SK shieldus

Threat Intelligence Report

EQST INSIGHT

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.

2025

04

SK shieldus

Contents

Headline

Increase in Cyber Attacks Targeting Medical Institutions and Security Response Strategies

Se-yong Kim / EQST Lab, Principal Research Engineer

■ Overview

The advancement of digital technology, while enhancing the quality of healthcare services, concurrently introduces new forms of security threats. Hospitals possess vast amounts of personal information and medical data, operating within a complex IT environment where various medical devices are interconnected through networks. Particularly Systems that operate 24/7 cannot be taken offline for maintenance, as even brief interruptions may jeopardize patient safety. This unique vulnerability makes such systems prime targets for attackers, who persistently attempt system paralysis through hacking, ransomware, and Distributed Denial of Service (DDoS) attacks.

Years	Total Cases	Incident Type		
		DDoS Attacks	Malware Infection & Distribution (Ransomware)	System Hacking
2020	5	0	5(5)	0
2021	5	2	1(1)	2
2022	23	1	5(5)	17
2023	39	1	7(6)	31
2024	57	0	4(4)	53
Total	129	4	22(21)	103

* Source: Boan News (Data: Korea Internet & Security Agency, Unit: Cases)

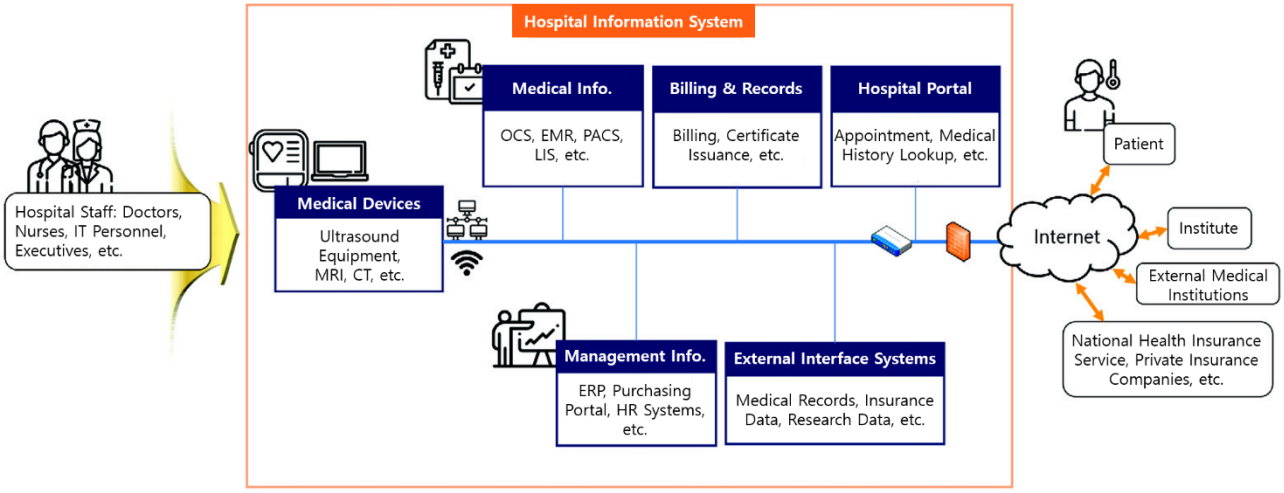
Table 1. Security Breaches in Medical Institutions

Trend of Increasing Incidents

In recent years, cyber-attacks targeting hospitals have been on a steady rise both domestically and internationally. Abroad, there have been instances where large hospital networks were infected with ransomware, leading to the cancellation of surgical schedules and the obstruction of access to patient records. Domestically, incidents involving the leakage of patient information have occurred. These incidents underscore the urgent need for hospitals to establish robust cybersecurity frameworks and safeguard critical informational systems and assets.

System Complexity

The information technology infrastructure of hospitals comprises Electronic Medical Records (EMR), internal network systems, and medical equipment systems. These have continually been digitized to enhance clinical convenience and operational efficiency. However, these advancements conceal numerous security vulnerabilities. For instance, integration with external partner systems, internet-facing equipment, and the prevalence of legacy systems and unsupported software—which create blind spots for patching and threat detection—have become principal avenues of infiltration for hackers.



* Source: National Intelligence Service Hospital Information System Security Guidelines (April 2025)

Figure 1. Hospital Information System

Status of Policy Responses

As cyber threats materialize, governmental responses are being fortified. Agencies such as the Ministry of Health and Welfare, the National Intelligence Service, and the Korea Internet & Security Agency (KISA) are devising policy guidelines to enhance the security capabilities of medical institutions. Among these, the 'Hospital Information System Security Guidelines' released in April 2025 provide concrete standards that practitioners in the field can reference. The purpose of these guidelines is to establish a foundation that enables hospitals to audit their security frameworks, identify their assets, and respond to a variety of threats.

■ Structural Vulnerabilities in Hospital Security

Constraints of the Operational Environment

Medical institutions, inherently linked to life-preserving medical practices, necessitate the absolute continuity of system operations. Consequently, hospitals, unlike typical corporations, face considerable challenges in freely conducting system inspections, network modifications, and the application of security patches. Additionally, the structure whereby various clinical departments and external collaborative organizations access hospital systems complicates the consistent implementation of security policies.

Aging of Technological Infrastructure

Certain information systems employed in some hospitals have been developed long ago, resulting in the utilization of outdated operating systems or the operation under conditions where security patches are no longer provided. Systems such as Electronic Medical Records (EMR), Order Communication Systems (OCS), and Picture Archiving and Communication Systems (PACS) have been constructed independently, complicating integrated management. Moreover, the interfaces designed for interlinking these systems elevate the potential for the emergence of new security vulnerabilities.

Security Limitations of Medical Devices

Numerous medical devices within hospitals are employed for diagnostic, therapeutic, and monitoring purposes and are often connected to networks. However, many of these devices were never security-hardened by design and often run outdated operating systems with default credentials still intact. Additionally, firmware is frequently not updated, leading to vulnerabilities being neglected for extended periods. Given that medical devices represent costly equipment, replacement is not a feasible option in many cases. Consequently, if compromised, these devices pose a risk of impacting the entire hospital system through the internal network.

Risks of External Linkage Systems

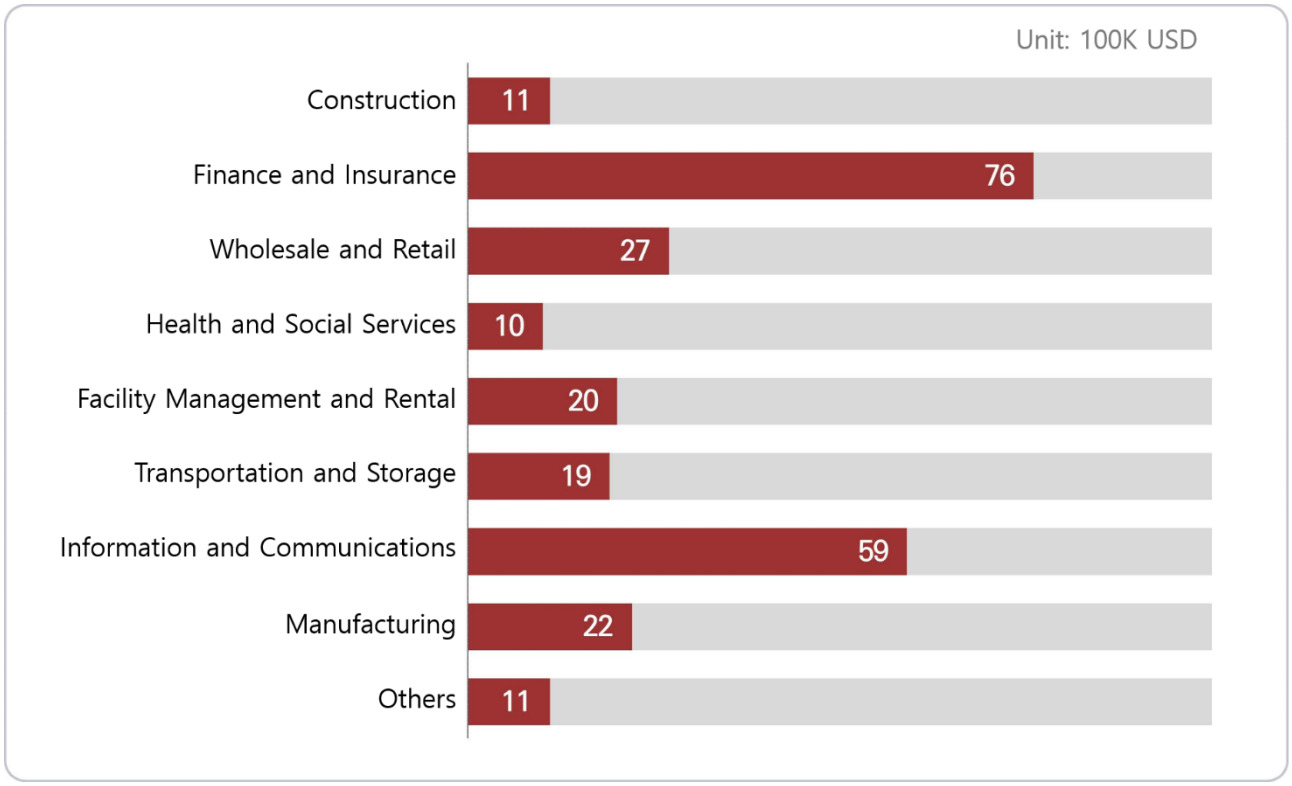
Telemedicine, cloud-based platforms, and internal and external collaboration systems are progressively expanding the boundaries of hospital networks, and in this process, new points of attack are being formed. External usage of VPNs, API integration points, and medical information exchange systems can become susceptible environments to hacking due to issues such as insufficient authentication, lack of encryption, and absence of access control.

Absence of a Security Responsibility Framework

Awareness of information security within hospitals is comparatively low. Additionally, there is often an absence of professional and independent security organizations, and dedicated budgets are frequently not secured. This is particularly problematic for small to medium-sized hospitals, where it is challenging to respond swiftly in crisis situations, and systematic responses for incident analysis or recurrence prevention are not easily implemented. Such organizational limitations can pose a more fundamental security threat than technical vulnerabilities.

Limitations of Security Investment and Awareness

Despite the increasing recognition of the importance of information security, the level of security investment in medical institutions remains lamentably low. According to a survey published by the Korea Internet & Security Agency (KISA) in December 2024, the amount invested in information security within the healthcare sector was reported to be the lowest among eight major industries. The Korean Hospital Association has also highlighted the challenging reality faced by small to medium-sized hospitals, which struggle to allocate budgets for security and to recruit specialized personnel. Investments are often made temporarily for the purpose of external audits or to obtain security certifications, yet there is a lack of organizational culture that manages security strategically as part of daily operations. There is a pressing need for a paradigm shift to view security not as an optional item but as essential infrastructure.



* Source: 2024 Information Security Disclosure Status Analysis Report

Figure 2. Average Investment in Information Security by Industry Sector

■ Legal Liability and Compensation Structure

One of the reasons why investment and management systems for hospital security have not been adequately established is the relatively weak legal and moral accountability associated with information leakage incidents. Despite medical information being among the most sensitive personal data, substantial compensation for victims is often not forthcoming following an incident, and the scope of responsibility that institutions are expected to bear is comparatively low.

Korean Case Study

In South Korea, incidents often result merely in the imposition of fines or recommendations for prevention of recurrence. Between 2018 and 2020, patient information breaches occurred in 17 general hospitals; however, the Personal Information Protection Commission imposed fines on 16 of these hospitals and issued corrective recommendations, yet direct compensation for the victims was not forthcoming.

Foreign Case Studies

Conversely, in foreign jurisdictions, actual compensation for the leakage of patient personal information is being realized. In the United States, Hospital A agreed to pay approximately $650,000 (about 900 million KRW) to a group of victims following a hacking incident in 2023 that resulted in the disclosure of patient information. Additionally, they provided free credit monitoring services for two years.

In another instance, the American healthcare corporation B is currently undergoing settlement procedures that include compensations of up to $10,000 (approximately 14 million KRW) per victim, pertaining to a personal information breach incident that occurred within the same year.

■ Security Incident Analysis

Cyberattacks targeting medical institutions manifest in various forms such as system paralysis, interruption of medical services, and leakage of patient information, with the consequences extending beyond mere financial loss to potentially life-threatening situations for patients. Prominent cases in domestic and international hospitals illustrate the security challenges faced by these institutions. Notably, there is ongoing evidence that North Korean hacking groups are targeting hospitals, and the National Intelligence Service has announced that North Korea is focusing on healthcare institutions as primary targets for multiple infiltration attempts.

Korean Case Study

In 2021, Hospital C fell victim to a large-scale security breach perpetrated by a North Korean hacking group, resulting in the compromise of its internal network. According to the National Police Agency, the attackers infiltrated the hospital's network for approximately two months, during which they exfiltrated personal information pertaining to roughly 810,000 patients and about 17,000 current and former employees. The assault was executed by exploiting vulnerabilities within the hospital's systems and routing through domestic and international external servers. The National Police Agency identified the origin of the attack based on the IP address, techniques for obfuscating the IP address, methods of system intrusion and management, and the use of North Korean terminology, thereby attributing the incident to the North Korean hacking group.

In July 2023, an incident occurred where approximately 185,000 cases of patient personal information were leaked from 17 general hospitals. According to police investigations, it was confirmed that hospital or pharmaceutical company employees accessed specific prescription information and subsequently leaked it externally via emails or USB drives. This was revealed during the investigation process related to violations of pharmaceutical sales regulations.

Foreign Case Studies

In September 2020, a hospital in Germany, referred to here as "D Hospital," suffered a paralysis of its computer network due to an external hacking attack, resulting in the disruption of both emergency services and appointment scheduling systems. Tragically, this led to the death of a patient during transfer to another facility. The intrusion was subsequently traced to a vulnerability in the Citrix VPN equipment.

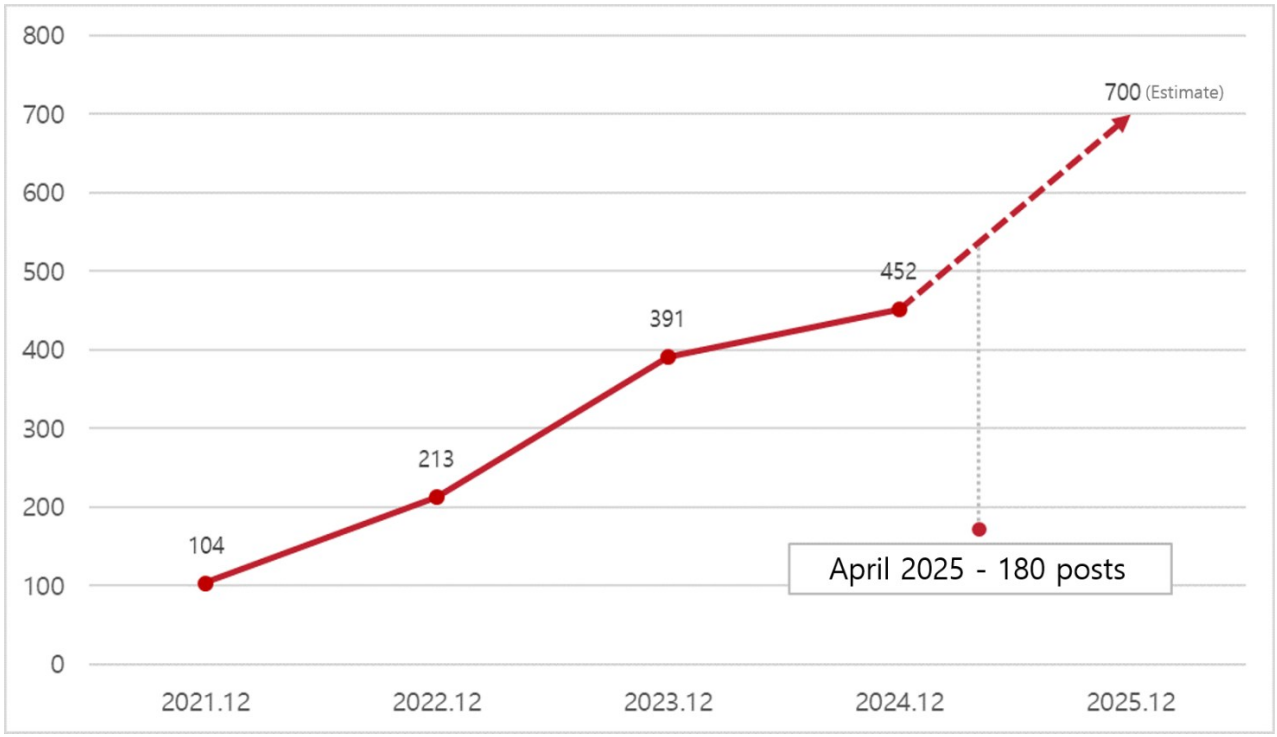
In October 2022, E Corporation, a non-profit healthcare organization in the United States, fell victim to a ransomware attack that simultaneously paralyzed the systems of 164 hospitals and clinics across 21 states, resulting in the breach of approximately 620,000 patients' data. It is conjectured that the attack was perpetrated through a system integration pathway with an external collaborator.

