. ItemID는 크기를 나타내는 ItemIDSize와 실제 데이터를 포함하는 가변 길이의 Data로 구성되어 있으며, ItemIDData 영역이 곧 Shell Item 구조체의 본체에 해당한다. 이때 Shell Item의 첫 바이트인 Class Type 값에 따라 해당 항목이 의미하는 대상의 유형을 구분할 수 있다.



[그림 7] Shell Item의 Class Type별 ItemIDData 영역 구조

Root ItemID

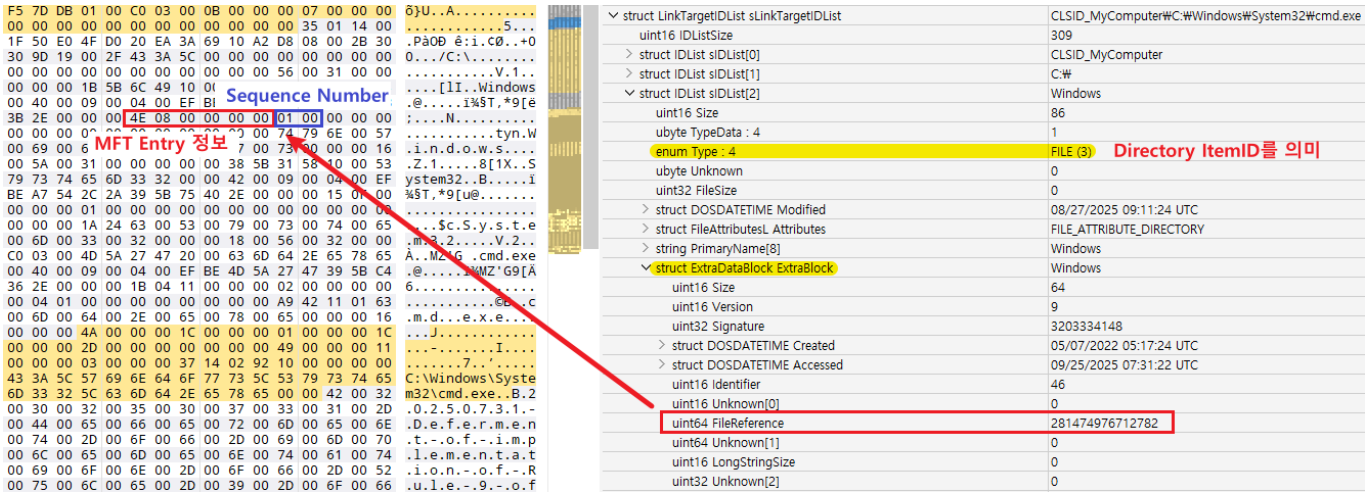
Root ItemID는 Type 값이 0x1F인 데이터 구조로, IDList 내에서 가장 먼저 위치한다. 이때 CLSID 필드는 항상 내 컴퓨터를 나타내는 값으로 20D04FE0-3AEA-1069-A2D8-08002B30309D 값으로 고정되어 있다.

VolumeItemID

Volume ItemID는 Type 값이 0x2F인 데이터 구조로, Volume의 문자열이 포함된다.

Directory ItemID

Directory ItemID는 Type 값이 0x3F인 데이터 구조로, 디렉터리의 수정 시간과 속성 정보를 포함한다. Directory ItemID 내의 ExtraDataBlock에는 디렉터리의 생성 시간과 접근 시간이 함께 저장된다. 또한 ExtraDataBlock에 포함된 File Reference 필드에는 하위 6비트는 MFT Entry 정보가, 상위 2비트는 Sequence Number가 저장된다.



[그림 8] Shell Item의 ExtraDataBlock의 FileReference 데이터

File ItmeID

File ItemID는 Type 값이 0x32인 데이터 구조로, Directory ItemID와 동일한 구조를 가진다.

3) LinkInfo

LinkInfo는 바로가기 대상이 원래 위치에서 발견되지 않을 경우, 대상을 식별하기 위해 필요한 바로가기 파일 생성 환경 정보를 저장한다. 이 구조는 LinkFlags 중 'HasLinkInfo' 비트 값이 설정된 경우에만 존재한다.

LinkInfo 영역은 데이터의 위치를 지정하는 오프셋 값과 실제 경로 및 볼륨 문자열 등을 저장하는 데이터 영역으로 구성된다. 이때 LinkInfo 내 각 데이터 구조는 해당 오프셋 값이 설정된 경우에만 존재한다.

LinkInfo의 상세 구조는 다음과 같다. LinkInfo의 영역 중 추후 구조별 데이터 분석 및 특징 도출 단계에서 식별 데이터로 활용되는 필드는 별도의 색상으로 구분하여 표시하였다.

[표 8] LinkInfo의 영역별 의미

범위	크기	필드	설명
0 – 3	4 bytes	LinkInfoSize	• LinkInfo 크기
4 – 7	4 bytes	LinkInfoHeaderSize	• LinkInfo 헤더의 크기
8 – 11	4 bytes	LinkInfoFlags	• LinkInfo 구조 정의
12 – 15	4 bytes	VolumeIDOffset	• VolumeID 필드의 위치
16 – 19	4 bytes	LocalBasePathOffset	• LocalBasePath 필드의 위치
20 – 23	4 bytes	CommonNetworkRelativeLinkOffset	• CommonNetworkRelativeLink 필드의 위치

24 – 27	4 bytes	CommonPathSuffixOffset	• CommonPathSuffix 필드의 위치
28 – 31	4 bytes	LocalBasePathOffsetUnicode	• LocalBasePathUnicode 필드의 위치
32 – 35	4 bytes	CommonPathSuffixOffsetUnicode	• CommonPathSuffixUnicode 필드의 위치
36 -	가변	VolumeID	• 바로가기 파일이 생성된 볼륨의 정보
…	가변	LocalBasePath	• 바로가기 대상의 로컬 경로
…	가변	CommonNetworkRelativeLink	• 바로가기 대상이 저장된 네트워크 경로
…	가변	CommonPathSuffix	• LocalBasePath 뒤에 연결되어 바로가기 대상의 전체 경로 구성
…	가변	LocalBasePathUnicode	• CommonPathSuffix와 결합해 바로가기 대상의 전체 경로 구성
…	가변	CommonPathSuffixUnicode	• LocalBasePathUnicode 뒤에 연결되어 바로가기 대상의 전체 경로 구성

LinkInfoFlags

LinkInfoFlags는 총 2개의 32bit 크기의 비트 값을 이용해 LinkInfo 영역의 구조를 정의한다.

[표 9] LinkInfoFlags 비트별 상세 의미

비트	값	설명
1	VolumeIDAndLocalBasePath	• LinkInfo 내 VolumeID 영역과 LocalBasePath 영역 존재 • 만약 LinkInfoHeaderSize가 0x24일 경우에는 LocalBasePath에 Unicode영역만 존재함
2	CommonNetworkRelativeLinkAndPathSuffix	• LinkInfo 내 CommonNetworkRelativeLink 영역 존재

VolumeID

VolumeID는 바로가기 파일이 생성된 볼륨의 정보 즉, 바로가기 대상이 존재하는 경로의 볼륨 정보를 저장하고 있다.

[표 10] VolumeID의 상세 구조

범위	크기	필드	설명
0 – 3	4 bytes	VolumeIDSize	• VolumeID의 크기
4 – 7	4 bytes	DriveType	• 바로가기 대상이 존재하는 경로의 드라이브 형식
8 – 11	4 bytes	DriveSerialNumber	• 바로가기 대상이 존재하는 경로의 드라이브 시리얼 번호
12 – 15	4 bytes	VolumeLabelOffset	• 볼륨 레이블 위치
16 – 19	4 bytes	VolumeLabelOffsetUnicode	• 유니코드 형식의 볼륨 레이블 위치
20 –	가변	Data	• 볼륨 레이블의 데이터

CommonNetworkRelativeLink

CommonNetworkRelativeLink는 바로가기 대상이 저장된 네트워크 경로에 대한 정보를 저장하며, 매핑된 드라이브 문자와 UNC(Universal Naming Convention) 경로 접두사 등의 정보를 포함한다. 이를 통해 사용자별로 네트워크 드라이브 문자가 다르거나 변경되더라도 바로가기 파일이 대상 경로를 올바르게 인식할 수 있도록 한다.

[표 11] CommonNetworkRelativeLink의 상세 구조

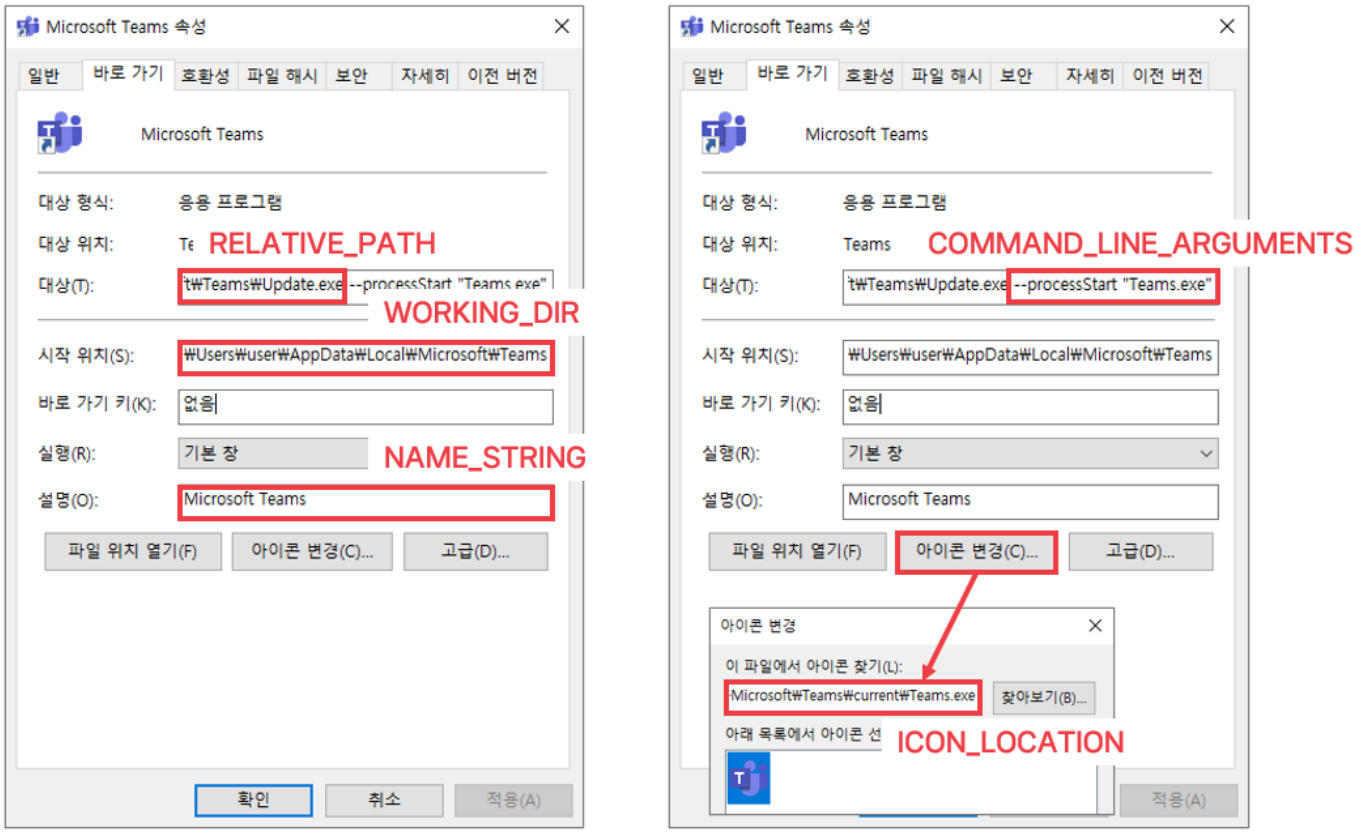
범위	크기	필드	설명
0 – 3	4 bytes	CommonNetworkRelativeLinkSize	• CommonNetworkRelativeLink의 크기
4 – 7	4 bytes	CommonNetworkRelativeLinkFlags	• CommonNetworkRelativeLink의 구조 정의
8 – 11	4 bytes	NetNameOffset	• NetName의 위치
12 – 15	4 bytes	DeviceNameOffset	• DeviceName의 위치
16 – 19	4 bytes	NetworkProviderType	• 네트워크 제공자의 유형
20 – 21	4 bytes	NetNameOffsetUnicode	• NetNameUnicode의 위치
22 – 25	4 bytes	DeviceNameOffsetUnicode	• DeviceNameUnicode의 위치
26 -	가변	NetName	• 바로가기 대상이 위치한 공유 폴더 경로
…	가변	DeviceName	• 매핑된 드라이브 문자열
…	가변	NetNameUnicode	• 유니코드 형태의 공유 폴더 경로
…	가변	DeviceNameUnicode	• 유니코드 형태의 매핑된 드라이브 문자열

4) StringData

StringData는 바로가기 파일의 작업 수행 디렉터리, 아이콘 경로 정보와 실행 시 수행되는 명령 인자와 같은 바로가기 파일의 문자열 정보를 저장하며, LinkFlags의 특정 비트 값 설정 여부에 따라 StringData의 구조 존재 여부가 결정된다.

[표 12] StringData 영역별 의미

번호	필드	설명	관련된 LinkFlags
1	NAME_STRING	• 바로가기 파일에 대한 설명	HasName
2	RELATIVE_PATH	• 바로가기 대상의 경로 표시	HasRelatviePath
3	WORKING_DIR	• 바로가기 파일의 작업 수행 디렉터리	HasWorkingDir
4	COMMAND_LINE_ARGUMENTS	• 실행 시 수행되는 명령 인자	HasArguments
5	ICON_LOCATION	• 바로가기 파일의 아이콘 경로	HasIconLocation



[그림 9] 바로가기 파일 속성 내 StringData 영역 해당 값

- NAME_STRING

LinkFlags의 HasName 비트 값이 설정되어 있어야만 존재하는 영역으로 바로가기 파일의 속성 정보에서 '설명' 값에 해당한다.

- COMMAND_LINE_ARGUMENTS

LinkFlags의 HasArguments 비트 값이 설정되어 있어야만 존재하는 영역으로, 바로가기 파일 실행 시 수행되는 명령줄 인자를 포함한다.

5) ExtraData

ExtraData 영역은 선택적으로 존재하는 영역으로, 바로가기 파일의 주요 속성 정보가 아닌 시스템 환경, 사용자 정보 등과 같이 바로가기 대상의 부가적인 속성 정보를 저장한다. 이 영역은 다른 구조와 달리 내부에 포함되는 하위 데이터 영역들이 고정된 순서로 저장되지 않으며, 생성 시점에 삽입된 순서대로 임의적으로 배치된다.

ExtraData에 포함되는 영역들의 상세 의미는 다음과 같다. ExtraData의 영역 중 추후 구조별 데이터 분석 및 특징 도출 단계에서 식별 데이터로 활용되는 필드는 별도의 색상으로 구분하여 표시하였다.

[표 13] ExtraData에 포함되는 데이터 영역 목록

번호	영역	시그니처	설명
1	EnvironmentVariableDataBlock	0xA0000001	• 바로가기 대상이 해당 환경 변수와 연결된 경로를 참조할 때, 환경 변수 정보의 경로를 지정함
2	ConsoleDataBlock	0xA0000002	• 콘솔 기반 프로그램의 바로가기 파일에 대한 설정을 저장함
3	TrackerDataBlock	0xA0000003	• 바로가기 대상의 경로가 변경되었을 때, Windows의 Distributed Link Tracking 서비스를 이용해 찾을 수 있는 정보를 저장함
4	ConsoleFEDataBlock	0xA0000004	• 바로가기 대상이 콘솔에서 실행되는 애플리케이션일 경우, 어떤 글꼴을 따를지 지정함
5	SpecialFolderDataBlock	0xA0000005	• 내 컴퓨터, 휴지통 등과 같은 특수 폴더를 가리킬 때 사용됨
6	DarwinDataBlock	0xA0000006	• 바로가기 대상이 설치되어 있지 않을 때 WindowsInstaller 기술과 관련된 정보를 저장함
7	IconEnvironmentDataBlock	0xA0000007	• 아이콘 경로를 지정하며, 시스템 마다 실제 경로가 다르더라도 환경 변수를 통해 동일한 아이콘 위치를 식별할 수 있도록 함
8	ShimDataBlock	0xA0000008	• Windows 구 버전에서 현재 버전으로의 호환성 정보를 저장함
9	PropertyStoreDataBlock	0xA0000009	• 바로가기 파일 내 추가적인 속성(ex. 사용자 SID)을 저장함
10	KnownFolderDataBlock	0xA000000B	• 다운로드, 음악, 사진, 바탕화면 등 알려진 폴더를 가리킬 때 사용됨
11	VistaAndAboveIDListDataBlock	0xA000000C	• Windows Vista 이상의 시스템에서 생성된 바로가기에 포함되며, 바로가기 대상의 IDList를 저장함

TrackerDataBlock

TrackerDataBlock는 바로가기 대상이 원래 위치에서 찾을 수 없을 때 해당 바로가기 대상을 추적하기 위해 사용되는 데이터를 정의하는 구조체이다. 이는 Windows의 분산 링크 추적 서비스(Distributed Link Tracking, DLT)[9]에 필요한 데이터를 저장한다.

[표 14] TrackerDataBlock의 상세 구조

범위	크기	필드	설명
0 – 3	4 bytes	BlockSize	• TrackerDataBlock의 크기
4 – 7	4 bytes	BlockSignature	• TrackerDataBlock의 시그니처 값
8 – 11	4 bytes	Length	• Length 필드 이후 데이터의 크기(항상 0x00000058 임)
12 – 15	4 bytes	Version	• 반드시 0x00000000 값을 가짐
16 – 31	16 bytes	MachineID	• 바로가기 파일이 생성된 호스트 명

[9] 분산 링크 추적 서비스(Distributed Link Tracking, DLT) : NTFS 파일 시스템에서 파일이나 폴더의 위치가 바뀌었을 때 링크를 자동 추적하고 업데이트하는 Windows의 기능

32 – 63	32 bytes	Droid	• 바로가기 대상의 UUID 값 (File/Volume)
64 – 95	32 bytes	DroidBirth	• 바로가기 대상이 처음 생성됐을 때의 UUID 값 (File/Volume)

PropertyStoreDataBlock

PropertyStoreDataBlock은 바로가기 파일의 추가적인 데이터를 저장할 수 있도록 하는 속성을 정의하는 구조체로, Windows Vista 이후 운영체제에서 속성 기반의 메타데이터를 담기 위해 추가된 구조이다.

[표 15] PropertyStoreDataBlock의 상세 구조

범위	크기	필드	설명
0 – 3	4 bytes	BlockSize	• PropertyStoreDataBlock의 크기
4 – 7	4 bytes	BlockSignature	• PropertyStoreDataBlock의 시그니처 값
8 -	가변	PropertyStore	• 바로가기 파일의 추가적인 속성을 저장하는 구조체

이 블록은 PROPERTY_STORE 개체 집합을 가지며, 각각의 Store는 다음과 같은 정보를 가진다.

[표 16] PropertyStore 개체의 정보

번호	값	설명
1	format_id	• 어떤 속성 그룹인지 나타내는 GUID
2	serialized_property_values	• 실제 속성 값 목록
3	version	• 보통 0x53505331로 고정
4	storage_size	• 헤더를 포함한 저장된 전체 크기

이때, format_id가 "46588AE2-4CBC-4338-BBFC-139326986DCE" 값을 갖는 경우 User 속성을 의미하며, 이때 속성 값 목록에는 사용자의 SID 값이 저장된다.


```
{
  "format_id": "46588AE2-4CBC-4338-BBFC-139326986DCE",
  "serialized_property_values": [
    {
      "id": 4,
      "value": "S-1-5-21-2446330629-4082116464-2183459656-1000",
      "value_size": 113,
      "value_type": "VT_LPWSTR"
    }
  ],
  "version": "0x53505331"
}
```


[그림 10] PROPERTY_STORE 개체 예시

2.3. 악성 바로가기 파일 특징

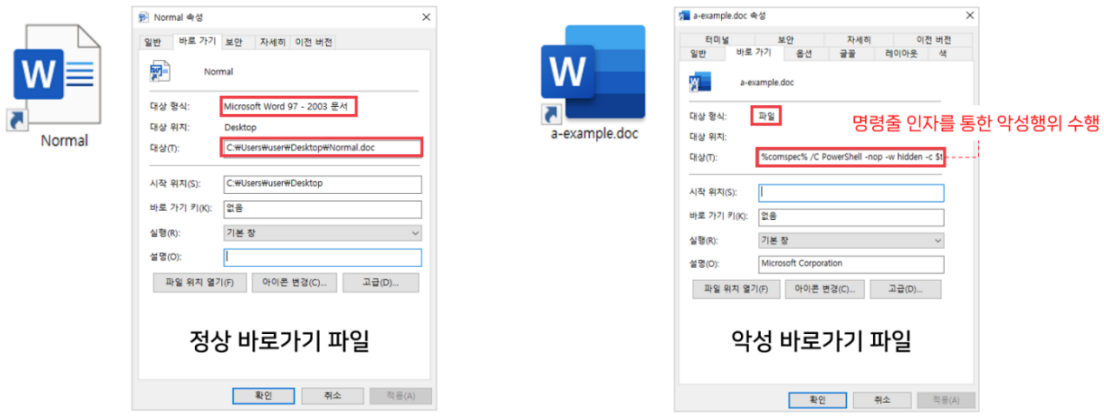
바로가기 파일은 본래 사용자의 편의를 위해 설계된 기능이지만, 파일 내부에 명령줄 인자나 스크립트 호출 구문을 삽입할 수 있다는 점을 악용해 악성 행위 수행에도 활용될 수 있다. 바로가기 파일은 외형상 일반 문서나 폴더로 위장되기 쉽고, 실행 시 사용자의 인지 없이 시스템 명령어를 수행하도록 구성할 수 있어 최근 수년간 다양한 공격 캠페인에서 초기 침투 수단으로 활용되고 있다.

1) 악성 명령 인자 삽입 및 실행 구조

일반적인 바로가기 파일은 사용자가 자주 사용하는 문서.이미지.응용 프로그램 등의 실행을 편리하기 위해 만들어진 반면, 악성파일은 겉보기에는 일반 파일을 실행하는 것처럼 위장되어 있지만, 실제로는 명령줄 인자를 통해 공격자가 의도한 악성 행위를 수행하도록 설계되어 있다. 다만, 바로가기 파일의 대상이 되는 경로가 반드시 정상 파일과 악성 파일을 구분하는 기준이 되는 것은 아니다. 일반적인 상황에서도 cmd.exe, powershell.exe, mshta.exe 등과 같은 시스템 프로그램에 대한 바로가기가 생성될 수 있기 때문이다. 다만, 악성 바로가기 파일은 명령 실행이 가능한 시스템을 대상으로 지정한 뒤 그 실행 인자에 악성 스크립트 다운로드, 외부 서버 통신, 파일 생성 등의 명령을 포함시키는 방식으로 악성 행위를 수행한다는 점에서 정상 파일과 구분된다.

[표 17] 정상 바로가기 파일과 악성 바로가기 파일의 차이

구분		설명
정상파일	바로가기 대상	• 일반 파일 또는 프로그램 (ex: MS Office 문서, 한컴 오피스 문서, 응용 프로그램)
	바로가기 파일의 아이콘	• 실제 가리키고 있는 대상의 아이콘
	바로가기 대상 경로	• 사용자가 실행하려는 대상 리소스 경로
악성파일	바로가기 대상	• cmd.exe, powershell.exe, mshta.exe 등과 같은 명령 실행이 가능한 시스템 프로그램을 대상으로 지정하고, 명령줄 인자를 통해 악성 행위를 수행하도록 구성
	바로가기 파일의 아이콘	• 이중 확장자의 아이콘
	바로가기 대상 경로	• 정상 경로처럼 보이나 실제로 명령줄 인자를 포함해 악성 명령 실행



[그림 11] 정상 바로가기 파일과 악성 바로가기 파일의 차이

위와 같이 삽입되는 명령줄 인자의 악성 스크립트는 StringData 영역의 'COMMAND_LINE_ARGUMENTS' 필드에 저장된다. 해당 필드는 글자 수나 용량에 제한이 없어 공격자가 원하는만큼 복잡한 명령을 추가할 수 있으며, 결과적으로 별도의 실행 파일 없이도 악성 행위를 수행할 수 있다. 예를 들어, 다음과 같은 명령줄 인자를 포함할 수 있다.

```
powershell -ExecutionPolicy Bypass -WindowStyle Hidden -Command "IEX (New-Object Net.WebClient).DownloadString('http://malicious[.]site/payload.ps1')"
```

이 경우 사용자는 단순한 문서 아이콘을 클릭했다고 생각하지만 실제로는 PowerShell을 통해 공격자가 지정한 스크립트가 실행되어 외부 서버의 스크립트가 다운로드 및 실행된다. 이러한 악성 행위는 단일 명령으로 종료되지 않고 여러 단계에 걸쳐 수행되거나 서로 조합되어 동작하기도 한다.

