KARA
Korea Anti Ransomware Alliance

2025.4Q

KARA 랜섬웨어 동향 보고서

KARA 랜섬웨어 동향 보고서

EQST Lab 팀 이호석, 정민수, 조효제, 이현아, 신동석, 이승호

랜섬웨어 트렌드

1. 4 분기 TREND

TREND

- **Weaxor :** React2Shell 취약점(CVE-2025-55182)을 악용
- **Clop :** Oracle E-Business Suite 취약점(CVE-2025-61882)을 악용

THREAT

- **4 분기 Top5 랜섬웨어 :** Qilin, Akira, Sinobi, INC, Clop
- **Clop 랜섬웨어 :** Oracle E-Business Suite 취약점을 악용한 대규모 공격

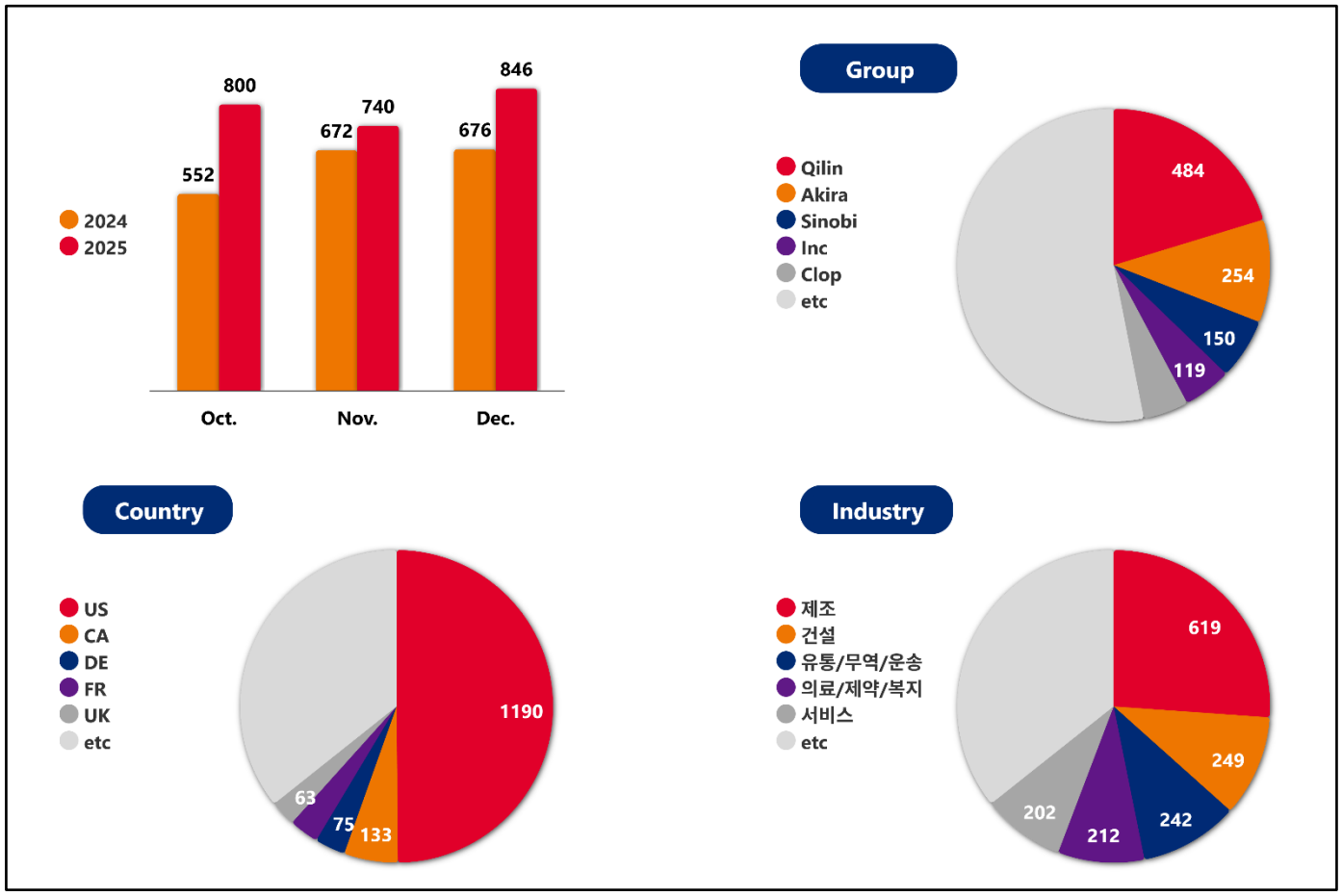
EXPLOIT

- **0-day :** CVE-2025-61882
- **1-day :** CVE-2024-1086, CVE-2025-10035, CVE-2025-55182

TARGET

- Oracle E-Business Suite 취약점을 통한 초기 침투
- 전체 공격 중 제조 26%, 미국 50%

2. 4 분기 랜섬웨어 활동 통계



[그림 1] 2025 년 4 분기 랜섬웨어 그룹 활동

2025 년 4 분기 랜섬웨어 피해 사례는 총 2,386 건으로 2024 년 동기 대비 약 26% 증가했으며, 2025 년 3 분기(1,620 건)와 비교했을 때는 대폭 상승한 수치를 기록했다. 이러한 증가는 주요 랜섬웨어 그룹의 활동이 전 분기 대비 증가하면서 전반적인 공격 건수 증가에 영향을 준 것으로 보인다. 특히 Qilin 그룹은 지난 3 분기(239 건) 대비 약 102% 증가한 484 건의 활동량을 보여주며 4 분기 전체 공격 증가의 주요 요인으로 확인된다.

Qilin 그룹은 2022 년 7 월 Agenda 라는 이름으로 활동하기 시작한 RaaS[1] 그룹으로 올해 2 분기부터 활동이 눈에 띄게 증가한 것으로 나타났다. Qilin 의 증가세는 RansomHub 공격자 일부가 Qilin 그룹에 합류한 영향으로 분석되며, 이러한 상승세는 단기적인 현상에 그치지 않고 2025 년 2 분기부터 4 분기까지 가장 많은 피해 사례를 기록했다.

Akira 랜섬웨어 그룹의 공격으로 발생한 2025 년 4 분기 피해 사례 수는 지난 3 분기(160 건)에 비해 약 59% 증가한 254 건을 기록했다. Akira 그룹은 2025 년 1 분기부터 꾸준히 소프트웨어 취약점을 이용한 초기 침투 전략을 지속하고 있으며, 피해 규모 또한 분기별로 계속해서 확대되고 있다.

[1] RaaS(Ransomware as a Service): 서비스형 랜섬웨어의 약어로, 금전을 대가로 랜섬웨어를 서비스 형태로 제공하는 수익 모델

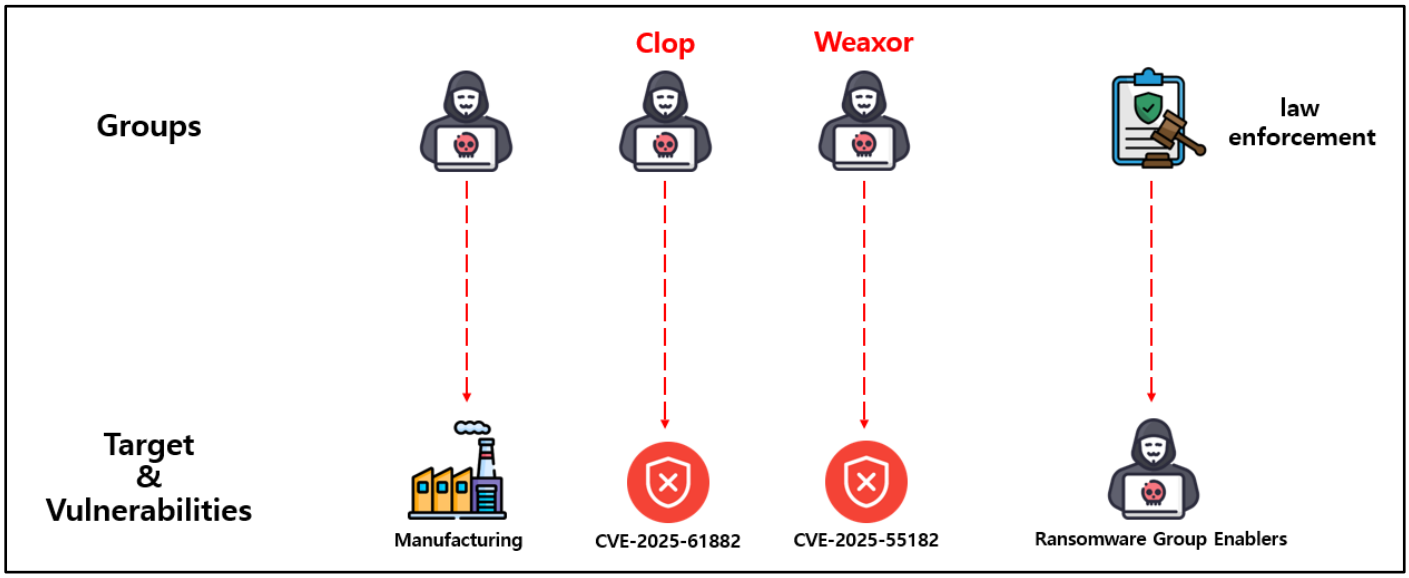
2025 년 4 분기 150 건의 피해 사례가 확인된 Sinobi 그룹은 2025 년 7 월에 등장한 랜섬웨어 그룹이다. 분석 결과, INC–Lynx–Sinobi 간 코드 및 구조적 유사성이 확인되었으며, Lynx 는 INC 와 코드 구성과 기능 흐름에서 유사한 특징을 보였고, Sinobi 또한 Lynx 와 암호화 방식 및 실행 인자 구조가 유사하게 확인되어 동일 계열로 판단된다.

INC 랜섬웨어 그룹은 2023 년 7 월부터 활동을 시작한 그룹으로, 2024 년 4 월 소스코드를 판매하며 활동을 중단한 것처럼 보였으나 이후에도 지속적인 공격 활동이 관찰되었다. 특히 기존 C/C++로 제작된 버전뿐만 아니라 Rust 로 개발된 변종을 활용해 공격을 이어가고 있다. INC 랜섬웨어는 4 분기 119 건의 피해를 기록해 지난 3 분기(136 건) 대비 활동 규모가 소폭 감소했지만 Qilin, Akira, Sinobi 그룹에 이어 네 번째로 많은 공격 건수를 기록한 그룹으로 확인됐다.

Clop 그룹은 2025 년 4 분기 112 건의 피해 사례가 확인되었으며, Oracle E-Business Suite 의 취약점(CVE-2025-61882)을 악용해 다수의 기업을 공격했다. 대부분의 침해는 2025 년 11 월 이전에 이뤄진 것으로 추정되나, Clop 은 11 월 이후 다크웹 유출 사이트를 통해 피해 조직을 일괄 공개하였다. 또한 Clop 그룹은 해당 취약점을 악용한 공격을 현재까지 지속하고 있어, 관련 시스템의 패치 적용이 필수적이다.

2025 년 4 분기 가장 많은 랜섬웨어 공격을 받은 국가는 미국과 캐나다로 나타났다. 미국은 이번 분기 1,190 건의 피해가 집계되어 전 분기 851 건 대비 증가했다. 또한 캐나다는 4 분기 133 건의 피해가 집계되어 전 분기 3 위에서 2 위로 상승했다. 한편, 산업별 피해 현황을 살펴보면 제조업은 지난 분기(401 건) 대비 약 54% 상승한 619 건으로 집계됐으며, 그 뒤를 이어 건설, 유통, 의료, 서비스 순으로 피해가 확인되었다.

3. 랜섬웨어 트렌드



[그림 2] 2025 년 4 분기 랜섬웨어 트렌드

✓ 국제 공조를 통한 랜섬웨어 인프라 제공자 및 공모자 제재 강화

2025 년 4 분기에는 랜섬웨어 그룹뿐만 아니라 공격에 필요한 인프라를 제공하거나 랜섬웨어 그룹과 공모한 인력까지 대상으로 한 제재가 확대되었다.

2025 년 11 월 19 일 미국, 영국, 호주는 랜섬웨어 그룹들의 인프라를 제공한 러시아 호스팅 업체 Media Land 에 대한 공동 제재를 발표했다. Media Land 는 LockBit, BlackSuit, Play 등 랜섬웨어 그룹에 운영 인프라를 제공한 업체로 지목됐으며, 공격에 사용된 서버와 네트워크를 호스팅해 온 것으로 확인됐다. 이에 따라 각 정부는 Media Land 와 계열사 ML Cloud 를 포함한 관련 법인 및 연계 인물들을 제재 명단에 등재했다.

미국 법무부는 2025 년 12 월 BlackCat(ALPHV) 랜섬웨어 그룹과 공모한 Ryan Goldberg 와 Kevin Martin 을 미국 내 피해자를 공격해 금전을 갈취한 혐의로 기소했다. 수사 내용에 따르면 Goldberg 는 사이버 보안 기업에서 사고 대응 업무를 수행한 경력이 있으며, Martin 은 랜섬웨어 사고 대응을 지원하는 기업에서 협상 업무를 담당했던 것으로 확인됐다. 이들은 관련 업무를 통해 얻은 경험을 바탕으로 피해 조직의 네트워크를 암호화하고 몸값을 요구했으며, 사전에 수익 분배를 합의한 뒤 피해자가 지급한 몸값 일부를 운영진에게 전달하는 방식으로 공모 관계를 유지한 정황이 확인된다.