■ The Impact of Patient Personal Information Leakage

The information handled by hospitals transcends mere names or contact details, encompassing highly sensitive medical records such as patients' diagnoses, medical histories, mental health records, surgical histories, and infectious disease details. Should such information be divulged, the resultant harm does not merely cease in the short term; rather, it has the potential to exert a long-lasting impact on the patient's life in various aspects, including employment, insurance, and social stigmatization.

In recent times, the dark web has been a persistent conduit for the circulation of sensitive medical information, which harbors a high potential to be utilized as material for financial crimes or social engineering attacks. Once medical data is leaked, it becomes virtually impossible to delete or retrieve, and tracking its extent of dissemination proves to be a formidable challenge. Consequently, victims are exposed to secondary damages over an extended period, with these damages accumulating and intensifying as time progresses.

In fact, the number of medical data records posted on the dark web has escalated approximately fourfold, from 104 cases in 2021 to 452 cases by 2024, and as of April 2025, 180 cases have already been confirmed. Should this upward trend persist, it is projected that the number of medical information postings will exceed 700 by the end of 2025.



* Source: SK Shieldus

Figure 3. Number of Medical Data Posts on the Dark Web

■ Hospital Information Security and Personal Data Protection Compliance Obligations

Obligations to Protect Patient Information Under the Medical Law

The Medical Law explicitly mandates that medical institutions are responsible for the stringent protection of patients' medical records and personal information. Article 19 of the Medical Law stipulates that medical professionals must not disclose or publish any confidential information about patients that they become aware of during the course of treatment or prescription. Violation of this provision subjects the offender to criminal prosecution. Furthermore, Article 21 permits access to medical records solely to the patient or an individual possessing legitimate legal authority, and Article 22 mandates the preservation of medical records and the implementation of secure management measures for Electronic Medical Records (EMR). These clauses provide a fundamental legal framework for the protection of patient information, with violations resulting in legal sanctions such as fines or administrative penalties.

Application of the Personal Information Protection Act in Hospitals

Hospitals are classified as 'personal information processors' under the Personal Information Protection Act, thereby bearing legal responsibility throughout the entire process of collecting, storing, utilizing, and destroying patient information. According to Article 29 of the Personal Information Protection Act and Article 30 of its Enforcement Decree, medical institutions must establish administrative, technical, and physical safeguards to securely protect personal information. Furthermore, pursuant to Article 34, in the event of a data breach, immediate notification to relevant authorities and measures to prevent further damage are mandatory. Non-compliance can result in penalties, administrative fines, and even criminal prosecution, thus necessitating that hospital information security systems meet the requirements stipulated by this law.

Mandatory Certification of ISMS and ISMS-P

The ISMS (Information Security Management System) certification and the ISMS-P (Information Security and Personal Information Management System) certification are national accreditation schemes jointly implemented by the Ministry of Science and ICT and the Personal Information Protection Commission. According to Article 47, Paragraph 2 of the Act on Promotion of Information and Communications Network Utilization and Information Protection, etc., institutions such as upper-level general hospitals designated under Article 3-4 of the Medical Act, or entities with an annual revenue from information communication services exceeding KRW 100 billion, or medical institutions with a daily average of over one million users, are mandatorily required to obtain this certification if their total annual sales or revenue exceeds KRW 150 billion based on the previous year's figures. The certification criteria encompass the entire spectrum of information security, including the establishment and operation of management systems, protective measures requirements, and specific demands at each stage of personal information processing, divided into three main sections.

Security Requirements for Electronic Medical Records (EMR) According to the Electronic Signature Act

Electronic Medical Records (EMR) encompass sensitive data aggregated from patient treatment information, thus representing a pivotal focus for information security. When medical institutions operate an EMR system, the application of digital signature technology is imperative to ensure the legal validity of this information. Article 23 of the Medical Law stipulates that when medical records and other documents are created electronically, they must include a digital signature in accordance with the Electronic Signature Act. Furthermore, Article 16 of the Enforcement Rules of the Medical Law mandates that facilities must be equipped with the capabilities to generate, store, and verify digital signatures on EMRs. These legal requirements signify that EMR systems must fulfill criteria such as prevention of tampering and alteration, and integrity, from an information security perspective. Consequently, hospitals are obliged to establish a security framework that incorporates digital signature technology to meet these standards.

■ Technical Response Strategies

Application Security Assessment and Vulnerability Diagnosis

External linkage systems operated by hospitals can serve as potential points of penetration for attackers. To counter such threats, regular security audits and vulnerability assessments are imperative, and the history of these inspections must be systematically recorded and managed. Particularly, the utilization of external security experts for scenario-based diagnostics effectively complements the internal security auditing framework. These activities should not conclude as isolated incidents; rather, they must be established as recurring security verification routines, to be performed repetitively with every system modification or service expansion.

Network Architecture Security

Hospitals operate within a complex environment where various networks, including clinical, administrative, medical device, and external networks, are intermixed. This structure inherently facilitates the proliferation of security threats. Consequently, it is imperative that networks within the hospital are segregated either logically or physically. Additionally, when necessary, communication domains specific to particular operations should be isolated using VLANs and dedicated networks.

Additionally, when accessing externally, security communications based on VPN should be the standard, accompanied by condition-based access policies such as device authentication, and restrictions on access time and location. This is a crucial defensive measure designed to block indiscriminate remote access and minimize unnecessary exposure of the system.

System Security and Patch Management

The information systems of hospitals are comprised of various components including operating systems (OS), databases (DB), electronic medical records (EMR), Picture Archiving and Communication Systems (PACS), and web servers, each necessitating independent security configurations and the regular application of patches. For systems where patching proves challenging, alternative technologies such as network access blocking, application control, and execution restrictions based on whitelists must be employed.

Medical Device Security

Medical devices predominantly operate on closed operating systems (OS) or firmware-based platforms, with modifications feasible solely through manufacturers or maintenance providers. This structure often impedes the smooth implementation of security patches and inspections, resulting in the prolonged neglect of outdated systems. Security measures include the segregation of dedicated networks, installation of security gateways, access control, log recording, and the segregation of privileges. Particularly, equipment featuring wireless connections or remote diagnostic capabilities must invariably incorporate encrypted communications and authentication systems.

Account Management and Access Control

Hospital computer systems constitute environments utilized by a diverse array of professional groups, and the tracking of user-specific activities is complicated due to shift work and the use of shared accounts. To address this issue, it is imperative to implement Role-Based Access Control (RBAC) and to mandatorily configure Multi-Factor Authentication (MFA) for administrator accounts.

Additionally, the privileges granted to external collaborators or dispatched personnel must be restricted based on conditions such as time, path, and commands, and all access records should be meticulously preserved in audit logs for regular inspection.

Log Analysis and Anomaly Detection

Security logs should not merely be stored; they must be utilized to detect anomalous activities at an early stage. It is imperative to establish a monitoring system capable of real-time detection of unusual accesses based on system logs and user behavior data. Additionally, a systematic approach should be developed for the periodic analysis of access histories, account usage details, and system modifications. Such frameworks serve as effective means to preemptively thwart security threats, including insider threats and ransomware infections.

■ Administrative Response Strategies

Security Personnel and Internal Capabilities

The cornerstone of hospital information security lies in the stable acquisition of dedicated personnel. As of 2024, the majority of hospital-level medical institutions are operated with fewer than five security staff members, and the average number of dedicated information security personnel across all healthcare institutions is a mere 2.8. Particularly in the case of smaller hospitals, it is commonplace to entrust the entirety of security tasks to external IT maintenance firms due to the absence of full-time security staff. This arrangement not only complicates the ability to respond promptly to security breaches but also diminishes the sustainability of security operations. It is advisable for hospitals of a certain size to secure their own dedicated security personnel and to establish an internal structure capable of performing specialized roles in separated domains such as network, system, and security management.

Security Education and Awareness Enhancement

Hospitals are multifaceted organizations where a variety of professions coexist. Merely distributing simple educational materials is insufficient to adequately disseminate security awareness; instead, customized training tailored to specific job functions is required. For instance, medical staff need training on detecting phishing emails, administrative personnel require education on the risks associated with account management, and those responsible for equipment must be guided on security procedures for medical devices. Such training should be structured to include at least annual regular sessions complemented by simulated exercises, and through continuous practice, a culture of security must be entrenched across the hospital.

Outsourcing and Supply Chain Security

Hospitals possess a structure interconnected with various external entities. Collaborators maintain a certain level of access to the hospital systems, which can potentially be exploited as security vulnerabilities, serving as indirect pathways for cyberattacks. Consequently, it is imperative that contracts with external vendors explicitly include clauses related to information security, and that these agreements are complemented by regular security audits and compliance assessments. Moreover, external access should be regulated based on factors such as time, permissions, and session logs, thereby establishing a framework that enables transparent management of the activities of external personnel.

Response Training and Incident Response System Management

As of 2024, 82.1% of all medical institutions have implemented training exercises to prepare for ransomware and hacking incidents, with the healthcare sector achieving the highest compliance rate at 96.2%. This indicates that regular training exercises substantially enhance security capabilities. Cyber drills should simulate complex, multi-layered incidents — such as EMR outages, device tampering, malfunctioning medical devices, and failed backups — rather than rely solely on phishing simulations. The outcomes of such training not only elevate security awareness but also serve as the foundation for establishing a comprehensive organizational crisis response system.

■ Hospital Information System Security Guidelines

In April 2025, the National Intelligence Service, in collaboration with the Ministry of Health and Welfare and the Korea Internet & Security Agency (KISA), unveiled a new Hospital Information System Security Guide targeted at medical institutions nationwide. This initiative stemmed from the recognition of the limitations in the existing security frameworks, particularly in light of the escalation of sophisticated cyber-attacks aimed at hospitals. These attacks have become increasingly prevalent as new forms of IT environments, including medical devices and external linkage systems, continue to permeate hospital settings.

Should hospital systems become incapacitated, the consequences extend beyond mere service disruptions, potentially endangering patients' lives. Consequently, it is imperative that hospitals autonomously establish security standards and practical guidelines to address a multitude of threat environments.

Guide Configuration

This guide is composed of three domains: network, system and application, and administrative security measures. It is designed to harmonize technical and administrative measures tailored to the hospital environment, and it boasts a flexible structure that can be adaptively applied depending on the size of the hospital.

Field	Description
Network Security Measures	Establish a robust protection framework for the hospital's network environment by logically and physically segmenting its various internal networks (e.g., clinical, medical-device and third-party networks); safeguarding the internal LAN through a demilitarized zone (DMZ) for all external access; and mandating encryption and authentication for every inter-network communication channel, including VPNs and APIs.
System and Application Security Measures	Require security configurations and regular patch management for all components of the hospital information system—OS, database, EMR, PACS, and web servers; establish dedicated security provisions for medical devices, emphasizing internet isolation, access control, and patch verification; and include measures to mitigate vulnerabilities in telemedicine systems and cloud environments.
Administrative Security Measures	Require the designation of an Information Security Officer; assignment of organizational roles and responsibilities; establishment of account and privilege management standards; access control for third-party vendors; implementation of security training; retention and periodic review of security logs; and other administrative controls to maintain and operate the hospital's overall security posture.

Table 2. Overview of Security Guidelines for Hospital Information Systems

■ Conclusion

Cyber threats targeting hospitals transcend mere technical issues. They constitute a grave matter directly linked to patient safety, the continuity of medical care, and the public's trust.

Recent security incidents at hospitals both domestically and internationally have meticulously exploited the structural vulnerabilities and operational flaws of medical institutions, resulting in situations where a single intrusion can lead to the paralysis of the entire system and, at times, even jeopardize the lives of patients.

Due to the intrinsic nature of medical institutions, where patient care is of paramount importance, there are considerable constraints on system inspections and the implementation of enhanced security measures. These limitations foster a perception among attackers that hospitals represent targets with a high likelihood of successful breaches. In reality, compared to other industries, hospitals tend to have slower incident responses and suffer broader extents of damage.

As medical services increasingly transition to digital platforms, the flow of patient information will continue to expand beyond the confines of individual hospitals to interconnected systems. In such an environment, maintaining operational stability and societal trust in hospitals necessitates that security be recognized not as an option, but as an imperative condition.

Keep up with Ransomware

The Rapidly Evolving Vanhelsing Ransomware

■ Overview

In March 2025, the number of ransomware incident cases recorded a decline of approximately 28% to 773 cases, compared to 1067 cases in February. The reduction in incidents during March can be attributed to the Clop group having disclosed all victims affected by the exploitation of vulnerabilities in Cleo's file transfer solution in February. Although there was a decrease from the previous month, the figure of 773 cases remains significantly high. This is due to the emergence and activity of numerous new ransomware groups.

In March, the South Korean threat of ransomware was once again confirmed. The NightSpire group, which emerged as a new entity in March, posted 15 victims during the month alone, including a domestic video content production company. The sample data released comprised a single episode script produced by the company, and despite the scheduled release date having passed, the full data set has not been disclosed. As of April, the dark web leak site has been deactivated.

The group operating BlackLock has announced a new ransomware-as-a-service named Mamona. Initially appearing in September 2023 under the alias LostTrust, the group underwent subsequent rebrandings, adopting the name ElDorado in June 2024, and later rebranding to BlackLock in September 2024. Concurrently, they established a separate entity called Mamona RIP and commenced promotional activities on Russian hacking forums. However, due to inadequate security configurations on the Mamona servers, the Mamona dark web leak site was compromised, leading to the deactivation of the BlackLock leak site. Presently, only the BlackLock leak site remains accessible.

It has been confirmed that the RansomHub ransomware was recently distributed through a malware service framework known as SocGholish (FakeUpdates). SocGholish emerged in 2018 and operates by injecting malicious scripts into legitimate websites, subsequently hijacking user traffic. When users visit these compromised websites, they are redirected to a counterfeit page disguised as a browser update notification, where they are prompted to download a ZIP-compressed SocGholish script file. Upon execution of this script file, attackers gain the capability to exfiltrate data or execute remote commands, a method through which the distribution of RansomHub ransomware has been facilitated.

It has been confirmed that the Akira Group recently succeeded in executing a ransomware attack by exploiting webcams. The group employed their customary strategy of infiltrating systems either by utilizing exposed credentials in vulnerable remote access solutions or through brute force attacks, subsequently attempting to deploy ransomware payloads. However, their attempts to distribute ransomware were thwarted by an Endpoint Detection and Response (EDR)[1] solution. In response, they identified vulnerable webcams within the accessible systems, capitalizing on the fact that these webcams lacked EDR protection. By gaining remote shell access, they successfully deployed the ransomware.

[1] EDR: A solution for real-time detection, analysis, and response to malicious activities on endpoints (computers, mobile devices, and servers), preventing damage spread.

■ Ransomware News

RansomHub Group Distributes Ransomware via SocGholish Malware-as-a-Service

- SocGholish intercepts user traffic by injecting scripts into legitimate websites.
- Visiting a compromised site may trigger a SocGholish script disguised as a browser update.
- Executing the file allows the attacker to steal data or execute remote commands.