[표 18] 악성 바로가기 파일 스크립트의 주요 동작 형태

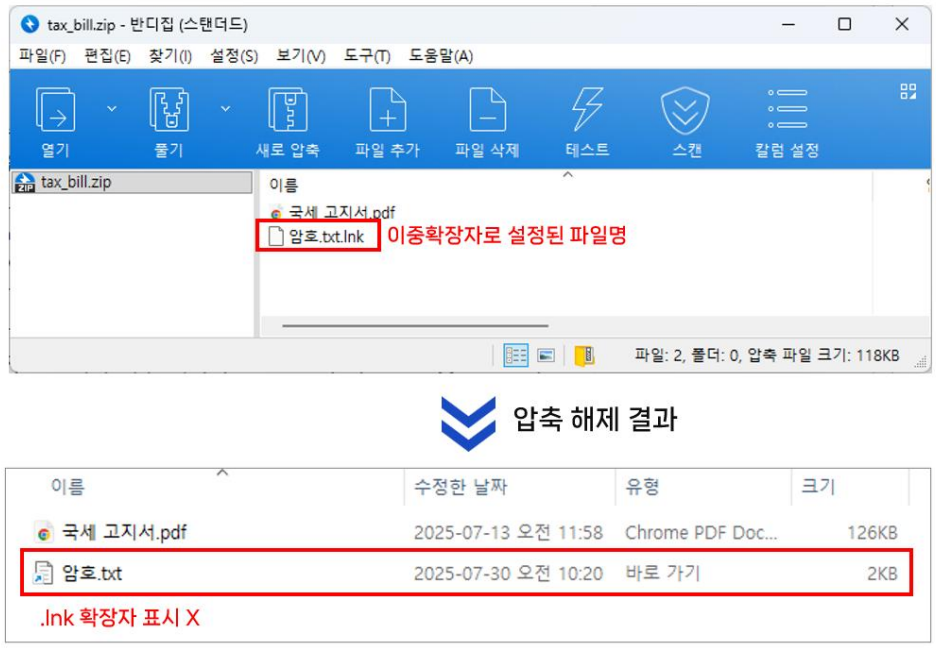
번호	동작 형태	설명
1	공격자 명령 제어 서버와의 통신	• 파일 실행으로 명령어가 수행되면 공격자 서버와 통신하여 원격 명령을 수신하거나 수집 데이터를 전송함
2	공격자가 활용하는 정상 서비스 접속	• Google Drive, Dropbox, MEGA, Github 등 외부 서비스의 접속 URL과 접근 토큰을 통해 추가 페이로드를 다운로드 및 실행함
3	내부 오버레이/데이터 블록 파싱	• 바로가기 파일 내부의 오버레이 또는 데이터 블록에 포함된 페이로드를 파싱하여 추가 악성파일을 생성 및 실행

명령줄 인자에 포함된 명령은 가장 먼저 실행되며 이후의 악성 행위의 흐름은 고정되지 않는다. 예를 들면, 내부 페이로드를 먼저 실행한 뒤 공격자 서버와 접속을 시도하거나, 반대로 공격자 서버 또는 정상 서비스에서 추가 페이로드를 다운로드한 후 내부 명령을 수행하는 방식 등 다양한 조합으로 동작할 수 있다. 이 과정에서 공격자가 활용한 서버나 서비스가 비활성화되면 분석가는 이후 행위를 분석하기 위한 페이로드를 확보하기 어려워지며, 결과적으로 행위 기반 분석이나 공격자 식별에 어려움이 발생한다.

2) 위장을 위한 사회공학적 기법

또한, 이처럼 악성 명령 인자가 실행되기 위해서는 사용자가 해당 파일을 실행하도록 유도하는 과정이 필수적이다. 이를 위해 공격자는 다양한 사회공학적 기법을 활용한다. 대표적인 방법으로는 아이콘을 변경해 정상적인 문서 파일처럼 보이게 하는 방식이 있다. 예를 들어, Word, Excel, PDF 문서 또는 폴더 아이콘 등으로 위장하면 사용자는 해당 파일이 일반적인 업무 문서라고 인식하고 실행하기 쉽다.

또한, Windows 탐색기에서 '.lnk' 확장자를 기본적으로 표시하지 않는 점을 악용해 파일명을 'document.pdf.lnk'와 같이 이중 확장자 형태로 설정하는 기법도 자주 사용된다. 이 경우 사용자에게는 'document.pdf'처럼 보이기 때문에 실제로는 바로가기 파일임에도 불구하고 일반 문서 파일로 오인할 가능성이 높다.



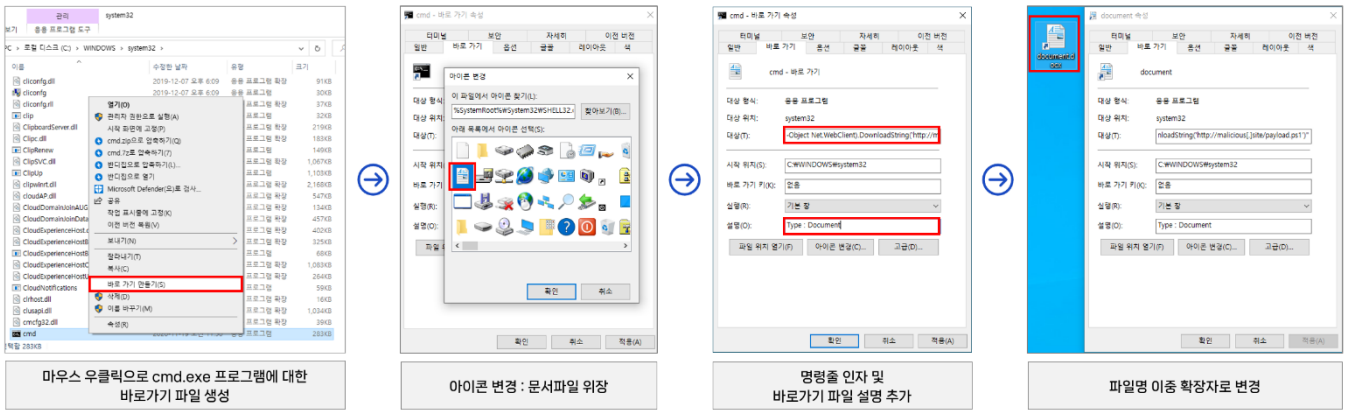
[그림 12] 이중 확장자 설정으로 정상 파일로 위장

이러한 사회공학적 기법은 악성 명령이 포함된 바로가기 파일을 사용자가 실행하도록 유도하는데 결정적인 역할을 하며, 그 결과 내부에 삽입된 악성 명령 인자가 실행되어 침해 행위가 시작된다.

3) 악성 바로가기 파일의 제작 방식

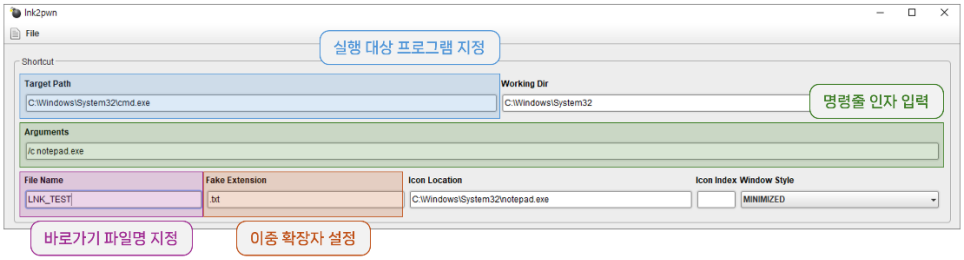
악성 바로가기 파일은 단순히 명령줄 인자를 삽입하거나 아이콘을 위장하는 행위만으로 제작되는 경우도 있지만, 실제 공격자는 보다 정교한 제작 도구와 자동화된 도구를 활용해 다수의 샘플을 일괄 생성하는 등의 방식으로 공격을 수행한다.

초기에는 공격자들이 다음과 같이 Windows 운영체제의 기본 기능을 활용해 GUI 환경에서 바로가기 파일을 직접 생성하고 속성 편집을 통해 실행 명령어를 삽입하거나 아이콘을 변경하는 방식으로 제작했다. 이러한 수작업 기법은 상대적으로 단순하지만, 사용자가 파일을 실행하도록 유도하기에는 충분히 효과적이기 때문에 지금도 여전히 활용되고 있다.



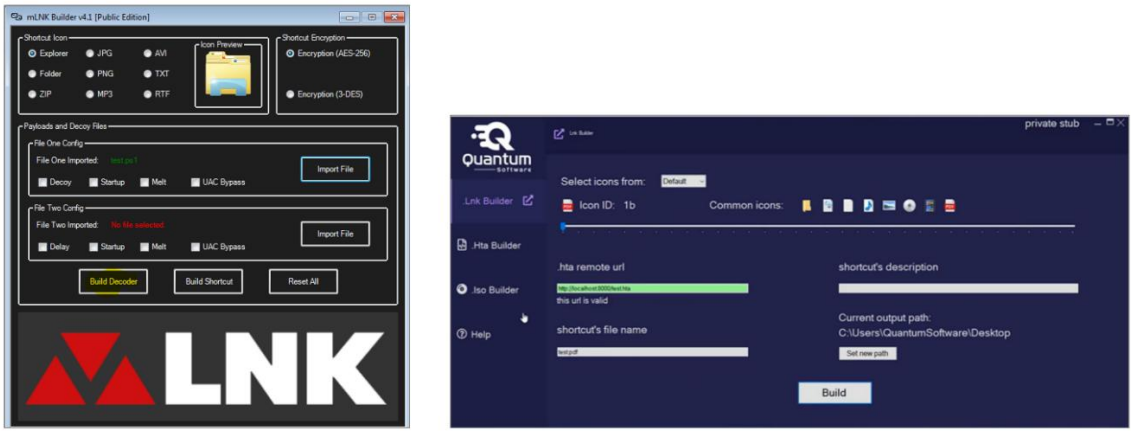
[그림 13] Windows의 기본 기능을 활용한 악성 바로가기 파일 제작

하지만, 최근 공격자들은 이런 수동 제작 단계를 넘어 자동화된 제작 환경과 전용 빌더 도구를 활용해 악성 바로가기 파일을 체계적으로 생성하고 있다. 예를 들어 lnk2pwn 10 , LNKUp 11 , Lnk_Generator 등과 같은 오픈소스 스크립트는 명령 인자, 아이콘, 파일 경로 등을 자동으로 삽입할 수 있으며 수십 개의 변형된 악성 바로가기 파일을 빠르게 생성할 수 있다.



[그림 14] lnk2pwn 프로그램을 이용한 악성 바로가기 파일 제작

더 나아가 다크 웹 등에서 유통되는 Quantum Builder, MLNK Builder 등의 상용 빌더 프로그램은 GUI 인터페이스를 통해 명령 인자 삽입, 아이콘 위장, UAC 우회, 탐지 회피 기능 등을 자동화한다. 이들 빌더는 .lnk, .hta, .vbs, .iso 등 다양한 형식으로 페이로드를 포장할 수 있으며 일부는 난독화나 암호화 기능을 포함해 백신 탐지를 어렵게 만든다. 특히 Quantum Builder는 AgentTelsaㅏ RedLine Stealer 등 정보탈취형 악성코드 유포에 활용된 사례가 보고[12]되었고, MLNK Builder는 4.2버전 이후 다중 페이로드 내장과 안티 바이러스 우회 기능이 강화되어 다크 웹에서 활발히 거래되고 있다[13].



[그림 15] 악성 바로가기 파일 제작을 위한 상용 빌더 프로그램

10 https://github.com/it-gorillaz/lnk2pwn
11 https://github.com/Plazmaz/LNKUp
12 CYBLE, "Quantum Software: LNK file-based builders growing in popularity", 2022-06-22, https://cyble.com/blog/quantum-software-lnk-file-based-builders-growing-in-popularity/
13 Rsecurity, "Shortcut-based (LNK) attacks delivering malicious code on the rise", 2022-07-17, https://www.resecurity.com/blog/article/shortcut-based-lnk-attacks-delivering-malicious-code-on-the-rise

이와 같은 자동화 빌더의 등장은 공격자가 악성 바로가기 파일을 단순한 사회공학적 미끼 수준에서 벗어나 대량 생산 가능한 공격 도구로 발전시켰음을 의미한다. 즉, 공격자는 한 번의 설정만으로 수십~수백 개의 악성 바로가기 파일을 일괄 생성하고 각 파일마다 다른 명령 인자, 아이콘, 파일명을 조합해 탐지를 회피하거나 캠페인 단위로 유포 전략을 세밀하게 수립할 수 있다.

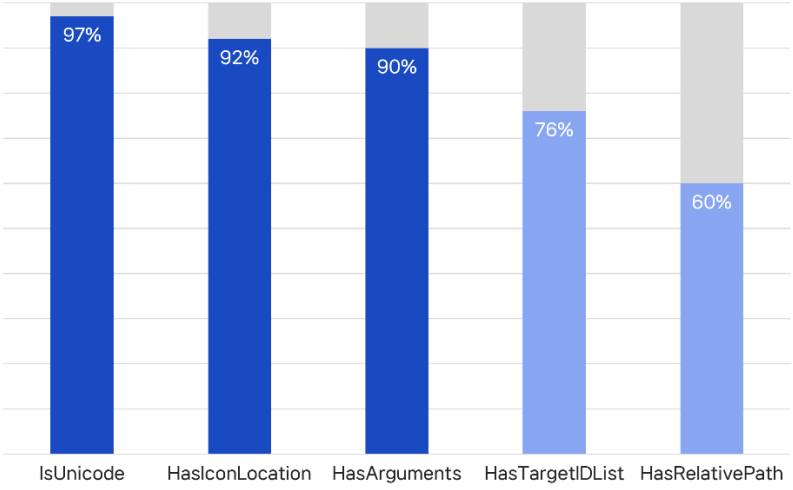
4) 수집된 악성 바로가기 샘플의 구조 통계 분석

앞서 제시한 악성 바로가기 파일의 구조적, 기능적 특징들이 실제 공격 사례에서도 공통적으로 나타나는지 확인하기 위해 본 연구에서는 총 1,135개의 악성 바로가기 파일 샘플을 수집하고 이에 대한 구조 분석을 수행했다. 분석은 각 샘플의 Shell Link Header에 포함된 LinkFlags 필드를 중심으로 진행되었으며, 해당 필드는 바로가기 파일 내에 어떤 구성 요소와 기능이 포함되어 있는지를 비트 단위로 명시하고 있어 이를 통해 명령줄 인자 삽입 여부, 아이콘 설정 여부 등 악성 파일 특성과 밀접하게 연관된 속성들을 확인할 수 있다.

LinkFlags 필드에 설정된 값을 통계적으로 분석한 결과, 가장 높은 비율로 확인된 항목은 'IsUnicode' 비트로 전체 샘플 중 약 97%에서 설정되어 있었다. 이는 바로가기 파일 내 문자열 데이터가 유니코드 형식으로 저장되어 있다는 것을 의미하며, Windows 2000 이후 운영체제에서 유니코드를 기본적으로 채택하고 있다는 점에서 제작 환경의 특성이 반영된 결과로 해석된다.

또한, 약 92%의 샘플에서 'HasIconLocation' 비트, 90%에서 HasArguments 비트가 설정되어 있었다. 이러한 값은 앞서 설명한 바와 같이 공격자가 아이콘을 정상 문서나 폴더처럼 위장하거나 명령줄 인자를 활용해 악성 행위를 수행하도록 설계하는 방식이 실제 샘플에서도 광범위하게 사용되고 있음을 시사한다.

이외에도 'HasLinkTargetIDList' 비트는 76%, 'HasRelativePath' 비트는 60%로 나타났다. 이는 악성 바로가기 파일이 실행 대상을 보다 정밀하게 설정하고 다양한 환경에서 실행할 수 있도록 상대 경로를 지정하거나 링크 대상 ID 명시 등의 구성을 적용하고 있음을 보여준다.



[그림 16] 악성 바로가기 샘플 내 LinkFlags 비트 상위 5개

3. 연구 내용

3.1. 바로가기 파일 구조 내 공격자 식별 데이터

본 연구는 기존의 공격자 프로파일링이 주로 침해지표(IoC)에 의존해왔다는 한계를 극복하고자 바로가기 파일의 구조적 특성을 기반으로 공격 주체를 식별하는 방법론을 제안한다. 이를 위해 수집한 악성 바로가기 파일 샘플의 구조를 분석하여 그 안에 포함된 공격자 환경 정보 또는 공격자 특성이 반영된 필드들을 주요 식별 데이터로 도출했다.

식별 데이터 선정은 두 가지 기준에 따라 수행되었다. 첫째, 해당 필드가 악성 바로가기 파일의 제작 방식과 관련된 특성을 담고 있는지 여부, 둘째, 악성 바로가기 파일이 생성된 시스템 환경의 정보를 담고 있는지 여부이다. 또한 각 필드가 자동 생성되는 시스템 정보인지, 공격자에 의해 변경 가능성이 있는지를 고려해 데이터의 신뢰도에 따라 가중치를 상, 중, 하로 구분해 부여했다.

[표 19] 주요 식별 데이터의 가중치 부여 기준

가중치	기준	설명
상	수정이 어려운 고유 식별 값	• 시스템에 의해 자동 부여되는 값으로, 사용자가 임의로 수정 불가능함
중	제한적 수정 가능	• 사용자가 직접 수정할 수는 없으나, 특정 도구를 사용하면 변경할 수 있음
하	수정이 용이함	• 별도의 도구 없이도 사용자가 임의로 수정할 수 있음

예를 들어, ExtraData 영역 내 TrackerDataBlock에 포함된 MachineID는 바로가기 파일이 생성된 환경의 호스트 이름을 의미하는 값으로, 일반 사용자도 손쉽게 변경할 수 있기에 가중치를 '하'로 분류했다. 반면, ExtraData 영역 내 PropertyStoreDataBlock의 PropertyStoreList에 포함된 사용자 SID는 운영체제 계정 생성 시 자동으로 부여되는 고유 식별자로, 사용자가 임의로 변경할 수 없어 가중치 '상'으로 분류했다.

[표 20] 바로가기 파일 구조 내 공격자 식별에 활용 가능한 데이터

가중치	식별 데이터	바로가기 구조 내 위치	의미
상	LinkFlags	• ShellLinkHeader 영역	• 바로가기 파일의 구조 정보
	FileReference	• LinkTargetIDList 영역 내 ItemID가 'Folder/File Entry'일 때 확장 영역의 FileReference	• 바로가기 파일이 가리키는 대상 프로그램이 위치했던 경로의 MFT Entry와 Sequence 번호
	SID	• ExtraData 영역 내 PropertyStoreDataBlock의 PropertyStoreList	• 바로가기 파일을 생성한 사용자 SID
중	VolumeID	• ExtraData 영역 내 TrackerDataBlock의 BirthDroidVolumeIdentifier 값	• 바로가기 파일이 가리키는 대상 프로그램이 원래 존재했던 볼륨의 정보
	FileID	• ExtraData 영역 내 TrackerDataBlock의 BirthDroidFileIdentifier 값	• 바로가기 파일이 가리키는 대상 프로그램이 처음 생성될 때의 고유 식별 값

	DriveSerialNumber	• LinkInfo 영역 내 VolumeID 구조의 DriveSerialNumber 값	• 바로가기 파일이 생성된 볼륨 정보
하	MAC 주소	• ExtraData 영역 내 TrackerDataBlock의 BirthDroidFileIdentifier	• 바로가기 파일이 생성된 시스템의 MAC 주소
	MachineID	• ExtraData 영역 내 TrackerDataBlock의 MachineID 값	• 바로가기 파일이 생성된 호스트명

이와 같이 식별 데이터를 가중치 기준에 따라 체계적으로 분류함으로써, 향후 구조 기반의 분석 과정에서 각 데이터의 신뢰도 및 공격자 추적 기여도를 보다 객관적으로 활용할 수 있다. 본 절에서 도출된 데이터는 후속 분석에서 실제 샘플에 적용되어 공격자 식별 가능성을 평가하는 기준 지표로 활용된다.

다음 항목에서는 앞서 식별 대상으로 제시한 주요 데이터 각각이 실제로 어떤 정보를 담고 있으며, 공격자 환경 또는 제작 방식의 특성과 어떤 방식으로 연관되는지를 구체적으로 설명하고자 한다. 특히 각 데이터가 위치한 구조적 영역, 생성 원리, 변경 가능성 등을 바탕으로 공격자 식별에 어떻게 기여할 수 있는지를 중심으로 기술한다.

1) LinkFlags

위 2.3절에서 언급된 다양한 방식과 도구를 통해 제작된 악성 바로가기 파일은 외형상 유사하게 보이지만 내부 구조의 세부 구현에는 차이가 존재한다. 특히 바로가기 파일의 핵심 구조 중 하나인 LinkFlags 필드는 각 바로가기 파일이 어떤 기능을 포함하고 있는지, 그리고 어떤 방식으로 제작되었는지를 보여주는 중요한 단서가 될 수 있다.

LinkFlags는 Shell Link Header 영역에 포함된 플래그 값으로, 바로가기 파일에 어떤 구성 요소를 포함하고 있는지를 비트 단위로 표현한다. 예를 들어, HasArguments 비트는 명령줄 인자가 삽입되었음을, HasIconLocation은 아이콘 정보가 포함되었음을 의미한다. LinkFlags는 총 32비트 중 25비트가 실제로 사용되며, 악성 바로가기 파일의 경우 이러한 플래그 값을 통해 공격자가 어떤 정보를 포함시켰는지, 제작 방식에 따라 어떤 특성이 반복되는지를 유추할 수 있다.

가령, 악성 바로가기 파일에 공통적으로 포함되는 HasArguments, HasIconLocation, IsUnicode 세 비트가 반드시 설정되고, 나머지 비트가 선택적으로 구성된다고 가정할 때 동일한 LinkFlags 조합이 반복될 확률은 약 0.00002384%로 매우 희박하다. 이는 LinkFlags 조합이 제작 도구의 고유한 패턴을 반영할 수 있음을 시사한다.

이에 따라, Windows의 기본 GUI 기능, 오픈소스 스크립트(lnk2pwn), 상용 빌더(Quantum Builder)를 각각 활용해 악성 바로가기 파일을 제작한 뒤 생성된 파일의 LinkFlags 값을 비교한 결과 다음과 같이 제작 방식에 따라 서로 다른 구조적 특징을 지님을 확인할 수 있었다.

[표 21] 악성 바로가기 파일 제작 방식에 따른 LinkFlags 구성 차이

도구명	유형	특징	LinkFlags
Windows GUI	수작업 방식	• 운영체제 기본 기능 사용	HasLinkTargetIDList, HasLinkInfo, HasRelativePath, HasWorkingDir, **HasArguments, HasIconLocation, IsUnicode**, HasExpIcon, EnableTargetMetaData
lnk2pwn	오픈소스 스크립트	• JAVA 또는 Python 기반 자동화 스크립트	HasLinkTargetIDList, HasLinkInfo, HasWorkingDir, **HasArguments, HasIconLocation, IsUnicode**
Quantum Builder	상용 빌더 도구	• GUID 제공, 다양한 형태의 페이로드 지원	HasLinkTargetIDList, HasLinkInfo, HasRelatvePath, **HasArguments, HasIconLocation, IsUnicode**

LinkFlags는 바로가기 파일 생성 시 자동으로 결정되며 공격자가 설정한 기능적 요소의 조합을 기반으로 하므로 제작 환경과 공격자의 선호 방식을 직간접적으로 반영한다. 특히 동일한 공격 그룹이 일관된 도구나 템플릿을 사용할 경우 특정 LinkFlags 조합이 반복적으로 나타날 수 있으며, 이는 공격자 프로파일링 및 도구 식별의 중요한 단서로 활용될 수 있다. 또한, 직접 수정이 제한되며 자동화된 도구에서 고정적으로 포함되는 조합이기 때문에 신뢰도가 높은 식별 정보로 간주할 수 있다.