✓ 제조업 분야에 대한 랜섬웨어 공격 확산

2025 년 1 분기부터 4 분기까지 누적 집계 기준으로 제조업은 전체 산업군 가운데 랜섬웨어 피해가 가장 많은 것으로 확인됐다. 특히 4 분기 제조업 피해 사례는 601 건으로, 지난 분기(401 건) 대비 약 50% 증가했다. 이는 제조 현장에서 생산 라인이 중단될 경우 납기 지연과 매출 손실이 즉각 발생해, 협상 과정에서 기업이 받는 피해가 크기 때문이다.

2025 년 10 월 31 일, 일본의 자동차 서스펜션 제조업체 TEIN 은 본사 메인 서버가 랜섬웨어에 감염되면서 그룹 내부 시스템 장애가 발생했고, 그 영향으로 일본 공장은 1 일간, 중국 자회사 공장은 약 1 주간 가동이 중단됐다.

2025 년 11 월 28 일, 독일의 물류 장비 제조업체인 Gerd Bär GmbH 는 PayoutsKing 그룹의 공격을 받아 직원 개인정보, 재무 정보, 설계도 등이 포함된 데이터 유출이 발생했으며, 12 월 5 일까지 약 1 주간 온라인 주문이 중지되고 배송이 지연됐다.

이처럼 제조업 분야에 대한 랜섬웨어 공격은 단순한 데이터 암호화와 데이터 유출을 넘어 생산 라인 중단, 배송 차질 등 운영 전반의 장애로 이어져 조직에 직접적인 피해를 초래한다.

✓ 취약점을 이용한 랜섬웨어 공격

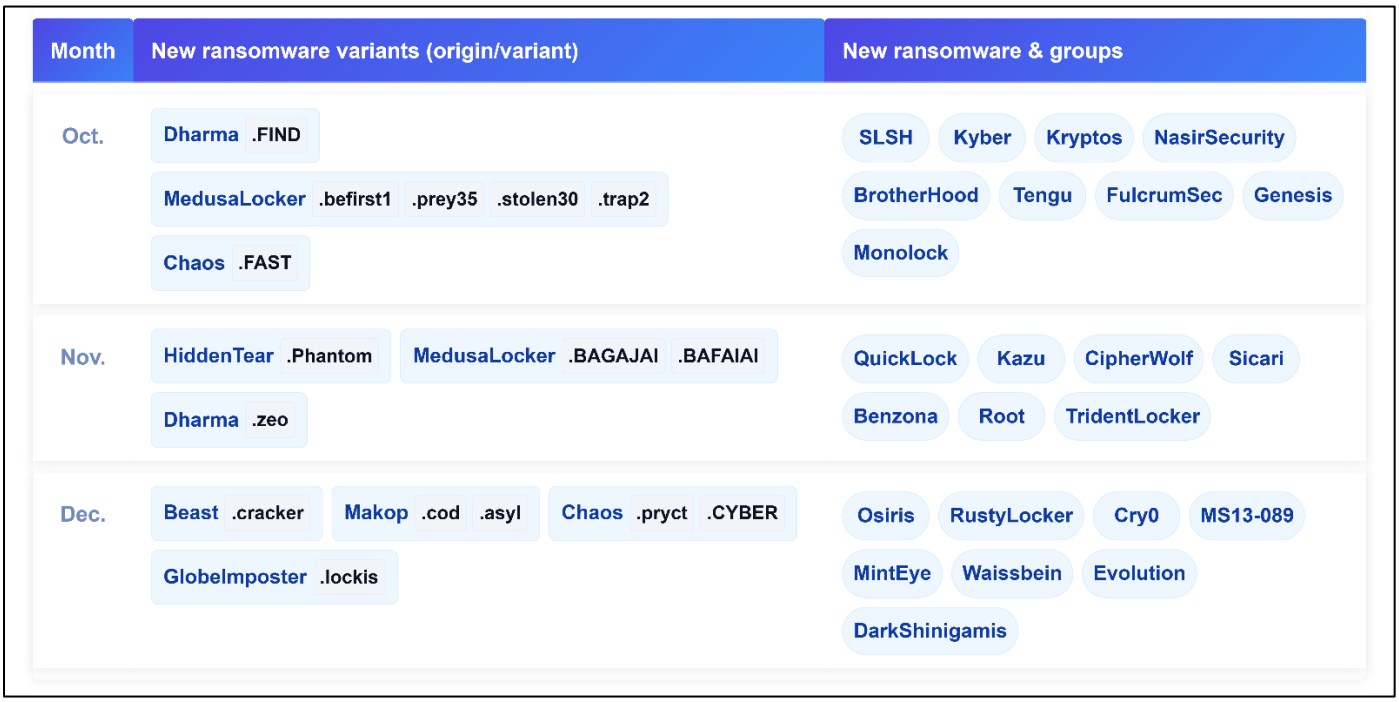
2025 년 11 월 Clop 그룹은 Oracle E-Business Suite 의 취약점(CVE-2025-61882)을 악용해 대규모 공격을 수행한 것으로 확인된다. 해당 취약점은 인증 없이 원격에서 서버의 요청 처리 흐름을 악용해 공격자의 코드가 서버에서 실행되도록 유도하고, 이를 통해 명령 실행 및 시스템 접근 권한을 확보한 뒤 추가 공격을 수행하는 방식으로 악용되었다.

2025 년 12 월 5 일 React2Shell 취약점(CVE-2025-55182)을 악용해 Weaxor 랜섬웨어가 배포된 사례가 확인됐다. 해당 취약점은 12 월 3 일 공개된 이후 불과 이틀 만에 실제 공격에 악용되었으며, 이는 취약점 공개 직후 랜섬웨어 공격자가 신속히 무기화해 실제 공격에 활용했음을 의미한다. 해당 취약점은 인증 절차 없이 원격 코드 실행이 가능하며, 공격자는 이를 악용해 초기 침투에 성공한 뒤 PowerShell 을 이용해 Cobalt Strike[2] 기반 C&C 서버[3]를 구성하고 Weaxor 랜섬웨어를 실행해 시스템 내 파일 암호화를 수행한 것으로 확인된다.

[2] Cobalt Strike: 공격자가 원격 명령 실행, 추가 페이로드 투하, 내부 정찰 및 수평 이동을 수행하기 위해 사용하는 C&C 프레임워크.

[3] C&C(Command & Control) 서버: 감염된 시스템과 통신하며 명령 전달, 추가 페이로드 배포, 정보 수집 등 공격자의 원격 제어를 수행하는 서버.

4. 신규 랜섬웨어 및 그룹 활동



[그림 3] [신규/변종 랜섬웨어]

2025 년 4 분기에도 신규 랜섬웨어 그룹들의 등장이 활발하게 이루어졌다. 특히나 주목할만한 그룹은 SLSH(Scattered Lapsus$ Hunters)로 해당 그룹은 Scattered Spider, LAPSUS$, ShinyHunters 로 이루어진 사이버 범죄 연합 그룹이다. Vishing[4], OAuth 토큰 탈취[5] 등을 통한 공격뿐만 아니라 최근에는 자체 랜섬웨어를 개발하고 유출 데이터 게시를 통한 이중 협박 전략을 본격적으로 채택하고 있다. 이들은 2025 년 11 월 무렵, 한 보안 외신과의 인터뷰에서 RaaS 운영 모델을 기반으로 Windows 환경뿐만 아니라 Linux 와 ESXi 환경을 타깃으로 한 랜섬웨어도 개발 중임을 밝힌 바 있다. 또한, SLSH 의 다크웹 유출 사이트에 기재된 바에 따르면 실질적 운영은 ShinyHunters 가 담당하되, RaaS 는 SLSH 라는 브랜드 이름으로 운영하며 Scattered Spider, LAPSUS$와의 협력 관계를 핵심 전략으로 유지하겠다는 입장을 표명했다. SLSH 는 다양한 공격 전술을 병행하며 공격 대상을 지속적으로 확장하겠다는 전략을 공공연히 밝히고 있는 만큼, 향후 행보에 대한 지속적인 모니터링과 분석이 요구된다. SLSH 를 제외하더라도, 2025 년 4 분기에 새롭게 확인된 신규 랜섬웨어 그룹은 총 23 개에 달하며, 이는 신규 위협 그룹의 등장 빈도가 매우 높은 수준임을 시사한다. 아래는 4 분기에 발견된 신규 랜섬웨어 그룹 중 주목할 만한 그룹에 대한 주요 특징이다.

- **Sicari**

  Sicari 는 2025 년 말에 발견된 신규 RaaS 그룹으로, 현재까지 공개한 피해 조직은 1 개에 불과하다. 이들은 자신들을 이스라엘·유대 정체성을 가진 집단으로 소개하며, 관련 상징과 메시지를 적극적으로 활용하고 있다. 해당 랜섬웨어는 피해 시스템이 이스라엘 언어 환경으로 설정되어 있을 경우 파일 암호화를 수행하지 않고 종료되는 구조를 가지고 있어, 이러한 정체성 주장을 기술적으로 반영하고 있다. 그러나 다크웹 포럼 활동은 주로 러시아어 기반으로 이루어지고 있으며, 사용된

[4] Vishing: 전화(voice) 기반 사회공학적 공격기법으로, 사람을 속여 인증 정보나 내부 정보를 빼내는 수법

[5] OAuth 토큰 탈취: 비밀번호 없이도 계정 접근이 가능한 OAuth 접근 토큰을 훔쳐서 계정을 장악하는 공격

히브리어 역시 문법 오류와 번역기 기반 문체가 다수 관측되고 있다. 이를 종합할 때, Sicari 는 실제 정체성을 은폐하기 위해 의도적으로 조작하는 방식을 사용한 것으로 판단되며, 이는 배후 추적을 어렵게 하기 위한 전략의 일환으로 해석된다.

- **Osiris**

  Osiris 는 2025 년 말에 등장한 신규 랜섬웨어 그룹으로, ECC + AES 기반 암호화 알고리즘을 사용하며, 파일 단위 고유 암호화 키를 적용하는 구조적 특징을 가진다. Osiris 는 공격 사전 단계부터 매우 체계적인 전략을 보였다. 랜섬웨어 배포 수일 전, 피해 시스템에서 Rclone 을 이용해 내부 데이터를 Wasabi 클라우드 스토리지 버킷으로 업로드해 외부 유출을 선행했다. Wasabi 는 AWS S3 호환 API 를 제공하는 정상 상용 클라우드 서비스로, 공격자 입장에서는 별도의 불법 인프라 구축 없이 안정적인 데이터 저장 및 관리가 가능하며, 탐지 위험 역시 상대적으로 낮다는 장점이 있다. 이와 유사한 데이터 탈취 전략은 2025 년 10 월경 발생한 Inc 랜섬웨어 공격 사례에서도 동일하게 관측된 바 있다.

  또한 자격 증명 탈취 과정에서 사용된 것으로 보이는 Mimikatz 실행 파일명이 Inc 랜섬웨어 사례에서 사용된 파일명과 일치하는 정황이 확인됐으며, 이를 통해 Osiris 가 Inc 그룹과의 연관성 또는 전략적 모방 관계에 있을 가능성이 제기된다. Osiris 는 Netscan, Netexec, MeshAgent 등을 활용해 내부 네트워크를 식별하고, Ruskdesk 기반 커스텀 원격 데스크톱 도구를 통해 원격 접근 권한을 유지했다. 특히 주목할 부분은 BYOVD(Bring Your Own Vulnerable Driver) 기법을 활용한 회피 전략으로, 이 과정에서 Malwarebytes 의 anti-exploit 드라이버를 가장한 Poortry 드라이버가 사용되었다. 해당 드라이버는 보안 솔루션보다 높은 권한으로 동작하고 정상 서명된 특성을 지니고 있어, Osiris 는 이를 악용해 랜섬웨어 실행 이전 단계에서 보안 솔루션 무력화를 수행한 것으로 판단된다.

  현재 Osiris 의 다크웹 유출 사이트는 접근이 불가능한 상태이나, 클리어웹 기반 유출 사이트는 접근이 가능하며, 피해자별 고유 주소를 통해서만 협상 채널 접속이 가능하도록 구성되어 있다. 또한 이들은 수사 혼선을 유도하기 위해 호스팅 업체 및 IP 를 빈번하게 변경하고 있으며, 사용된 인프라 다수가 방탄 호스팅(Bulletproof Hosting) 업체 기반으로 확인된다. 아직 공개된 피해 사례 수는 제한적이나, 사용 중인 공격 기법과 인프라 구조의 고도화를 고려할 때, Osiris 는 향후 더 큰 위협 그룹으로 성장할 가능성이 존재하며, 지속적인 추적 및 분석이 요구된다.

- **RustyLocker**

  RustyLocker 는 2025 년 12 월 발견된 신규 랜섬웨어 그룹으로, ChaCha20 알고리즘을 이용해 파일을 암호화하고, 해당 키를 RSA 알고리즘으로 보호하는 구조를 가진다.