Akira Group Successfully Executes Ransomware Attack Webcams

- The attacker deployed the ransomware payload via network-connected webcams to evade EDR.
- The attacker exploited webcams to gain remote shell access, taking advantage of the absence of EDR solutions.

North Korean threat group Moonstone Sleet observed distributing Qilin ransomware

- The group Moonstone Sleet has a history of distributing its proprietary ransomware, FakePenny.
- The group began distributing Qilin ransomware in late February 2025.

Mamona Ransomware-as-a-service(RaaS) was hacked by the DragonForce

- Mamona is a newly unveiled RaaS operated by the group known as BlackLock.
- Insufficient security settings led to the exposure of the administrator interface and management panel.
- The darkweb leak site of Mamona was Hacked by DragonForce, rendering BlackLock's DLS inactive.
- BlackLock's DLS was recovered roughly two weeks following the hack.

The newly emerged Vanhelsing group begin group partners

- The advertisement for RaaS partner recruitment was posted on RAMP, a prominent Russian hacking forum.
- The scope of targeted platforms expanded within just three weeks of its initial release.
- Total of eight victims were posted over a period of about one month.

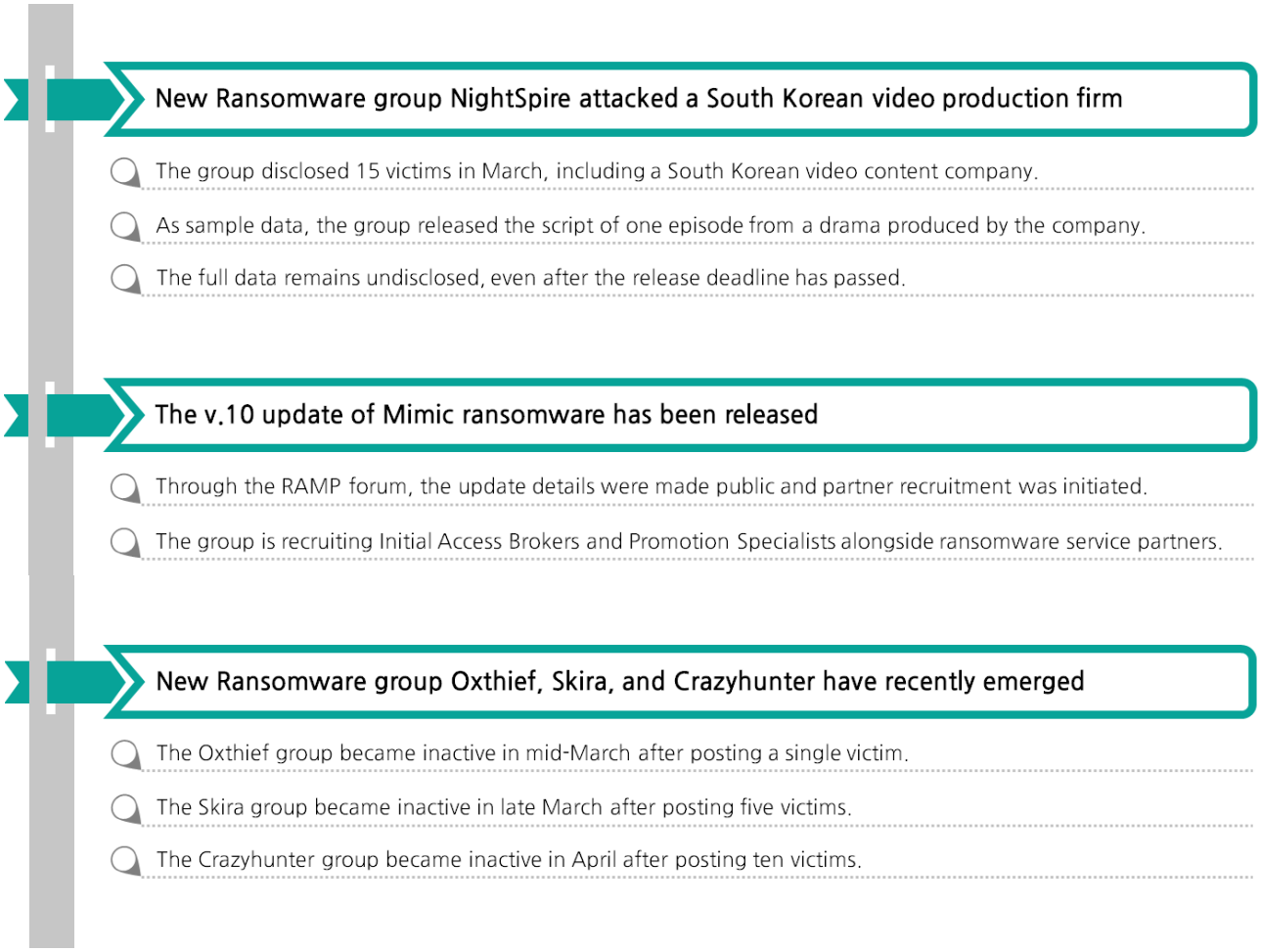


Figure 1. Trends in Ransomware

Ransomware Threats

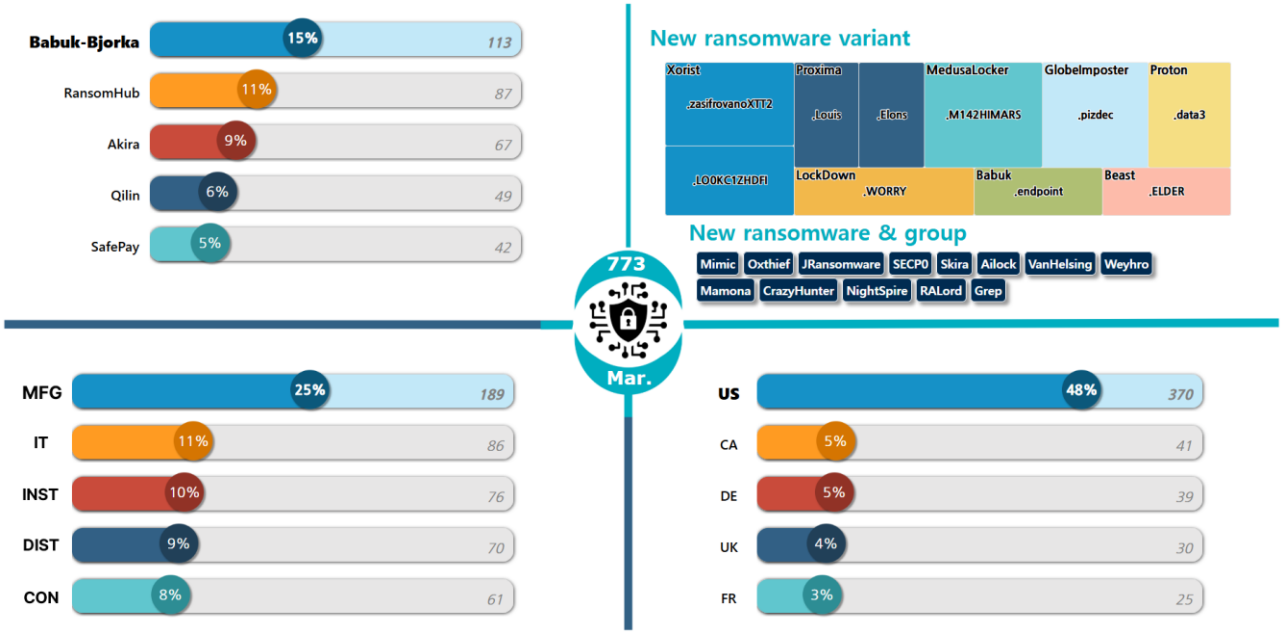


Figure 2. Ransomware Threat Landscape as of March 2025

New Threats

In March, updates concerning existing ransomware groups were observed, alongside the identification of several new ransomware collectives. A total of six new groups were detected, of which three—Oxthief, Skira, and CrazyHunter—are currently inaccessible as of April. Initially emerging in early March, the Oxthief and Skira groups uploaded one and five victims respectively; however, their dark web leak sites became inactive starting from March. Similarly, the leak site for CrazyHunter was deactivated in April.

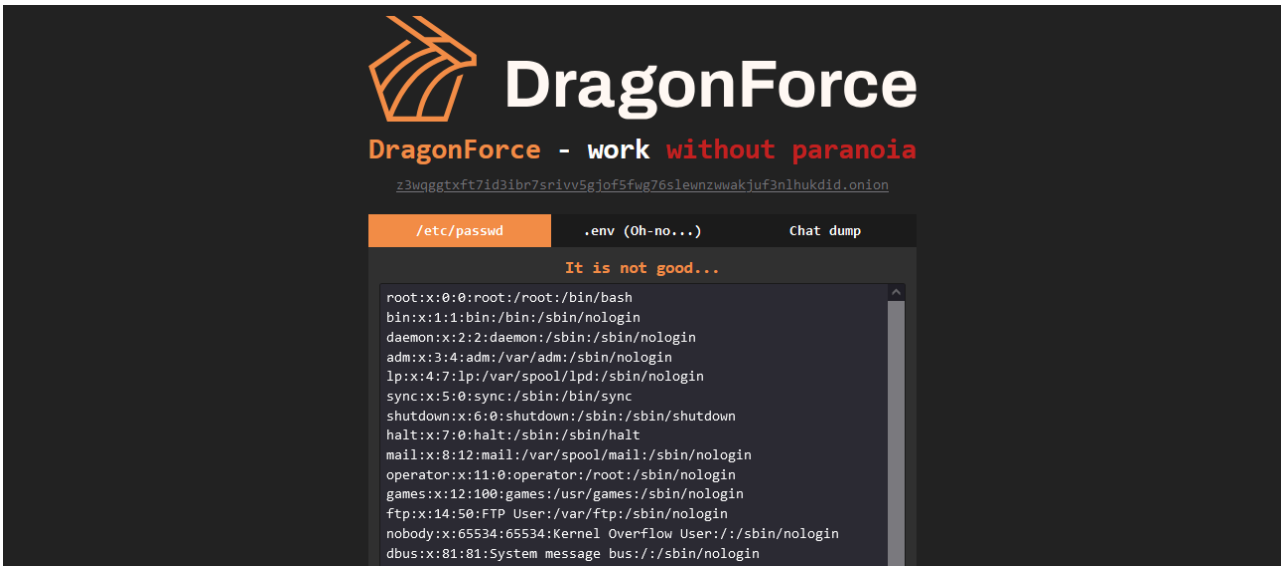


Figure 3. The Mamona dark web leak site hacked by DragonForce

The group operating BlackLock recently unveiled a new ransomware-as-a-service called Mamona, which has been compromised due to inadequate security configurations. Not only was the Mamona administrator page exposed, but also its management panel. This issue was shared on a Russian hacking forum, enabling forum users to access the service unauthorized and view data. Subsequently, BlackLock's leak site was deactivated, and the DragonForce group even tampered with Mamona's dark web leak site. Approximately two weeks later, BlackLock restored the deactivated dark web leak site; however, Mamona is no longer operational.

A new group has been identified on a Russian hacking forum promoting their ransomware services. The Vanhelsing group commenced recruitment of partners in early March to utilize their Ransomware-as-a-Service (RaaS)[2] on the Russian hacking forum, and within three weeks of launch, they expanded their target platforms. Additionally, they posted eight victims over the course of a month.

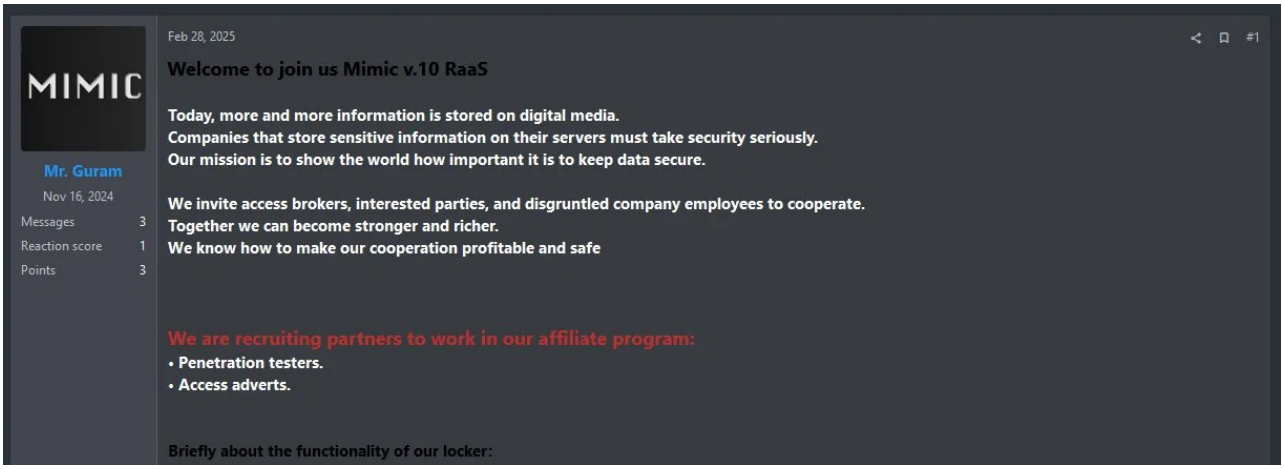


Figure 4. Promotional Material for Mimic v.10

[2] RaaS (Ransomware-as-a-Service): business model that offers ransomware as a service, enabling anyone to easily create and launch ransomware attacks.

Since June 2022, the Mimic ransomware, known for its activities, has launched v.10 and is currently recruiting initial penetration experts and access advertisement managers on Russian hacking forums. The ransomware they offer supports a diverse range of operating systems including Windows, ESXi[3] ,NAS[4] ,FreeBSD[5]. In addition to providing ransomware, they also offer services such as making extortionate calls to victims and supplying software necessary for various operations.

Two ransomware groups have been identified that, although they are ransomware operators, do not manage separate data leak sites but solely operate chat pages for ransom negotiations. These groups are JRansomware and Ailock. They provide a chat page URL and a session ID required for login in the ransom note, thereby administering a chat page individually for each victim.

[3] ESXi: UNIX-based logical platform developed by VMware that allows a host computer to run multiple operating systems concurrently.

[4] NAS: Storage solution comprising multiple storage devices that are accessible over a network.

[5] FreeBSD: Unix-inspired OS distributed as open-source software.

Top 5 Ransomware

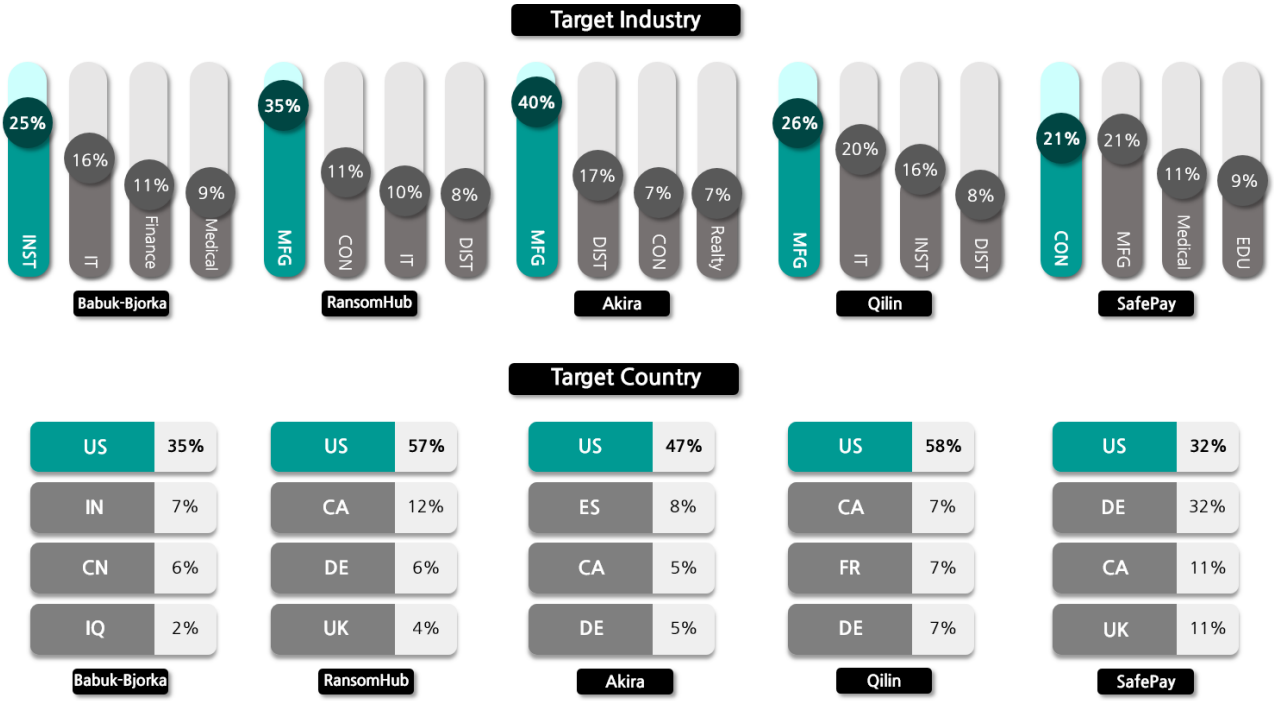


Figure 5. Current Status of Major Ransomware Attacks by Industry/Nation

The group known as Babuk-Bjorka, which claims to be Babuk2, commenced their operations in January. They exhibited their most prolific activity in March by posting 113 victims, many of whom had previously been compromised by other groups such as RansomHub, Meow, Everest, Babuk, Funksec, and LockBit, with their data already made public. Consequently, it is imperative to ascertain whether these incidents represent genuine attacks where data was exfiltrated by the group itself, or whether they merely recycled previously disclosed data to extort ransom.