2) FileReference

바로가기 파일이 참조하는 대상이 로컬에 존재하는 파일 또는 디렉터리인 경우, 해당 정보는 LinkTargetIDList 영역 내 ItemID 중 'Folder Entry' 또는 'File Entry' 구조로 저장된다. 이러한 구조에는 NTFS 파일 시스템에서 관리되는 고유 식별 정보인 File Reference 필드가 포함될 수 있다. 다만, 이 필드는 바로가기 파일이 생성된 운영체제의 버전과 파일 시스템 유형에 따라 포함 여부가 달라질 수 있으며[14] NTFS 기반 환경에서 생성된 경우에만 관찰된다.

FileReference는 NTFS 파일 시스템에서 파일이나 디렉터리를 고유하게 식별하기 위해 사용하는 정보로, 해당 항목의 MFT Entry 번호 6bytes와 Sequence 번호 2bytes를 조합한 총 8bytes 값이다. 즉, 이 값은 바로가기 파일이 생성될 당시 참조하던 파일 또는 폴더가 디스크 내 어디에 위치하고 있었는지를 직접적으로 나타내는 파일 시스템 수준의 고유 식별자다.

[14] libfwsi, "Windows Shell Item format specification", 2014-12-11, https://github.com/libyal/libfwsi/blob/main/documentation/Windows%20Shell%20Item%20format.asciidoc#extension_block_0xbeef004 (최종수정일 : 2024-03-18)

[그림 17] FileReference 값의 구성

이 정보는 단순한 경로 문자열이 아니라 운영체제가 내부적으로 관리하는 식별 정보를 반영하기 때문에 외형상 동일한 경로를 참조하더라도 시스템이나 파일이 다르면 FileReference 값은 달라진다. 반대로, 동일한 시스템 환경에서 동일한 대상 파일을 참조해 생성된 바로가기 파일은 동일한 FileReference 값을 가질 수 있으며, 이러한 반복은 공격자가 같은 PC 또는 동일한 이미지 환경에서 악성 바로가기 파일을 반복적으로 생성했음을 시사한다.

[표 22] 악성 바로가기 파일 제작 환경에 따른 FileReference 구성 차이

값	제작 환경	대상 프로그램 경로		
		Windows	System32	cmd.exe
FileReference	A	2301-1	3938-1	55562-1
	B	515-1	3286-1	45929-1

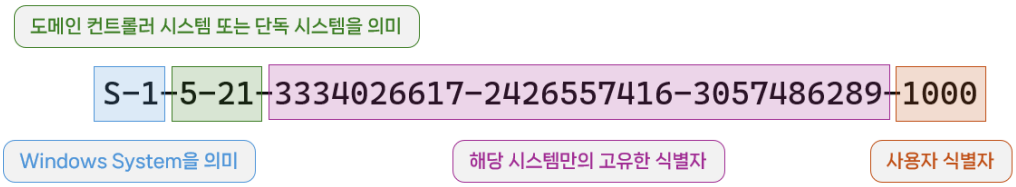
이처럼 FileReference는 일반 사용자가 직접 수정하거나 위조하기 어려운 정보이며, 운영체제의 파일 시스템 수준에서 자동 생성되는 값으로 높은 신뢰도를 가진다. 또한, 특정 환경 내 반복적으로 생성되는 악성파일에서 동일한 FileReference 값이 나타날 수 있으므로 이는 공격자가 사용하는 제작 환경 식별에 유의미한 단서로 활용될 수 있다.

3) 사용자 SID

바로가기 파일의 ExtraData 영역에는 다양한 확장 블록 유형이 존재하며, 그 중 PropertyStoreDataBlock은 Windows Shell 환경에서 파일, 폴더, 바로가기 등 다양한 객체에 메타데이터 형식의 속성 값을 저장하기 위해 사용되는 구조이다. 이 구조 내부에는 'PropertyStoreList'라는 서브 구조가 존재하며, 다양한 속성 값들이 formatID를 키로 하여 나열된다. 이 중 formatID가 '46588AE2-4CBC-4338-BBFC-139326986DCE'인 항목의 value 값에는 바로가기 파일을 생성한 사용자 계정의 SID 정보가 저장된다.

- formatID가 46588AE2-4CBC-4338-BBFC-139326986DCE일 경우, 값은 VT_LPWSTR 형식으로 저장된 사용자 SID 문자열이며, 이 구조는 Windows 7 이상에서 생성된 바로가기 파일에서 주로 관찰됨

사용자 SID는 Security Identifier의 약자로, 이는 운영체제에서 로컬 사용자 계정 또는 도메인 계정에 고유하게 부여하는 보안 식별자이다. 일반적으로 SID는 다음과 같은 형식을 가지며, 해당 사용자가 속한 보안 주체를 명확하게 식별할 수 있다.



[그림 18] Windows SID 구조

SID는 사용자가 수동으로 변경할 수 없고 시스템 또는 도메인 컨트롤러에 의해 자동으로 생성되기 때문에 위.변조가 어려운 고유 식별 값에 해당한다. 이에 따라 악성 바로가기 파일에 포함된 SID는 해당 파일을 생성한 시스템의 계정 정보를 파악할 수 있는 실질적인 단서가 될 수 있다.

특히, 다수의 악성 바로가기 샘플에서 동일한 SID가 반복적으로 확인될 경우 이는 공격자가 동일한 PC 또는 가상머신 환경에서 여러 개의 악성 바로가기 파일을 일괄적으로 생성했을 가능성을 시사한다. 이러한 맥락에서 SID는 공격자의 제작 환경을 식별하거나 상관 분석을 수행하는 데 유의미한 정보로 활용될 수 있다.

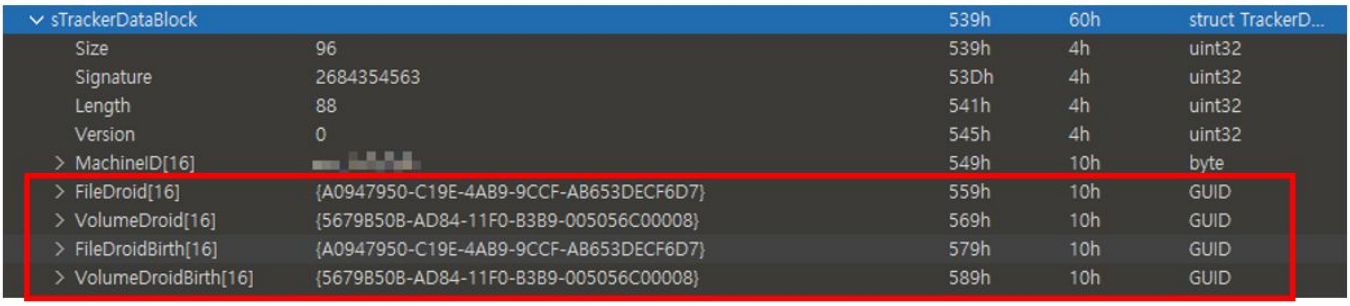
4) VolumeID와 FileID

바로가기 파일의 ExtraData 영역에 포함된 TrackerDataBlock 구조는 바로가기 파일이 가리키는 대상 파일이 위치했던 시스템 및 파일에 대한 식별 정보를 담고 있다. 이 블록에는 다음과 같은 네 개의 GUID 값이 존재하며, 각각 파일과 볼륨의 고유 식별자를 나타낸다.

[표 23] TrackerDataBlock에 포함된 파일 및 볼륨 식별자 항목

• BirthDroidFileIdentifier • BirthDroidVolumeIdentifier	• DroidFileIdentifier • DroidVolumeIdentifier

이 중 'BirthDroid*' 값은 참조 대상이 처음 생성되었을 때의 환경 정보를 의미하며, 'Droid*' 값은 바로가기 파일이 생성될 당시 운영체제가 인식한 현재의 파일 및 볼륨 정보를 나타낸다. 다음은 한 악성 바로가기 파일에서 추출한 관련 데이터의 예시이다.



[그림 19] TrackerDataBlock 내 파일과 볼륨의 고유 식별자 예시

이 값들은 Windows 링크 추적(Link Tracking) 기능에 의해 자동으로 생성되며, 대상 파일이 이동되거나 이름이 변경되더라도 원본 파일을 추적할 수 있도록 설계되어 있다[15]. 그 중에서도 BirthDroid 계열 식별자는 파일이 최초로 생성된 시점의 환경 정보를 보존하므로 공격자가 악성 바로가기 파일을 제작한 환경을 식별하기 위한 기준 데이터로 채택했다.

즉, BirthDroidFileIdentifier와 BirthDroidVolumeIdentifier는 악성 바로가기 파일이 처음 생성된 시스템의 환경을 식별할 수 있으며, 이후 파일이 복사되거나 이동되더라도 해당 값은 변하지 않는다. 반면, Droid* 값은 파일이 새로운 시스템으로 이동되거나 복사될 경우 갱신될 수 있으므로 공격자 식별 지표로서는 상대적으로 신뢰도가 낮다고 판단했다.

[표 24] VolumeID 및 FileID 관련 주요 필드 요약

선택	항목명	의미	구조 내 위치	데이터 형식
✓	BirthDroidFileIdentifier	• 대상 파일이 최초로 생성된 시점의 고유 식별자	TrackerDataBlock	GUID
✓	BirthDroidVolumeIdentifier	• 대상 파일이 처음 저장된 볼륨 식별자		
	DroidFileIdentifier	• 바로가기 파일 생성 시 인식된 파일 식별자		
	DroidVolumeIdentifier	• 바로가기 파일 생성 시 인식된 볼륨 식별자		

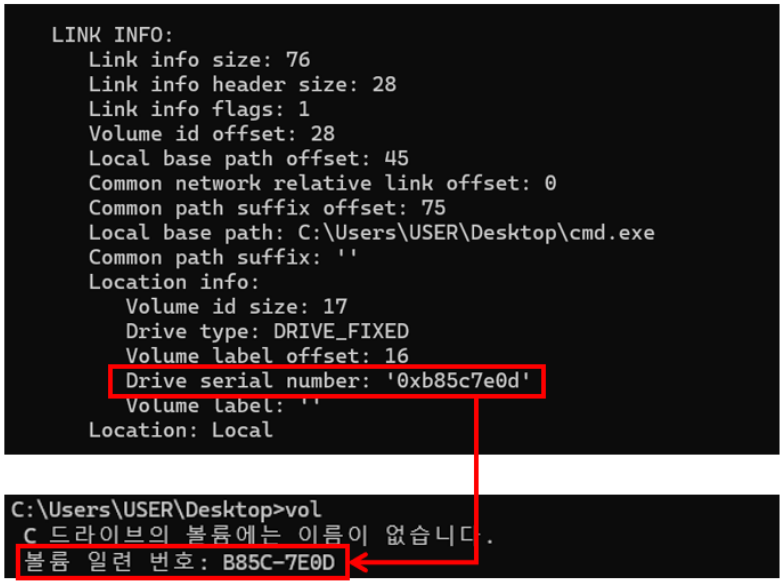
이처럼 BirthDroid* 값은 파일 시스템 수준에서 자동 생성되는 정보로, 일반적인 사용자 환경에서는 임의 수정이 어려워 높은 신뢰도를 가진다. 다만, 고급 악성코드 제작 환경에서는 TrackerDataBlock 전체를 바이너리 수준에서 조작하거나 빌더 도구를 통해 GUID 값을 임의로 조작하는 것이 기술적으로 가능하다. 따라서, BirthDroid 값은 일반 사용자 환경에서는 위변조 가능성이 매우 낮지만, 공격자가 고의적으로 조작할 가능성은 존재하므로 반복성과 맥락 분석을 병행해야 한다. 그럼에도 불구하고 동일한 BirthDroid 값이 여러 샘플에서 반복 발견될 경우, 이는 공격자가 동일한 환경에서 악성 바로가기 파일을 대량 생산했을 가능성을 시사하며, 공격자가 악성 바로가기 파일을 제작한 환경을 추적하는데 유의미한 근거로 활용될 수 있다.

5) DriveSerialNumber (VolumeSerialNumber)

바로가기 파일의 LinkInfo 구조에는 대상 파일이 위치한 드라이브의 정보를 담는 VolumeID 구조가 포함되며, 이 안에는 드라이브를 식별하기 위한 DriveSerialNumber가 저장된다. 이 값은 일반적으로 디스크가 포맷될 때 시스템에 의해 자동으로 생성되며, 8자리의 16진수 값으로 표현된다.

[15] Microsoft, “Distributed Link Tracking and Object Identifiers”, 2021-12-31, https://learn.microsoft.com/en-us/windows/win32/fileio/distributed-link-tracking-and-object-identifiers

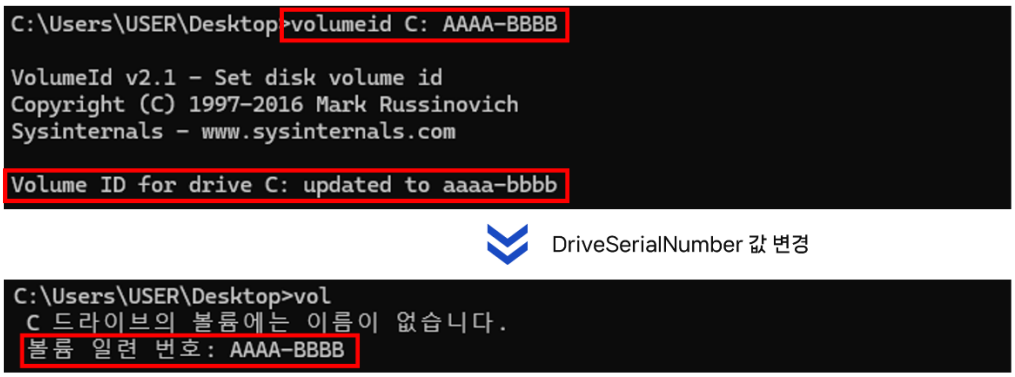
DriveSerialNumber는 명령어 'vol' 또는 'fsutil fsinfo volumeinfo'를 통해 확인할 수 있으며, 운영체제는 이를 통해 각 디스크 또는 파티션을 내부적으로 식별한다. 바로가기 파일이 참조하는 대상이 로컬 디스크에 존재할 경우, 이 값은 파일이 위치했던 디스크의 고유 식별자 역할을 하며, LinkInfo 구조 내에 다음과 같이 저장된다.



[그림 20] VolumeID 구조 내 DriveSerialNumber

이 값은 Windows 운영체제가 디스크를 포맷할 때 기본적으로 생성하는 고유 식별자로, 사용자가 별도로 지정하지 않는 이상 동일한 값이 중복될 가능성은 낮다. 따라서, 여러 악성 바로가기 파일 간에 동일한 DriveSerialNumber가 반복적으로 등장할 경우, 공격자가 동일한 디스크 환경에서 악성파일을 생성했을 가능성을 유추할 수 있다.

그러나 DriveSerialNumber는 'volumeid'와 같은 유틸리티[16]를 이용해 조작이 가능한 값이므로, 다른 식별 지표와 결합해 판단해야 할 보조적 지표로 보는 것이 타당하다.



[그림 21] volumeid 유틸리티를 활용한 DriveSerialNumber 변경 결과

[16] https://learn.microsoft.com/ko-kr/sysinternals/downloads/volumeid

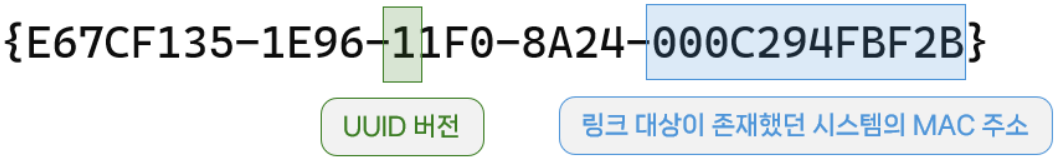
그럼에도 불구하고 DriveSerialNumber는 운영체제 수준에서 자동 생성되는 값으로 일반적인 파일 생성에서는 변경되지 않으며, 의도적으로 도구를 사용해야 변경이 가능하기 때문에 공격자의 제작 환경을 유추할 수 있는 유의미한 근거로 활용될 수 있다. 특히, FileReference, 사용자 SID 등과 같은 고유 값과 함께 동일하게 반복되는 경우, 해당 환경의 고유성을 식별하는데 보다 신뢰성 있는 단서로 작용할 수 있다.

6) MAC 주소

바로가기 파일의 ExtraData 영역에 포함된 TrackerDataBlock 구조는 참조 대상 파일의 위치 정보뿐 아니라 파일 생성 당시의 시스템 식별 정보를 함께 담고 있다. 이 중 BirthDroidFileIdentifier, BirthDroidVolumeIdentifier, DroidFileIdentifier, DroidVolumeIdentifier 항목은 각각 GUID 형식으로 저장되며, 해당 GUID의 구성 중 UUID 버전 1 방식으로 생성된 경우, 내부에 바로가기 파일 생성 시스템의 MAC 주소가 포함된다.

- UUID(Universally Unique Identifier)와 GUID(Globally Unique Identifier)는 모두 16bytes 길이의 고유 식별자를 의미하며, 형식과 기능 면에서는 동일하다. 다만, UUID는 IETF의 RFC 4122에 정의된 표준 명칭으로, 플랫폼 간 범용적으로 사용된다. 반면, GUID는 Microsoft에서 내부적으로 사용되는 명칭으로, COM 객체 식별자 등 Windows 환경에서 주로 사용된다. 따라서, 본 문서에서 바로가기 파일의 구조 형식을 언급할 때는 GUID라는 용어를 사용하지만, 기술 설명을 위해서는 공식 용어인 UUID라는 표현을 사용한다.

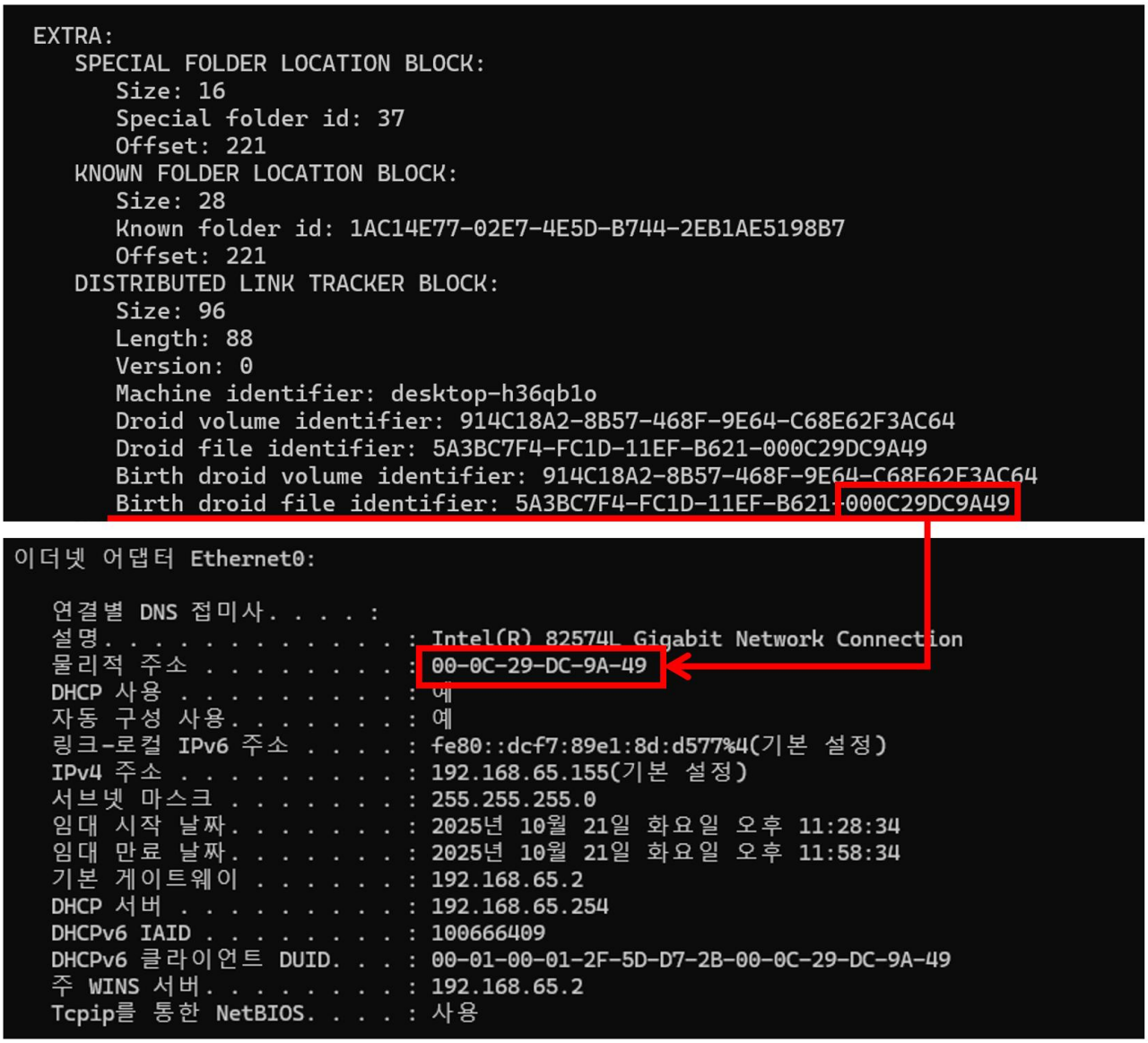
UUID는 총 16bytes 길이의 고유 식별자로, 일반적으로 '8-4-4-4-12' 형태로 표현된다. UUID의 버전은 세 번째 블록의 첫 문자에 저장되며, 이 값이 '1'일 경우 버전 1로 판별할 수 있다. UUID 버전 1은 시간 기반 식별자로, 생성 시간과 함께 해당 시스템의 MAC 주소 정보를 포함하도록 설계되어 있다[17].



[그림 22] UUID 버전과 MAC 주소 식별 (UUID 1일 때로 제한)

예를 들어, 다음과 같은 값이 BirthDroidFileIdentifier로 저장된 사례를 살펴보면 이 GUID 값의 세 번째 블록이 '11EF'로 시작되며, 이 중 첫 문자 '1'은 UUID의 버전이 1 버전임을 나타낸다. 마지막 12자리인 '000C29DC9A49'는 시스템의 MAC 주소에서 파생된 값으로, 일반적인 MAC 주소 표현 방식에 따라 00-0C-29-DC-9A-49로 해석할 수 있다.