  해당 그룹의 다크웹 유출 사이트는 파나마 기반 호스팅 업체 IP 에서 서비스되었으며, 랜섬웨어를 호스팅하던 웹 서버에는 VPN 도구, Mimikatz, UAC 우회 테스트 스크립트 등 공격 준비 단계에 사용된 도구들이 다수 존재했다. 또한 여러 버전의 랜섬웨어 샘플과 복호화 도구까지 함께 존재했으며, 이는 공격자가 실제 공격 이전에 다양한 빌드 테스트를 수행한 정황으로 해석된다.

  현재까지 게시된 피해 조직은 없으며, 다크웹 유출 사이트 역시 12 월 초 이후 접속 불가 상태이다. 다만, 서버 인프라 내에 실제 공격 도구들이 사전에 준비되어 있었던 점을 고려할 때, 향후 재등장 또는 리브랜딩 가능성을 배제할 수 없는 그룹으로 평가된다.

- **Evolution(Aware)**

  Evolution 그룹은 Global 랜섬웨어 그룹의 리브랜딩 조직으로, 두 랜섬웨어 샘플 간 코드 유사도는 약 99% 수준으로 사실상 동일한 구조이다. Evolution 랜섬웨어 실행 시 배경화면은 Global 감염 문구가 삽입된 이미지로 변경되지만, 랜섬노트에 포함된 링크는 Global 이 아닌 Aware 명칭의 다크웹 사이트로 연결된다.

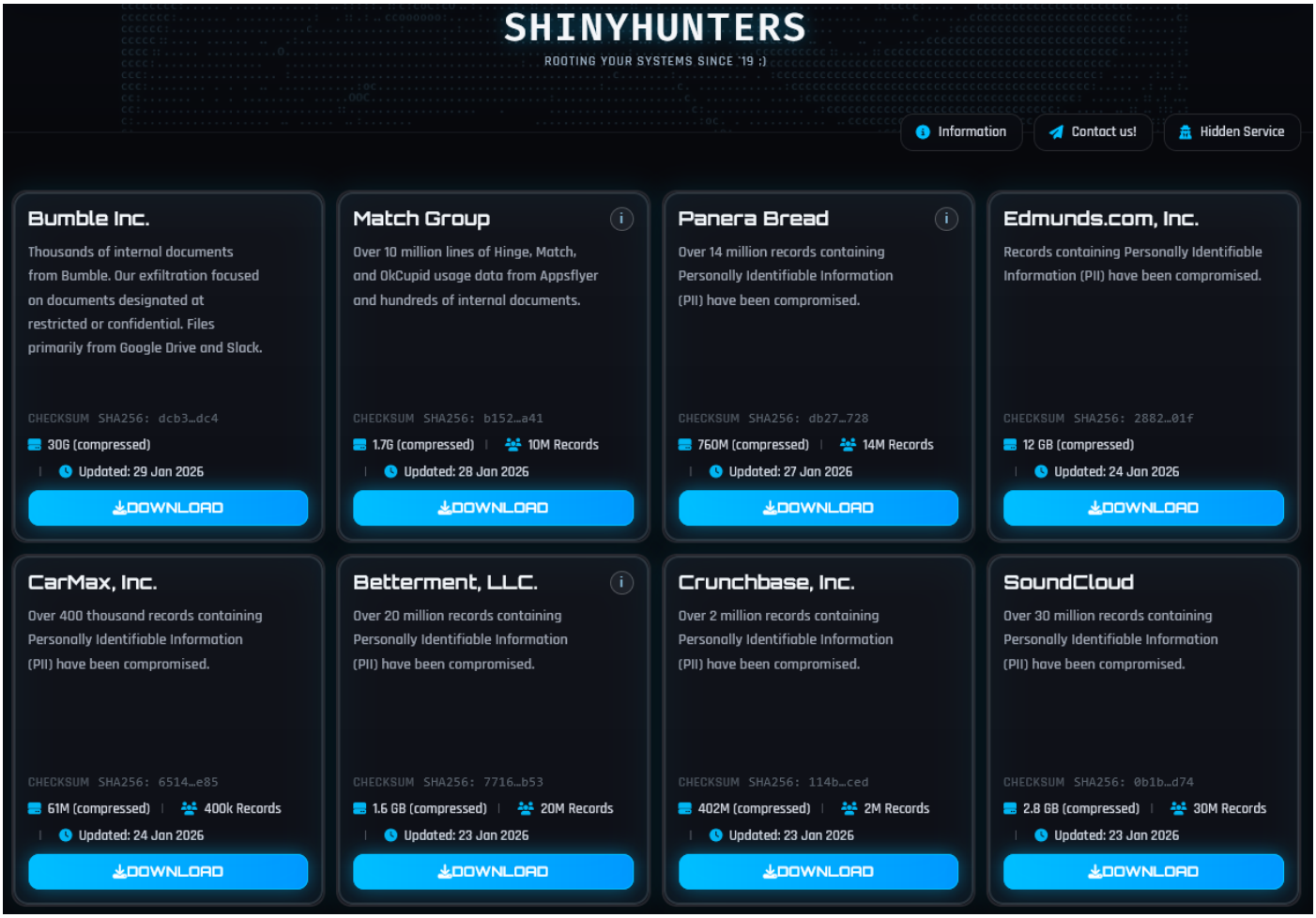
  Evolution 랜섬웨어가 발견된 웹 서버에서는 코인 마이너, Phorpiex 계열 봇넷(Twizt 변종), 다운로더형 악성코드 등이 함께 확인됐으며, 이들 악성코드는 zip 파일 내 lnk 파일 기반 유포 구조를 사용하고 있었다. 이는 2024 년 LockBit 그룹이 사용한 유포 방식과 구조적으로 유사하다.

  Evolution 그룹은 2025 년 말 Aware 다크웹 페이지 공개 이후 본격적으로 'Evolution'이라는 명칭을 랜섬노트에 사용하며 활동을 전개하고 있으며, 해당 조직은 MetaEncryptor → LostTrust → Eldorado → BlackLock → Mamona → Global → Evolution 으로 이어지는 지속적인 리브랜딩 이력을 보유하고 있다.

  다크웹 포럼 상 해당 그룹 관련 인물은 다수의 랜섬웨어 프로젝트를 제작하고 운영한 이력이 있으며, 여러 랜섬웨어 그룹과의 교류 정황도 확인된다. 이는 리브랜딩이 단순한 명칭 변경이 아닌, 공격 전략, 운영 구조, 생태계 내 역할 변화까지 포함한 구조적 전환 과정임을 시사하며, 장기적 위협 행위자 추적 관점에서 중요한 분석 지표로 활용될 수 있다.

SLSH 그룹 상세 분석

1. 개요



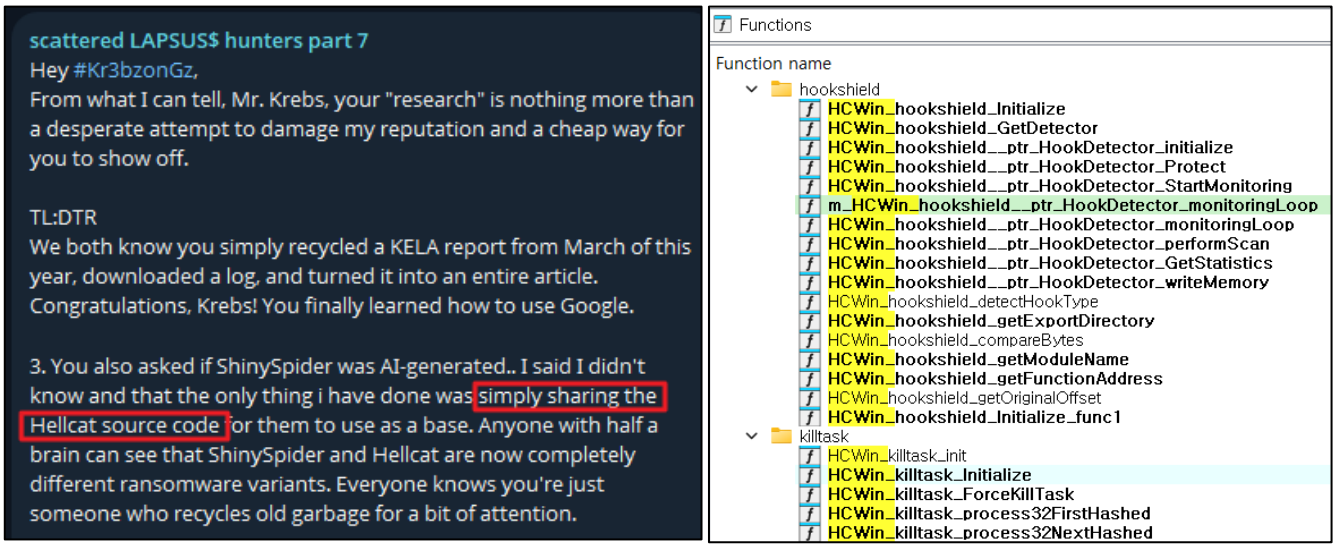
[그림 4] SLSH DLS

25 년 8 월초 등장한 SLSH(Scattered LAPSUS$ Hunters, SLH)는 기존의 세 그룹 Scattered Spider, LAPSUS$, ShinyHunters의 일부 구성원이 참여한 조직으로, 실질적인 운영은 ShinyHunters가 주도하고 있다. 등장 직후 텔레그램 채널을 개설해 범죄 예고, 내부자 모집, 피해 기업 명단 공개는 물론 협박과 조롱 같은 메시지를 연달아 퍼뜨리며 빠르게 존재감을 키워나갔다. 이들이 운영하는 텔레그램 채널은 운영정책 위반으로 수차례 차단됐으나, 새로운 채널을 개설하며 활동을 이어가고 있다. 이들은 또한 25 년 9 월과 10 월 두 차례에 걸쳐 활동 중단을 선언했는데, 실제로는 아주 짧은 공백기를 가진 뒤 새로운 텔레그램 채널을 개설해 활동을 재개했다. 이런 반복적인 채널 복구와 운영 중단 선언은 수사 회피와 관심을 분산시키기 위한 전략으로 보인다.

SLSH 의 가장 대표적인 침해 사례는 Salesforce 생태계를 겨냥한 공급망 침해 공격이다. Salesforce 는 기업 고객이 사용하는 고객 관리 플랫폼으로, 다양한 외부 마케팅·영업 도구와 연동해 활용된다. 공격자는 이러한 구조를 악용해 Salesforce 와 연동된 외부 마케팅 도구인 Salesloft Drift 를 공격 경로로 삼았으며, 이를 통해 기업 계정에 접근해 고객 정보를 탈취했다고 주장했다. 이들은 그 규모가 약 15억 건에 달한다고 밝혔지만, 실제로는 일부 데이터만 유출된 것으로 확인된다. 9월에는 텔레그램과 별도의 데이터 유출 사이트를 통해서 협박을 시작했으며, 10 월에는 협상에 응하지 않은 기업의 데이터를 일부 공개해 압박의 수위를 높였다. 10 월 말 이들의 데이터 유출 사이트는 폐쇄되었고 유출 중단을 선언하며 일단락 됐지만, SLSH 의 대표적인 공격 사례로 남아 이들의 전략과 수법이 주목을 받게 됐다.

앞선 데이터 유출 사례 이후, SLSH 는 자체 데이터 유출 기반 협박 서비스를 EaaS(Extortion-as-a-Service) 형태로 제공할 것이라고 2025 년 10 월에 발표했다. 이는 암호화 기능 없이 데이터 탈취와 유출 협박만을 제공하는 서비스 모델로, RaaS(Ransomware-as-a-Service)와 유사한 형태지만 목적과 수단이 제한된 구조다. 다만 해당 서비스는 발표 이후 현재까지 공식적으로 운영되거나 배포된 정황은 확인되지 않고 있다. 한편, 이보다 앞선 8 월에는 SLSH 가 자체 RaaS 개발에 착수했다고 밝힌 바 있다. 여기에 사용될 랜섬웨어인 ShinySpider 를 LockBit 및 DragonForce 계열과 비교하며 기술적 우위를 주장하고 ESXi 가상화 플랫폼까지 공격할 수 있도록 개발 중이라고 홍보했으나, 이 역시 실제 배포 여부나 구체적 기능은 공개되지 않았다. 그러던 중 11 월, 'ShinySpider 3.0'이라는 이름의 샘플이 외부에서 발견되며 해당 랜섬웨어의 실체가 일부 드러나기 시작했다.

2. HellCat 소스코드 기반의 ShinySpider 3.0 랜섬웨어

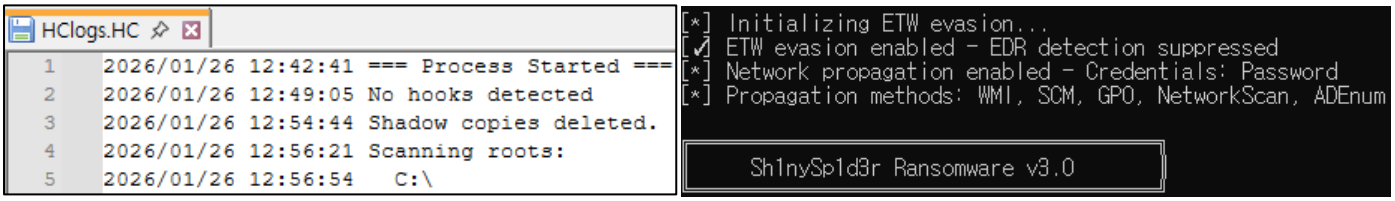


[그림 5] HellCat 소스코드 사용 주장 및 ShinySpider 3.0 에서 사용된 함수명

SLSH 는 공개된 ShinySpider 3.0 랜섬웨어가 HellCat 의 소스코드를 기반으로 만들어졌음을 밝혔다. Go 기반의 HellCat 랜섬웨어는 기능이 경량화된 버전과 더 많은 기능이 포함된 버전 총 2 개가 확인됐는데, HellCat 과 ShinySpider 에 공통적으로 존재하는 코드와 동작 방식이 거의 동일한 것이 확인됐다. 그 외에도 ShinySpider 일부 샘플에서 "HellCat Windows version"의 약자로 추정되는 "HCWin" 문자열이 포함된 함수가 다수 발견됐다는 점을 미루어 보아, ShinySpider 가 HellCat 랜섬웨어 소스코드를 활용해 개발된 버전임을 알 수 있다. 각 버전의 기능은 다음 표와 같다.