The RansomHub group orchestrated an attack on the Malaysian engineering services corporation, HexcoSys Group, resulting in the exfiltration of 336GB of data, which included contracts, blueprints, source codes, and product development data. Additionally, they targeted Japan Rebuilt, a Japanese automotive parts manufacturer, from which they leaked 200GB of data encompassing production data, financial information, payment details, and customer records.

In March, the Akira Group launched an attack on CS Plastics, a Belgian industrial machinery manufacturer, resulting in the leakage of sensitive data, including audit reports, financial statements, and personal information of employees and customers. Additionally, recent observations have confirmed that they are employing novel attack strategies, such as using webcams to circumvent the detection capabilities of EDR (Endpoint Detection and Response) solutions. Detailed information on Akira Group's specific attack strategies and countermeasures can be found in the SK Shields KARA Ransomware Trend Report for the fourth quarter of 2024.

It has been recently revealed that the Qilin group is associated with the North Korean threat group known as Moonstone Sleet. Moonstone Sleet, a threat group with a history of distributing its self-developed FakePenny ransomware, has been identified as deploying the Qilin ransomware payload since the end of February 2025.

On March 30th alone, the SafePay Group disclosed a batch of 31 victims. Among these, Sir William Ramsay School, a secondary school located in High Wycombe, UK, suffered a data breach involving the leakage of 183GB of data. Similarly, Brighton Australia, a construction contracting firm in Australia, experienced the exfiltration of 160GB of data, which included financial statements, accounting records, personnel files, and customer documents.

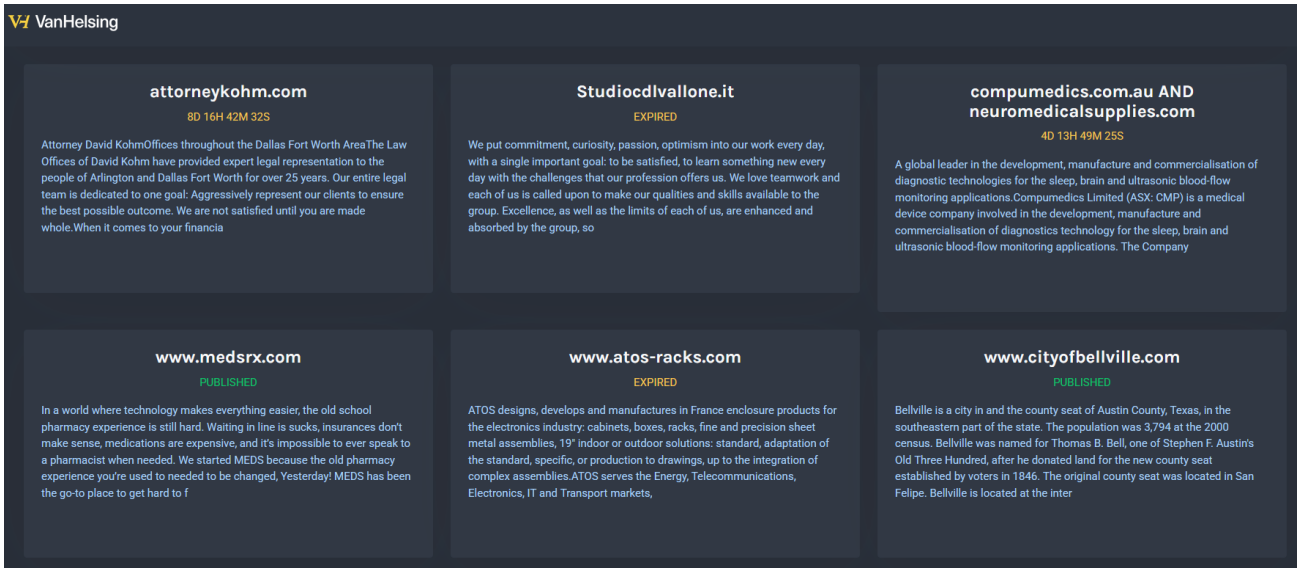


Figure 6. Vanhesling Dark Web Leak Site

The Vanhelsing group, which emerged on March 7, is a ransomware collective that began recruiting affiliates on the Russian hacking forum RAMP. They do not require a membership fee from individuals with a certain level of reputation; others, however, must pay a deposit equivalent to 5,000 USD to join. According to promotional posts on the forum, the group prohibits attacks against CIS countries and demands only a 20% commission from the ransom obtained.

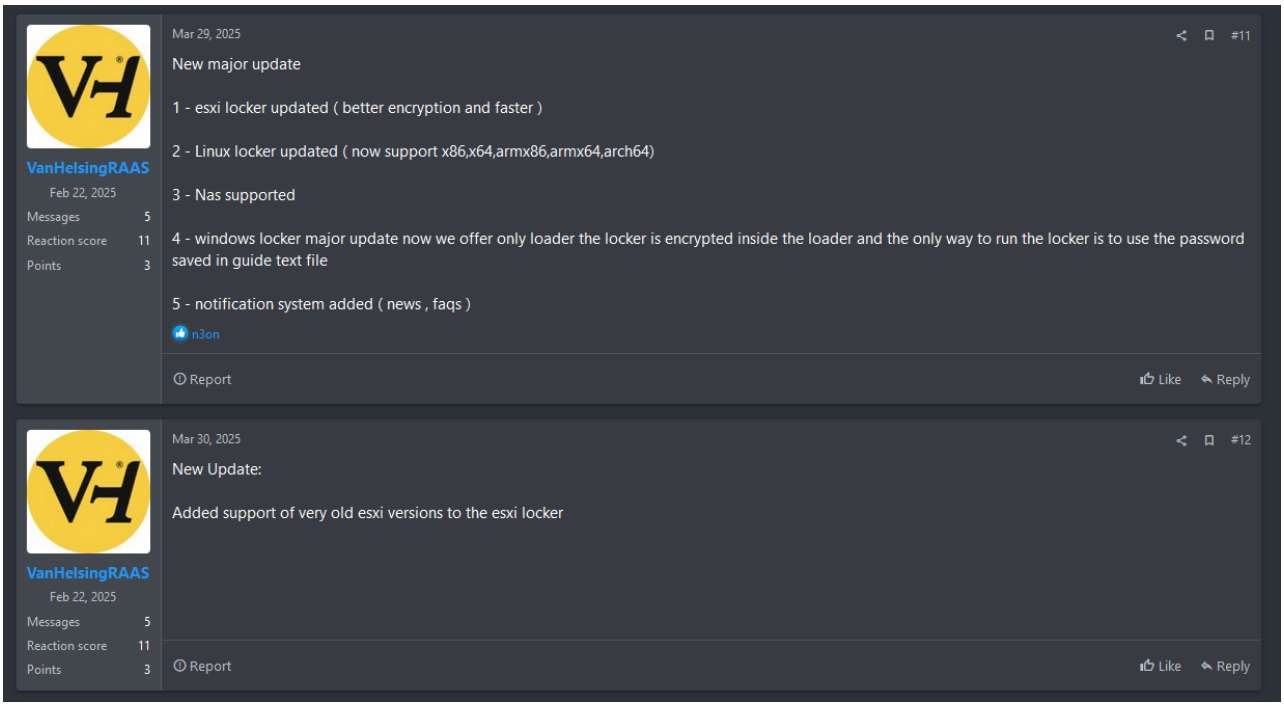


Figure 7. Vanhelsing Feature Update

From the outset, the developers have introduced encryption support targeting a diverse array of platforms including Windows, Linux, BSD[6] , ARM[7] and ESXi. According to the update history posted by the operator at the end of March, the capability to attack a broader spectrum of versions and architectures has been enhanced for Linux and ESXi, and Network Attached Storage (NAS) has also been added to the list of supported targets for attacks.

To date, only versions of Windows ransomware utilizing the encryption extensions 'vanhelsing' and 'vanlocker' have been identified. These two versions of ransomware were developed approximately five days apart, with the latest iteration featuring a newly added capability for propagating across internal networks—a feature absent in earlier versions. Additionally, although not yet implemented, parameters related to the propagation through vCenter[8] have also been detected. The rapid pace of updates to the ransomware's functionalities is not only evident but, according to promotional materials for the ransomware, it appears likely that future versions will be capable of targeting a broader array of platforms. Consequently, it is imperative to prioritize the examination of the contents pertaining to the Windows version of the ransomware in preparation for the impending threats.

[6] BSD: suite of Unix-inspired operating systems originally developed at the UC Berkeley.

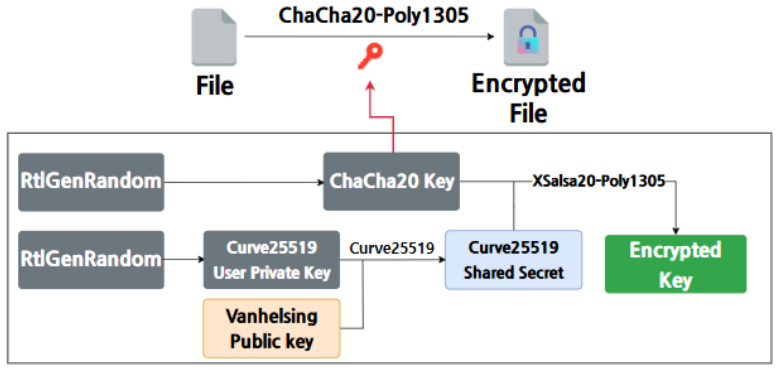
[7] ARM: low-power, high-performance processor architecture widely adopted in smartphones, tablets, and IoT devices.

[8] vCenter: unified management solution from VMware for administering clusters of ESXi hosts along with associated infrastructure.

Vanhelsing Ransomware

Encryption Key

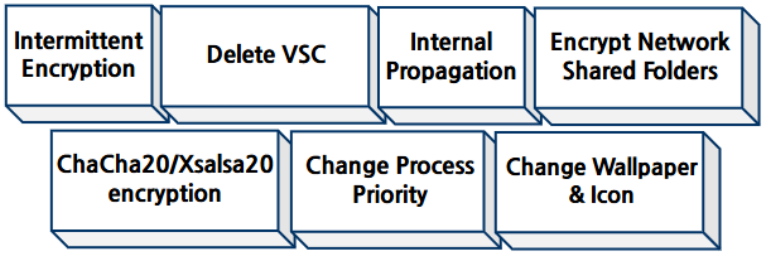
Encrypt files with ChaCha20-Poly1305 algorithm and protect the key with Curve25519 shared secret



Encryption Method

1GB or less: Encrypt the entire file
More than 1GB: Encrypt 30% of total size

Features



Ransom Note

README.txt - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

--= No news is a good news ! =--

Your network has been breached and all your files Personal data, financial reports and important documents has been stolen , encrypted

if you willing to continue your bussines and make more money and keep bussines secret safe you need to restore your files first, And to r
don't bother your self and wast your time or make it more harder on your bussines , we developed a locker that can't be decrypted using

making your self geek and trying to restore the files with third part decrypter this will leads to lose all your date ! and then the even you

to chat with us :

1 - Download tor browser https://www.torproject.org/download/
2 - go to one of these links above
http://vanhelcbxqt4tqie6fuevfng2bsdtxgc7xslo2yo7nitaacdfrlpxnqd.onion
http://vanhelqmjstkvlhrjwzgjzpq422iku6wlggiz5y5r3rmfdeiaj3ljaid.onion
http://vanhelsokskrlaacilyfmtuqqa5haikubsjaokw47f3pt3uoivh6cgad.onion
http://vanheltarnbfjhuvggbncniap56dscnzz5yf6yjmxqivqmb5r2gmllad.onion

3 - you will be asked for your ticket id to enter the chat this for you : TICKET ID

usefull links :
#OUR TOR BLOG :
http://vanhelvuuo4k3xsiq626zkqvp6kobc2abry5wowxqysibmqs5yjh4uqd.onion
http://vanhelwmbf2bwzw7gmseg36qqm4ekc5uuhqbsew4eihzcahyq7sukzad.onion
http://vanhelxjo52qr2ixcmtjayqqrcodkuh36n7uq7q7xj23ggotyr3y72yd.onion

README.txt

Extension

.vanhelsing, .vanlocker

Production Language

C/C++

Figure 8. Overview of Vanhelsing Ransomware

Vanhelsing Ransomware Strategy

Figure 9. Vanhelsing Ransomware Attack Strategy

The Vanhelsing ransomware is capable of utilizing a variety of execution parameters to determine the targets and methods of encryption, as well as to decide whether to enable functions such as changing the desktop background or deleting backup copies. However, there is a discrepancy between the help messages displayed by the ransomware and the actual parameters used; some parameters are merely verified without being employed, and in some instances, the functionalities are not implemented. The parameters and functionalities that are actually verified are as follows in the table below.

Parameters	Description
-h	Display help for command-line parameters
-v	Display logs
--skipshadow	Skip deleting Volume Shadow Copy(VSC)
--Driver <driver>	Encrypt only specified drives
--Directory <directory>	Encrypt only specified folders
--File <file>	Encrypt only specified files
--Force	Allow concurrent execution of the ransomware
--no-priority	Skip setting ransomware priority
--no-wallpaper	Skip changing the desktop background
--no-local	Skip encrypting local files
--no-mounted	Encrypt only fixed local drives
--no-network	Skip encrypting network shared folders
--spread-smb	Propagate within the internal network
--no-logs	Disable log output
--no-admin	Execute regardless of administrative privileges
--Silent	Bulk convert file extensions after encrypting all target files
--system	Feature not implemented
--no-autostart	Feature not implemented
--spread-vcenter	Feature not implemented

Table 1. Execution Parameters of Vanhelsing Ransomware

Upon verifying the execution parameters, several measures are undertaken to prevent errors during the encryption process. These include changing the desktop background and icon, and accessing network shared folders, which necessitate administrative privileges. Consequently, it is essential to ascertain whether the ransomware is currently running with administrative rights. If it is not executing under administrative privileges, the ransomware will terminate, unless the "--no-admin" Parameter is employed, which allows the ransomware to continue operating without administrative rights. Additionally, to prevent the ransomware from executing multiple instances, a mutex[9] is created using the string "Global\\VanHelsing". To enhance the encryption speed, the ransomware's process priority is set to the highest level. Both of these functionalities can be deactivated using the "--Force" and "--no-priority" parameters, respectively.