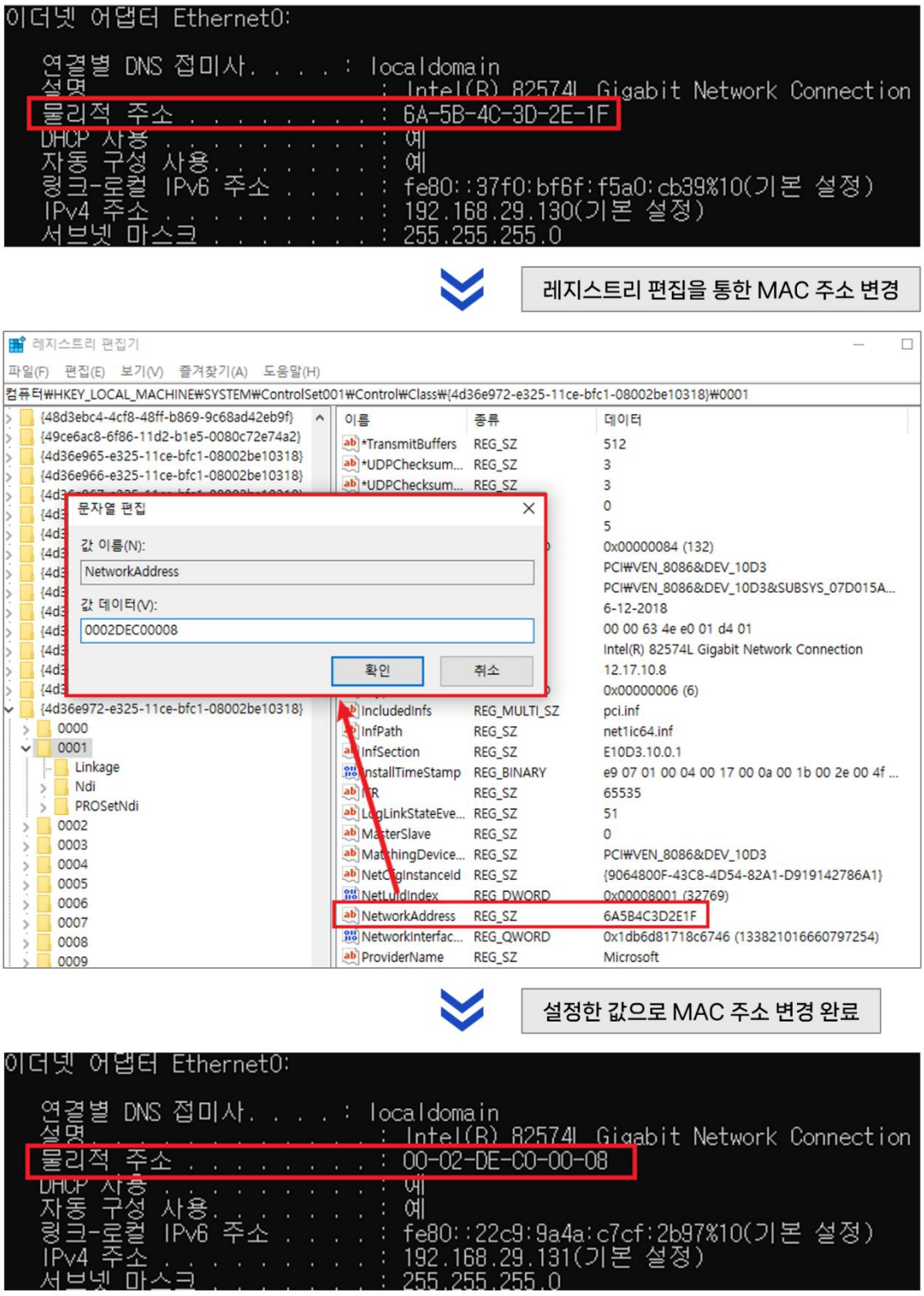
[17] P. Leach, M. Mealling, R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", Network Working Group, 2005-07, https://datatracker.ietf.org/doc/html/rfc4122



[그림 23] BirthDroidFileIdentifier 값 내 MAC 주소 확인

MAC 주소는 네트워크 인터페이스 장치에 고유하게 부여되는 48bit 값으로, 대부분의 운영체제에서는 사용자가 수동으로 변경하지 않는 한 지속적으로 동일한 값을 유지한다. 또한, 제조사별로 할당된 OUI(Organizationally Unique Identifier)를 통해 장치 제조사까지 추정할 수 있다는 점에서 악성 바로가기 파일을 제작한 환경을 식별하는데 유용하다.

다만, 레지스트리 조작을 통해 MAC 주소 변경이 가능하고 가상 환경에서도 MAC 주소가 임의로 생성되거나 수동 설정이 가능하므로, 해당 값만으로 공격자의 환경을 단정하기에는 한계가 존재한다.

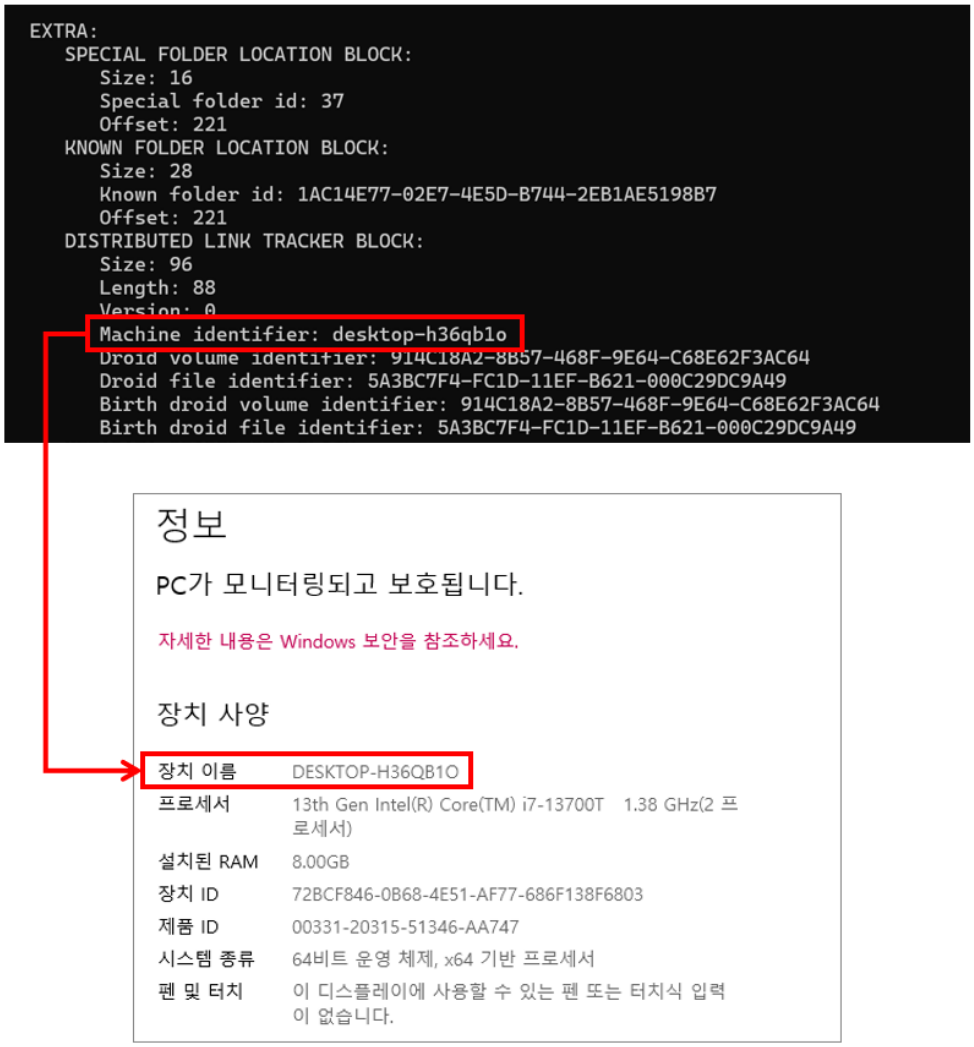


[그림 24] 레지스트리 편집을 통한 MAC 주소 변경 예시

따라서 본 연구에서는 MAC 주소를 단독 지표가 아닌 SID, FileReference, VolumeID 등 다른 구조 기반 식별자와 함께 교차 분석하는 방식으로 공격자의 제작 환경을 추정했다.

7) MachineID

바로가기 파일의 ExtraData 영역 중 TrackerDataBlock에는 바로가기 파일이 생성된 시스템의 호스트명을 의미하는 MachineID가 저장된다. 이 값은 일반적으로 Windows 운영체제에서 컴퓨터 이름으로 사용되는 값으로, 바로가기 파일이 생성된 컴퓨터를 식별할 수 있는 단서를 제공한다.



[그림 25] ExtraData 영역의 TrackerDataBlock 내 MachineID

MachineID는 운영체제 설치 시 자동으로 생성되거나, 사용자가 직접 지정한 컴퓨터 이름이 저장된다. 이 값은 네트워크나 로컬 환경에서 시스템을 구분하기 위해 사용되며, 동일한 시스템에서 생성된 바로가기 파일은 동일한 Machine ID 값을 갖는다는 점에서 악성 바로가기 파일이 제작된 환경을 식별하는데 보조 지표로 활용될 수 있다.

다만, MachineID는 Windows 설정 메뉴의 '이 PC 이름 바꾸기' 기능을 통해 언제든지 사용자가 직접 변경할 수 있는 값으로, SID나 VolumeID처럼 시스템 내부적으로 고정된 식별자에 비해 신뢰도가 낮다.

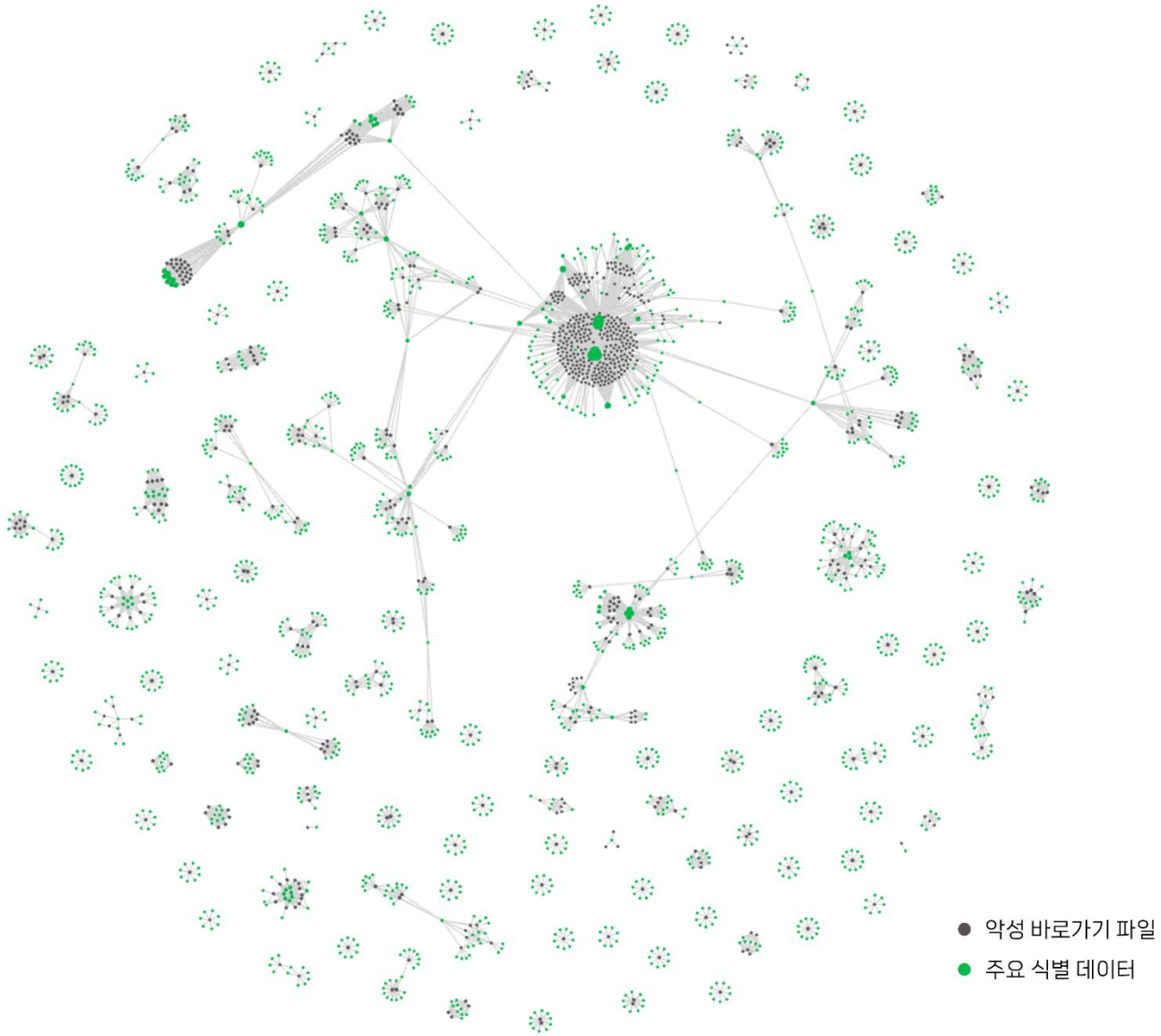
[그림 26] Windows 설정 메뉴의 '이 PC 이름 바꾸기' 기능을 통한 MachineID 변경

따라서, 본 연구에서는 MachineID를 사용자가 변경이 가능한 보조 식별 지표로 분류하고, 단독으로 공격자 환경을 특정하는데 사용하기보다는 FileReference, VolumeID, SID 등 고유 식별자와 함께 교차 분석하는 방식으로 활용했다.

3.2. 악성 바로가기 샘플 데이터 간 상호 연관성 분석

앞서 3.1절에서는 악성 바로가기 파일 구조 내에서 악성 바로가기 파일을 제작한 환경을 식별할 수 있는 주요 데이터를 정리했다. 본 절에서는 이러한 주요 식별 데이터를 실제 수집한 1,135개의 악성 바로가기 파일에 적용해 공격자의 제작 환경이나 공격 도구의 공통점 등을 식별할 수 있는지 분석했다.

분석은 각 악성 샘플에서 주요 식별 데이터가 확인된 경우, 해당 값을 개별 샘플에 태그화하는 방식으로 진행했다. 이후, 태그화 된 데이터를 Obsidian의 그래프 뷰 기능을 활용해 시각화함으로써 샘플 간의 구조적 유사성, 식별자 반복 여부, 그리고 특정 공격 그룹 또는 도구와의 연관성을 파악했다.

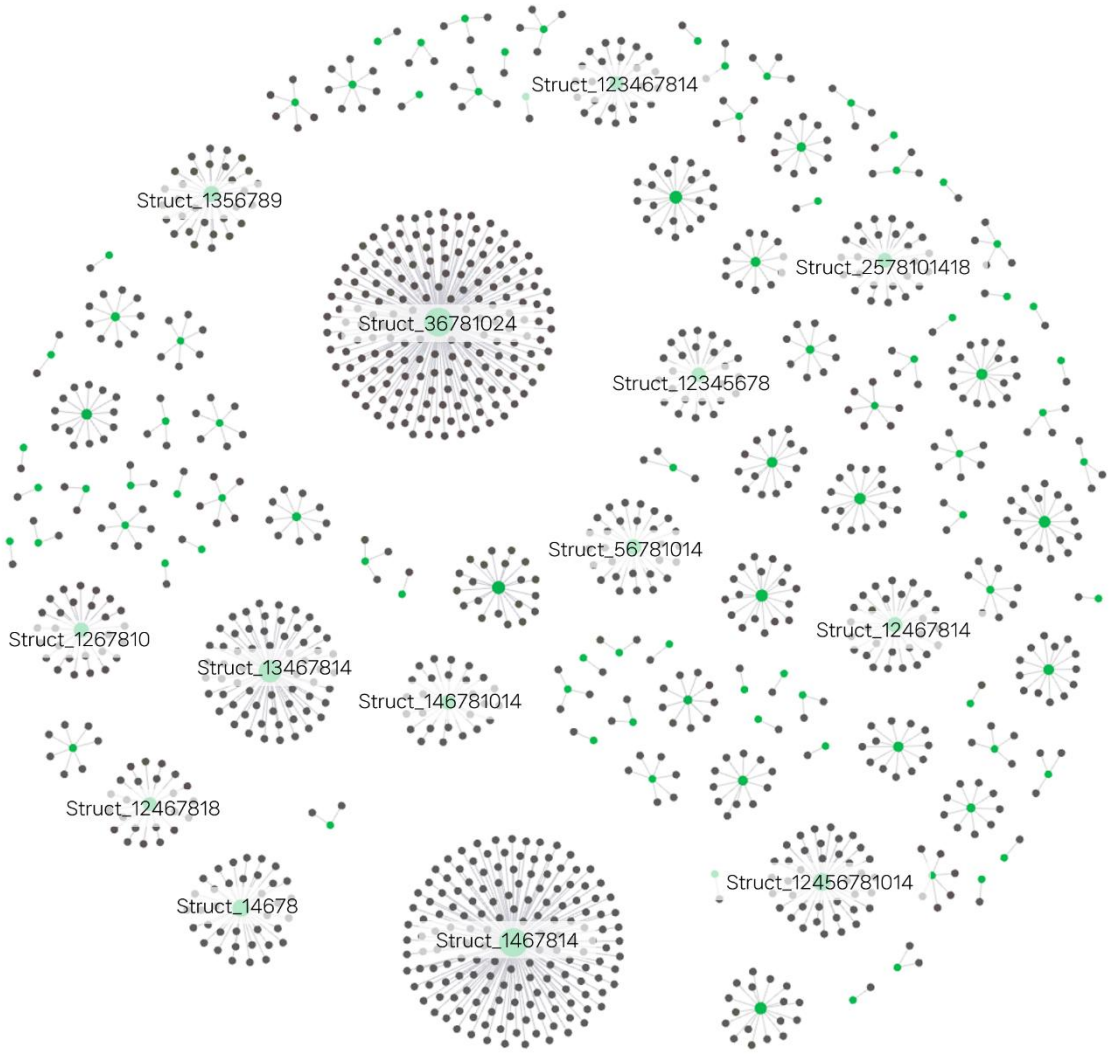


[그림 27] 수집한 악성 바로가기 샘플 대상 주요 식별 데이터 태깅 결과

1) LinkFlags 구조 분석

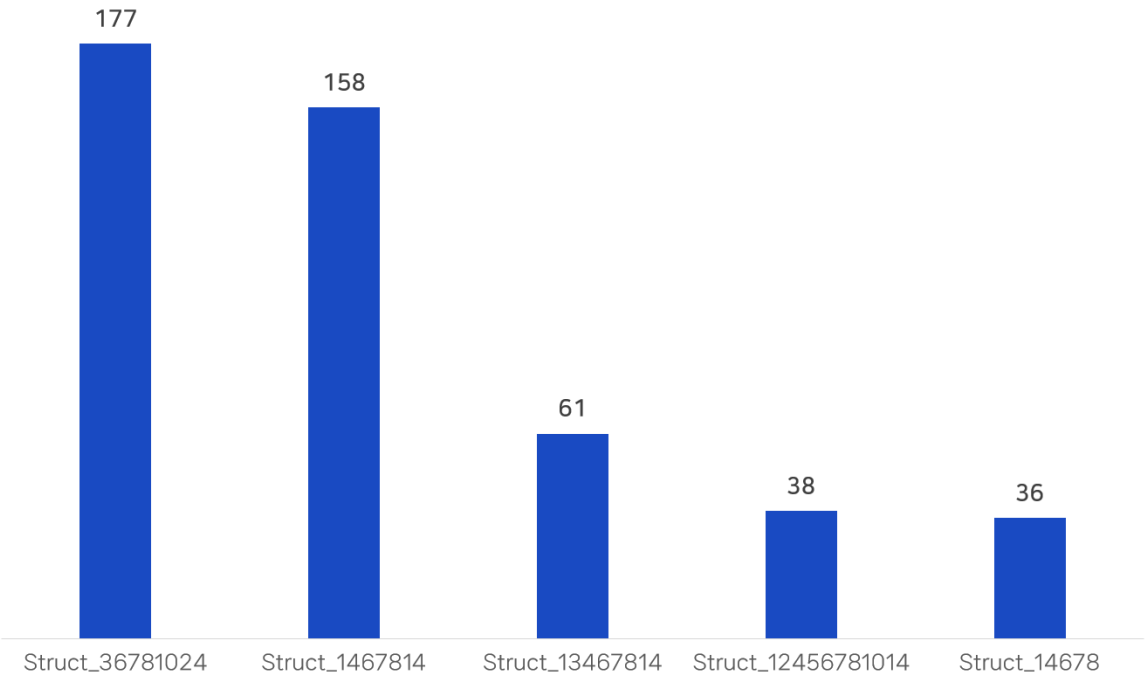
앞서 선정한 주요 식별 데이터 중 하나인 LinkFlags 구조를 분석하기 위해 항목에 대한 유형화 작업을 수행했다. 바로가기 파일의 LinkFlags는 여러 개의 플래그가 비트 단위로 설정된 값으로, 이를 해석하기 위해 플래그의 비트 위치를 추출하고 이를 조합한 식별자로 변환했다. 구체적으로, 설정된 비트 위치를 오름차순으로 나열한 뒤, 접두사 'Struct_'를 붙여 태깅하는 방식을 사용했다. 예를 들어, 'HasName'은 비트 위치 3번, 'HasArguments'는 비트 6번이 설정되어 있으므로 해당 플래그를 가지는 구조는 'Struct_36'과 같은 형태로 표기했다. 이와 같은 방식은 각 샘플의 LinkFlags 구조를 고정된 식별자로 정규화해 분석과 분류를 용이하게 하기 위한 목적이다.

이러한 태깅 결과를 바탕으로 Obsidian의 그래프 뷰 기능을 활용해 수집된 악성 바로가기 파일 샘플의 LinkFlags 구조를 시각화한 결과, 동일한 LinkFlags 구조를 가진 샘플들이 그룹화되는 것이 관찰되었다.



[그림 28] Obsidian의 그래프 뷰 기능을 통해 확인한 LinkFlags 구조

또한, 전체 샘플 중에서 가장 빈번하게 나타난 LinkFlags 구조 상위 5개는 다음과 같다.



[그림 29] 데이터셋 내 동일 LinkFlags 구조 분포 통계 – 상위 5개

[표 25] 데이터셋 내 동일 LinkFlags 구조

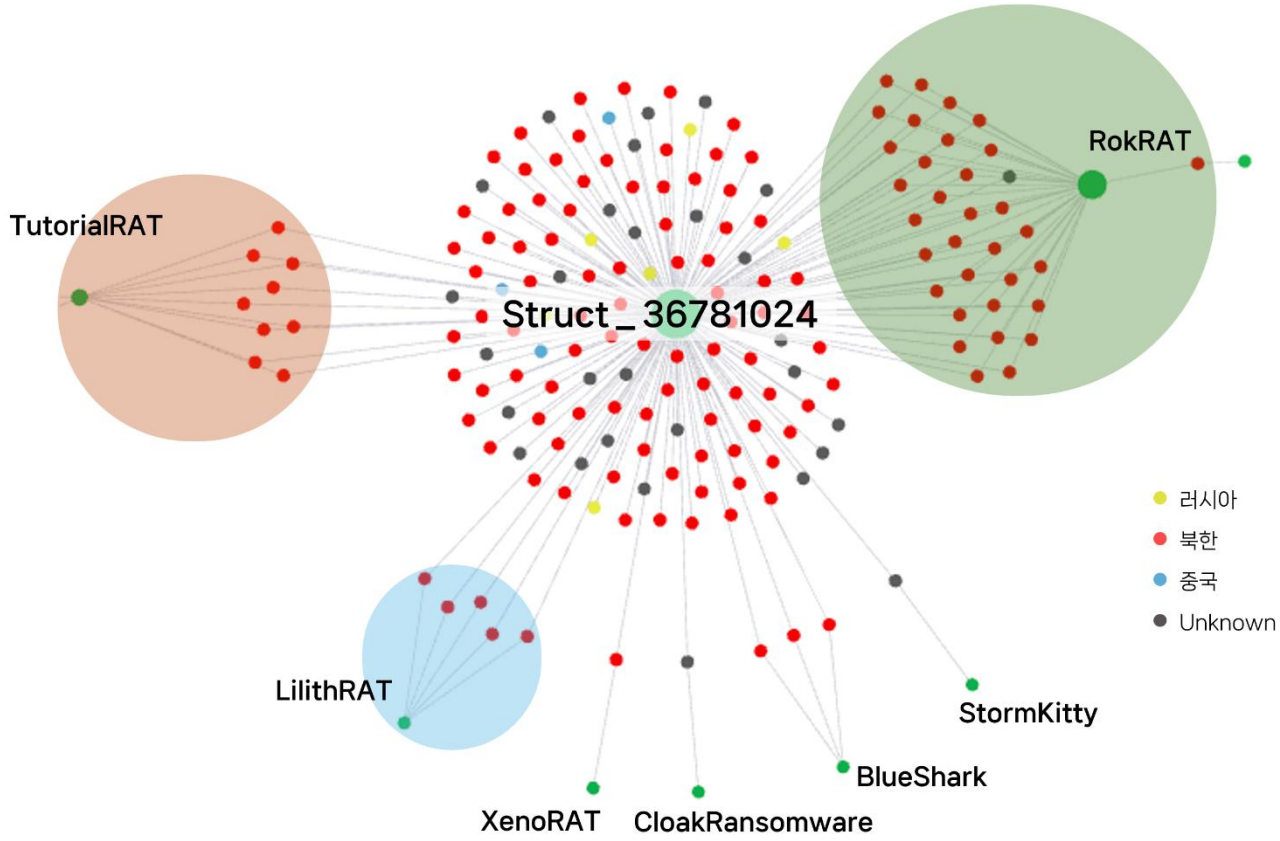
Struct ID	포함된 LinkFlags	빈도 수
Struct_36781024	• HasName, HasArguments, HasIconLocation, IsUnicode, HasExpString, PreferEnviromentPath	177
Struct_1467814	• HasTargetIDList, HasRelativePath, HasArguments, HasIconLocation, IsUnicode, RunAsUser	158
Struct_13467814	• HasTargetIDList, HasName, HasRelativePath, HasArguments, HasIconLocation, IsUnicode, RunAsUser	61
Struct_12456781014	• HasTargetIDList, HasLinkInfo, HasRelativePath, HasWorkingDir, HasArguments, HasIconLocation, IsUnicode, HasExpString, RunAsUser	38
Struct_14678	• HasTargetIDList, HasRelativePath, HasArguments, HasIconLocation, IsUnicode, RunAsUser	36

가장 많이 확인된 Struct_36781024 구조는 바로가기의 설명과 아이콘을 통해 정상 파일로 위장하는 동시에 명령행 인자를 통해 페이로드 실행을 유도한다. 또한 %APPDATA%와 같은 환경변수 확장을 사용하고 환경변수 경로를 우선 사용해 실행 경로를 동적으로 지정하도록 한다. 즉, 동일한 바로가기 파일로 서로 다른 시스템 환경에서도 자동으로 적절한 위치를 참조해 악성 행위를 수행할 수 있도록 설계한 것이다.