HellCat (경량화)	HellCat	ShinySpider 3.0
백업 복사본 삭제, 로컬 디스크 암호화	백업 복사본 삭제, 로컬 디스크 암호화, 프로세스/서비스 종료, 디버깅 모드, Anti-Hooking	백업 복사본 삭제, 로컬 디스크 암호화, 프로세스/서비스 종료, 디버깅 모드, Anti-Hooking, Anti-Injection, Anti-Debug, 메모리 보호, ETW 우회, 네트워크 공유 폴더 암호화, 네트워크 전파, 이벤트 로그 삭제, 바탕화면 변경, 자가 삭제, 디스크 빈 공간 삭제

각 버전별로 기능의 유무가 다를 뿐이지 공통적으로 존재하는 기능은 동일하게 동작하지만, 디버깅 모드에 차이가 존재하는 것이 확인됐다. 디버깅 모드가 활성화된 경우 HellCat은 "HClogs.HC" 라는 파일에 실행 로그를 저장하며, ShinySpider 3.0의 경우 별도의 로그 파일에 저장하는 것이 아닌 실행 중에 명령 프롬프트에 출력한다.



[그림 6] 디버깅 모드 비교(좌: HellCat, 우: ShinySpider 3.0)

3. ShinySpider 랜섬웨어 상세 분석

ShinySpider 3.0은 Windows 운영체제를 타깃으로 하는 Go 기반 랜섬웨어로, 기존 HellCat 소스코드를 바탕으로 여러 기능이 추가된 형태다. 공격자들은 ESXi 플랫폼까지 타깃을 확장할 예정임을 암시한 바 있으며, 이는 향후 위협 범위가 더욱 넓어질 수 있음을 의미한다. 이에 따라 본 보고서에서는 현재까지 공개된 Windows 버전 랜섬웨어를 분석 대상으로 삼아, ShinySpider 3.0의 구조와 동작 방식을 설명하며 이를 통해 향후 파생 변종 및 플랫폼 확장 시 활용 가능한 인사이트를 제공하고자 한다.

ShinySpider 는 별도의 실행 인자 확인 없이 내장된 설정값을 기반으로 동작한다. 대부분의 설정 값은 True(1)/False(0)로 이루어져 각종 기능의 실행 여부를 결정하며, 그 외에도 암호화 모드와 전파에 사용할 인증 정보와 같은 정보가 포함되어 있다. 공식적으로 각 내장된 옵션 항목이 명명된 것은 아니지만, 편의상 저장된 내장 옵션을 순서대로 구분하면 아래와 같다.

내장 옵션	옵션 설명
delete_eventlog	이벤트 로그 삭제 여부
change_wallpaper	바탕화면 변경 여부
check_mutex	Mutex 확인 여부(중복 실행 방지)
encrypt_share	네트워크 공유 폴더 암호화 여부
kill_service_process	서비스/프로세스 종료 여부
network_propagation	네트워크 전파 여부
self_destruction	자가 삭제 여부
cleanup_disk	디스크 빈 공간 정리 여부
etw_evasion	ETW(Event Tracing for Windows) 우회 여부
debug	디버깅 로그 출력 여부
encryption_mode	암호화 방식 (normal: 전체, partial: 부분, automatic: 크기에 따른 자동 선택)
encryption_mode_len	암호화 모드 문자열 길이
credential_list	호스트 인증 정보 리스트
credential_list_len	호스트 인증 정보 리스트 길이
propagation_wmi	WMI(Windows Management Instrumentation) 기반 네트워크 전파 여부
propagation_scm	SCM(Service Control Manager) 기반 네트워크 전파 여부
propagation_gpo	GPO(Group Policy Object) 기반 네트워크 전파 여부
propagation_netscan	서브넷 탐색 여부
propagation_adenum	AD(Active Directory) 탐색 여부

랜섬웨어는 행위 기반 엔드포인트 보안 솔루션에 의한 탐지 및 차단을 회피하기 위해 다양한 보호 및 우회 기법을 사용한다. 행위 기반 보안 솔루션은 악성 행위를 탐지하기 위해 탐지용 DLL 을 대상 프로세스에 주입하고, Nt 또는 Zw 계열의 시스템 호출 함수를 변조하여 해당 함수 호출 시 관련 정보를 탐지용 DLL 로 전달함으로써 프로세스의 행위를 모니터링하거나 분석한다. 또한 윈도우 이벤트 로깅 추적 프레임워크(ETW)를 활용해 행위를 수집·분석하는 방식도 사용된다. 이러한 환경에서 ShinySpider 는 ETW 변조뿐만 아니라 메모리 보호 기법, Anti-Injection, Anti-Hooking 기법을 함께 사용해 보안 솔루션이 랜섬웨어 프로세스를 모니터링하지 못하도록 방해하거나, 모니터링 여부를 사전에 확인 후 악성 행위 자체가 실행되지 않도록 제어하는 방식을 사용한다.

SetProcessMitigationPolicy의 ProcessExtensionPointDisablePolicy를 활성화해 외부 DLL이 랜섬웨어 프로세스에 인젝션 되는 것을 방지한다. 그다음으로는 데이터 실행 방지(DEP) 정책을 활성화하고, 힙 손상 시 프로세스가 종료되도록 설정해 프로세스의 메모리 변조와 코드 실행을 막아 보안 솔루션이 접근하지 못하도록 한다.

```
golang_org_x_sys_windows_NewLazyDLL("kernel32.dll", 12);
v7 = v0;
SetProcessDEPPolicy = golang_org_x_sys_windows__ptr_LazyDLL_NewProc(v0, "SetProcessDEPPolicy", 19);
if ( !golang_org_x_sys_windows__ptr_LazyProc_Find(SetProcessDEPPolicy).tab )
{
  p__1_uintptr = runtime_newobject(&RTYPE__1_uintptr);
  (*p__1_uintptr)[0] = 1;                       // PROCESS_DEP_ENABLE
  golang_org_x_sys_windows__ptr_LazyProc_Call(SetProcessDEPPolicy, p__1_uintptr, 1, 1);
}
HeapSetInformation = golang_org_x_sys_windows__ptr_LazyDLL_NewProc(v7, "HeapSetInformation", 18);
if ( !golang_org_x_sys_windows__ptr_LazyProc_Find(HeapSetInformation).tab )
{
  p_security_HeapInformation = runtime_newobject(&RTYPE_security_HeapInformation);
  p_security_HeapInformation->HeapClass = 1;  // HeapEnableTerminationOnCorruption
  v9 = p_security_HeapInformation;
  p__4_uintptr = runtime_newobject(&RTYPE__4_uintptr);
  (*p__4_uintptr)[2] = v9;
  (*p__4_uintptr)[3] = 8;
  golang_org_x_sys_windows__ptr_LazyProc_Call(HeapSetInformation, p__4_uintptr, 4, 4);
}
```

[그림 7] DEP 정책 설정 및 힙 오류 설정

시스템 호출 함수가 보안 솔루션에 의해 변조되어 모니터링 되는지 확인하기 위해 시스템 호출 함수의 원본 코드 16Bytes 를 저장한 뒤, 정기적으로 원본 코드와 현재 사용중인 시스템 호출 함수를 비교해 후킹 여부를 판단한다.

```
debug094:000000C00011C178 dq offset unk_C00000A6D0              ; NtAccessCheck
debug094:000000C00011C180 dq 10h
debug094:000000C00011C188 dq 10h
debug094:000000C00011C190 dq offset aNtaccesscheck              ; "NtAccessCheck"
debug094:000000C00011C198 dq 0Dh
debug094:000000C00011C1A0 dq offset unk_C00000A6F0
debug094:000000C00011C1A8 dq 10h
debug094:000000C00011C1B0 dq 10h
debug094:000000C00011C1B8 dq offset aNtaccesschecka
debug094:000000C00011C1C0 dq 1Ah
debug094:000000C00011C1C8 dq offset unk_C00000A700
debug094:000000C00011C1D0 dq 10h
debug094:000000C00011C1D8 dq 10h
debug094:000000C00011C1E0 dq offset aNtaccesscheckb
debug094:000000C00011C1E8 dq 13h
debug094:000000C00011C1F0 dq offset unk_C00000A710
debug094:000000C00011C1F8 dq 10h
debug094:000000C00011C200 dq 10h
debug094:000000C00011C208 dq offset aNtaccesscheckb_0

debug065:000000C00000A6D0 unk_C00000A6D0 db  4Ch ; L
debug065:000000C00000A6D1 db  8Bh
debug065:000000C00000A6D2 db 0D1h
debug065:000000C00000A6D3 db 0B8h
debug065:000000C00000A6D4 db    2
debug065:000000C00000A6D5 db    0
debug065:000000C00000A6D6 db    0
debug065:000000C00000A6D7 db    0
debug065:000000C00000A6D8 db 0F6h
debug065:000000C00000A6D9 db    4
debug065:000000C00000A6DA db  25h ; %
debug065:000000C00000A6DB db    8
debug065:000000C00000A6DC db    3
debug065:000000C00000A6DD db 0FEh
debug065:000000C00000A6DE db  7Fh ; □
debug065:000000C00000A6DF db    1
```

[그림 8] 메모리에 저장된 시스템 호출 함수 원본 코드

보안 솔루션이 ETW 를 활용해 악성 행위를 탐지하는 것을 우회하기 위해 로드된 EtwEventWrite 함수의 코드가 이벤트가 기록되지 않도록 0 을 반환하는 Callback 함수로 점프하도록 패치한다.

```
ntdll.dll:00007FFA4EA00300 ntdll_EtwEventWrite proc near
ntdll.dll:00007FFA4EA00300
ntdll.dll:00007FFA4EA00300 var_38= word ptr -38h
ntdll.dll:00007FFA4EA00300
ntdll.dll:00007FFA4EA00300 mov     r11, rsp
ntdll.dll:00007FFA4EA00303 sub     rsp, 58h
ntdll.dll:00007FFA4EA00307 mov     [r11-18h], r9
ntdll.dll:00007FFA4EA0030B xor     eax, eax
ntdll.dll:00007FFA4EA0030D mov     [r11-20h], r8d
ntdll.dll:00007FFA4EA00311 xor     r9d, r9d
ntdll.dll:00007FFA4EA00314 mov     [r11-28h], rax
ntdll.dll:00007FFA4EA00318 xor     r8d, r8d
ntdll.dll:00007FFA4EA0031B mov     [r11-30h], rax
ntdll.dll:00007FFA4EA0031F mov     [rsp+58h+var_38], ax
ntdll.dll:00007FFA4EA00324 call    sub_7FFA4EA00388
ntdll.dll:00007FFA4EA00329 add     rsp, 58h
ntdll.dll:00007FFA4EA0032D retn
ntdll.dll:00007FFA4EA0032D ntdll_EtwEventWrite endp
```

```
ntdll.dll:00007FFA4EA00300 ntdll_EtwEventWrite proc near
ntdll.dll:00007FFA4EA00300 mov     rax, 0C8AE0Ah
ntdll.dll:00007FFA4EA0030A jmp     rax
ntdll.dll:00007FFA4EA0030A ntdll_EtwEventWrite endp
ntdll.dll:00007FFA4EA0030A
ntdll.dll:00007FFA4EA0030A ; ---
ntdll.dll:00007FFA4EA0030C db  90h
ntdll.dll:00007FFA4EA0030D db  90h
ntdll.dll:00007FFA4EA0030E db  89h
ntdll.dll:00007FFA4EA0030F db  43h ; C
ntdll.dll:00007FFA4EA00310 db 0E0h
ntdll.dll:00007FFA4EA00311 db  45h ; E
ntdll.dll:00007FFA4EA00312 db  33h ; 3
ntdll.dll:00007FFA4EA00313 db 0C9h
ntdll.dll:00007FFA4EA00314 db  49h ; I
ntdll.dll:00007FFA4EA00315 db  89h
ntdll.dll:00007FFA4EA00316 db  43h ; C
ntdll.dll:00007FFA4EA00317 db 0D8h
```

jmp callback function

[그림 9] 패치 전 EtwEventWrite 함수(좌) 및 패치 후 EtwEventWrite 함수(우)

보안 솔루션을 우회하는 것뿐만 아니라, 분석을 방해하기 위한 목적으로 현재 실행 중인 랜섬웨어 스레드에 ThreadHideFromDebugger 옵션을 활성화해 연결된 디버거를 해제한다.

```
golang_org_x_sys_windows_NewLazyDLL("ntdll.dll", 9);
NtSetInformationThread = golang_org_x_sys_windows__ptr_LazyDLL_NewProc(v0, "NtSetInformationThread", 22);
p__4_uintptr = runtime_newobject(&RTYPE__4_uintptr);
(*p__4_uintptr)[0] = -2;
(*p__4_uintptr)[1] = 17;                    // ThreadHideFromDebugger
(*p__4_uintptr)[2] = 1;                     // True
(*p__4_uintptr)[3] = 0;
v9 = golang_org_x_sys_windows__ptr_LazyProc_Call(NtSetInformationThread, p__4_uintptr, 4, 4);
```

[그림 10] Anti-Debug

본격적인 암호화에 앞서, 사용자가 임의로 시스템을 복구하지 못하도록 현재 시스템에 존재하는 백업 복사본을 삭제한다.

```
v5 = fmt_Sprintf(
       "cmd.exe /c C:\\Windows\\System32\\wbem\\WMIC.exe shadowcopy where \"ID='%s'\" delete",
       78,
       v13,
       1,
       1);
v6 = HCWin_shadow_runCommand(v5.part_0, v5.part_1);
if ( v6.part_0 )
{
  v12 = 0u;
  runtime_convTstring(v3.ptr, v3.len);
  v11[0] = &RTYPE_string;
  v11[1] = v7;
  *&v12 = *(v6.part_0 + 8LL);
  *(&v12 + 1) = v6.part_1;
  log_Printf("Failed to delete shadow copy %s: %v", 35, v11, 2, 2);
}
```

[그림 11] 백업 복사본 삭제 명령어

원활한 암호화를 위해서 특정 서비스와 프로세스를 사전에 종료한다. 종료 대상 서비스와 프로세스는 동일한 리스트를 참고하지만 서비스의 경우 리스트와 완전히 일치하는 경우에만 종료하며, 프로세스는 아래 정규 표현식에 따라 문자열이 포함된 모든 프로세스를 종료한다.