Additionally, the encrypted files are rendered irrecoverable by the user through the deletion of backup copies. Similarly, employing the "--skipshadow" Parameter prevents the deletion of these backup copies. The command used to delete the backup copies is as follows.

cmd.exe /c C:\\Windows\\System32\\wbem\\WMIC.exe shadowcopy where "ID='%s'" delete

Table 2. Command for Deleting VSC

When the "--spread-smb" Parameter is employed, propagation within the internal network becomes feasible. To execute ransomware on PCs or servers connected to the internal network, psexec[10] is utilized. This program is stored alongside the ransomware, hence it is saved in a temporary folder before being deployed.

```
network_list = WSAStartup_sub_40CA90(pMemoryBlock: pMemoryBlock_1);
GetTempPathW(nBufferLength: 0x1F4u, lpBuffer: temp_path);
m_format_string_sub_40D890(Buffer: psexec_path, Format: L"%s\\psexec.exe", temp_path);
m_print_message_sub_4011D0(L"[*]\tpsexec_path : %s \n");
memset(buf: stream, value: 0, n0x20: 0xB0u);
m_wfsopen_sub_40BB30(buf: stream, Buffer: psexec_path, n48: 0x30, v43, Buffer: psexec_path);// open %TEMP%psexec.exe stream
m_write_stream_sub_40BD50(this: stream, &psexec_data, 0xAED90i64);
m_close_stream_sub_406660(buf: stream);
```

Figure 10. Storage of psexec

[9] Mutex: synchronization mechanism that prevents multiple threads or processes from accessing a shared resource simultaneously, often used in ransomware to block duplicate executions.

[10] psexec: Tool designed for remotely managing and launching processes on Windows platforms via the command line.

After saving psexec, the next step is to retrieve the IP address of the system running the ransomware. Then, the final octet[11] is varied from 1 to 255 to determine if any internal network addresses are reachable. Once a reachable address is identified, the ransomware is copied to a network folder on that address with write permissions and renamed 'vanlocker.exe'. Using psexec, the ransomware is then executed on PCs or servers connected to the internal network. During this operation, the execution parameters '--no-mounted' and '--no-network' are used, and the following command is executed:"

```
cmd.exe /c %TEMP%psexec.exe -accepteula \\${shared_folder} -c -f
${shared_folder}\vanlocker.exe -d --no-mounted --no-network < NUL
```

Table 3. Internal Propagation Commands

In addition to propagation within internal networks, the configuration of targets for file encryption can be facilitated through various execution parameters, which are primarily categorized into files, folders, local drives, and network shared folders. Utilizing the "--File" Parameter encrypts only a single specified file, whereas the "--Directory" Parameter encrypts the designated folder along with all its subfolders, and the "--Driver" Parameter encrypts the entirety of the specified drive. Should no encryption-related execution parameters be employed, all drives and network shared folders are encrypted. Moreover, the use of the "--no-network" Parameter deactivates the encryption of network shared folders, the "--no-local" Parameter omits the encryption of local files, and the "--no-mounted" Parameter enables the encryption of solely the fixed local drives.

[11] Octet: 8-bit unit used to represent a 32-bit IP address by partitioning it into 8-bit segments.

Once the encryption targets have been established, each directory is traversed to ascertain whether it corresponds to an exception item. Initially, it is imperative to verify if the target directory itself is an exception item. Upon completion of this directory verification, it is then necessary to determine whether each file within the directory qualifies as an exception item. The encryption exceptions under consideration are outlined in the table below.

Directory Name	File Extension and File Name
tmp, winnt, temp, thumb, $Recycle.Bin, $RECYCLE.BIN, System Volume Information, Boot, Windows, Trend Micro, program files, program files(x86), tor browser, windows, intel, all users, msocache, perflogs, default, microsoft	.vanlocker, .exe, .dll, .lnk, .sys, .msi, .bat, .bin, .com, .cmd, .386, .adv, .ani, .cab, .ico, .bod, .msstyles, .msu, .nomedia, .ps1, .rtp, .syss, .prf, .deskthemepack, .cur, .cpl, .diagcab, .diagcfg, .dll, .drv, .hlp, .pdb, .hta, .key, .lock, .ldf, .icns, .ics, .idx, .mod, .mpa, .msc, .msp, .nls, .rom, .scr, .shs,. spl, .theme, .themepack, .wpx, boot.ini, autorun.inf, bootfont.bin, bootsect.bak, desktop.ini, iconcache.db, ntldr, ntuser.dat, ntuser.dat.log, ntuser.ini, thumbs.db, GDIPFONTCACHEV1.DAT, d3d9caps.dat, LOGS.txt, .README.txt

Table 4. Exceptions to Encryption

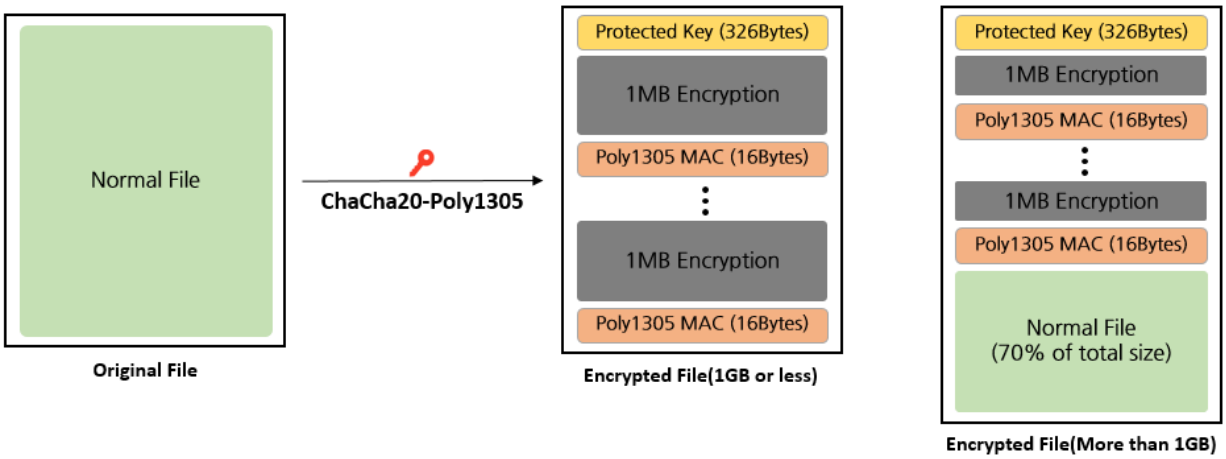


Figure 11. File Encryption Methods by Size

File encryption involves encrypting the entirety of files smaller than 1GB, while for files exceeding 1GB, only 30% of their total size is encrypted. For each file, a random 32-byte key and a 12-byte nonce are generated, and the file is encrypted using the ChaCha20-Poly1305 algorithm. The encryption process is conducted in 1MB increments. Owing to the characteristics of the ChaCha20-Poly1305 algorithm, a Message Authentication Code (MAC) is generated to ensure data integrity. Consequently, each 1MB segment of the encrypted file includes a 16-byte MAC stored alongside it.

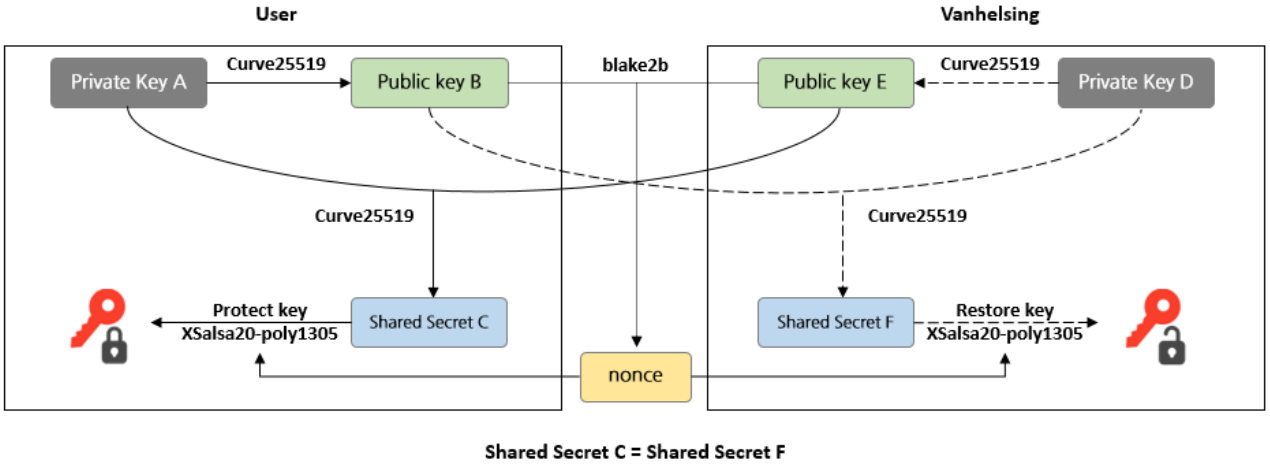


Figure 12. Key Protection Methods

Additionally, the key used for encryption is safeguarded by being stored at the beginning of the file, employing a method that utilizes a shared secret generated via Curve25519 to protect the encryption key and nonce, while concurrently storing the user's public key, which enables key recovery. Separate from the file encryption key, a unique random private key is generated for each file, thereafter, a shared secret can be created using the hardcoded public key of Vanhelsing. This leverages the characteristic of Curve25519, where the shared secret produced using one's private key and another's public key is identical to that generated using one's public key and another's private key. For key protection, the XSalsa20-Poly1305 algorithm is employed, which, in addition to the key, requires supplementary data to be used as a nonce. The nonce is formed by concatenating the user's public key with Vanhelsing's public key, followed by generating a 12-byte hash using the blake2b algorithm.

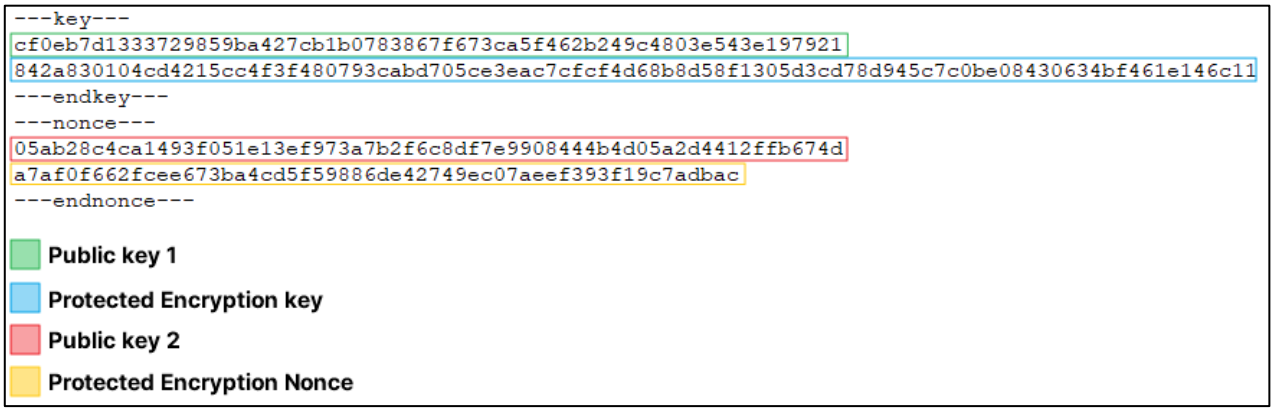


Figure 13. Protected Key Storage Methods

The protected key is stored at the forefront of the encrypted file in text form, alongside the public key intended for recovery. Distinctions are made between the utilized key and nonce, indicated as ---key---, ---nonce---, and the public key and protected key are sequentially saved.

Figure 14. Altered Desktop Background

Following the encryption of files, the ransomware modifies the desktop and the icons of encrypted files to those stored within its own repository, specifically using image and icon files. The desktop wallpaper is altered to "vhlocker.png," and the icon image is changed to "vhlocker.ico." Utilizing the "--no-wallpaper" Parameter precludes alterations to the desktop wallpaper and icon imagery.

Strategies for Responding to Vanhelsing Ransomware

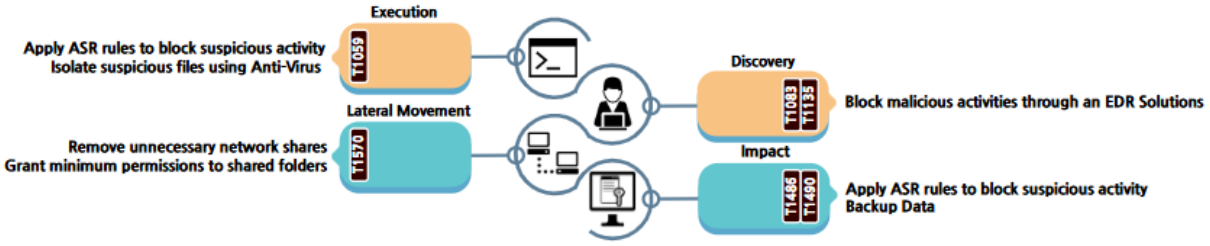


Figure 15. Response Strategies for Vanhelsing Ransomware

The Vanhelsing ransomware utilizes the Windows command prompt to execute commands for deleting backup copies and facilitating internal propagation. Consequently, by activating ASR (Attack Surface Reduction) [12] rules, one can obstruct anomalous processes and thus thwart malicious activities. Additionally, since the ransomware stores programs in temporary folders or replicates itself in network-shared folders, employing Anti-Virus software enables the isolation of suspicious files.

To facilitate encryption for internal dissemination and network shared folders, the current system's internal network bandwidth is explored, and attempts are made to access connectable network addresses and shared folders. Moreover, in the case of file encryption, all drives are scanned and, based on execution parameters, the drives to be encrypted are distinguished. Through the implementation of an Endpoint Detection and Response (EDR) solution, it is possible to thwart the malicious activities of attackers.

Additionally, internal propagation is feasible as ransomware is copied to a shared folder with write permissions before execution; thus, if no separate access rights are granted, it is possible to block internal propagation. Therefore, one strategy involves minimally granting access permissions to network shared folders. Furthermore, if network services such as network shared folders are unnecessary, it is imperative to either disable the service entirely or block the SMB port (445) to minimize damage.

To prevent unauthorized restoration of encrypted files, the ransomware deletes all system backup copies before starting encryption. Enabling ASR rules can block both the deletion of backups and the encryption process. In addition, backups should be stored on separate networks or storage systems to ensure recovery even if the system is compromised."

[12] ASR (Attack Surface Reduction): protective feature that blocks specific processes used by attackers and executable processes.