다음으로 많이 확인된 Struct_1467814 구조는 링크 대상 경로를 포함하고 있으며 Struct_36781024 구조와 동일하게 바로가기의 설명과 아이콘을 통해 정상 파일로 위장하는 동시에 명령행 인자를 통해 페이로드 실행을 유도한다. 또한 바로가기 파일 실행 시 특정 사용자 권한으로 실행되도록 설정함으로써 권한 상승 우회 수단으로 활용될 가능성이 있다.

이처럼 동일한 LinkFlags 조합이 반복적으로 나타나는 것은 공격자가 동일한 제작 도구 또는 스크립트를 사용해 악성 바로가기 파일을 일괄 생성했을 가능성을 시사한다. 특히 LinkFlags 조합은 자동화된 도구에 의해 고정된 형식으로 설정되는 경우가 많으므로 특정 조합이 자주 등장한다면 악성 바로가기 파일을 제작하는데 사용된 도구를 특정하는데도 활용이 가능하다.

다음은 실제 가장 많이 확인된 Struct_36781024 구조의 악성 바로가기 샘플들에 대해 공격 그룹의 추정 배후 국가 및 최종적으로 실행된 악성코드 계열을 정리한 시각화 결과이다. 공격자 배후 국가와 악성코드 계열은 서로 다르지만 동일한 바로가기 파일 구조가 반복되었다는 점은 여러 공격 그룹이 동일한 제작 도구 또는 코드 베이스를 공유하고 있을 가능성을 시사한다.



[그림 30] Struct_36781024 구조를 악용한 악성 바로가기 파일의 공격자별 분포 및 악성코드 계열

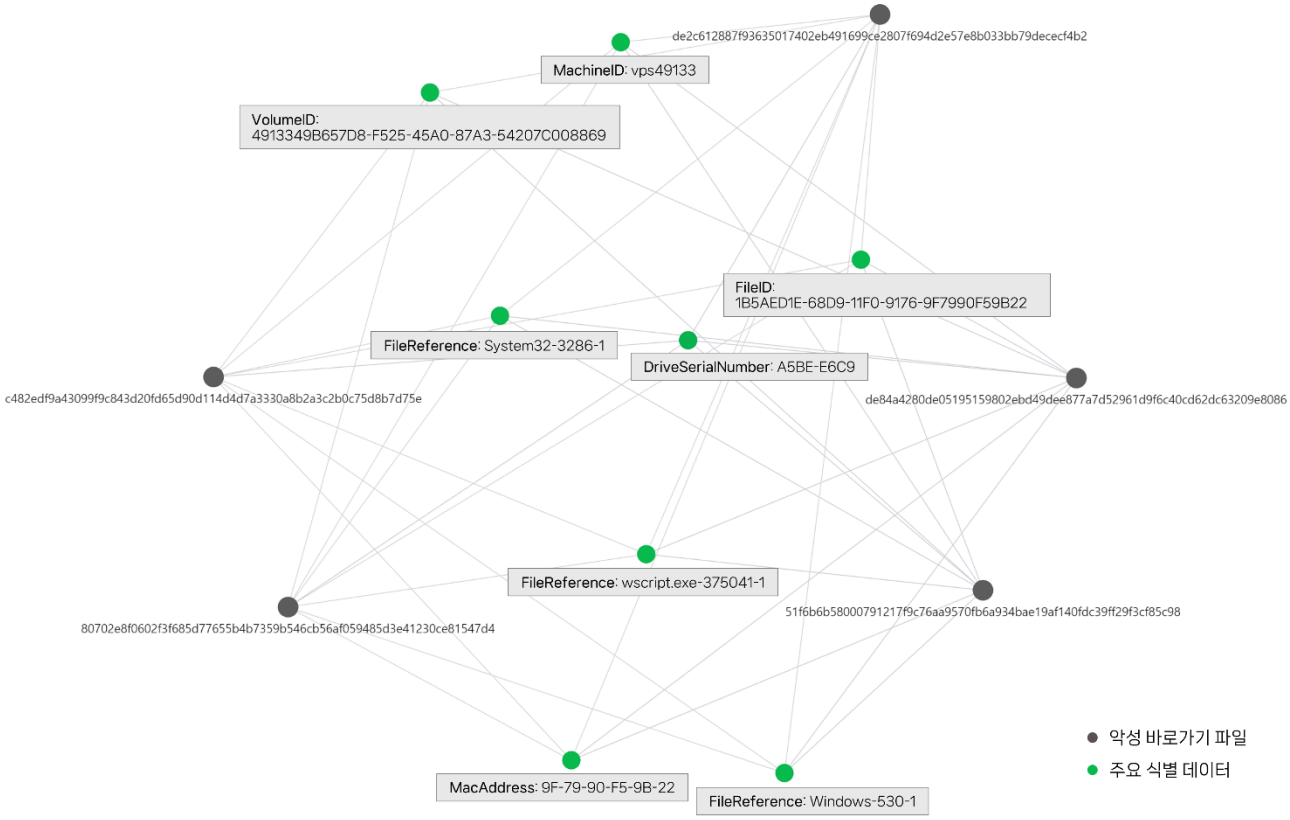
2) 악성 바로가기 파일 제작 환경 정보 기준 분석

악성 바로가기 파일은 공격자가 사용하는 시스템 환경의 흔적을 포함하고 있을 수 있으며, 이러한 식별 정보가 반복적으로 사용된 경우 동일한 제작 환경에서 생성된 악성 샘플일 가능성이 높다. 이에 따라 본 항목에 대한 분석 시 바로가기 파일의 제작 환경을 식별하기 위해 구조 내 포함한 주요 식별자 중 신뢰도가 높은 항목인 'FileReference'와 '사용자 SID' 값을 기준으로 분석을 진행했다.

FileReference 기반 분석

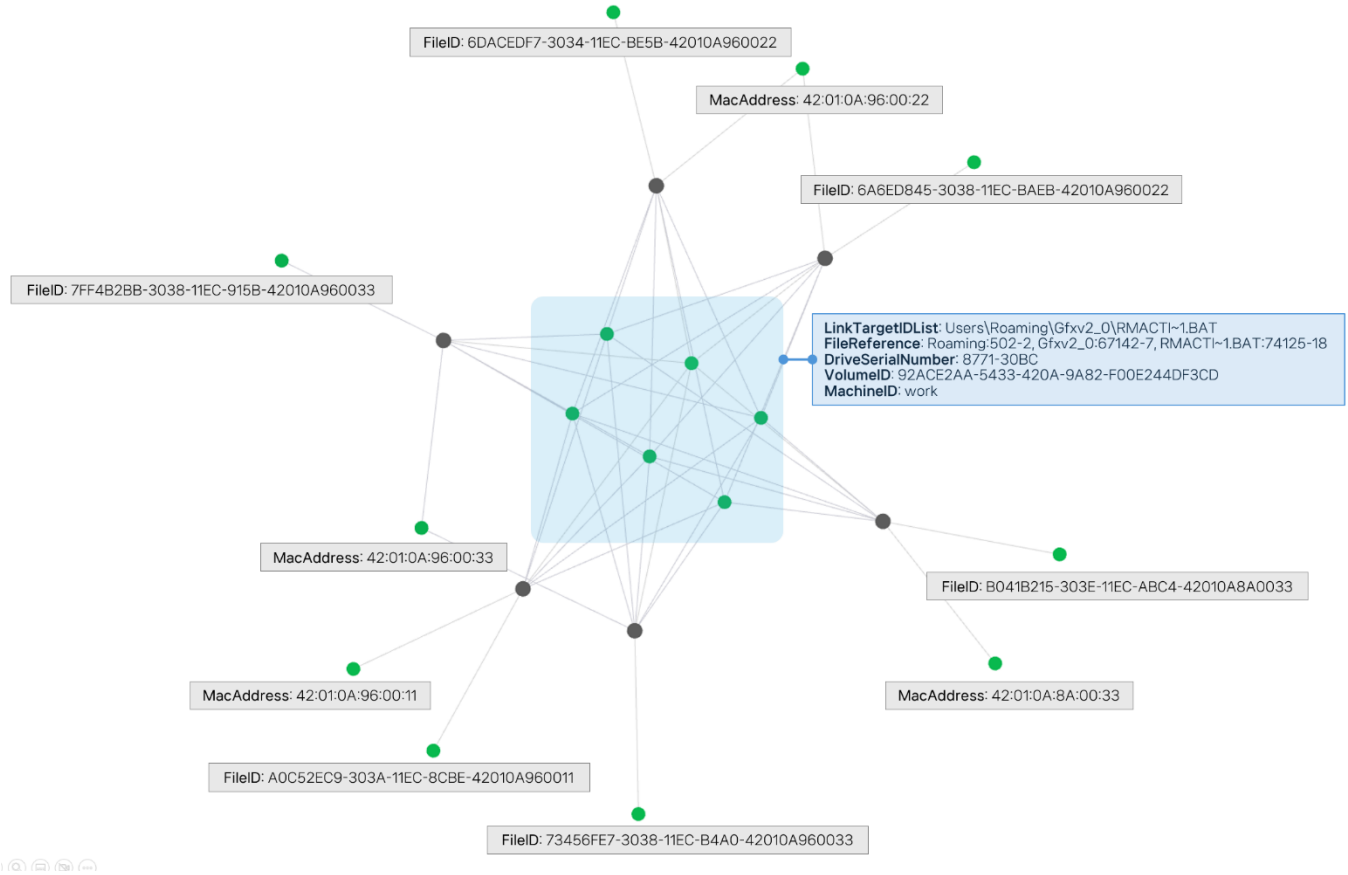
먼저 1,135개의 악성 바로가기 샘플 중 FileReference의 값이 완전히 동일한 샘플들을 추출했다. FileReference는 바로가기 파일이 생성된 시점과 경로, MFT 내 위치 등의 정보를 기반으로 산출되는 식별자로, 일반적인 파일 복사나 이동으로 변경되지 않기 때문에 동일 시스템에서 생성된 파일일 가능성을 강하게 시사한다.

바로가기 파일의 대상 프로그램의 폴더 및 파일의 FileReference 값을 공유하는 샘플들을 그룹화한 결과, 다음과 같이 FileReference 뿐만 아니라 VolumeID, DriveSerialNumber, MAC 주소, VolumeID, FileID 등 다른 주요 데이터 값들도 동일하게 반복되는 사례가 확인되었다. 이는 공격자가 동일한 제작 환경을 그대로 복제하거나 복수의 악성 파일을 자동 생성하는 과정에서 동일한 시스템을 활용했을 가능성을 보여준다.



[그림 31] 악성 샘플 간 제작 환경 유사성 시각화 – FileReference 기준 (1)

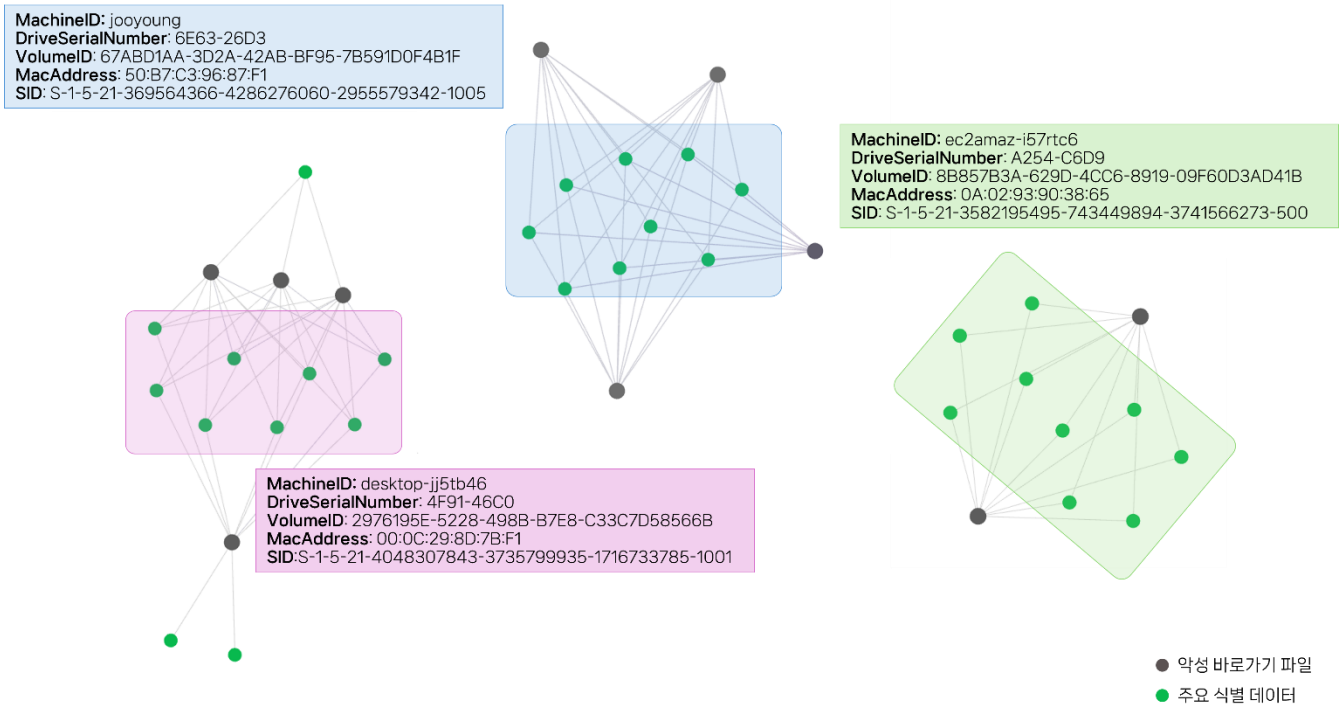
반면, FileReference 값은 동일하지만 나머지 식별 데이터가 상이한 사례도 일부 확인되었다. 다음 그림에서는 바로가기 파일이 가리키는 대상 파일과 FileReference가 완전히 동일하며, 해당 파일이 생성된 볼륨의 DriveSerialNumber 및 VolumeID 역시 일치한다. 그러나 FileID와 MAC 주소는 서로 달라 바로가기 파일이 생성된 시점이나 해당 데이터의 변조의 가능성이 있다. 특히, MAC 주소의 경우 OUI(Organizationally Unique Identifier) 기반으로 검색했을 때 유효한 제조사 정보가 확인되지 않아, 일부 변조되었거나 무작위로 생성했을 가능성이 높다고 판단된다.



[그림 32] 악성 샘플 간 제작 환경 유사성 시각화 – FileReference 기준 (2)

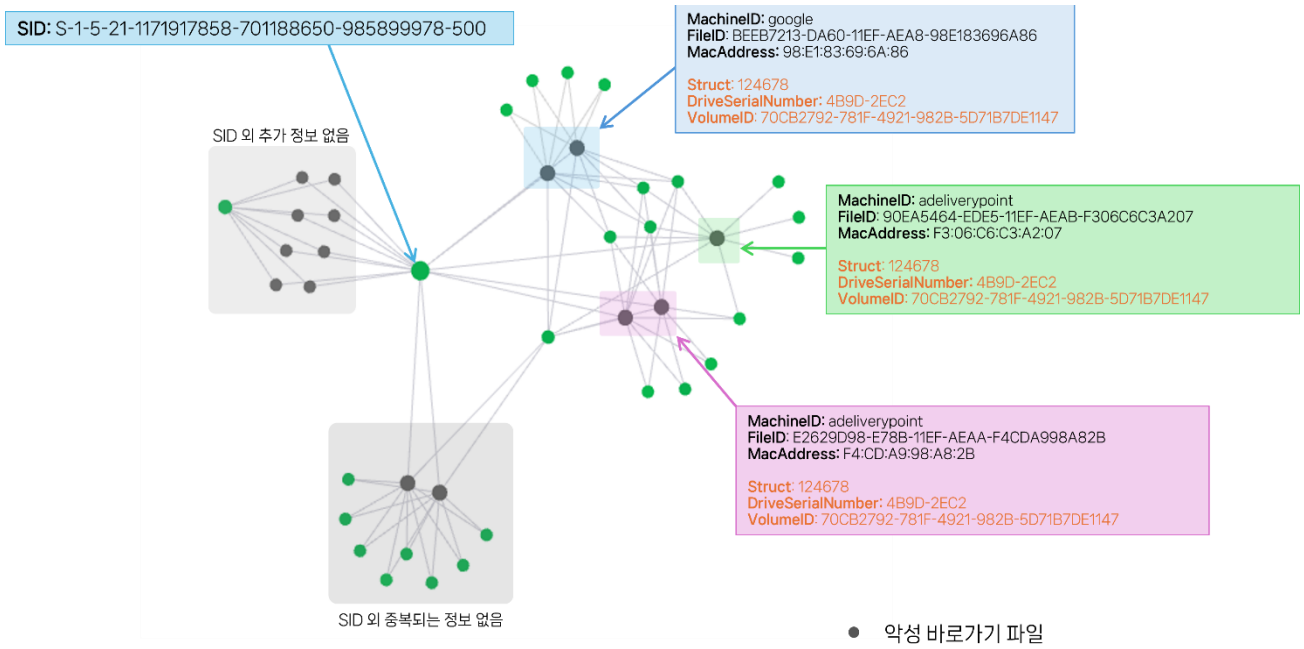
사용자 SID 기반 분석

FileReference와 함께 신뢰도가 높은 항목으로 분류되는 사용자 SID로도 분석을 진행했다. 마찬가지로 동일한 SID 값을 공유하는 악성 샘플들 역시 특정 그룹을 형성했으며, 이들 샘플 중 대다수는 DriveSerialNumber, VolumeID, MAC 주소, MachineID 등 다른 주요 식별 데이터들도 일치하여 동일한 시스템에서 제작된 악성 파일일 가능성을 시사한다.



[그림 33] 악성 샘플 간 제작 환경 유사성 시각화 – 사용자 SID 기준 (1)

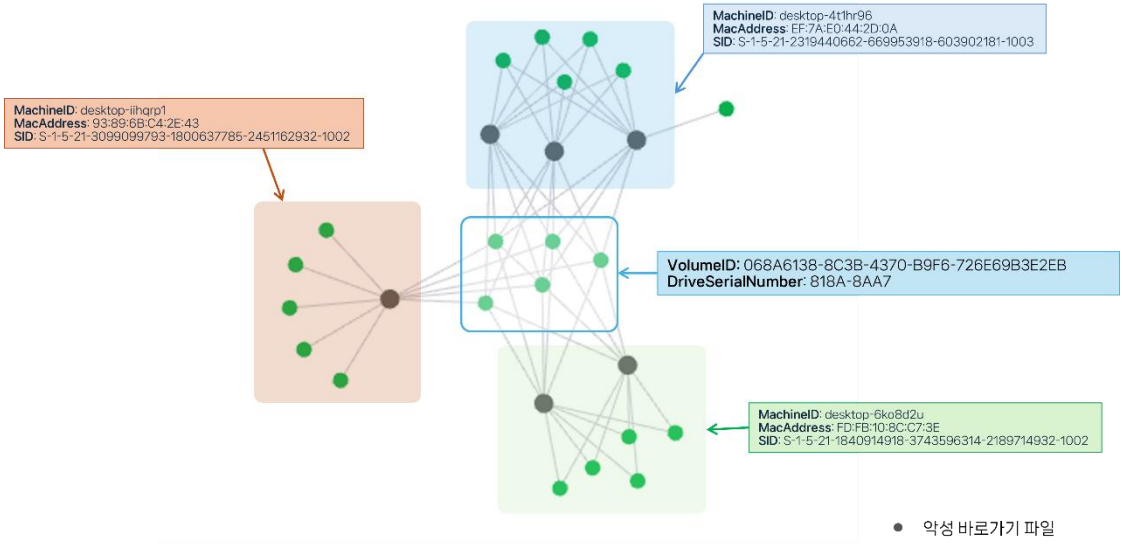
반면, 사용자 SID 값은 같지만 그 외의 식별 데이터 값이 상이한 경우도 일부 확인되었다. 다음 그림에서는 바로가기 파일을 생성한 PC의 사용자 계정의 SID와 해당 파일이 생성된 볼륨의 DriveSerialNumber 및 VolumeID 역시 일치한다. 그러나 FileID, MAC주소, MachineID는 서로 달라 바로가기 파일이 생성된 시점이 달라 환경 정보가 변경되었거나, 공격자가 임의로 변조했을 가능성을 시사한다.



[그림 34] 악성 샘플 간 제작 환경 유사성 시각화 – 사용자 SID 기준 (2)

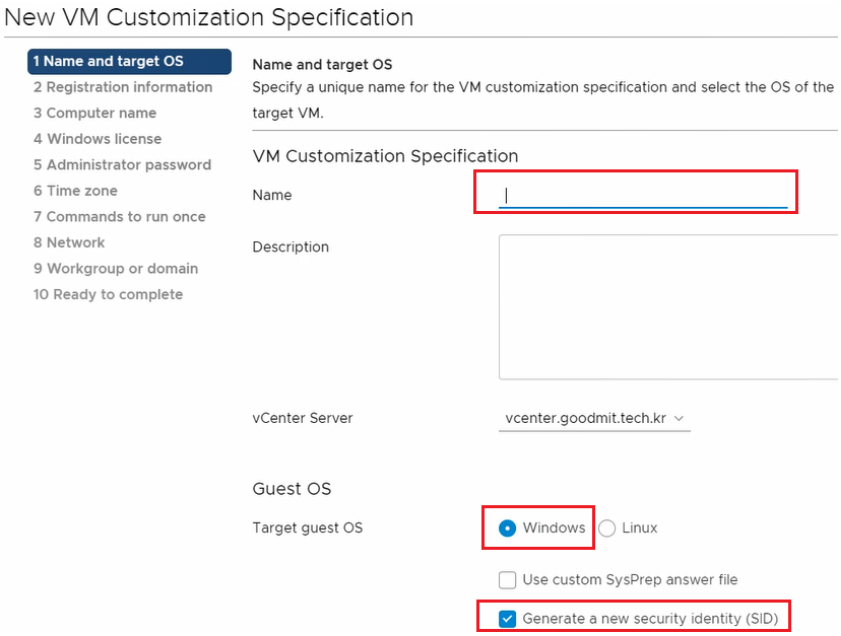
DriveSerialNumber 기반 분석

본 항목에서는 DriveSerialNumber를 중심으로 수집된 악성 바로가기 샘플 간의 상호 연관성을 분석했다. 그 결과, VolumeID와 DriveSerialNumber는 동일하지만 MachineID, MAC 주소, 사용자 SID가 서로 다른 샘플 그룹이 확인되었다.



[그림 35] 악성 샘플 간 제작 환경 유사성 시각화 – DriveSerialNumber 기준

이때, SID는 운영체제에서 사용자 계정 생성 시 자동으로 부여되는 고유 식별자임에도 불구하고 동일한 디스크 식별자를 가지고 있지만, SID만 다른 것은 공격자가 vSphere와 같은 가상화 관리 서버를 이용해 하나의 템플릿 이미지를 복제한 뒤 사용자 SID 값만을 변경해 다수의 제작 환경을 구성했을 가능성을 시사한다.