정규 표현식(프로세스)
(?i)*.<TARGET_NAME>.*(?:\.exe)?$

서비스/프로세스 종료 대상 리스트
sql, oracle, ocssd, dbsnmp, synctime, agntsvc, isqlplussvc, xfssvccon, mydesktopservice, ocautoupds, encsvc, firefox, tbirdconfig, mydesktopqos, ocomm, dbeng50, sqbcoreservice, excel, infopath, msaccess, mspub, onenote, outlook, powerpnt, steam, thebat, thunderbird, visio, winword, wordpad, notepad

ShinySpider 는 로컬 디스크를 암호화하기 전에, 현재 시스템에 연결된 네트워크를 감염시킨다. 우선 현재 시스템의 네트워크와 동일한 서브넷에 있는 호스트 중, SMB 포트가 열려있으며 접근이 가능한 호스트에 연결을 시도한다. 연결 시에는 현재 시스템의 사용자 계정 인증 정보를 사용하거나 하드코딩된 인증 정보 리스트를 활용해 인증을 시도한다. 연결에 성공한 호스트의 모든 공유 폴더를 확인하며, 폴더명 끝에 "$" 문자가 들어가는 관리자 권한 폴더를 제외한 모든 네트워크 공유 파일을 암호화한다.

```
if ( debug )
{
  v61[0] = &RTYPE_string;
  v61[1] = &off_EB3E00;                    // [Network Shares] Starting Phase 1: Network Discovery
  log_Println(v61, 1, 1);
}
shares = HCWin_shares_DiscoverNetworkHosts();

if ( debug )
{
  v61[0] = &RTYPE_string;
  v61[1] = &off_EB3E20;                   // [Network Shares] Starting Phase 2: Testing share access
  log_Println(v61, 1, 1);

}
v1 = credential_len;
t_res = HCWin_shares_TestShareAccess(shares.part_0, shares.part_1, shares.part_2, credential_ptr, credential_len);

if ( debug )
{
  v61[0] = &RTYPE_string;
  v61[1] = &off_EB3E30;                   // [Network Shares] Starting Phase 3: Share enumeration and encryption
  log_Println(v61, 1, 1);
}
while ( v4 > 0 )
{
  sync__ptr_WaitGroup_Add(p_sync_WaitGroup, 1);
  v5 = runtime_newobject(&unk_E357A0);
  v6 = main_encryptNetworkShares_func1;   // count shares
  *v5 = main_encryptNetworkShares_func1;

if ( debug )
{
  v61[0] = &RTYPE_string;
  v61[1] = &off_EB3E50;                  // [Network Shares] Starting Phase 4: File encryption
  log_Println(v61, 1, 1);
}
while ( v25 > 0 )
{
  v48 = ptr->NetName.ptr;
  v49 = *&ptr->NetName.len;
  Remark = ptr->Remark;
  HostIP = ptr->HostIP;
  sync__ptr_WaitGroup_Add(v23, 1);
  v27 = runtime_newobject(&unk_E258A0);
  *v27 = main_encryptNetworkShares_func2;// encrypt network shares (except *$ folders}
```

[그림 12] 네트워크 공유 폴더 암호화

네트워크 공유 폴더를 암호화한 뒤, 현재 시스템과 동일한 네트워크에 존재하는 다른 호스트에 랜섬웨어 전파를 시도한다. 네트워크 전파에도 인증 정보가 필요하기 때문에, 현재 계정의 인증 정보를 그대로 사용하거나 하드코딩된 인증 정보를 활용해 전파한다. 또한 설정값에 따라 3 가지 전파 방식을 활용할 수 있다. WMI 명령어를 사용해 원격으로 랜섬웨어를 실행시키거나 SCM 으로 랜섬웨어 서비스를 생성해 실행할 수 있으며, GPO 스크립트 등록으로 호스트 로그온 시 자동으로 랜섬웨어가 실행되도록 하는 방식도 존재한다. 이 외에도 내부망에 해당하는 서브넷 대역(10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16)이나 현재 시스템과 동일한 네트워크 내에서 암호화 대상을 탐색하거나, 동일한 AD 에 연결된 호스트를 탐색해 전파할 수도 있다.

설정값 “propagation_netscan”이 활성화된 경우, 네트워크 공유 폴더 암호화와 동일한 방식으로 현재 시스템이 사용중인 서브넷에서 SMB 포트로 접속이 가능한 호스트를 식별해 전파 대상에 추가한다.

```
if ( !net__ptr_IPNet_Contains(part_3, v37.part_0, v37.part_1, v37.part_2) || v31 >= 512 )
  break;
v14 = part_0;
LODWORD(ptr) = *(part_0 + 3);
if ( ptr && ptr != 0xFF )
{
  v18 = part_1;
  v19 = part_2;
  v20 = net_IP_String(*&v14);
  v36[0] = 0;
  v36[1] = 0;
  runtime_convTstring(v20.ptr, v20.len);
  v36[0] = &RTYPE_string;
  v36[1] = v21;
  v22 = fmt_Sprintf("%s:445", 6, v36, 1, 1);
  LODWORD(ptr) = 100000000;
  *(&v5 - 3) = net_DialTimeout("tcp", 3, v22.part_0, v22.part_1, 100000000);
```

[그림 13] SMB 포트 연결 시도

설정값 “propagation_adenum”이 활성화된 경우, 현재 시스템이 포함된 도메인에 연결된 호스트를 전파 대상에 추가한다. 도메인에 연결된 호스트를 확인하기 위해서 NetServerEnum 함수로 사용하거나 ADODB[6] 질의를 통해 도메인에 연결된 모든 서버 리스트를 가져온다.

```
if ( !qword_106BBF8 )
  HCWin_propagation_Initialize();
v21 = 0;
v17 = 0;
v15 = 0;
v22[0] = 0;
v22[1] = 101;
v23 = &v21;
v24 = 0xFFFFFFFFLL;
v25 = &v17;
v26 = &v15;
v27 = 3;
v28 = 0u;
v38 = HCWin_apihash__ptr_APIHash_CallAPI(qword_106BBF8, "netapi32.dll", 12, "NetServerEnum", 13, v22, 9, 9);
```

[그림 14] NetServerEnum 을 활용한 호스트 검색

[6] ADODB: COM 기반 데이터 접근 기술로, OLE DB Provider 를 통해 데이터베이스뿐만 아니라 LDAP Provider 를 이용한 Active Directory 질의에도 사용

```
v24 = HCWin_propagation__ptr_Propagator_domainToDN(a1, v87);
v83[0] = 0;
v83[1] = 0;
runtime_convTstring(v24.ptr, v24.len);
v83[0] = &RTYPE_string;
v83[1] = v25;
v26 = fmt_Sprintf("LDAP://%s", 9, v83, 1, 1);
v83[0] = 0;
v83[1] = 0;
runtime_convTstring(v26.part_0, v26.part_1);
v83[0] = &RTYPE_string;
v83[1] = v27;
v74 = fmt_Sprintf("SELECT dNSHostName, cn FROM '%s' WHERE objectCategory='computer'", 64, v83, 1, 1);
v80[0] = &RTYPE_string;
v80[1] = &v74;
v81 = &RTYPE__ptr_ole_IDispatch;
v82 = v70;
github_com_go_ole_go_ole_oleutil_CallMethod(v68, "Open", 4, v80, 2, 2);
```

[그림 15] ADODB 질의를 통한 호스트 검색

전파 대상이 추가된 이후에는 설정값에 따라 WMI 혹은 SCM 을 활용해 각 호스트에서 랜섬웨어를 실행한다. 두 방식 모두 우선 각 호스트의 임시 폴더에 랜덤한 8 글자의 이름으로 랜섬웨어 파일을 복사하며, WMI 는 해당 파일을 새로운 프로세스로 실행시키고 SCM 은 복사된 랜섬웨어를 서비스로 등록한 뒤 실행한다.

- 랜섬웨어 복사 경로: \\<Domain>\ADMIN$\TEMP\[a-zA-Z0-9]{8}.exe

```
v52 = HCWin_apihash__ptr_APIHash_CallAPI(qword_106BBF8, "kernel32.dll", 12, "CopyFileW", 9, v35, 3, 3);
if ( v52.part_0 )
{
  v25 = HCWin_propagation__ptr_Propagator_installService(a1, ptr, len, v37, v2.part_1, v40, v15.part_1);
  part_3 = v25.part_1;
  part_2 = v25.part_0;
  if ( !v25.part_0 )
  {
    started = HCWin_propagation__ptr_Propagator_startService(a1, ptr, len, v37, v2.part_1);
    if ( started.part_0 )
    {
      v43 = started.part_1;
      HCWin_propagation__ptr_Propagator_cleanupService(a1, ptr, len, v37, v2.part_1);
      part_2 = started.part_0;
      part_3 = v43;
    }
    else
    {
      time_Sleep(2000000000);
      HCWin_propagation__ptr_Propagator_cleanupService(a1, ptr, len, v37, v2.part_1);
      v44[0] = 0;
      v44[1] = 0;
      runtime_convTstring(ptr, len);
      v44[0] = &RTYPE_string;
      v44[1] = v27;
      log_Printf("[Propagation] SCM deployment successful on %s", 45, v44, 1, 1);
```

[그림 16] SCM 을 통한 랜섬웨어 전파

만약 “propagation_gpo” 옵션을 사용하면 랜섬웨어 파일을 AD 환경의 공유 경로에 복사한 뒤 GPO 시작/종료 스크립트를 활용해 호스트가 로그온할 때 자동으로 랜섬웨어가 실행되도록 한다.

- 랜섬웨어 복사 경로: \\<Domain>\SYSVOL\<Domain>\scripts\[a-zA-Z0-9]{8}.exe
- 스크립트 정의 경로: \\<Domain>\SYSVOL\<Domain>\Policies\<GUID>\Machine\Scripts\scripts.ini
- GUID: {31B2F340-016D-11D2-945F-00C04FB984F9}

```
DefaultGPOPath = HCWin_propagation__ptr_Propagator_getDefaultGPOPath(a1, *(&a3 - 1));
if ( DefaultGPOPath.len )
{
  *v25 = DefaultGPOPath;
  v27 = 7;
  v26 = "Machine";
  v29 = 7;
  v28 = "Scripts";
  v31 = 11;
  v30 = "scripts.ini";
  v7 = path_filepath_Join(v25, 4, 4);
  v13 = path_filepath_Abs_0_8(v7.part_0, v7.part_1, v8, a4, a5, v9, v10, v11, v12, v20, v21);
  os_MkdirAll(v13, v7.part_1, 493);
  v24[0] = 0;
  v24[1] = 0;
  runtime_convTstring(a4, a5);
  v24[0] = &RTYPE_string;
  v24[1] = v14;
  v15 = fmt_Sprintf("\n[Startup]\n0CmdLine=%s\n0Parameters=\n", 36, v24, 1, 1);
```