IoCs

Hash(SHA-256)
86d812544f8e250f1b52a4372aaab87565928d364471d115d669a8cc7ec50e17
99959c5141f62d4fbb60efdc05260b6e956651963d29c36845f435815062fd98

■ Reference

• BleepingComputer (https://www.bleepingcomputer.com/news/security/microsoft-north-korean-hackers-now-deploying-qilin-ransomware/)

• Cyber Daily (https://www.cyberdaily.au/security/11919-exclusive-contractor-brighton-australia-listed-on-safepay-s-ransomware-leak-site)

• S-rm (https://www.s-rminform.com/latest-thinking/camera-off-akira-deploys-ransomware-via-webcam)

• BleepingComputer (https://www.bleepingcomputer.com/news/security/ransomware-gang-encrypted-network-from-a-webcam-to-bypass-edr/)

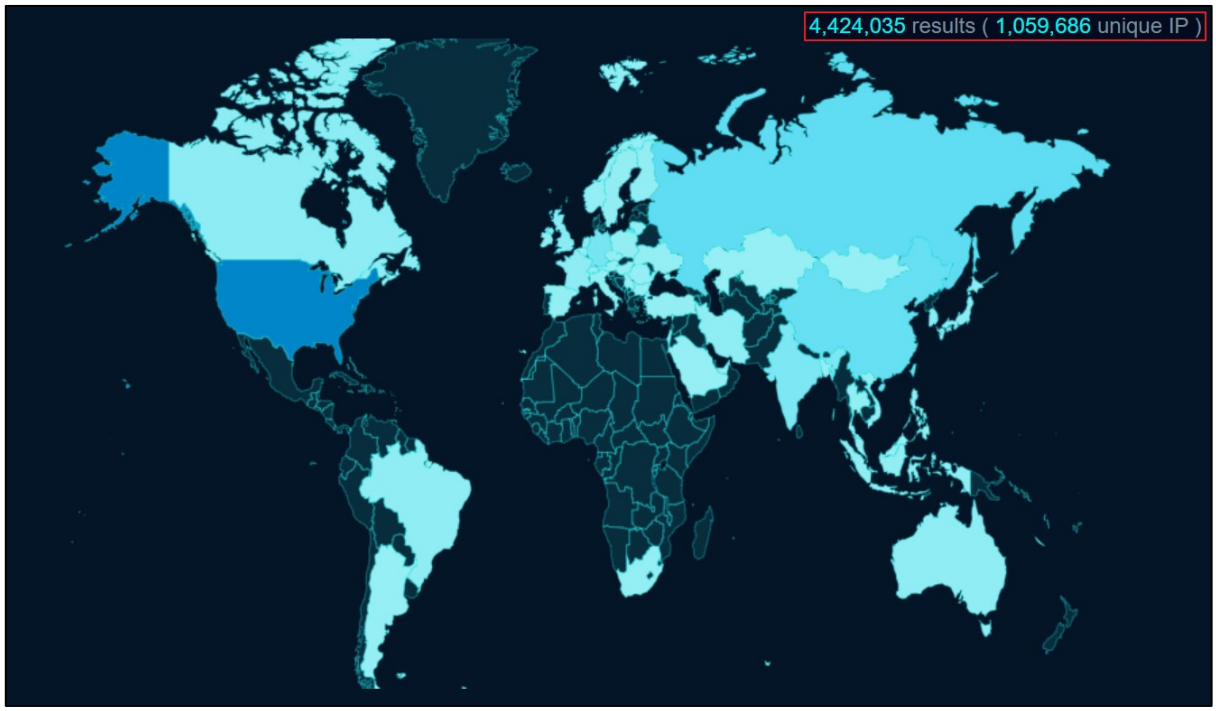
• Trend Micro (https://www.trendmicro.com/en_us/research/25/c/socgholishs-intrusion-techniques-facilitate-distribution-of-rans.html)

Research & Technique

Bypass Vulnerability in Next.js Middleware (CVE-2025-29927)

■ Introduction

Next.js is an open-source web framework based on Node.js that supports Server-Side Rendering (SSR)[13] and Static Site Generation (SSG)[14]. The globally popular JavaScript library React mentions Next.js as a recommended toolchain in its official documentation. An OSINT-based analysis of publicly available data showed that, as of April 15, 2025, Next.js powers approximately 4.42 million websites worldwide, including sites in the United States, Russia, and Germany.



Source: fofa.info

Figure 1. Usage Statistics of Next.js

[13] SSR (Server-Side Rendering): A web communication method in which the server generates all data and sends it to the client, and the client interprets that data to render the website.

[14] SSG (Static Site Generation): A method in which page HTML is generated at build time and reused for each request.

On March 21, 2025, a vulnerability in Next.js middleware[15] bypass, identified as CVE-2025-29927, was disclosed. This vulnerability arises from the exploitation of Next.js's internal logic, which checks whether the middleware has already been executed using specific header values. By manipulating these header values, attackers are capable of circumventing the middleware, and if the authentication process is implemented through middleware, this bypass enables access to restricted pages. Given the extensive utilization of Next.js as a web framework, it is imperative to conduct a thorough examination to determine if one's systems are susceptible to this vulnerability.

[15] middleware: concept in the request–response cycle that processes incoming requests before they reach their destination and processes responses before, they are sent.

■ Attack Scenario

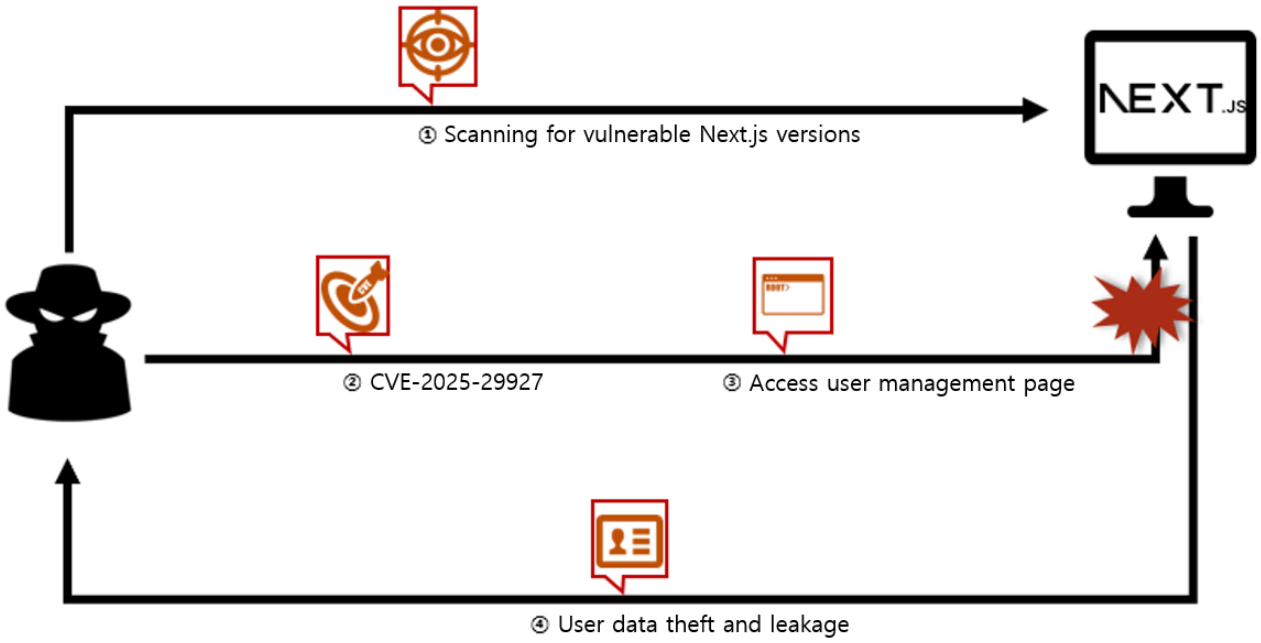


Figure 2. Attack Scenario for CVE-2025-29927

①	The attacker scans servers running vulnerable versions of Next.js.
②	Exploiting CVE-2025-29927, the attacker bypasses authentication.
③	Bypassing authentication, the attacker accesses the victim's user management page.
④	From there, the attacker harvests and exfiltrates large volumes of personal data.

■ Affected Software Versions

The software versions vulnerable to CVE-2025-29927 are as follows.

Software Component	Vulnerable Version
Next.js	Version prior to v15.2.3 Version prior to v14.2.25 Version prior to v13.5.9 Version prior to v12.3.5 All v11 Version

■ Configuration Information of the Test Environment

A test environment was established to scrutinize the operational process of CVE-2025-29927.

Identifier	Details
Victim	Next.js v15.1.7 (192.168.0.3)
Attacker	Kali Linux (192.168.216.133)

■ Vulnerability Assessment

Step 1. Configuration of the Environment

The victim's PC is configured with the Next.js v15.1.7 environment, and a test environment is established to replicate the vulnerability. Detailed configuration methods can be accessed in the GitHub repository provided below.

• URL: https://github.com/EQSTLab/CVE-2025-29927

```
# Clone a GitHub repository
git clone https://github.com/EQSTLab/CVE-2025-29927

# Move to the directory, then build and run the Docker image
cd next15
docker build -t nextjs .
docker run -p 3000:3000 --rm -it nextjs
```

Upon accessing the victim's address via a web browser, one verifies whether the Next.js server is operating correctly.

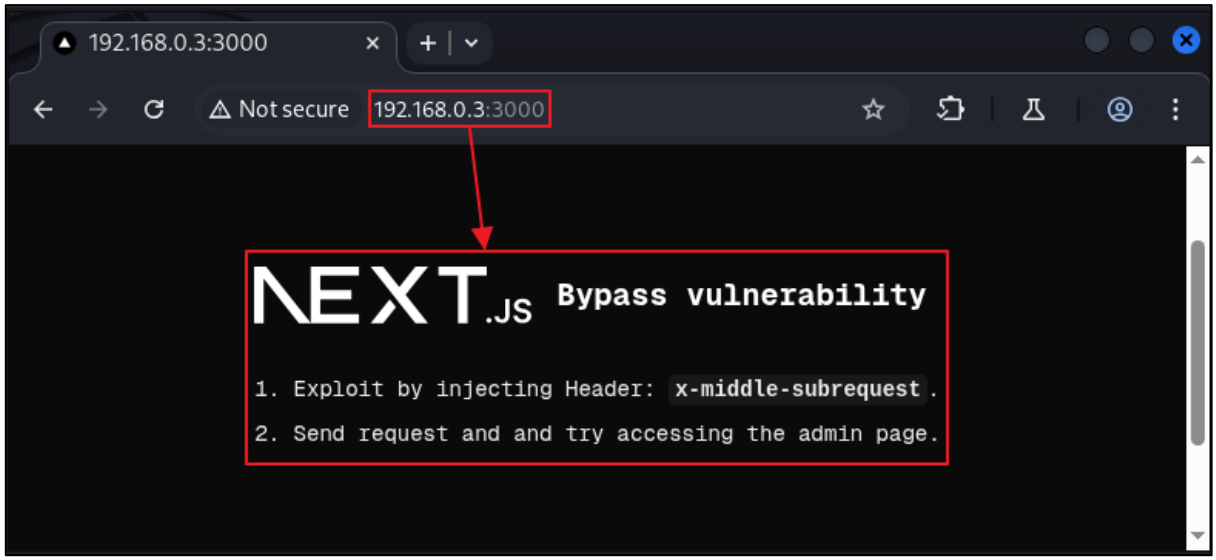


Figure 3. Verification of a Vulnerable Next.js Environment Setup

Step 2. Vulnerability Testing

The Next.js server is configured to block unauthorized access to the /admin endpoint.

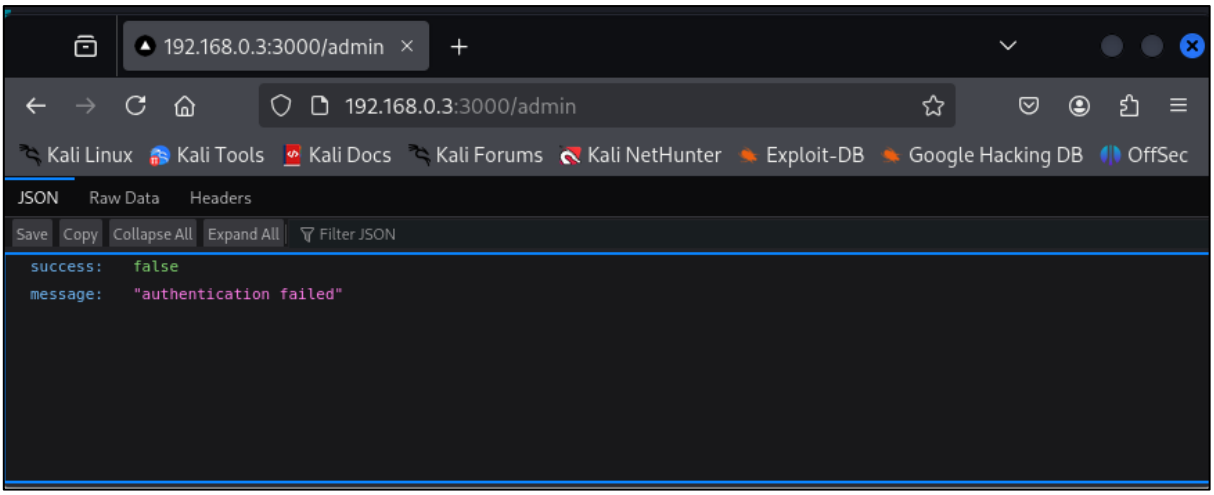


Figure 4. Blocking Access to the /admin Endpoint

However, the addition of the following headers to the request circumvents the middleware authentication process.

```
x-middleware-subrequest: middleware: middleware: middleware: middleware: middleware
```

When a request containing this header is transmitted, the middleware is treated as having already been executed, thereby enabling access to the /admin endpoint.

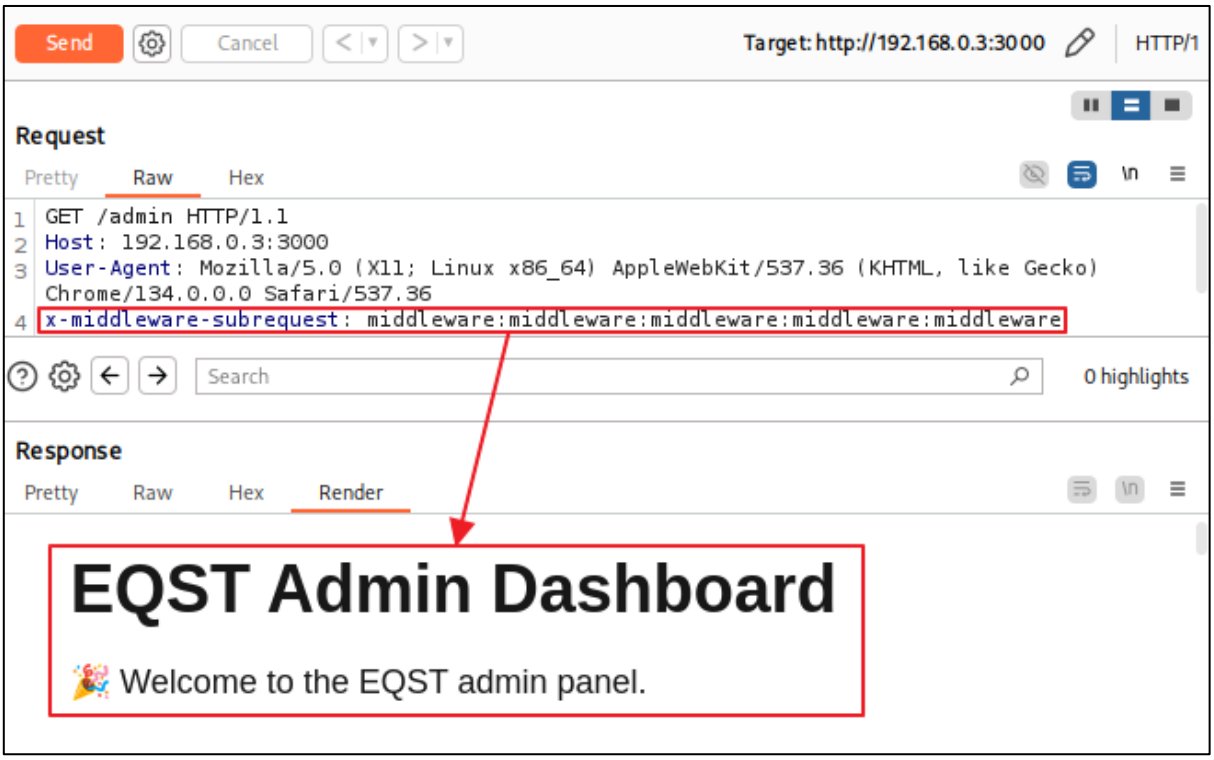


Figure 5. Verification of Middleware Authentication Bypass

■ Detailed Analysis of Vulnerabilities

In the detailed analysis of vulnerabilities, the report methodically elucidates the CVE-2025-29927 vulnerability, from its genesis to the process of authentication circumvention. Step 1 introduces the concept of middleware in Next.js, along with the characteristics unique to each version. Step 2 describes the vulnerabilities observed in the middleware implementations of each version and explains the techniques employed to circumvent authentication by exploiting these vulnerabilities.

Step 1. Implementation of Next.js Middleware

1) Characteristics of Next.js Middleware

Middleware acts as an intermediary layer that processes client requests before they reach the application within the request-response cycle, and is capable of performing additional processing prior to the transmission of responses. In Next.js, this functionality enables the implementation of pre-request validation, header rewriting, and redirections, and is defined through either a middleware.ts or middleware.js file. The middleware file may be positioned alongside the app and pages folders in the top-level directory of the project, or it can be included within the src directory.