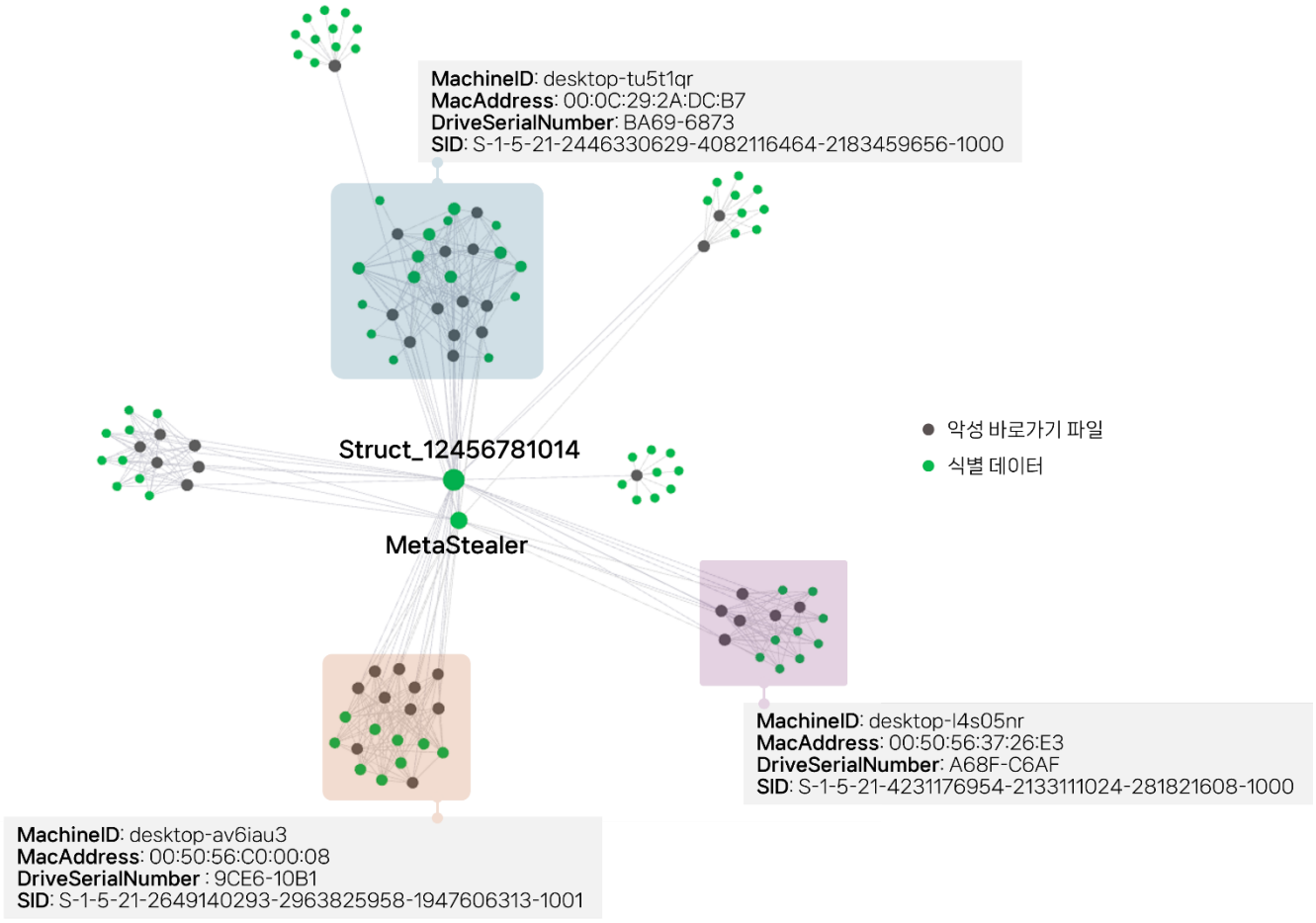
[그림 36] 가상화 서버(vSphere)를 활용한 템플릿 복제 시 SID 변경 옵션

3) LinkFlags 구조와 악성파일 제작 환경 교차 분석

앞서 분석에서는 LinkFlags 구조를 기준으로 악성 바로가기 파일을 유형화하여 공격자가 동일한 제작 프로그램이나 자동화 스크립트를 사용했을 가능성을 도출했다. 또한, FileReference, 사용자 SID, DriveSerialNumber 등의 신뢰도가 높은 주요 식별 데이터를 기준으로 MAC 주소, MachineID, FileID, VolumeID 등의 식별 데이터와 교차 분석하며 동일한 제작 환경의 반복 사용 정황도 확인할 수 있었다.

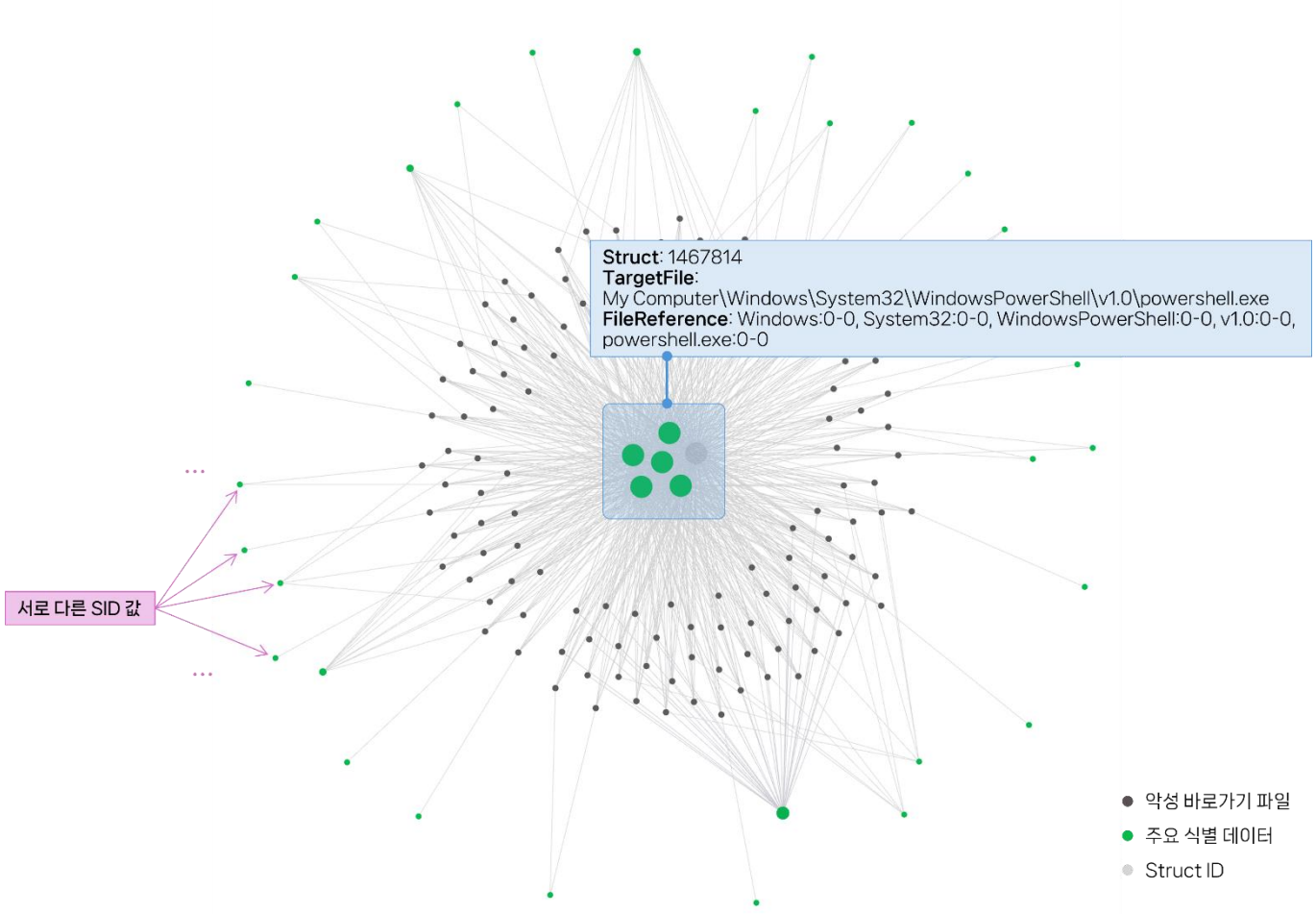
본 항목에서는 두 가지 분석 관점을 결합해 바로가기 파일 LinkFlags 구조와 제작 환경 정보 간의 상호 관계를 교차 분석했다. 이를 통해 공격자가 정형화된 제작 템플릿을 다양한 환경에 배포해 사용하는지, 혹은 환경을 특정적으로 반복 사용하는지 등 악성 파일 생성 방식의 특징을 파악하고자 했다.

분석 결과, 먼저 다음과 같이 LinkFlags 구조는 동일하지만 DriveSerialNumber, 사용자 SID, MAC 주소, MachineID 등 주요 식별 데이터는 모두 상이한 사례가 확인됐다. 이는 공격자가 동일한 제작 스크립트나 프로그램을 여러 개의 시스템 환경에 배포해 악성 파일을 생성한 것으로 해석된다.



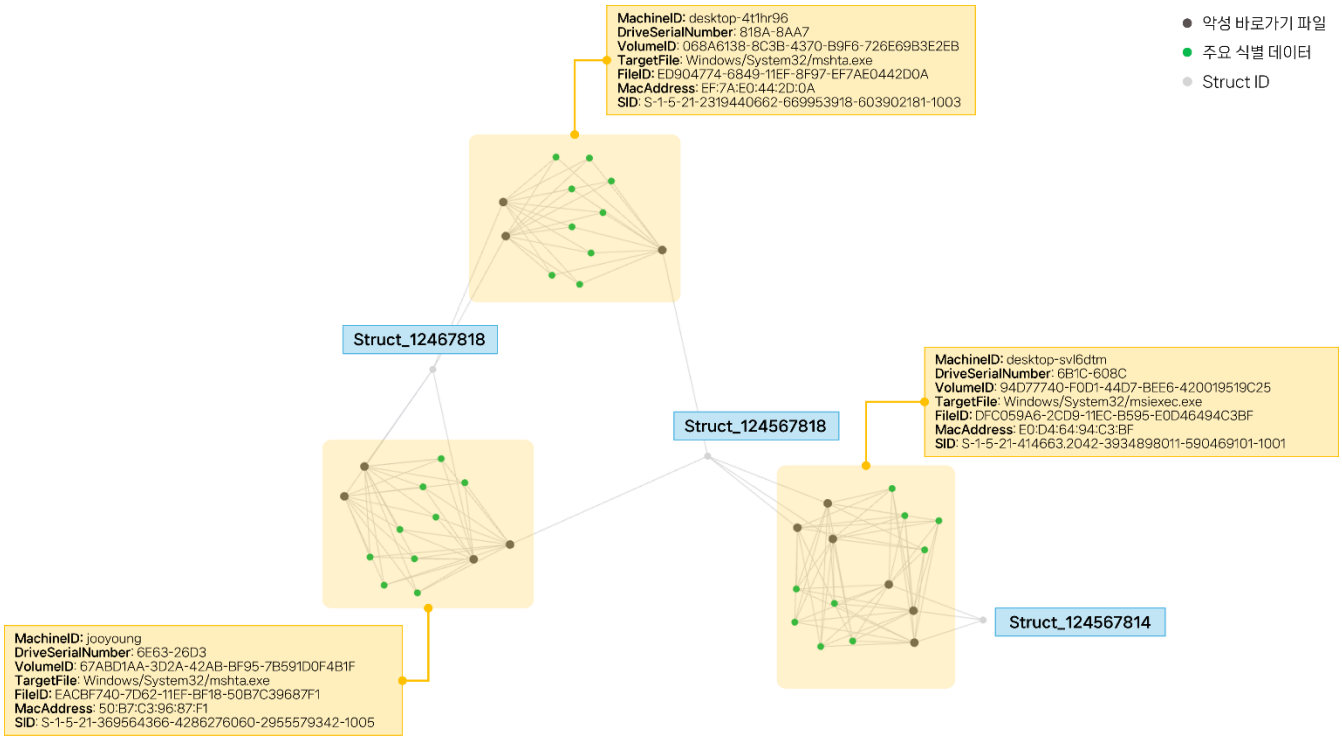
[그림 37] 악성 샘플 간 제작 환경 유사성 시각화 – LinkFlags 및 식별 데이터 교차 분석 (1)

또한, 동일한 Struct ID인 'Struct_1467814' 구조를 가지는 다수의 악성 바로가기 파일들의 LinkTargetIDList를 분석한 결과, 'My Computer\Windows\System32\WindowsPowerShell\v1.0\powershell.exe'를 참조하고 있었다. 그러나 이들의 FileReference 값은 모두 0-0으로 설정되어 있었고, 사용자 SID는 모두 상이한 값으로 확인됐다. 이는 악성 파일 제작 도구가 FileReference 값을 의도적으로 '0-0'으로 처리하는 기능을 포함하고 있으며, 사용자 SID는 서로 다른 것으로 보아 다른 시스템에서 파일이 생성되었음을 시사한다. 즉, 공통된 구조와 공통된 실행 경로를 가지지만 서로 다른 환경에서 생성된 악성 바로가기 파일들로 추정된다.



[그림 38] 악성 샘플 간 제작 환경 유사성 시각화 – LinkFlags 및 식별 데이터 교차 분석 (2)

추가적으로, 사용자 SID, FileReference, VolumeID, FileID, DriveSerialNumber, MAC 주소, MachineID 등 모든 주요 식별 데이터가 동일함에도 불구하고 LinkFlags 구조가 상이한 사례도 일부 확인되었다. 이는 하나의 제작 환경 내에서 서로 다른 바로가기 템플릿을 사용해 악성 바로가기 파일을 생성한 사례로 판단되며, 공격자가 제작 환경은 동일하게 유지하면서 배포 대상이나 목적에 따라 바로가기 파일의 구조만 변경했을 가능성을 보여준다.



[그림 39] 악성 샘플 간 제작 환경 유사성 시각화 – LinkFlags 및 식별 데이터 교차 분석 (3)

3.3. 북한 배후 공격 그룹 대상 샘플 적용

본 절에서는 3.1절에서 정의한 주요 식별 데이터와 3.2절에서 수립한 분석 방법론을 바탕으로 북한 배후 공격 그룹이 사용하는 악성 바로가기 파일의 구조적 특징 및 제작 환경의 특징을 도출하고자 한다.

전체 수집된 1,135개의 악성 바로가기 샘플 중에서 보안 블로그, 분석 보고서, 위협 인텔리전스 플랫폼 등에서 공격 그룹이 명시된 샘플을 선별한 결과, 북한 배후로 식별된 샘플은 총 215개였다. 이 중 공격 그룹별로 Kimsuky 75건, Konni 61건, Scarcruft 70건, 기타 그룹이 명시되지 않은 건 9건으로 나타났다.

[표 26] 수집된 악성 바로가기 샘플 중 북한 배후 공격 그룹별 개수

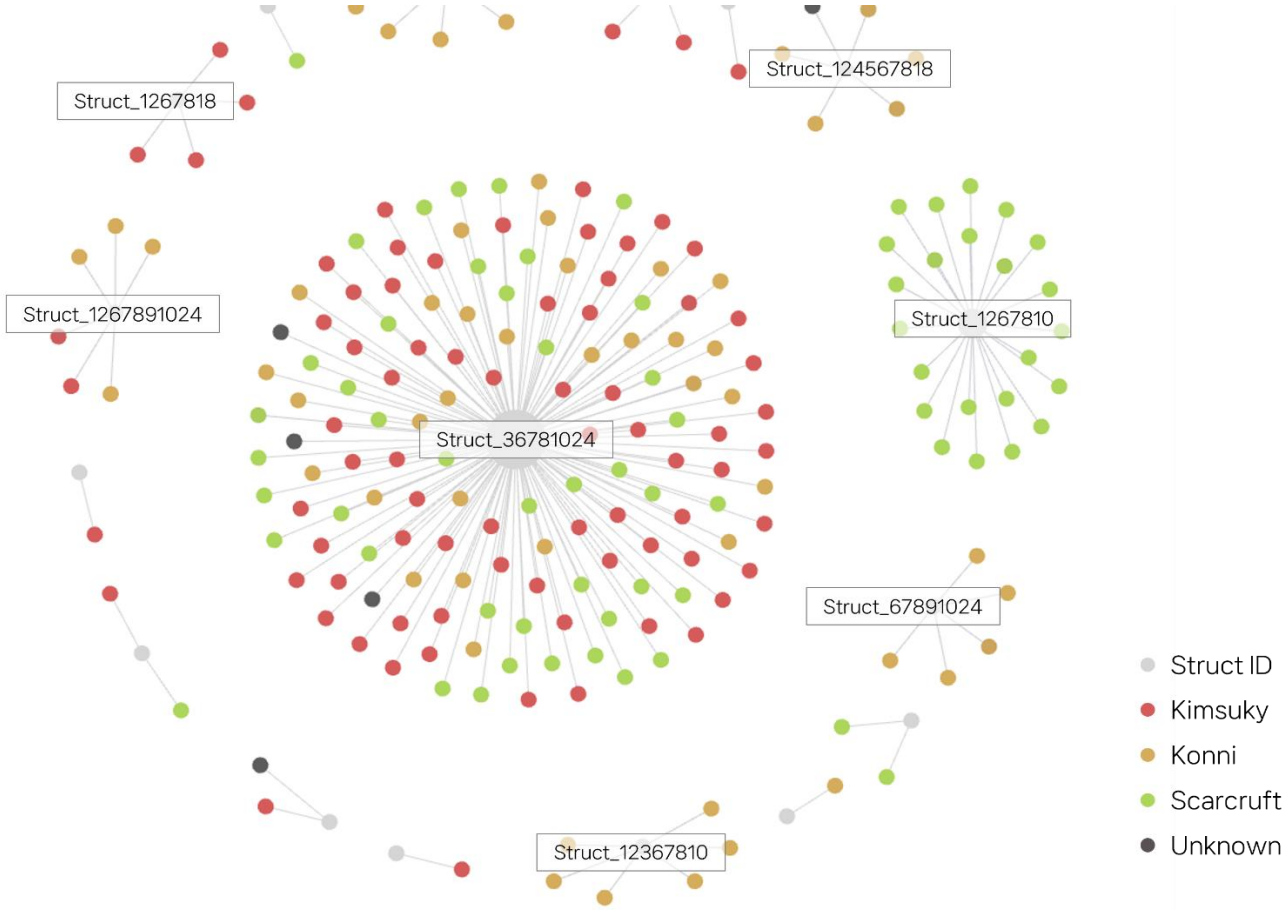
Kimsuky	Konni	Scarcruft	Unknown	총합
75개	61개	70개	9개	215개

참고로 각 공격 그룹의 명칭은 보안 업체 및 위협 인텔리전스 기관에 따라 APT37, APT43, Velvet Cheonlima, Reaper 등으로 다양하게 지칭되기도 하나, 본 보고서에서는 분석의 일관성과 독자의 이해를 고려하여 각각 Kimsuky, Konni, Scarcruft라는 명칭으로 통일하여 표기한다.

분석 방법은 앞서 3.2절과 동일하게 (1) LinkFlags 구조 기반 분석, (2) 악성 바로가기 파일 제작 환경 정보 기준 분석, (3) LinkFlags 구조와 제작 환경 정보 간의 교차 분석으로 구성하고, 추가적으로 북한 배후 공격 그룹 특유의 바로가기 파일 내 속성 값과 같이 주요 구조 외에 관찰된 속성 값들도 추가로 분석해 특징을 도출한다.

1) LinkFlags 구조 기반 분석

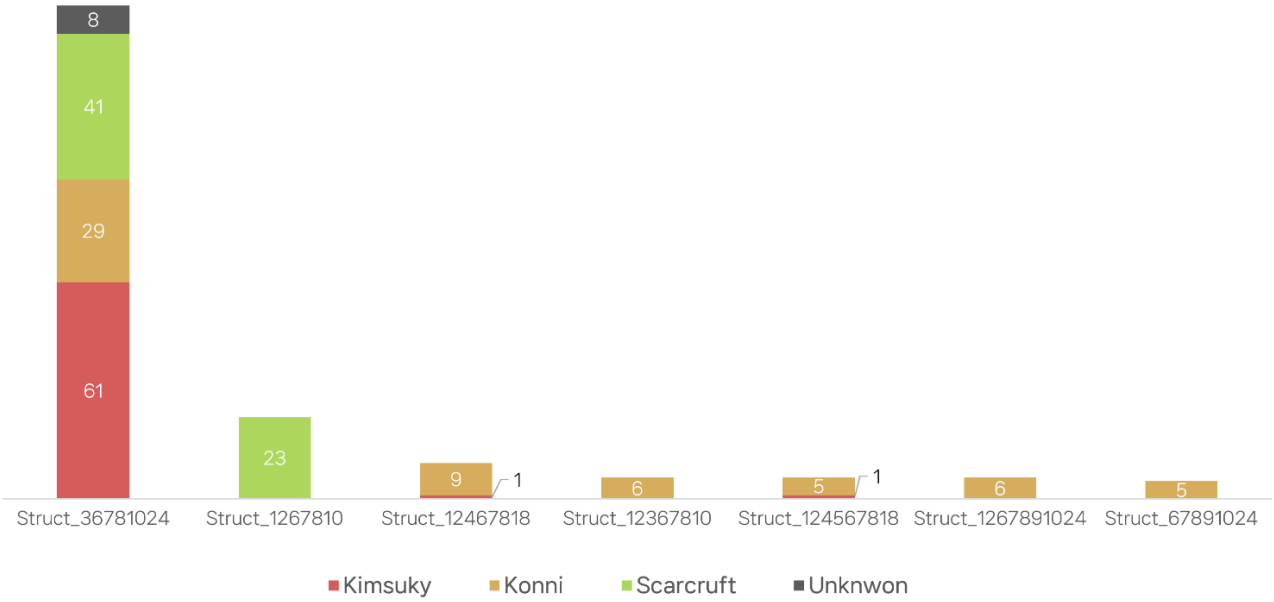
앞서 3.2절에서 전체 악성 샘플을 대상으로 수행한 분석과 동일한 방식으로 본 항목에서는 북한 배후 공격 그룹이 활용한 악성 바로가기 파일 215건에 대해 LinkFlags 구조 유형을 분석했다. 각 샘플의 LinkFlags 조합을 기준으로 유형화한 후, 해당 샘플이 어느 공격 그룹에 해당하는지에 따라 색상으로 구분했다.



[그림 40] 북한 배후 공격 그룹 악성 샘플의 LinkFlags 구조 유형 분포

분석 결과, 전체 215개 샘플 중 다수는 'Struct_36781024' 형태의 LinkFlags 조합을 반복적으로 사용하고 있는 것으로 나타났다. 이 구조는 Kimsuky, Konni, Scarcruft 세 그룹의 샘플 모두에서 공통적으로 확인되었으며, 이는 북한 배후 공격 그룹이 동일하거나 유사한 제작 도구 또는 자동화된 스크립트를 기반으로 악성 바로가기 파일을 생성하고 있음을 시사한다.

한편, 일부 LinkFlags 조합은 특정 그룹에서만 식별되는 특이성을 보였다. 예를 들어 'Struct_1267810' 구조는 Scarcruft 그룹에서만, 'Struct_12467818' 구조는 Konni 그룹의 샘플에서만 확인되었다. 이러한 고유한 LinkFlags 구조는 해당 공격 그룹이 독자적인 제작 방식을 유지하고 있을 가능성을 뒷받침하며, LinkFlags 조합 유형이 그룹별로 상이하게 분포된다는 점에서 공격 그룹 식별에 있어 중요한 구조적 단서로 활용될 수 있다.