[그림 17] 시작/종료 스크립트 생성

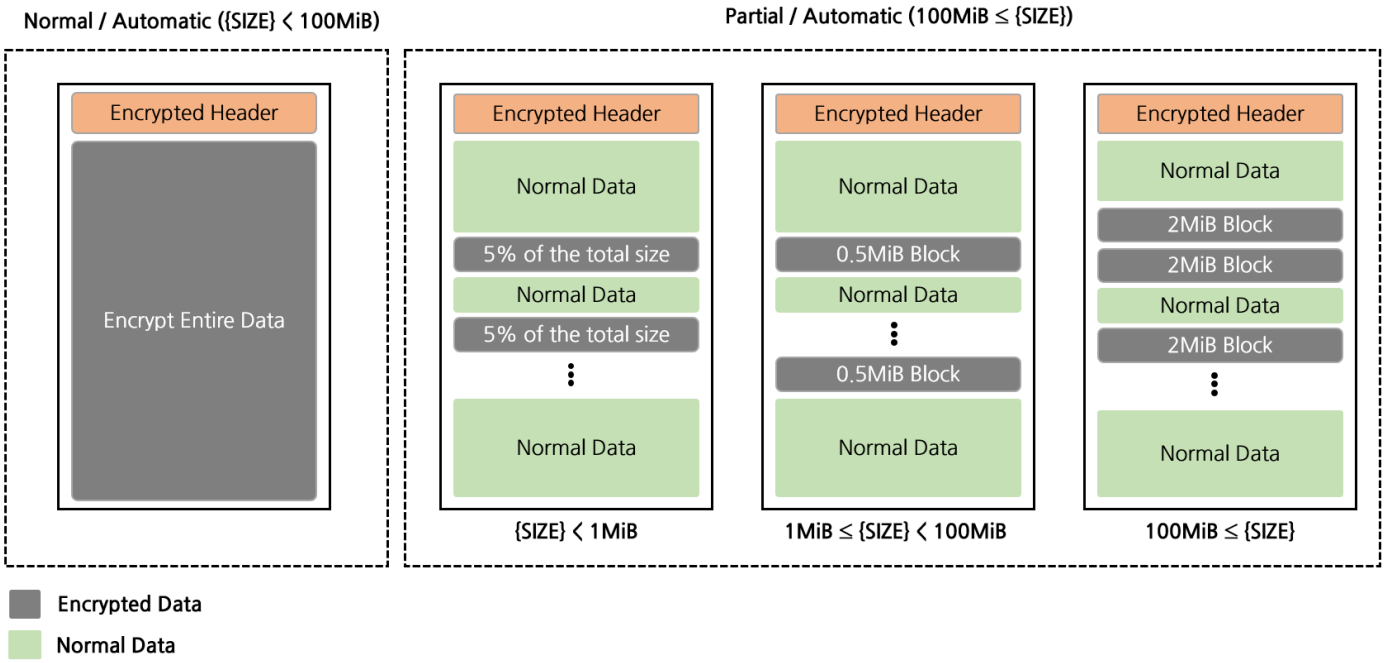
네트워크 공유 폴더 암호화와 네트워크 전파가 완료된 이후에는 현재 시스템에 있는 로컬 디스크를 암호화한다. 현재 시스템의 루트 디렉터리부터 모든 하위 디렉터리를 순회하면서 순차적으로 암호화하는데, 암호화 예외 디렉터리와 파일명, 파일 확장자는 제외한다.

암호화 예외 디렉터리
$recycle.bin, config.msi, $windows.~bt, $windows.~ws, windows, appdata, application data, boot, google, mozilla, program files, program files (x86), programdata, system volume information, tor browser, windows.old, intel, msocache, perflogs, x64dbg, public, all users, default

암호화 예외 파일
autorun.inf, boot.ini, bootfont.bin, bootsect.bak, desktop.ini, iconcache.db, ntldr, ntuser.dat, ntuser.dat.log, ntuser.ini, thumbs.db, R3ADME_1Vks5fYe.txt

암호화 예외 확장자
386, .adv, .ani, .bat, .bin, .cab, .cmd, .com, .cpl, .cur, .deskthemepack, .diagcab, .diagcfg, .diagpkg, .dll, .drv, .exe, .hlp, .icl, .icns, .ico, .ics, .idx, .ldf, .lnk, .mod, .mod, .msc, .msp, .msstyles, .msu, .nls, .nomedia, .ocx, .prf, .ps1, .rom, .rtp, .scr, .shs, .spl, .sys, .theme, .themepack, .wpx, .lock, .key, .hta, .msi, .pdb, .[A-Z0-9]{8}(encrypted_file)

파일 암호화는 기본적으로 설정값에 지정된 암호화 옵션에 따라 전체 암호화와 부분 암호화를 결정한다. 암호화 옵션은 파일 전체를 암호화하는 normal, 전체 크기의 28%만 암호화하는 partial, 파일 크기에 따라 암호화 방식을 설정하는 automatic, 총 3 개로 구분된다.



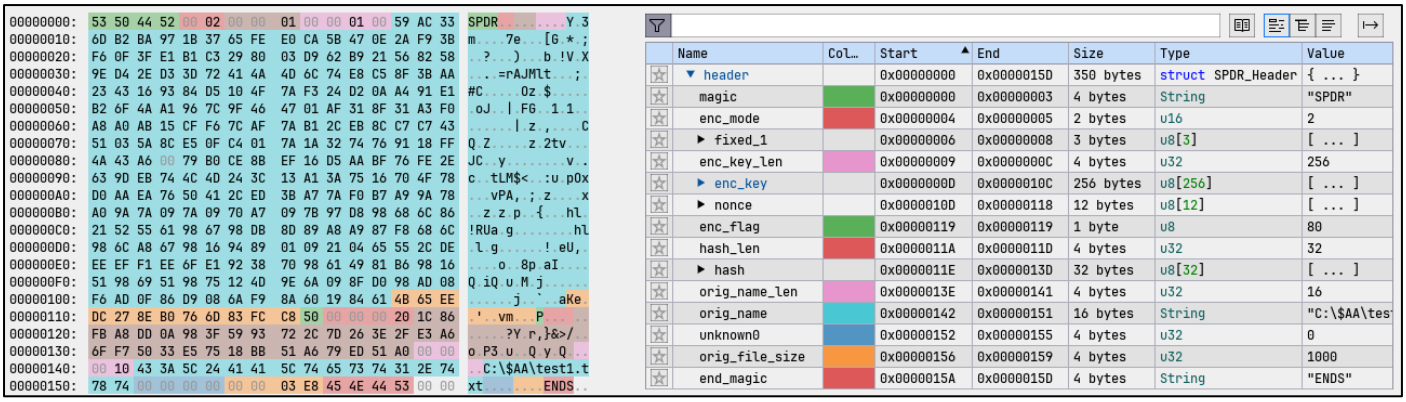
[그림 18] 암호화 방식

부분 암호화의 경우 전체 파일 크기의 28%만 암호화를 진행한다. 이를 위해 각 파일마다 랜덤한 위치를 선택하고 아래 블록 단위로 암호화를 진행해, 암호화된 블록의 누적 크기가 28%가 될 때까지 반복한다. 완전히 랜덤한 위치라면 복호화에 문제가 생길 수 있지만, 파일의 원본 경로에 기반한 해시 값의 상위 8Bytes 를 시드로 생성한 난수로 암호화 위치를 정하기 때문에 이론상 복구에도 문제가 없다.

```
encryption_size = 0.28 * filesize;
v11 = encryption_size;
if ( !encryption_size && filesize > 0 )
{
  v60 = num_32;
  v11 = math_Min(filesize, hashed_filepath, num_32, num_32_1, v4, v5, v6, v7, v8, *&v38[0]);
  num_32 = v60;
}
v41 = v11;
crypto_sha256_Sum256(hashed_filepath, num_32, num_32_1);// sha256(sha256(filepath))
v39[0] = v38[0];
v39[1] = v38[1];
parsed_hash = encoding_binary_bigEndian_Uint64(v39, 8, 32);// parse uppder 8 bytes
seed = text_template_parse__ptr_ListNode_Copy(parsed_hash);
math_rand_New(seed, 8);
v53 = v14;
if ( filesize >= 0x100000 )                    // select block size
{
  v1 = 0x200000;
  if ( filesize < 0x6400000 )
    v1 = 0x80000;
  block_size = v1;
}
else
{
  block_size = filesize / 20;
}
```

[그림 19] 부분 암호화 offset 설정

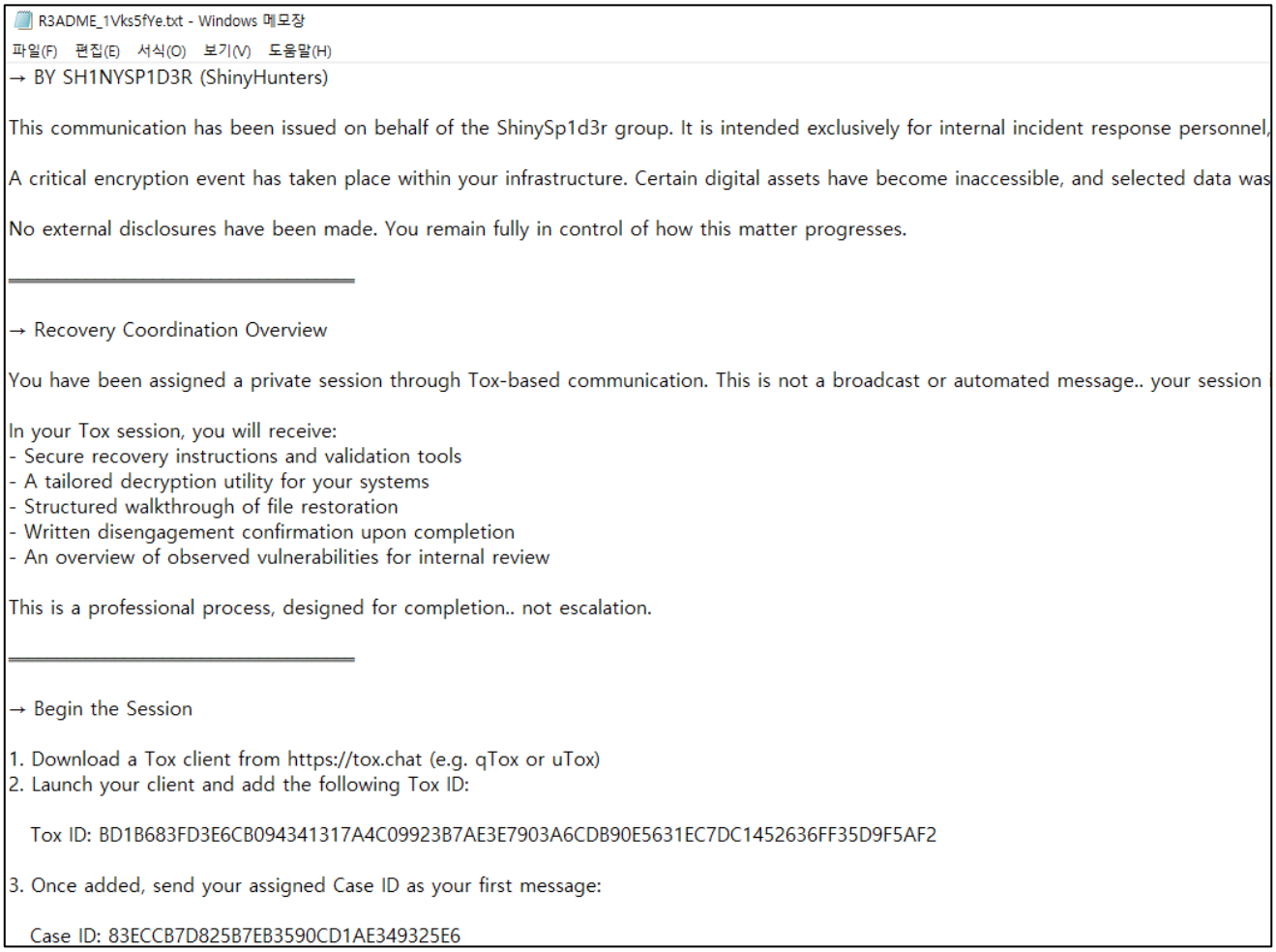
각 파일은 랜덤하게 생성된 32Bytes 키와 12Bytes Nonce 를 사용해 ChaCha20 알고리즘으로 암호화된다. 파일 암호화에 사용한 키는 256Bytes 크기의 RSA 공개키로 암호화된다. 암호화된 파일의 맨 앞에는 복호화를 위한 여러 메타데이터가 저장된다. 데이터 영역을 구분할 수 있는 구분자는 물론 RSA-2048 로 보호된 암호화 키, Nonce, 암호화 모드, 원본 파일명이 저장되며, 부분 암호화된 파일의 경우 암호화 오프셋을 찾을 수 있는 해시도 함께 저장된다.



Name	Col…	Start	End	Size	Type	Value
header		0x00000000	0x0000015D	350 bytes	struct SPDR_Header	{ ... }
magic		0x00000000	0x00000003	4 bytes	String	"SPDR"
enc_mode		0x00000004	0x00000005	2 bytes	u16	2
fixed_1		0x00000006	0x00000008	3 bytes	u8[3]	[...]
enc_key_len		0x00000009	0x0000000C	4 bytes	u32	256
enc_key		0x0000000D	0x0000010C	256 bytes	u8[256]	[...]
nonce		0x0000010D	0x00000118	12 bytes	u8[12]	[...]
enc_flag		0x00000119	0x00000119	1 byte	u8	80
hash_len		0x0000011A	0x0000011D	4 bytes	u32	32
hash		0x0000011E	0x0000013D	32 bytes	u8[32]	[...]
orig_name_len		0x0000013E	0x00000141	4 bytes	u32	16
orig_name		0x00000142	0x00000151	16 bytes	String	"C:\$AA\tes
unknown0		0x00000152	0x00000155	4 bytes	u32	0
orig_file_size		0x00000156	0x00000159	4 bytes	u32	1000
end_magic		0x0000015A	0x0000015D	4 bytes	String	"ENDS"

[그림 20] 암호화된 파일 헤더

모든 파일을 암호화하고 난 뒤에는 인코딩된 랜섬노트를 “0xDEADBEEFCAFEBABE” 라는 8Bytes 키로 디코딩해 각 폴더에 저장한다.