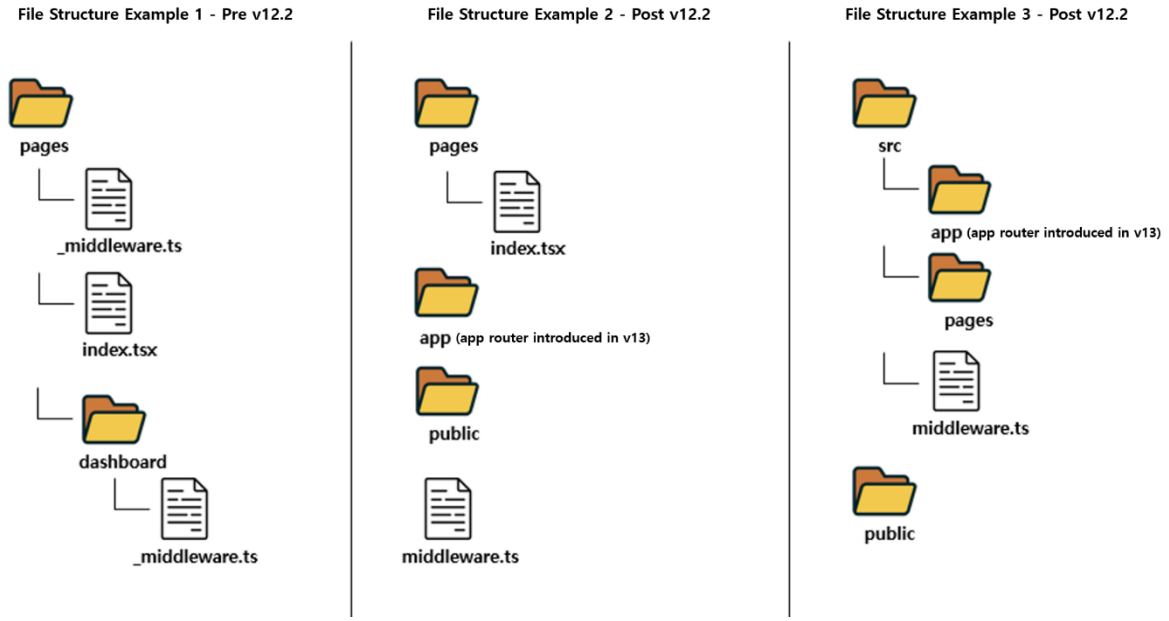


Figure 6. Example of Middleware Placement by File Structure

2) Characteristics of Middleware by Next.js Version

Since version 12.2, Next.js has upgraded and patched its middleware, resulting in significant changes to the manner in which middleware is utilized.

(1) Versions below 12.2

In versions prior to Next.js v12.2, it was possible to define middleware in any directory. In such instances, upon receiving a request, the server sequentially executes the middleware, commencing from the top-level directory and progressing to the subdirectories.

For instance, if middleware is defined in the /pages directory, it is applied to all requests for index.tsx, about.tsx, and teams.tsx within that directory.

```
- package.json
- /pages
    └ _middleware.ts       # Will run on all routes under /pages
    └ index.tsx
    └ about.tsx
    └ teams.tsx
```

Figure 7. The _middleware file within the /pages directory

Additionally, if middleware is defined in the directories /pages, /pages/about, and /pages/about/teams, upon receiving a request, the middleware is executed in a hierarchical order from the upper to the lower levels of the directory structure.

```
- package.json
- /pages
    └ _middleware.ts              # will run first
    └ index.tsx
    └ /about
        └ _middleware.ts          # will run second
        └ about.tsx
        └ /teams
            └ _middleware.ts      # will run third
            └ teams.tsx
```

Figure 8. Execution Sequence of Nested Middleware Files

(2) Versions subsequent to v12.2

From version 12.2 of Next.js, the use of underscores (_) in middleware filenames has been discontinued, and the filenames have been changed to either middleware.ts or middleware.js. Furthermore, the method of defining middleware in a nested manner within directories is no longer supported, and only a single middleware file can be placed in a structure parallel to the project root or the pages directory. As the application of middleware at the directory level becomes untenable, it is necessary to utilize matcher configurations to apply middleware to specific paths. For instance, to apply middleware to paths that begin with /admin, one must specify the corresponding pattern in the matcher as follows.

```
1   import { NextResponse } from 'next/server'
2   import type { NextRequest } from 'next/server'
3
4   export function middleware(request: NextRequest) {
5       return NextResponse.rewrite(new URL('/about-2',  request.url))
6   }
7
8   // Supports both a single string value or an array of matchers
9   export const config = {
10      matcher: ['/about/:path', '/dashboard/:path'],
11  }
```

Figure 9. Example Code for Matcher Configuration

Alternatively, conditional statements can be embedded within the middleware to bifurcate operations based on the request path. This approach proves particularly beneficial in scenarios requiring complex conditions that are challenging to manage solely through matcher configurations.

```
1   import { NextResponse } from 'next/server'
2   import type { NextRequest } from 'next/server'
3
4   export function middleware(request: NextRequest) {
5       if (request.nextUrl.pathname.startsWith('/about')) {
6           return NextResponse.rewrite(new URL('/about-2', request.url))
7       }
8
9       if (request.nextUrl.pathname.startsWith('/dashboard')) {
10          return NextResponse.rewrite(new URL('/dashboard/user', request.url))
11      }
12  }
```

Figure 10. Example Code for Conditional Branching

Step 2. Implementation of Authentication in Next.js Middleware and Methods to Circumvent by Version

1) Implementation of Authentication Using Middleware

Among the authentication implementation methods provided by Next.js, there is also an approach that utilizes middleware. For instance, the example below illustrates code that redirects to specific paths based on user permissions. Such a method proves particularly beneficial in scenarios requiring swift processing, such as concealing UI elements or controlling access based on permissions and roles.

```
1   import { NextRequest, NextResponse } from 'next/server'
2
3   // Protected and Public Routes
4   const protectedRoutes = ['/dashboard']
5   const publicRoutes = ['/login', '/signup', '/']
6
7   export default async function middleware(req: NextRequest) {
8     // Some authentication logic
9     return NextResponse.next()
10  }
11
12  // Routes Middleware should not run on
13  export const config = {
14    matcher: ['/(?!api|_next/static|_next/image|.*\\.png$).*'],
15  }
```

Figure 11. Example of Implementing Authentication in Next.js Using Middleware

After implementing authentication through middleware, it can be observed that attempts to access the /admin endpoint without authentication are subsequently blocked, as demonstrated below.

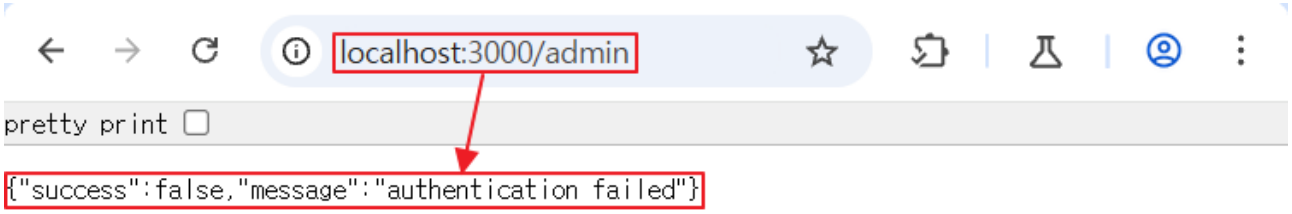


Figure 12. Verification Process of x-middleware-request

2) Versions below 12.2

Next.js internally utilizes the x-middleware-subrequest header to ascertain whether the middleware has already been executed for a particular request. This logic, as of version 12.0.1, can be verified within the /packages/next/server/next-server.ts file. The principal flow of the code is as follows.

```
630   const subreq = params.request.headers[`x-middleware-subrequest`]
631   const subrequests = typeof subreq === 'string' ? subreq.split(':') : []
632   const allHeaders = new Headers()
633   let result: FetchEventResult | null = null
634
...
650
651     if (subrequests.includes(middlewareInfo.name))
652       result = {
653         response: NextResponse.next(),
654         waitUntil: Promise.resolve(),
655       }
656       continue
657     }
```

Figure 13. x-middleware-request Verification Process

① The values transmitted via the x-middleware-subrequest header are segmented based on the colon (:) delimiter and are subsequently arrayed.
② Check whether the value of the middlewareInfo.name variable exists in the configured array.
③ If the specified value exists, the NextResponse.next() function is employed to bypass the middleware, thereby advancing to the subsequent processing stage to receive the response from the requested path.

At this juncture, the value of the middlewareInfo.name variable is loaded through the getMiddlewareInfo function located within the /packages/next/server/next-server.ts file. This function is tasked with retrieving middleware information corresponding to the requested path, and based on this information, it evaluates whether to execute the middleware by inspecting the x-middleware-subrequest header.

```
698       const middlewareInfo = getMiddlewareInfo({
699         dev: this.renderOpts.dev,
700         distDir: this.distDir,
701         page: middleware.page,
702         serverless: this._isLikeServerless,
703       })
```

Figure 14. Retrieving middlewareInfo through getMiddlewareInfo

The getMiddlewareInfo function is implemented in the /packages/next/server/require.ts file and returns information pertaining to the middleware corresponding to the specified path in the form of an object.

```
83  export function getMiddlewareInfo(params: {
84    dev?: boolean
85    distDir: string
86    page: string
87    serverless: boolean
88  }): { name: string; paths: string[] } {
89    const serverBuildPath = join(
90      params.distDir,
91      params.serverless && !params.dev ? SERVERLESS_DIRECTORY : SERVER_DIRECTORY
92    )
93
94    const middlewareManifest: MiddlewareManifest = require(join(
95      serverBuildPath,
96      MIDDLEWARE_MANIFEST
97    ))
...
```

Figure 15. Invocation of MIDDLEWARE_MANIFEST within getMiddlewareInfo

This logic references the .next/server/middleware-manifest.json file located within the .next directory generated subsequent to the build of Next.js, thereby importing the middleware information.

```
11  "middleware": {
12    "/": {
13      "env": [],
14      "wasm": [],
15      "files": [
16        "server/middleware-runtime.js",
17        "server/pages/_middleware.js"
18      ],
19      "name": "pages/_middleware",
20      "page": "/",
21      "regexp": "^/(?!_next).*$"
22    }
23  },
24  "version": 1
25  }
```

Figure 16. Information on middleware name within middleware-manifest.json

The "name" key in the middleware-manifest.json file denotes the location path of each middleware. Consequently, the middlewareInfo.name value ultimately becomes pages/_middleware. This is consistently verified when checked through debugging tools during execution.

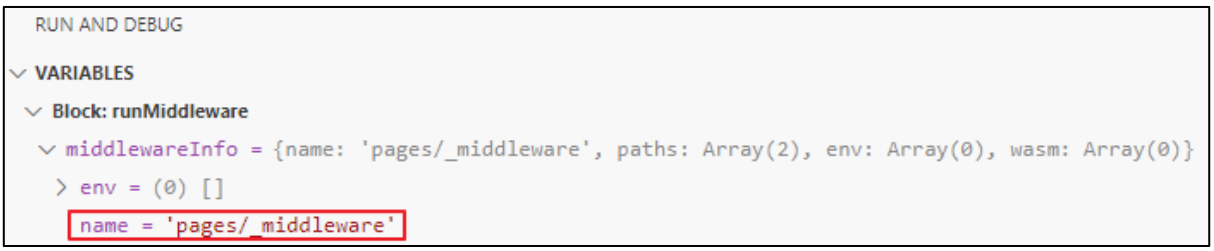


Figure 17. The Value of middlewareInfo.name as Verified through the Debugger

Thus, when the x-middleware-subrequest header value is set to pages/_middleware, the request is treated as having already passed through the middleware. Consequently, the middleware does not execute, and the authentication procedures implemented within can be bypassed, resulting in a vulnerability.

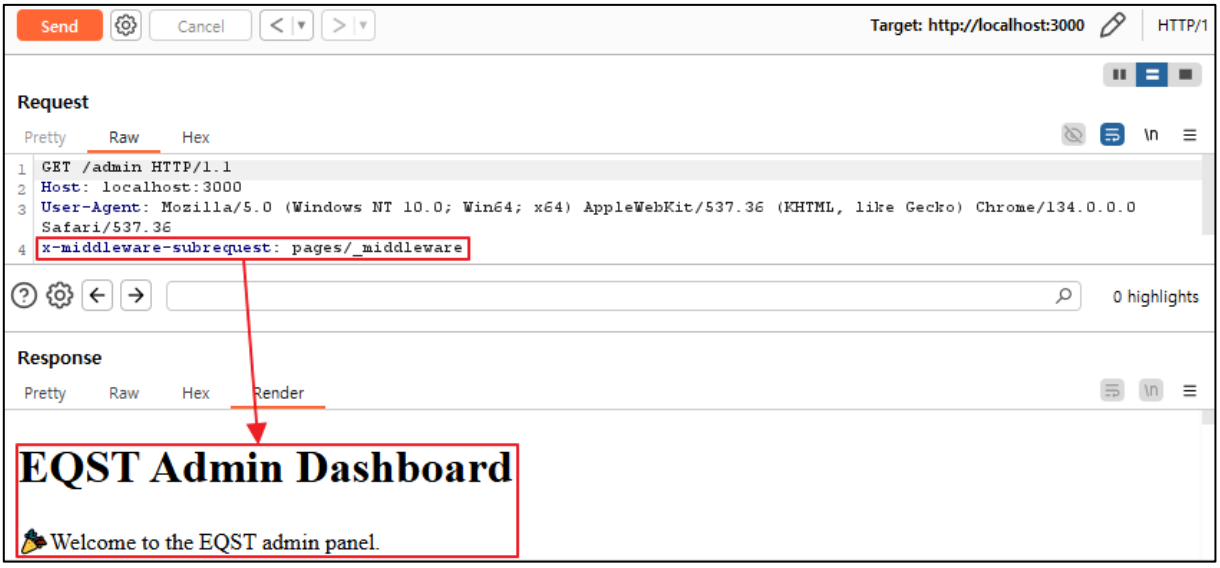


Figure 18. Verification of Middleware Authentication Process Bypass

As elucidated in Step 1, versions prior to v12.2 permit the individual definition of middleware for each directory. In such instances, by entering the path of each middleware defined under the 'pages' directory into the x-middleware-subrequest header value, it becomes feasible to circumvent the authentication process. For instance, as demonstrated in Figure 8, specifying either pages/about/_middleware or pages/about/teams/_middleware in the header enables the bypassing of the respective middleware.

3) Versions above 12.2 and below 14.2

The logic pertaining to x-middleware-subrequest has remained unaltered; consequently, as was the case previously, setting the middleware path in this header still facilitates the bypassing of authentication. From version 12.2 onwards, the naming convention for middleware files has been modified to either middleware.ts or middleware.js, and their location has been standardized to a directory parallel to, rather than inside, the pages directory. Thus, the middleware path has transitioned from the former pages/_middleware to simply middleware.

Such modifications can be observed in the .next/server/middleware-manifest.json file, through which it is discernible that the path of the middleware has been altered from pages/_middleware to middleware.

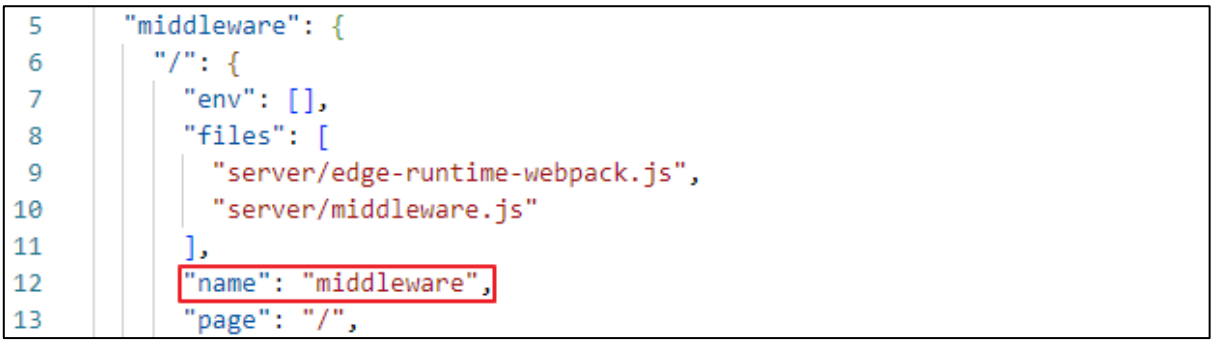


Figure 19. Information on middleware name within middleware-manifest.json

Similarly to the case of Next.js v12.2 mentioned above, by inspecting the values in execution through the debugger, it can be ascertained that the value in question is indeed middleware.