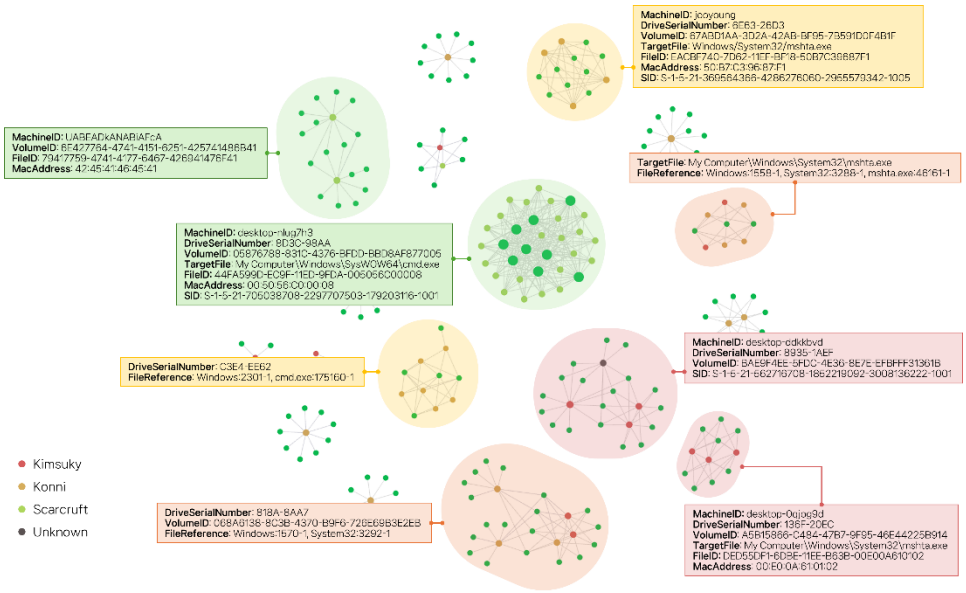


[그림 41] 북한 배후 공격 그룹 악성 샘플의 LinkFlags 구조 유형 통계

2) 악성 바로가기 파일 제작 환경 정보 기준 분석

이번 항목에서는 바로가기 파일의 제작 환경을 식별할 수 있는 사용자 SID, FileReference, DriveSerialNumber 등의 데이터를 기준으로 분석을 수행했다. 이 분석을 통해 북한 공격 그룹이 동일한 제작 환경을 반복적으로 사용하는지, 다양한 시스템에서 제작하는지 파악하고자 했다.

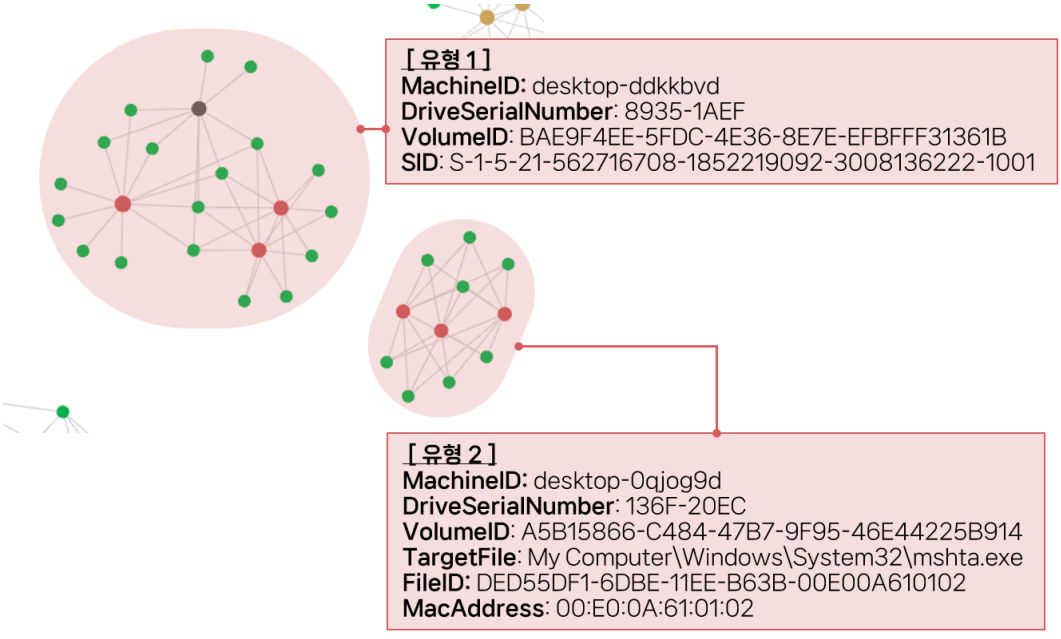
분석 결과, 공격그룹별로 다음과 같은 특징들이 확인되었다.



[그림 42] 북한 배후 공격 그룹 악성 샘플의 동일 주요 식별 데이터 유형 분포

Kimsuky 공격 그룹

Kimsuky 공격 그룹이 제작한 악성 바로가기 샘플의 제작 환경을 분석한 결과, 제작 환경 정보를 기준으로 두 가지 유형으로 분류되는 사례가 확인되었다. 각 유형은 MachineID, VolumeID, DriveSerialNumber, 사용자 SID, MAC 주소 등 주요 시스템 정보가 동일하게 반복되는 특징을 보여주며, 이를 통해 해당 샘플들이 동일한 시스템에서 제작되었음을 유추할 수 있다.



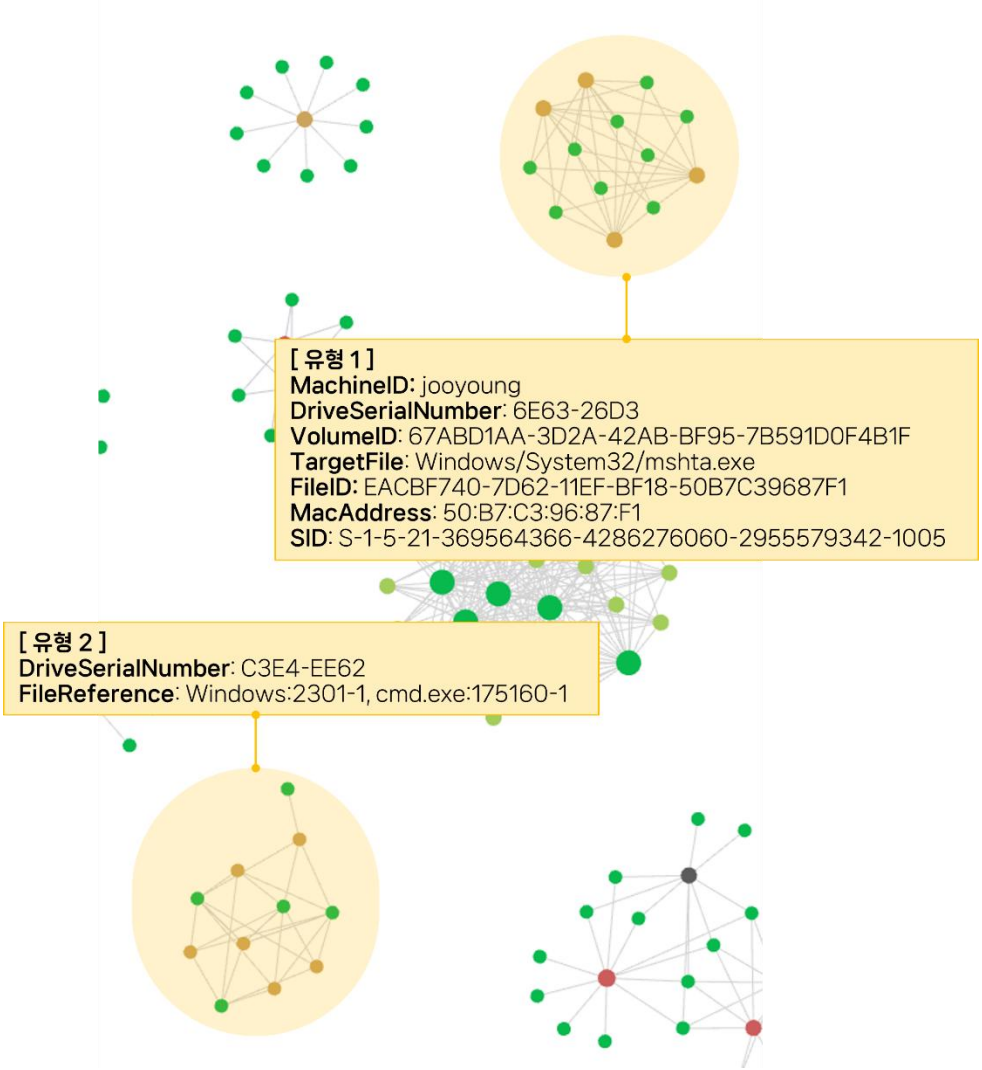
[그림 43] Kimsuky 공격 그룹의 제작 환경 유형

첫 번째 유형은 MachineID, DriveSerialNumber, VolumeID, 사용자 SID가 모두 샘플들로 구성되어 있으며, 이는 동일한 로컬 계정과 디스크 환경에서 파일이 생성되었음을 시사한다. 이러한 정황은 동일한 물리적 시스템 혹은 가상머신에서 해당 악성파일이 제작되었을 가능성이 있다.

두 번째 유형은 MachineID, MAC 주소와 바로가기 파일이 가리키는 대상 파일 경로, VolumeID, DriveSerialNumber가 모두 동일한 환경에서 생성된 사례들이다. 특히 이 유형에서 MAC 주소의 OUI를 확인한 결과, 'DIVA, INC.'로 등록된 네트워크 인터페이스 제조사로 식별되었다. 이는 일반적인 가상머신의 MAC 주소 대역과는 다르며, 해당 샘플들이 가상환경이 아닌 공격자가 실제로 사용하는 물리 시스템에서 제작되었을 가능성을 시사한다.

Konni 공격 그룹

Konni 공격 그룹이 활용한 악성 바로가기 샘플의 제작 환경 정보를 분석한 결과, 구조적으로 유사한 환경에서 제작된 것으로 보이는 두 가지 주요 유형이 확인되었다.



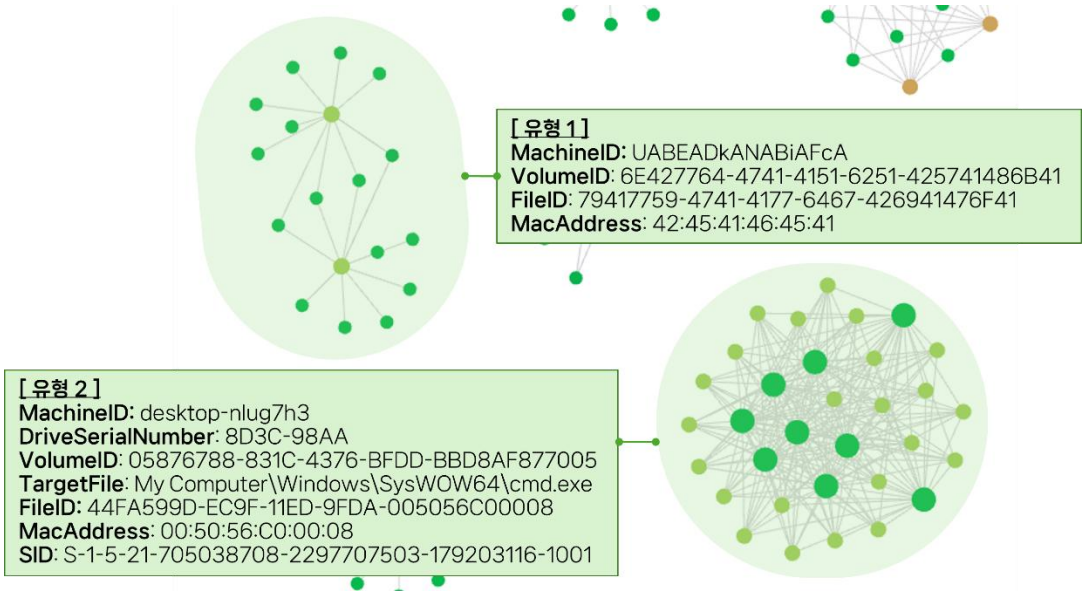
[그림 44] Konni 공격 그룹의 제작 환경 유형

첫 번째 유형의 샘플들은 MachineID, DriveSerialNumber, VolumeID, MAC 주소, 사용자 SID 등이 모두 동일하게 설정되어 있다. 바로가기 파일이 가리키는 대상 프로그램 역시 'System32\mshta.exe'로 동일하며, FileID 값과 FileReference 역시 일치한다. 이처럼 제작 환경에 대한 주요 식별 데이터가 일치한다는 것은 해당 악성파일들이 동일한 시스템에서 해당 악성파일이 제작되었을 가능성을 의미한다. 특히 MAC 주소의 OUI를 확인한 결과, 해당 주소의 네트워크 인터페이스 카드는 삼성전자에서 제작한 것으로 확인됐다. 이는 해당 샘플이 가상 환경이 아닌 공격자가 실제로 사용하는 물리 시스템에서 제작되었을 가능성을 시사한다.

두 번째 유형의 샘플들은 DriveSerialNumber와 FileReference 정보가 동일하나, 바로가기 파일이 가리키는 대상 경로가 System32와 SysWOW64로 차이를 보인다. 디스크를 식별하는 값과 FileReference 값이 일치하므로, 동일한 시스템에서 제작되었다고 볼 수는 있겠으나, 가상 환경의 동일한 템플릿을 복제해 활용하고 있을 가능성도 존재한다.

Scarcruft 공격 그룹

Scarcruft 공격 그룹이 제작한 악성 바로가기 파일을 분석한 결과, 마찬가지로 제작 환경 정보를 기준으로 두 가지 주요 유형의 사례가 확인되었다.



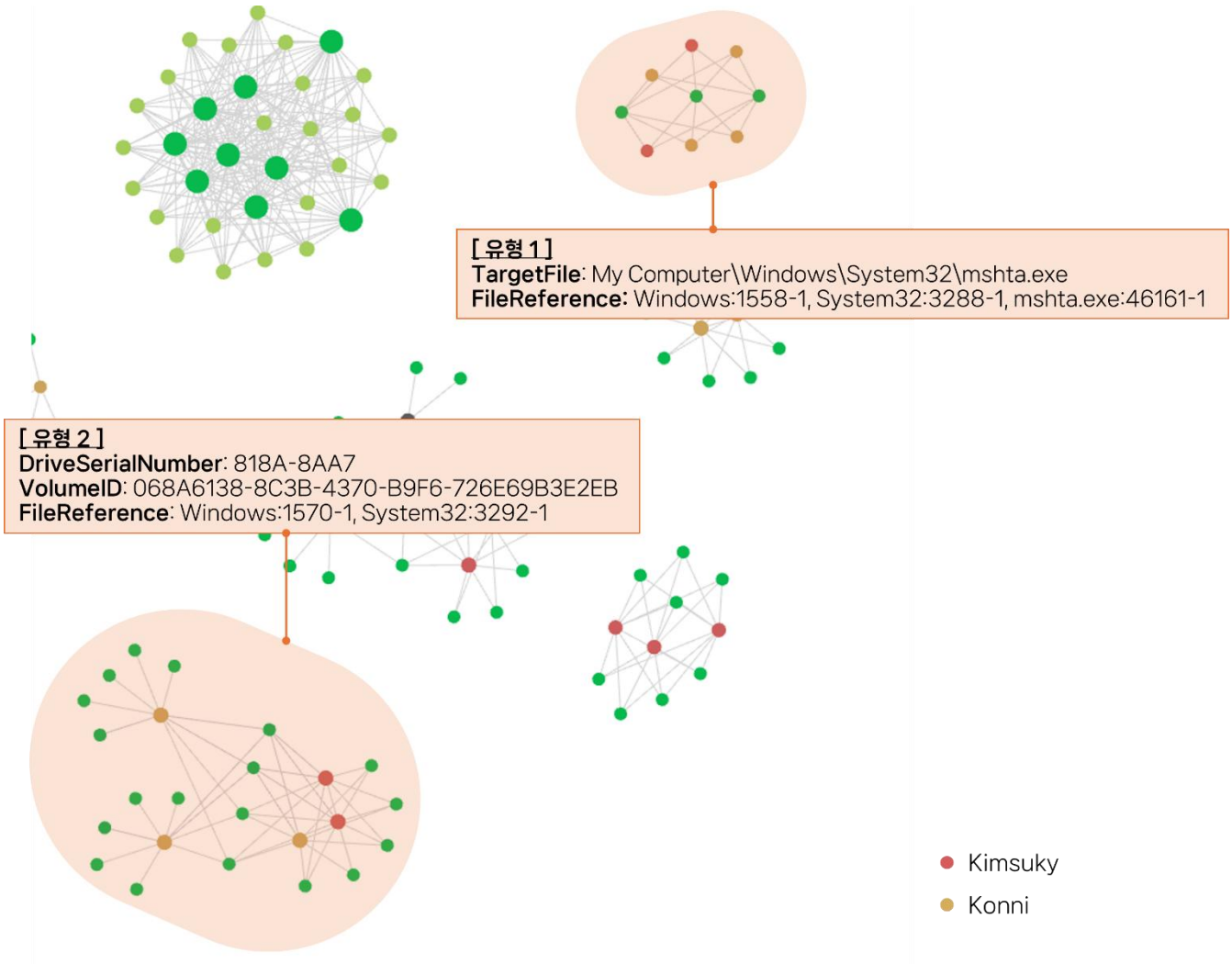
[그림 45] Scarcruft 공격 그룹의 제작 환경 유형

첫 번째 유형의 샘플은 MachineID, VolumeID, FileID, MAC 주소 등의 값이 모두 동일하며, 특히 MAC 주소는 ASCII 코드 기반으로 변조된 것으로 추정된다. 이 주소는 OUI 등록 정보에서는 확인되지 않아, 공격자가 제작 환경의 식별 값을 조작함으로써 분석가의 추적을 회피하려는 의도일 것으로 판단된다. 또한, 동일한 경로의 프로그램을 가리키고 있으나 FileReference가 상이하며, DriveSerialNumber 값도 다른 것으로 미루어보아 서로 다른 환경에서 제작되었음을 나타낸다. 따라서 이 유형의 샘플은 공격자가 여러 제작 환경을 활용하되, 구조 식별 값을 의도적으로 일치시키거나 변조해 흔적을 감추려는 시도가 있었던 것으로 판단된다.

두 번째 유형의 샘플은 MachineID, VolumeID, DriveSerialNumber, 대상 프로그램 및 FileReference, MAC 주소, 사용자 SID 등 거의 모든 주요 식별 값이 동일하다. 특히, MAC 주소는 VMware Inc.에 등록된 OUI로, 해당 샘플이 가상 환경에서 제작되었을 가능성을 시사한다. 이러한 특성은 동일한 가상머신 이미지를 활용했거나, 동일한 템플릿으로 생성된 가상머신을 복제하여 사용한 것으로 해석할 수 있으며 이를 통해 Scarcruft 그룹은 악성파일 제작 시 가상 환경을 주로 활용하고 있는 것으로 추정된다.

기타

악성 바로가기 파일의 구조 정보를 기반으로 공격그룹별 제작 환경을 분석하는 과정에서 일부 공격 그룹 간 동일한 가상머신 템플릿을 사용하는 것으로 추정되는 정황이 확인됐다. 다음과 같이 Kimsuky와 Konni로 분류된 악성 샘플 간에 제작 환경의 주요 식별 정보가 일치한 사례가 확인되었다.



[그림 46] Kimsuky와 Konni 공격 그룹의 인프라 공유 추정

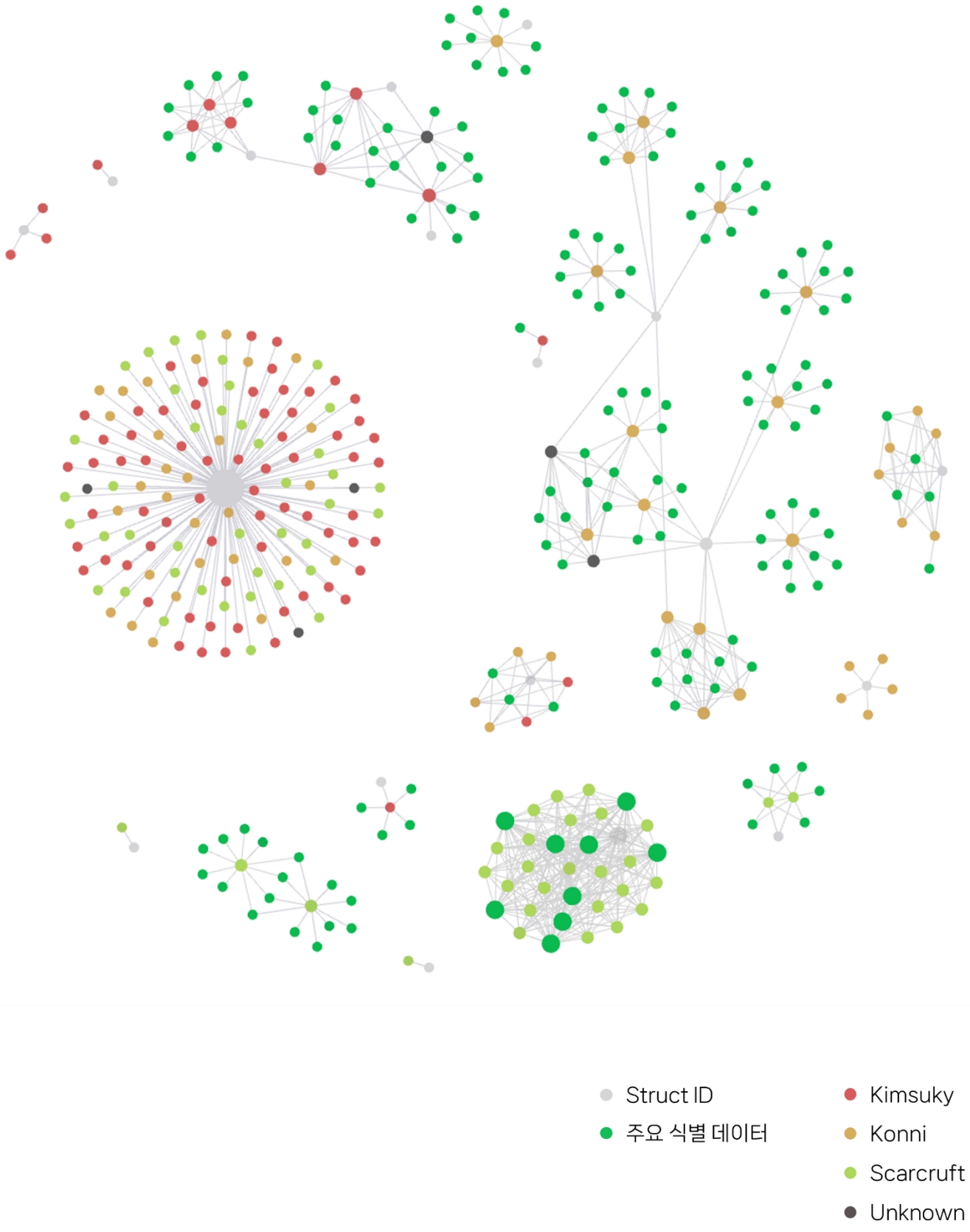
먼저, 첫 번째 유형의 샘플들에서는 바로가기 참조 대상 파일의 경로와 FileReference가 완전히 동일했으나, MachineID, 사용자 SID, MAC 주소 등 추가적으로 확인할 수 있는 제작 환경 식별 데이터가 포함되지 않았다. 이러한 정황은 동일한 제작 도구 또는 자동화 스크립트를 통해 생성된 바로가기 파일일 가능성을 시사하지만, 보다 정확한 판단을 위해서는 누락된 환경 식별 정보에 대한 보완 데이터 확보가 필요하다.

두 번째 유형의 샘플들은 동일한 DriveSerialNumber와 VolumeID, FileReference 구조를 가지고 있어 이는 해당 악성 샘플들이 동일한 디스크 이미지를 기반으로 제작되었을 가능성을 보여준다. 반면, MachineID, MAC 주소, 사용자 SID 값은 서로 상이한 것으로 미루어보아 가상 환경 내 동일한 템플릿을 기반으로 하되, 운영체제 계정 및 네트워크 설정 등 일부 구성 요소만 변경해 사용한 정황으로 해석할 수 있다.