R3ADME_1Vks5fYe.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

→ BY SH1NYSP1D3R (ShinyHunters)

This communication has been issued on behalf of the ShinySp1d3r group. It is intended exclusively for internal incident response personnel,

A critical encryption event has taken place within your infrastructure. Certain digital assets have become inaccessible, and selected data was

No external disclosures have been made. You remain fully in control of how this matter progresses.

→ Recovery Coordination Overview

You have been assigned a private session through Tox-based communication. This is not a broadcast or automated message.. your session

In your Tox session, you will receive:
- Secure recovery instructions and validation tools
- A tailored decryption utility for your systems
- Structured walkthrough of file restoration
- Written disengagement confirmation upon completion
- An overview of observed vulnerabilities for internal review

This is a professional process, designed for completion.. not escalation.

→ Begin the Session

1. Download a Tox client from https://tox.chat (e.g. qTox or uTox)
2. Launch your client and add the following Tox ID:

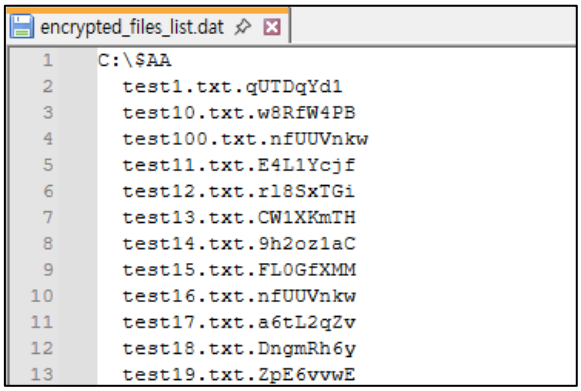
Tox ID: BD1B683FD3E6CB094341317A4C09923B7AE3E7903A6CDB90E5631EC7DC1452636FF35D9F5AF2

3. Once added, send your assigned Case ID as your first message:

Case ID: 83ECCB7D825B7EB3590CD1AE349325E6

[그림 21] 랜섬노트

또한 파일마다 랜덤한 8Bytes 문자로 이루어진 암호화 확장자를 추가하고, 암호화된 파일 리스트를 각 디렉터리 별로 정리한 로그 파일을 생성한다.



[그림 22] 암호화된 파일 리스트

설정값 "cleanup_disk"가 활성화되어 있으면 임시 폴더에 랜덤한 파일을 하나 생성한 뒤, 디스크의 빈 공간을 모두 채울 때까지 파일을 늘린 다음 삭제하는 방식으로 디스크의 빈 공간을 임의의 데이터로 덮어쓴다. 이를 통해서 디스크에 남아있을 원본 데이터의 메타 데이터를 삭제해 복구를 방해한다.

```
v42 = math_rand_Uint64(v30);              // create random filename
v105[0] = 0;
v105[1] = 0;
v43 = runtime_convT64(v42);
v105[0] = &RTYPE_uint64;
v105[1] = v43;
v44 = fmt_Sprintf("wipe-%016x.tmp", 14, v105, 1, 1);
v45 = runtime_concatstring2(v81, v114, v44);
v46 = golang_org_x_sys_windows_UTF16PtrFromString(v45.ptr, v45.len);
v47 = 0;
v48 = 2;
File = golang_org_x_sys_windows_CreateFile(v46, 0x40000000, 0, 0, 2, 0xA4000080, 0);
```

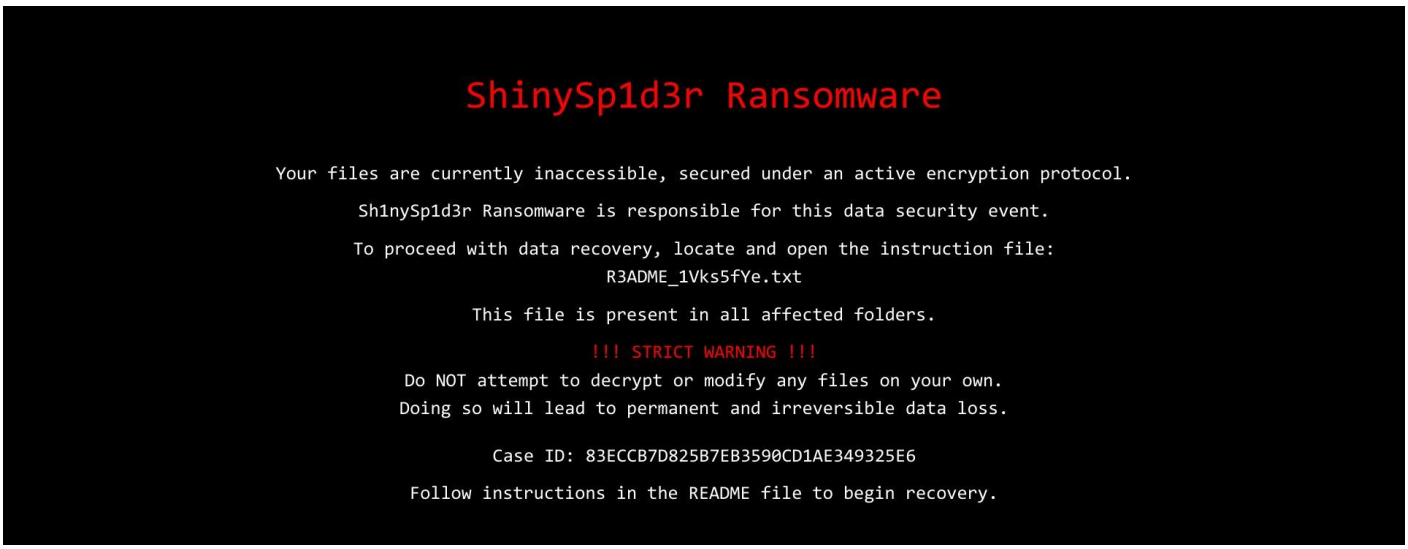
[그림 23] 덮어쓸 파일 생성

Windows 시스템의 이벤트 로그를 삭제하는 기능도 존재한다.

```
v26[1] = 11;
v26[0] = "Application";
v26[3] = 6;
v26[2] = "System";
v26[5] = 8;
v26[4] = "Security";
syscall_NewLazyDLL("advapi32.dll", 12);
v24 = v0;
OpenEventLogW = syscall__ptr_LazyDLL_NewProc(v0, "OpenEventLogW", 13);
ClearEventLogW = syscall__ptr_LazyDLL_NewProc(v24, "ClearEventLogW", 14);
CloseEventLog = syscall__ptr_LazyDLL_NewProc(v24, "CloseEventLog", 13);
```

[그림 24] 이벤트 로그 삭제

랜섬노트와 동일하게 인코딩된 이미지를 사용해 시스템의 바탕화면을 변경하며, 마지막으로 자가 삭제를 진행한 뒤 종료한다.



[그림 25] 변경된 바탕화면

```
v7 = fmt_Sprintf(
       "Set objWMI = GetObject(\"winmgmts:\\\\.\\root\\cimv2\")\n"
       "Set colProcesses = objWMI.ExecQuery(\"SELECT * FROM Win32_Process WHERE ProcessId = %d\")\n"
       "Do While colProcesses.Count > 0\n"
       "    WScript.Sleep 100\n"
       "    Set colProcesses = objWMI.ExecQuery(\"SELECT * FROM Win32_Process WHERE ProcessId = %d\")\n"
       "Loop\n"
       "Set objFSO = CreateObject(\"Scripting.FileSystemObject\")\n"
       "objFSO.DeleteFile \"%s\", True\n"
       "WScript.Quit",
       386,
       v22,
       3,
       3);
os_TempDir();
v19[1] = v7.part_1;
v19[0] = v8;
v21 = 11;
v20 = "cleanup.vbs";
v9 = path_filepath_Join(v19, 2, 2);
v32 = runtime_stringtoslicebyte(0, v7.part_0, v7.part_1);
os_WriteFile(v9.part_0, v9.part_1, v32.part_0, v32.part_1, v32.part_2, 420);
runtime_convTstring(v9.part_0, v9.part_1);
v18[0] = &RTYPE_string;
v18[1] = v10;
v11 = fmt_Sprintf("wscript.exe //B //Nologo \"%s\"", 29, v18, 1, 1);
v17 = syscall_UTF16PtrFromString(v11.part_0, v11.part_1);
p_syscall_StartupInfo = runtime_newobject(&RTYPE_syscall_StartupInfo);
```

[그림 26] 자가 삭제 VBScript

4. 결론

SLSH는 25년 8월 Salesforce 공격 이후 꾸준한 공격을 선보이고 과시적인 커뮤니케이션을 통해 존재감을 점차 키워가고 있다. 이들은 여러 차례 EaaS 및 RaaS 형태의 공개 서비스를 언급하며 공격 수단의 확장 가능성을 시사해온 만큼, 단순한 선언에 그치지 않고 실제 공격으로 이어질 가능성을 염두에 둘 필요가 있다.

또한 25 년 11 월 공개된 ShinySpider 3.0 랜섬웨어는 HellCat 소스코드를 기반으로 한 변종으로, 기존 코드 구조와 동작 방식을 상당 부분 공유하면서 일부 기능이 확장된 형태를 보인다. 현재까지는 Windows 환경을 대상으로 한 제한적인 샘플만 확인됐지만, 기존 소스코드에 기능을 추가해 개발되었다는 점과 지속적으로 ESXi 등 가상화 환경 공격을 언급해왔다는 점을 고려하면, 향후 언제든지 공격 범위가 새로운 플랫폼으로 확장될 가능성이 있어 이에 대한 주의가 필요하다.

5. IoCs

ShinySpider 3.0
3bf53cddf7eb98d9cb94f9aa9f36c211a464e2c1b278f091d6026003050281de
3e780ae2d67a024af3cd6242afbe6d907047d22769075776697e48f6ef677d6d
9ff1051bd0fa917ffa3829e67f5e056ef467333bd2a1d2e11fe527484257932c
50d18f4b11c5d9de7fc16cbc6ca71e65c5e8e9df7d8f3fb192565f035e5adf8a
62dc6ed7c83769648b5c59ad9cc2a4e26daec96a952eb44c93fd45f2011a3444
285aadef3f3cdb367bd89ecbaaa698cc12cdf29fa38f03bfa064de54ab9c6a12
670a269d935f1586d4f0e5bed685d15a38e6fa790f763e6ed5c9fdd72dce3cf2
d12e44a6c04ab4cafda1471a1204fbe3b6f0d01ca4017e3d8ae13fa8870c7689
e41dd341f317cb674ff12c83a17365e5c5aa3240d912ab3801ff4cf09a00ccb2
e225da072f5664bac05e5553ce9707e00e6b4773a24451d836150be6ab0d3c66

HellCat (Go version)
5646171e8e2e3b409293f1b97570264f0b236903c09341be4b42626fecd5d8bb
6924479c42b3732e0d57b34714b7210e14655ee1ca570ae4aab1d90c3f6c6428
b8e71845cc8ccd668a3436d1952a6c57649974bb8399e599dc33afc4c0843be7
dcd7995038ad4839e88e5bb3bf654b4f7c2ad09780a39c9d47596ce717fd4ac2