3) LinkFlags 구조와 제작 환경 정보 간의 교차 분석

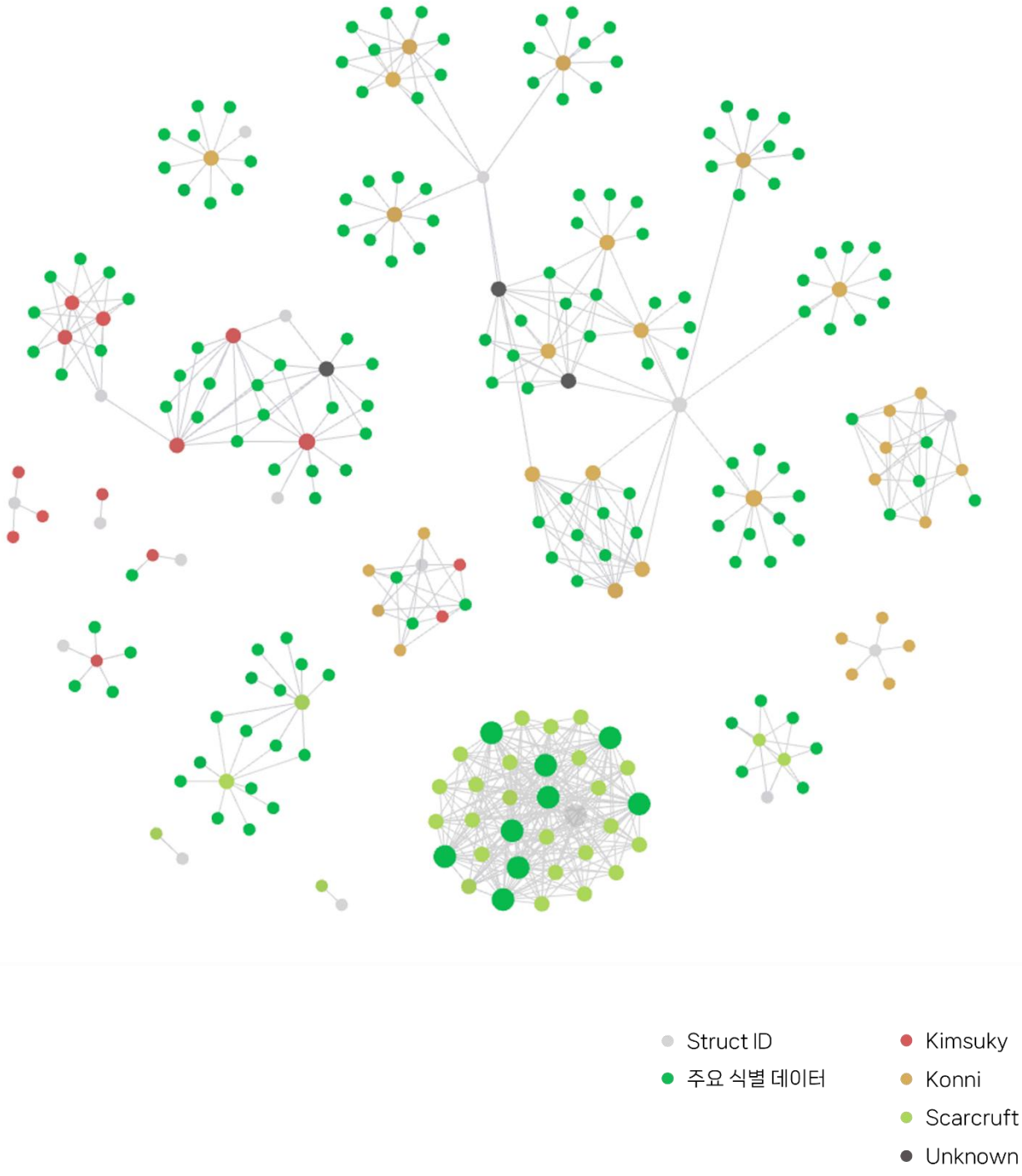
악성 바로가기 파일의 LinkFlags 구조와 제작 환경 정보를 결합해 교차 분석한 결과 북한 배후 공격 그룹이 활용하는 악성 바로가기 파일의 특징이 보다 명확하게 드러나는 것을 확인할 수 있었다. 다음 그림은 각 샘플의 LinkFlags 구조에 기반해 구조 유형별로 시각화하고 동시에 제작 환경 식별 데이터를 함께 매핑하여 그룹화한 결과이다.

우선적으로 본 절의 '(1) LinkFlags 구조 기반 분석' 항목에서 언급한 바와 같이 북한 배후의 공격 그룹들의 악성 바로가기 샘플들에서는 Struct_36781024 구조를 가지는 경우가 가장 많이 확인되었고, 이는 공격 그룹간 동일하거나 유사한 제작 도구 또는 자동화된 스크립트를 기반으로 악성 바로가기 파일을 생성하고 있음을 시사한다.



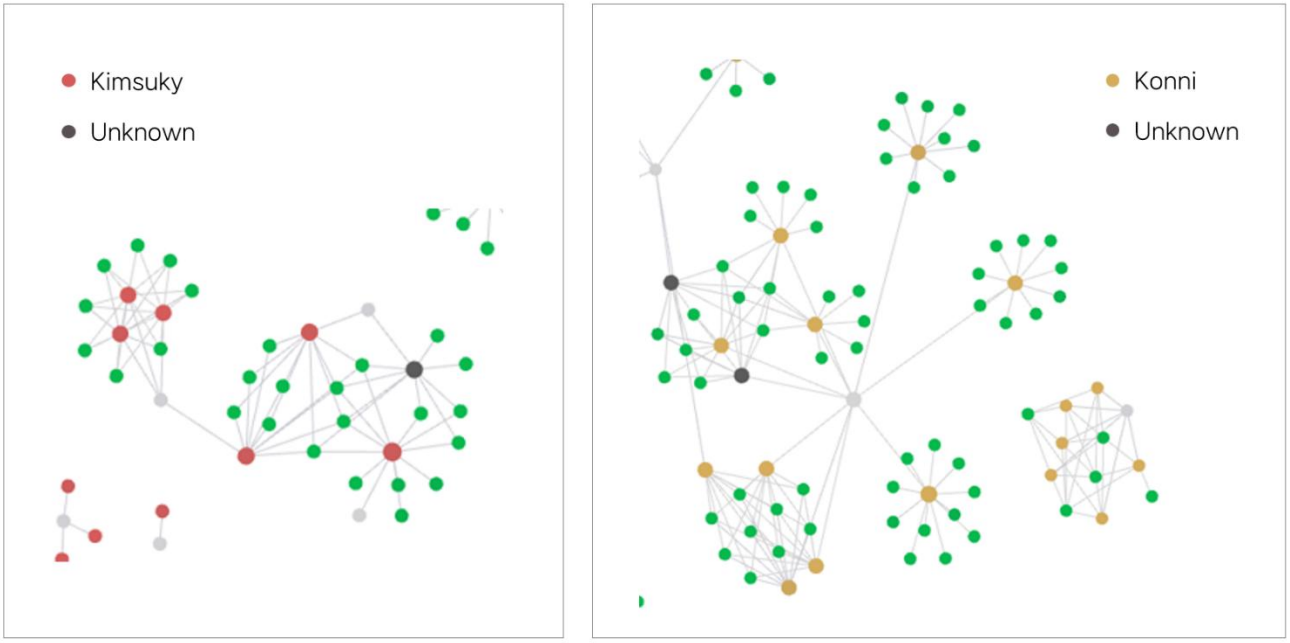
[그림 47] LinkFlags 구조 및 제작 환경 정보 간의 교차 분석 결과 (1)

또한, 제작 환경에 대한 주요 식별 데이터를 포함하지 않는 Struct_36781024 구조를 제외하고 분석한 결과 공격 그룹 간의 연관성이 보다 명확하게 드러나는 것을 확인할 수 있었다. Kimsuky, Konni, Scarcruft 등 주요 북한 배후 공격 그룹들은 서로 다른 LinkFlags 구조를 활용했으며, 그 구조에 대응하는 고유한 제작 환경 정보도 함께 반복적으로 나타났다.



[그림 48] LinkFlags 구조 및 제작 환경 정보 간의 교차 분석 결과 (2)

또한, 일부 샘플은 최초 분류 시 공격 그룹이 식별되지 않았으나, 해당 샘플이 사용하는 LinkFlags 구조와 제작 환경을 식별할 수 있는 주요 데이터를 종합적으로 분석한 결과, 기존 공격 그룹과 높은 유사성을 보이는 사례도 확인되었다. 이를 통해 LinkFlags 구조와 제작 환경 정보의 결합은 단일 지표만으로는 확인이 어려운 공격 그룹 식별에도 유효하게 활용될 수 있음을 시사한다.



[그림 49] LinkFlags 구조 및 제작 환경 정보를 활용한 Unknown 샘플의 그룹 식별 사례

4) 기타 특징 분석

다음은 수집한 215개의 샘플에서 식별된 바로가기 파일의 NAME_STRING 분포이다. NAME_STRING 값은 바로가기 파일의 표시 이름을 저장하는 문자열로, 파일 속성 정보의 '설명(Description)' 항목에 해당한다. 북한 공격 그룹의 악성 바로가기 샘플 중 98개에서 NAME_STRING 값이 설정되어 있는 것으로 확인되었으며, 포함된 문자열별 비율은 다음과 같다.

[표 27] 북한 배후 공격 그룹의 악성 바로가기 샘플의 NAME_STRING 값 통계

번호	NAME_STRING 값	빈도 수
1	hwp File	33
2	Type: Text Document Size:5.23KB Datemodified:01/02/202011:23	22
3	Type: Hangul Document Size:2.84KB Datemodified:10/20/202311:23	11

번호	NAME_STRING 값	빈도 수
12	test	1
13	Text File	1
14	Type: BMP File Size:358KB Datemodified:04/20/202411:23	1

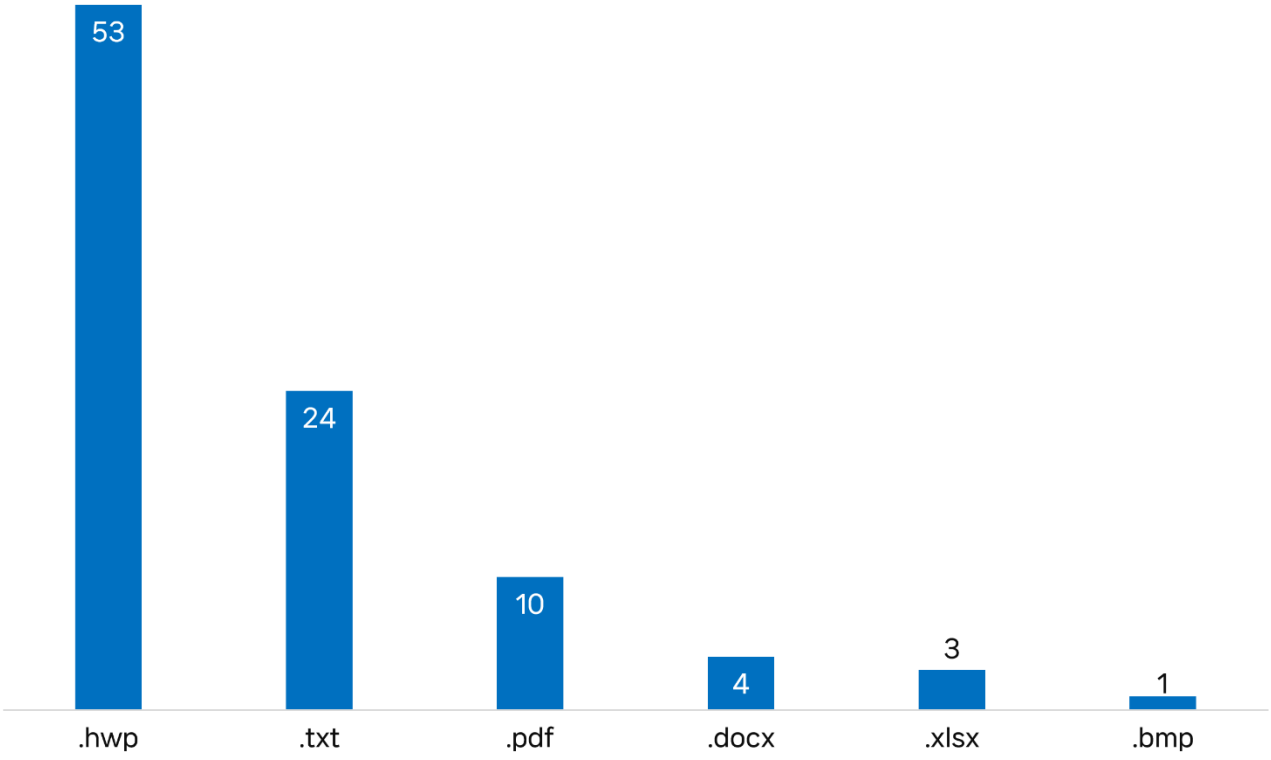
4	Type: PDF File Size:358KB Datemodified:04/20/202411:23	7
5	docx File	4
6	xlsx File	3
7	PDF File	2
8	Type: HWP 2018 Document Size:137KB Datemodified:03/31/202414:51	2
9	Type: HWP 2022 Document Size:27KB Datemodified:03/23/202514:51	2
10	[*] CB Tech	1
11	Date modified : 29/07/2025	1

15	Type: HWP 2018 Document Size:159KB Datemodified:03/31/202414:51	1
16	Type: HWP 2022 Document Size:1.4MB Datemodified:05/23/202414:51	1
17	Type: HWP 2022 Document Size:137KB Datemodified:05/23/202414:51	1
18	Type: HWP 2022 Document Size:140KB Datemodified:05/23/202414:51	1
19	Type: HWP 2022 Document Size:27KB Datemodified:05/23/202414:51	1
20	Type: PDF Document Size:559KB Datemodified:01/24/202310:23	1
21	Type: Text Document Size:5.23KB Datemodified:02/20/202311:23	1

식별된 NAME_STRING 값 중 가장 많이 관찰된 형태는 다음과 같다. 이들 문자열은 실제 바로가기 파일의 크기나 수정날짜와는 무관하게 사용자가 신뢰할만한 정상 문서로 인식하도록 자연스러운 크기와 최근 수정일을 표시해 클릭을 유도한다.

> Type : {위장된 파일 형식}
> Size : {위장된 파일의 크기}
> Date modified : {수정 일자}

또한, 위장된 파일 형식별 분포를 분석한 결과 한글 문서 파일(hwp) 형식이 53건으로 가장 많았으며, 그 뒤로 텍스트 파일, PDF, 워드, 엑셀 파일 순으로 확인되었다. 이러한 결과는 북한 배후 공격자들은 우리나라 문서 환경에 맞춰 공격을 수행하고 있음을 시사한다.



[그림 50] NAME_STRING 값을 통해 위장된 파일 형식별 분포도

4. 결과 및 시사점

본 연구는 Windows 바로가기 파일의 구조적 특성과 파일 구조 내 바로가기 파일을 생성한 PC의 정보를 식별할 수 있는 데이터를 기반으로 공격 그룹의 식별 및 프로파일링이 가능한지에 대한 분석 방법론을 수립하고 이를 북한 배후 공격 그룹이 사용하는 악성 샘플을 대상으로 검증했다.

연구 결과, 바로가기 파일의 내부 구조만으로도 악성 여부를 상당 부분 판단할 수 있는 정형화된 패턴이 존재한다는 것을 확인했다. 이러한 분석 방식은 안티 바이러스, EDR 등이 파일 실행 이전 단계에서 악성 행위를 탐지하거나 차단하는데 활용될 수 있다. 특히 LinkFlags 구조와 제작 환경 정보를 함께 분석함으로써 단일 악성 파일 분석에서 벗어나 공격자의 제작 습관 및 환경적 구성까지 식별할 수 있음을 확인했다.

또한, 일부 제작 환경 정보가 서로 다른 공격 그룹으로 분류된 샘플 간에서도 반복적으로 확인된 사례가 존재했다. 이는 공격 그룹 간 제작 인프라가 공유되었거나, 동일한 인물이 여러 그룹의 악성 파일 제작에 관여했을 가능성을 시사하며, 공격 배후의 조직적 연계성을 분석할 수 있는 간접적 단서로 해석된다.

다만, 동일 제작 환경의 반복이 반드시 동일한 공격 주체를 의미하는 것은 아니며, 악성 파일 제작 환경 공유, 재사용된 제작 스크립트 등 다양한 해석 가능성이 존재한다. 따라서 이러한 유사성은 공격 그룹 간의 연계성 추정에 참고 지표로 활용하되, 정량적·정성적 분석을 병행한 다각적 검증이 필요하다.

한편, 본 연구는 북한 배후 공격 그룹에 대한 악성 바로가기 샘플 총 216건을 대상으로 검증을 수행하였으나, 샘플 수의 한계로 인해 각 그룹별·시기별 변화를 충분히 반영하지 못한 점은 아쉬움으로 남는다. 향후에는 보다 다양한 시점과 출처의 데이터를 확보하여 그룹별 제작 행태의 진화를 장기적으로 추적하는 연구가 필요할 것으로 판단된다.

그럼에도 불구하고, 본 연구는 바로가기 파일이라는 비교적 단순한 파일 형식을 기반으로 공격자 프로파일링을 수행할 수 있음을 실증적으로 입증하였으며, 기존의 IoC 기반 사후 대응 중심 위협 인텔리전스에서 벗어나 파일 구조 기반의 사전적 위협 식별 접근법을 제시했다는 점에서 의의가 크다.

또한 본 보고서에서 제시한 방법론은 북한 배후 공격 그룹에 한정되지 않고, 추후 러시아, 중국 등 다양한 국가 배후 공격 그룹에도 확장 적용이 가능한 분석 체계로 발전시킬 수 있다. 이를 통해 단일 샘플의 악성 행위 분석을 넘어, 공격자의 제작 환경·행위 패턴·전술적 일관성까지 종합적으로 프로파일링하는 인텔리전스 체계로 확장할 수 있을 것으로 기대한다.

참고문헌

[1] Haizhou Wang, Ashkan Hosseini, Ashutosh Chitwadgi, “Windows Shortcut (LNK) Malware Strategies”, PaloaltoNetworks Unit42, https://unit42.paloaltonetworks.com/lnk-malware/, 2025-07-02

[2] ASEC, “2025년 8월 APT 공격 동향 보고서(국내)”, https://asec.ahnlab.com/ko/90103/, 2025-09-12

[3] ASEC, “2025년 7월 APT 공격 동향 보고서(국내)”, https://asec.ahnlab.com/ko/89578/, 2025-08-14

[4] 한국무역협회, ““사방이 적” 북·중·러發 해킹 82%…새 정부, 사이버안보 컨트롤타워 세워야, https://www.kita.net/board/totalTradeNews/totalTradeNewsDetail.do?no=92368&siteId=1, 2025-06-04

[5] CYBLE, “Quantum Software: LNK file-based builders growing in popularity”, https://cyble.com/blog/quantum-software-lnk-file-based-builders-growing-in-popularity/, 2022-06-22

[6] Rsecurity, “Shortcut-based (LNK) attacks delivering malicious code on the rise”, https://www.resecurity.com/blog/article/shortcut-based-lnk-attacks-delivering-malicious-code-on-the-rise, 2022-07-17

[7] libfwsi, “Windows Shell Item format specification”, https://github.com/libyal/libfwsi/blob/main/documentation/Windows%20Shell%20Item%20format.asciidoc#extension_block_0xbeef004, 2014-12-11

[8] Microsoft, “Distributed Link Tracking and Object Identifiers”, https://learn.microsoft.com/en-us/windows/win32/fileio/distributed-link-tracking-and-object-identifiers, 2021-12-31

[9] P. Leach, M. Mealling, R. Salz, “A Universally Unique IDentifier (UUID) URN Namespace”, Network Working Group, https://datatracker.ietf.org/doc/html/rfc4122, 2005-07

부록 : 북한 배후 공격 그룹의 주요 특징 정보

Kimsuky 공격 그룹

구분	값	빈도 수	비고
LinkFlags	Struct_1236781418	1	
	Struct_12367818	2	
	Struct_12368	1	
	Struct_124567818	1	
	Struct_12467818	1	
	Struct_1256781418	1	
	Struct_1267818	4	
	Struct_136789	1	
	Struct_167814	1	
	Struct_346781024	3	
	Struct_36781024	61	
DriveSerialNumber	0x1aef8935	4	
	0x20ec136f	3	
	0x8aa7818a	2	
	0x0	1	
	0xb412d645	1	
사용자 SID	S-1-5-21-562716708-1852219092-3008136222-1001	4	
	S-1-5-21-2319440662-669953918-603902181-1003	2	
	S-1-5-21-553791927-332201608-4149232231-1001	1	
MAC 주소	00:E0:0A:61:01:02	3	• 제조사 : DIVA, INC.
	BD:A4:3B:CA:E5:38	2	• 제조사 OUI 확인 불가
	C0:35:32:15:70:0E	2	• 제조사 : Liteon Technology Corporation
	EF:7A:E0:44:2D:0A	2	• 제조사 OUI 확인 불가
MachineID	desktop-0qjog9d	3	
	desktop-4t1hr96	2	
	desktop-ddkkbvd	4	

Konni 공격 그룹

구분	값	빈도 수	비고
LinkFlags	Struct_1234567818	1	
	Struct_12367810	6	
	Struct_124567818	5	
	Struct_12467818	9	
	Struct_1267891024	6	
	Struct_36781024	29	
	Struct_67891024	5	
DriveSerialNumber	0x26d36e63	4	
	0x2caf875e	2	
	0x7280f661	1	
	0x8aa7818a	3	
	0x90384211	1	
	0xaec18832	1	
	0xce8e6630	1	
	0xd8f2338c	1	
	0xee62c3e4	6	
	0xfda1026	1	
사용자 SID	S-1-5-21-1084486348-3098408493-2041479114-1002	1	
	S-1-5-21-1840914918-3743596314-2189714932-1002	1	
	S-1-5-21-2289896217-3033761267-3699187338-1000	1	
	S-1-5-21-2319440662-669953918-603902181-1003	1	
	S-1-5-21-2435115830-3329699541-325296845-1018	1	
	S-1-5-21-2471695828-269338241-4285109741-1002	1	
	S-1-5-21-2973277103-947954210-2262777577-1006	1	
	S-1-5-21-3099099793-1800637785-2451162932-1002	1	
	S-1-5-21-369564366-4286276060-2955579342-1005	4	

	S-1-5-21-468297479-1764105027-627121249-1004	1	
	S-1-5-21-876373686-1671278151-1943386753-1002	2	
MAC 주소	00:0E:C6:FC:25:72	1	• 제조사 : ASIX ELECTRONICS CORP.
	00:15:5D:15:0E:B7	1	• 제조사 : Microsoft Corporation
	1C:99:57:1D:D4:D0	2	• 제조사 : Intel Corporate
	24:F5:AA:E4:C0:C8	1	• 제조사 : Samsung Electronics Co.,Ltd
	50:B7:C3:96:87:F1	4	• 제조사 : Samsung Electronics Co.,Ltd
	93:89:6B:C4:2E:43	1	• 제조사 OUI 확인 불가
	A8:A1:59:A9:7B:FE	1	• 제조사 : ASRock Incorporation
	D0:50:99:91:CD:56	1	• 제조사 : ASRock Incorporation
	E0:D5:5E:8B:FB:D6	1	• 제조사 : GIGA-BYTE TECHNOLOGY CO.,LTD.
	EF:7A:E0:44:2D:0A	1	• 제조사 OUI 확인 불가
	FD:FB:10:8C:C7:3E	1	• 제조사 OUI 확인 불가
MachineID	14_g2_itl	2	
	¹Ú¼ºÇÑ	1	
	ÀçÈ°¿îµ¿4	1	
	cy-p1	1	
	desktop-06vseh4	1	
	desktop-0jpcpit	1	
	desktop-4t1hr96	1	
	desktop-6ko8d2u	1	
	desktop-iihqrp1	1	
	jooyoung	4	
	win-kejvo9cld80	1	

Scarcruft 공격 그룹

구분	값	빈도 수	비고
LinkFlags	Struct_124678	2	
	Struct_1267810	23	
	Struct_167814	1	
	Struct_36781024	41	
	Struct_4516171823	1	
	Struct_46781024	1	
DriveSerialNumber	0x8c6d29d7	1	
	0x98aa8d3c	23	
	0xb891595b	1	
사용자 SID	S-1-5-21-553791927-332201608-4149232231-1001	1	
	S-1-5-21-705038708-2297707503-179203116-1001	23	
MAC 주소	00:50:56:C0:00:08	23	• 제조사 : VMware, Inc.
	42:45:41:46:45:41	2	• 제조사 OUI 확인 불가
MachineID	desktop-nlug7h3	23	
	UABEADkANABiAFcA	2	