암호화된 파일 헤더 구조체

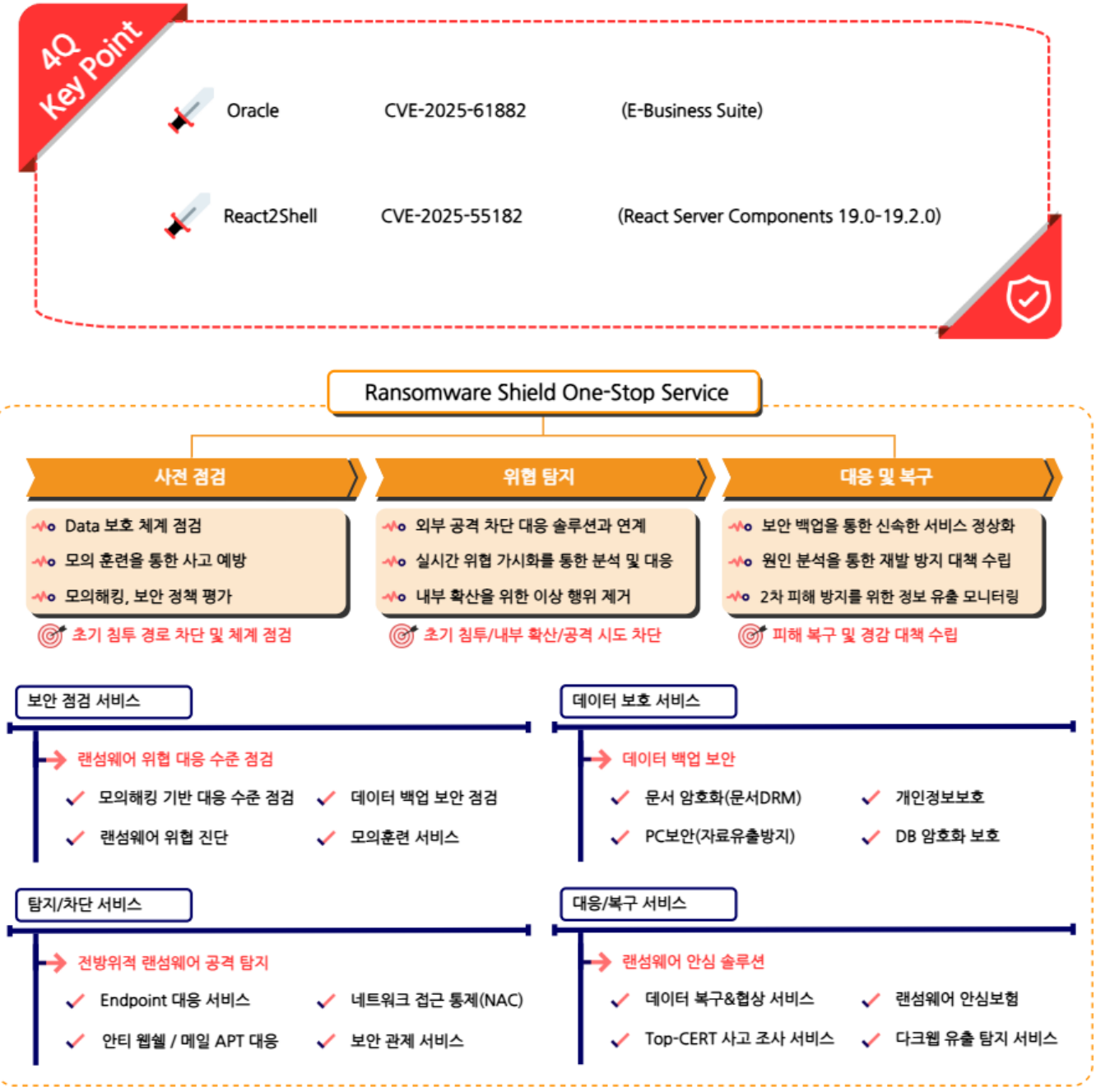
```
#pragma endian big

struct SPDR_Header {
    char magic[4];                  // "SPDR"

    u16   enc_mode;                 // 암호화 모드(1: 전체, 2: 부분)
    u8    fixed_1[3];               // 3 bytes (1 고정)

    u32   enc_key_len;              // 보호된 암호화 키 길이
    u8    enc_key[enc_key_len];     // 보호된 암호화 키
```

```
    u8    nonce[12];                    // 암호화 nonce

    u8    enc_flag;                     // 암호화 모드(0x43: 전체, 0x50: 부분)

    // enc_flag == 0x50일 때만 해시 블록 존재
    if (enc_flag == 0x50) {
        u32 hash_len;                   // 파일명 해시 길이
        u8   hash[hash_len];            // 파일명 해시
    }

    u32   orig_name_len;                // 원본 파일명 길이
    char orig_name[orig_name_len];      // 원본 파일명

    u32   unknown0;                     // 4 bytes (0 고정)
    u32   orig_file_size;               // 원본 파일 사이즈

    char end_magic[4];                  // "ENDS"
};

SPDR_Header header @ 0x0;
```

랜섬웨어 Mitigations

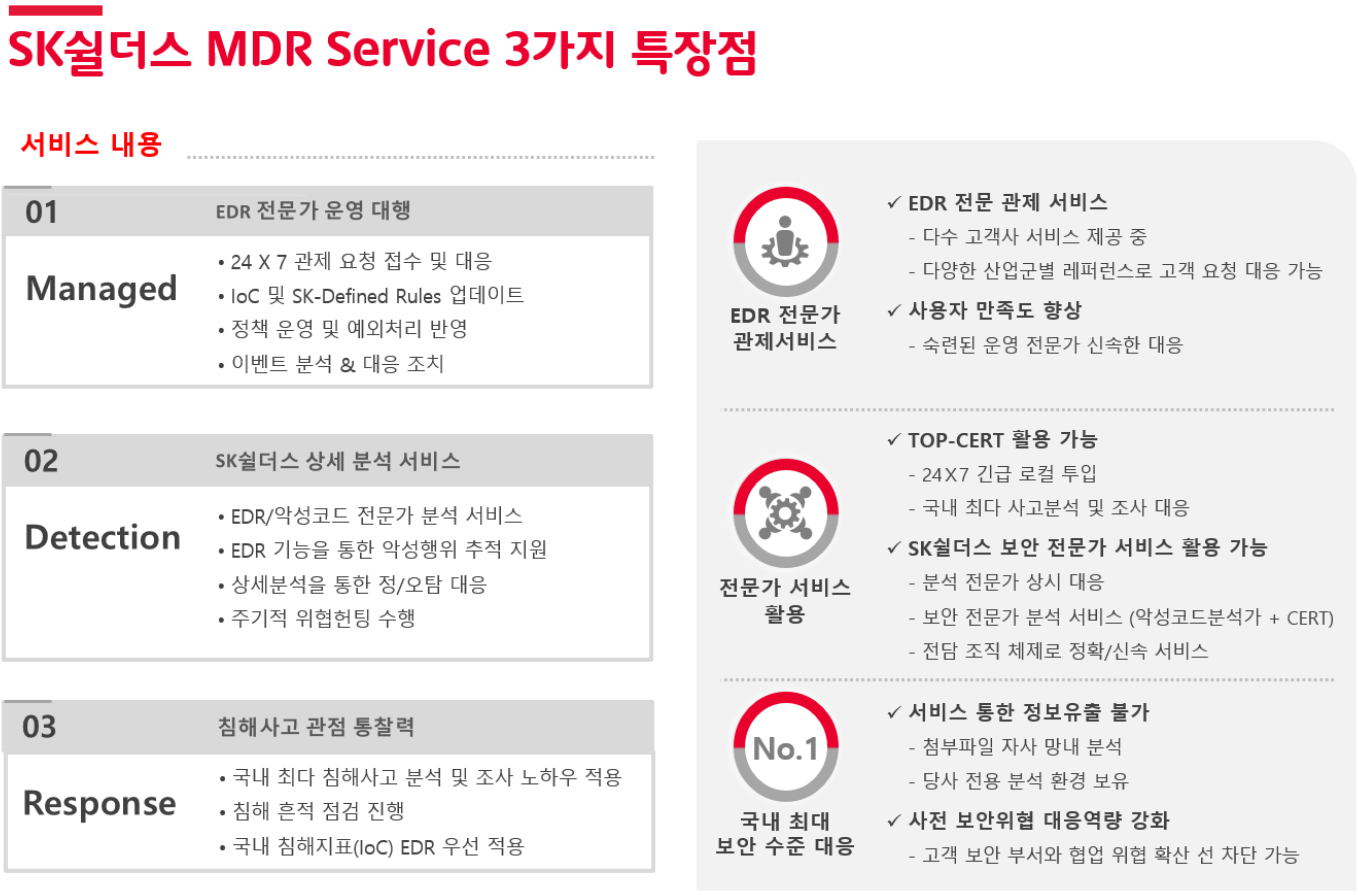
1. 랜섬웨어 대응방안 안내

이번 분기에는 Clop 그룹이 Oracle E-Business Suite 에서 발생한 취약점(CVE-2025-61882)을 이용해 침투를 수행했으며, Weaxor 랜섬웨어는 React2Shell 취약점(CVE-2025-55182)을 악용해 인증 없이 원격 코드 실행을 수행하고, 이를 통해 초기 침투 후 랜섬웨어를 실행했다. 이처럼 주요 랜섬웨어 그룹들의 공격들은 인프라의 취약점과 인증 미흡 지점을 노린 초기 침투 후, 시스템 권한 획득 및 데이터 탈취, 협상 실패 시 유출까지 이어지는 이중 갈취 전략을 사용한다. 이에 대응하기 위해서는 방화벽과 VPN 장비 등 외부 노출 자산에 대한 보안 패치를 신속히 적용하고, 계정 기반 인증 체계를 다중 인증으로 강화하는 한편, 내부 시스템에 대한 행위 기반 모니터링을 통해 수상한 권한 상승과 데이터 이동을 탐지할 수 있는 체계를 마련해야 한다.

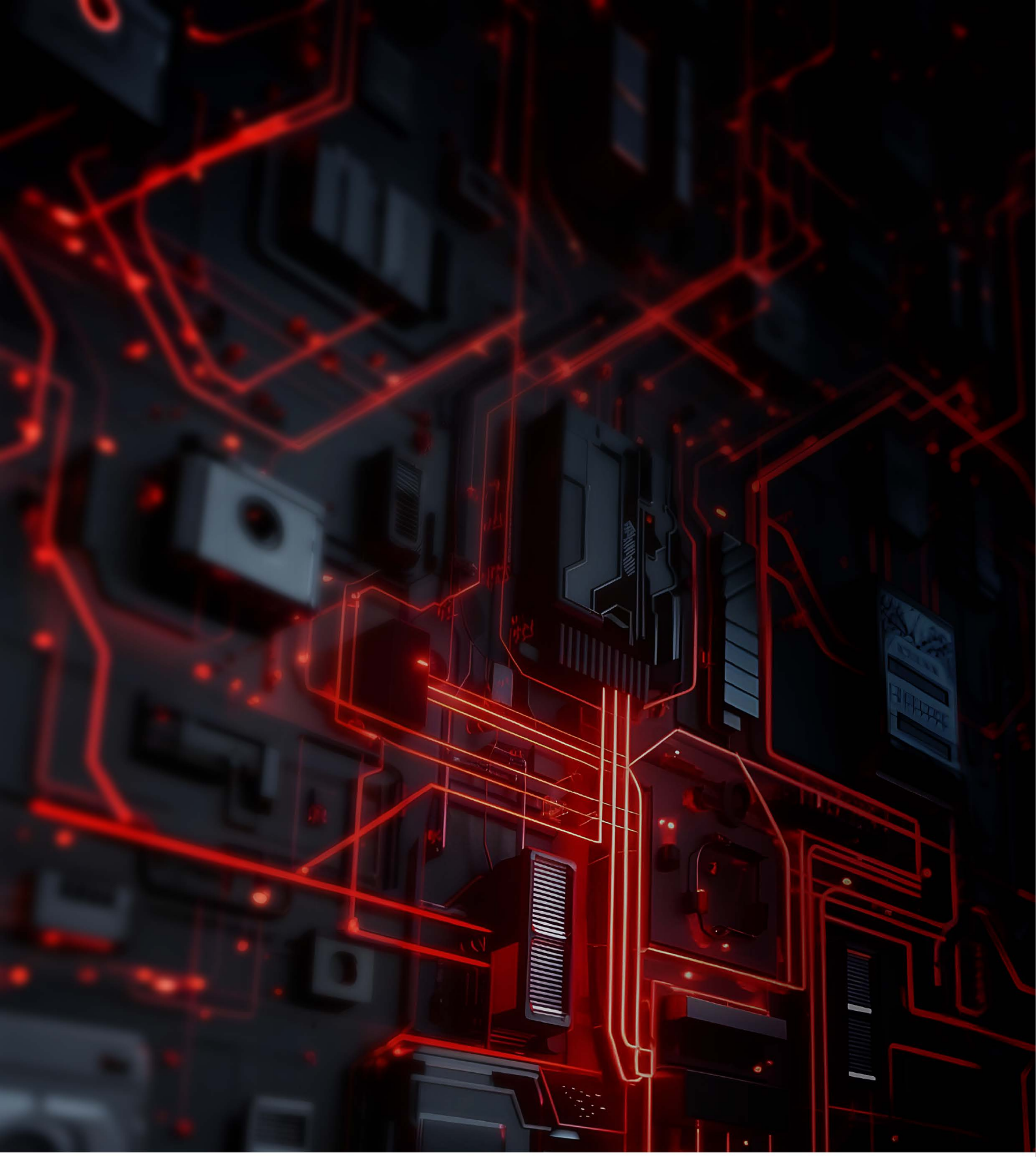


2. SK 쉴더스 MDR 서비스

랜섬웨어에 전문적으로 대응하기 위해서 SK 쉴더스의 MDR(Managed Detection and Response) 서비스[7]를 사용하는 것이 효과적인 방안이 될 수 있다. 최근 랜섬웨어 공격자들의 치밀한 전략과 고도화된 탐지 회피 기법으로 인해 기존의 방어 체계만으로는 위협에서 벗어나기 어려운 상황이다. 이를 해결하기 위해 SK 쉴더스는 실시간으로 네트워크를 모니터링하고 이상 징후를 감지하며 필요시 즉각적으로 대응할 수 있는 MDR 서비스를 제공하고 있다. 랜섬웨어 공격은 사전 예방이 무엇보다 가장 중요하지만, 피해가 발생했을 경우 신속한 조치를 통해 피해를 최소화하는 것 또한 매우 중요하다. 따라서 기업에서는 전담 조직의 신속하고 정확한 사고 조사와 분석을 토대로 맞춤형 보안 솔루션을 제공하는 SK 쉴더스의 MDR 서비스를 고려하는 것을 추천한다.



[7] MDR 서비스: 실시간 위협 감지와 대응을 통해 사이버 공격으로부터 조직을 보호하는 관리형 보안 서비스



안녕을 지키는 기술 | SK 쉴더스

SK쉴더스㈜ 13486 경기도 성남시 분당구 판교로227번길 23, 4&5층
https://www.skshieldus.com

발행인 : SK쉴더스 EQST/SI솔루션사업그룹 & KARA(Korea Anti Ransomware Alliance)
제　작 : SK쉴더스 마케팅그룹