Figure 20. The Value of middlewareInfo.name as Verified through the Debugger

Similarly, by setting the value of the x-middleware-subrequest header to middleware, it is possible to circumvent the authentication process.

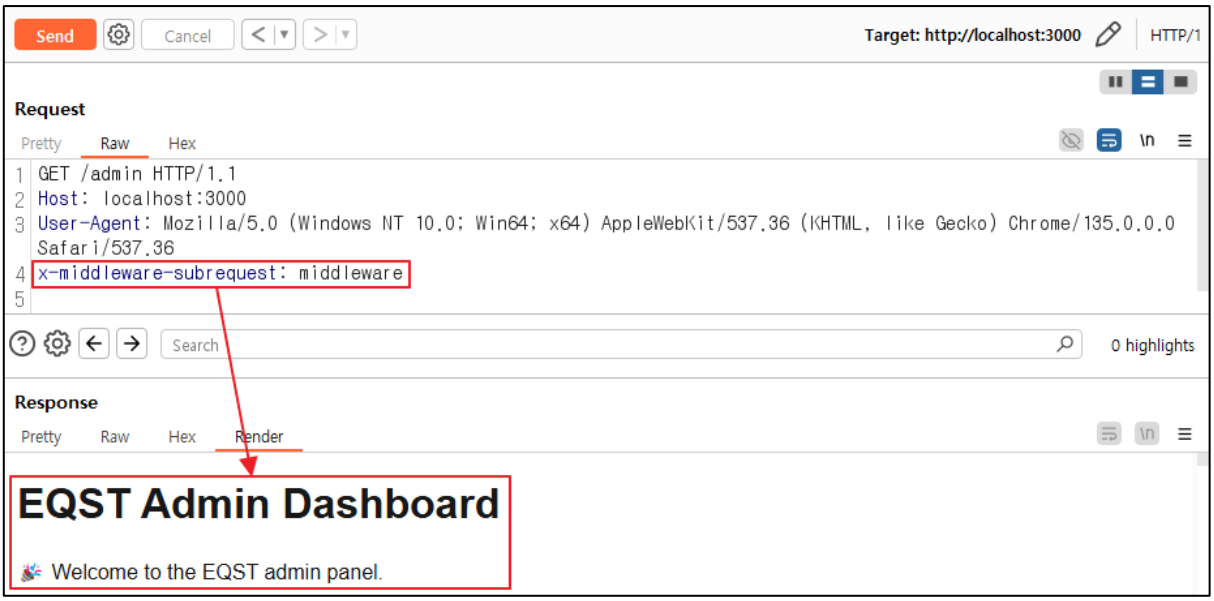


Figure 21. Verification of Bypassing the Middleware Authentication Process

4) Versions subsequent to v14.2

From version 14.2 of Next.js, the code has been modified such that if the value in the x-middleware-subrequest header of the middleware includes a middleware path that repeats beyond a certain number of times based on a colon (:), the request is engineered to bypass the middleware.
This can be verified through the code in /packages/next/src/server/web/sandbox/sandbox.ts.

```
96  const subreq = params.request.headers[`x-middleware-subrequest`]
97  const subrequests = typeof subreq === 'string' ? subreq.split(':') : []
98
99  const MAX_RECURSION_DEPTH = 5
100 const depth = subrequests.reduce(
101   (acc, curr) => (curr === params.name ? acc + 1 : acc),
102   0
103 )
104
105 if (depth >= MAX_RECURSION_DEPTH) {
106   return {
107     waitUntil: Promise.resolve(),
108     response: new runtime.context.Response(null, {
109       headers: {
110         'x-middleware-next': '1',
111       },
112     }),
113   }
114 }
```

Figure 22. Modified x-middleware-request Verification Process

① The values transmitted via the x-middleware-subrequest header are segmented based on the colon (:) delimiter and subsequently structured into an array.
② The configured array is scrutinized to ascertain the number of occurrences of the value of the variable params.name.
③ If the value of the variable params.name is repeated more than five times, the middleware is bypassed, and the response from the requested path is received.

In this context, params.name corresponds to the previously mentioned middleware.name, a congruence verifiable through the .next/server/middleware-manifest.json file and subsequent debugging.

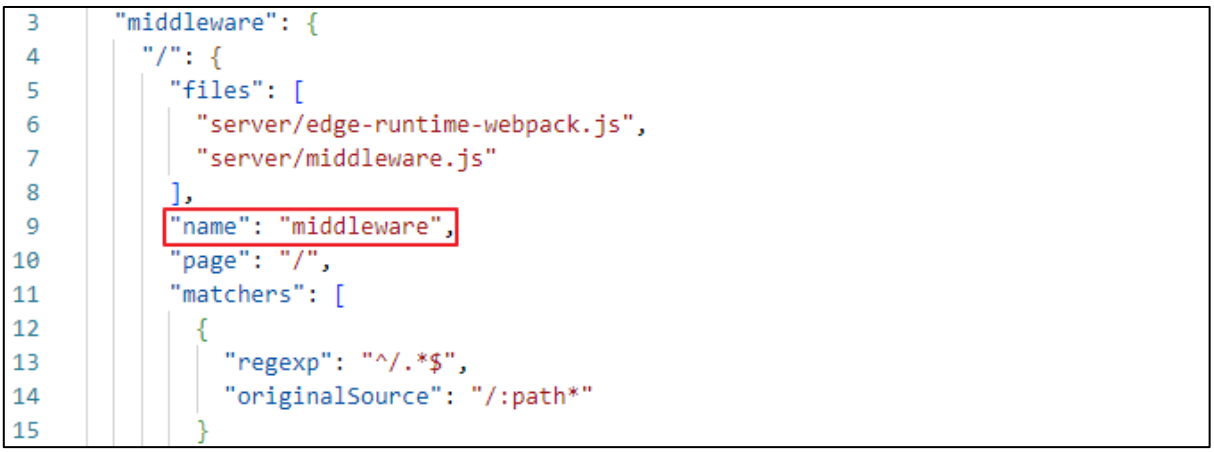


Figure 23. Information on middleware name within middleware-manifest.json

Figure 24. The Value of params.name as Verified Through the Debugger

Consequently, subsequent to version 14.2, it is requisite that the middleware path be reiterated no fewer than five times, delineated by colons (:), as in middleware:middleware:middleware:middleware:middleware. By entering such a value into the x-middleware-subrequest header, one can circumvent the middleware verification process.

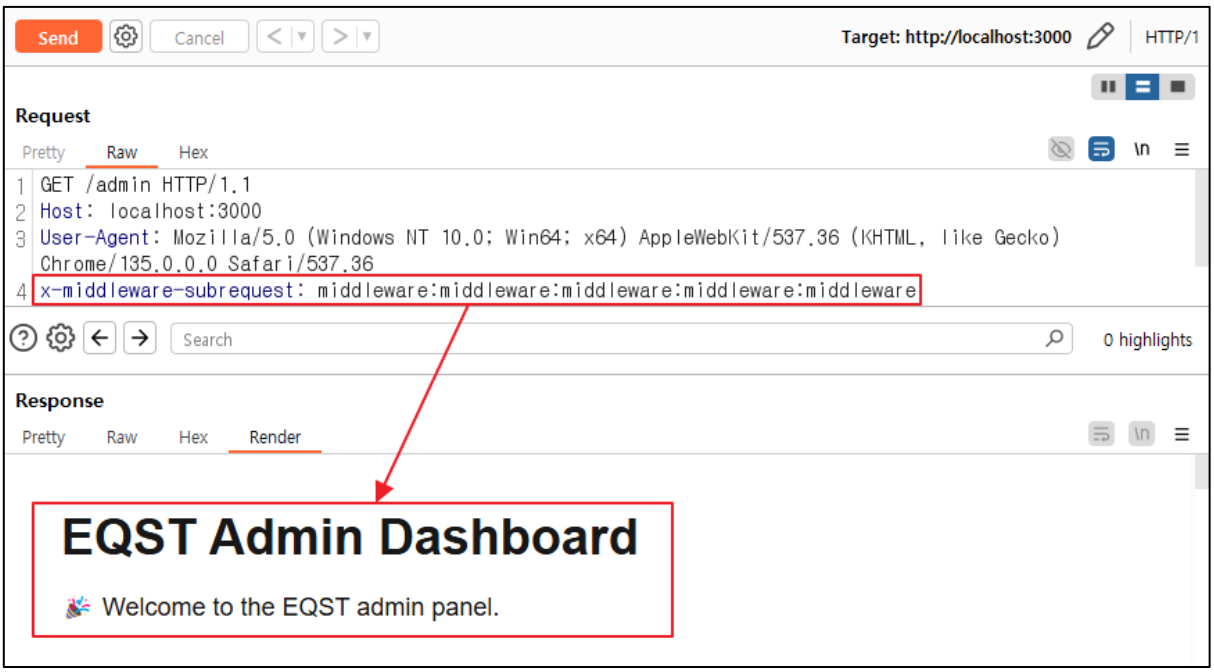


Figure 25. Verification of Bypassing the Middleware Authentication Process

5) Version-specific Middleware Evasion Payloads

As illustrated in Figure 6, the middleware may be situated within the src directory. Consequently, the middleware bypass payload that can be entered via the x-middleware-subrequest header value varies depending on the version as follows.

Version	Bypass Payload Example
version < v12.2	pages/[SubDirectory]/_middleware or ages/[subdirectory]/_middleware
v12.2 ≤ version < v14.2	middleware or src/middleware
version ≥ v14.2	middleware:middleware:middleware:middleware:middleware or src/middleware:src/middleware:src/middleware:src/middleware:src/middleware

■ Response Measures

The vulnerability CVE-2025-29927, which pertains to the bypassing of authentication in Next.js middleware, was identified by researchers zhero; and inzo_. It was initially reported on February 27, 2025.

•URL:https://zhero-web-sec.github.io/research-and-things/nextjs-and-the-corrupt-middleware#section-7

The vulnerability in question was patched on March 18, 2025, and the corresponding security advisory was made public on March 21, 2025.

•URL: https://github.com/vercel/next.js/security/advisories/GHSA-f82v-jwr5-mffw

Upon examining the patch notes for version 15.2.3, one can ascertain the specific security measures that have been implemented.

•URL: https://github.com/vercel/next.js/compare/v15.2.2...v15.2.3

Initially, in packages/next/src/server/lib/router-server.ts, the function crypto. getRandomValues is utilized to generate an 8-byte random value. Subsequently, this value is stored as a global variable[16] for the @next/middleware-subrequest-id.

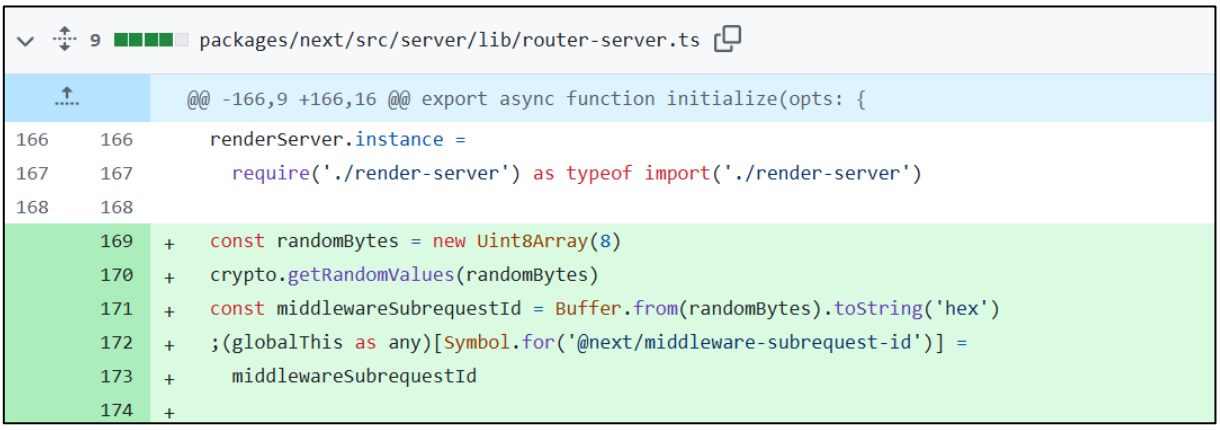


Figure 26. Modifications to packages/next/src/server/lib/router-server.ts

[16] global variable: A variable declared outside of a function that can be accessed from anywhere in the program.

Subsequently, the verification process implemented in packages/next/src/server/lib/server-ipc/utils.ts ascertains the presence of the x-middleware-subrequest and x-middleware-subrequest-id headers. It further scrutinizes whether the value of the x-middleware-subrequest-id header corresponds with the @next/middleware-subrequest-id stored previously in a global variable. Should there be a discrepancy in the values, the x-middleware-subrequest header is expunged to preclude its utilization.

11 packages/next/src/server/lib/server-ipc/utils.ts

```
@@ -57,5 +57,16 @@ export const filterInternalHeaders = (
57  57        if (INTERNAL_HEADERS.includes(header)) {
58  58          delete headers[header]
59  59        }
    60  +
    61  +     // If this request didn't origin from this session we filter
    62  +     // out the "x-middleware-subrequest" header so we don't skip
    63  +     // middleware incorrectly
    64  +     if (
    65  +       header === 'x-middleware-subrequest' &&
    66  +       headers['x-middleware-subrequest-id'] !==
    67  +         (globalThis as any)[Symbol.for('@next/middleware-subrequest-id')]
    68  +     ) {
    69  +       delete headers['x-middleware-subrequest']
    70  +     }
```

Figure 27. Modifications to packages/next/src/server/lib/server-ipc/utils.ts

Users are incapable of predicting the 8-byte random values stored internally; consequently, the server side can securely utilize the x-middleware-subrequest header to ascertain whether execution within the middleware is taking place.

The determination of whether vulnerable versions are in use can be ascertained by examining the package.json file within the source code. Versions below 15.2.3, 14.2.25, 13.5.9, 12.3.5, and all 11.x versions are susceptible to vulnerabilities. Consequently, it is imperative to apply patches to these versions if they are in use, to prevent the exploitation of x-middleware-subrequest by malicious actors.

```
{
  "name": "case4",
  "version": "0.1.0",
  "private": true,
  ▷Debug
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  },
  "dependencies": {
    "react": "^19.0.0",
    "react-dom": "^19.0.0",
    "next": "15.2.2"
  },
  "devDependencies": {
    "typescript": "^5",
    "@types/node": "^20",
    "@types/react": "^19",
    "@types/react-dom": "^19",
    "postcss": "^8",
    "tailwindcss": "^3.4.1"
  }
}
```

Figure 28. Example of Using a Vulnerable Next.js Version

It should be noted that version 11.x is no longer supported, rendering patching unfeasible; thus, migration to the continuously supported version 15 is recommended. Should patching to a secure version prove challenging, it is advisable to block external user requests that include the x-middleware-subrequest header from reaching the Next.js application.

■ Reference Sites

• Wikipedia (Next.js) : https://en.wikipedia.org/wiki/Next.js
• Wikipedia (Static web page) : https://en.wikipedia.org/wiki/Static_web_page
• Wikipedia (Server-side rendering) : https://en.wikipedia.org/wiki/Server-side_scripting
• Project structure and organization (Next.js docs) : https://nextjs.org/docs/app/getting-started/project-structure
• middleware (Next.js docs) : https://nextjs.org/docs/app/building-your-application/routing/middleware
• Next.js and the corrupt middleware: the authorizing artifact (zhero_web_security) : https://zhero-web-sec.github.io/research-and-things/nextjs-and-the-corrupt-middleware
• Authorization Bypass in Next.js Middleware (GitHub Security Advisory) : https://github.com/vercel/next.js/security/advisories/GHSA-f82v-jwr5-mffw

EQST INSIGHT

2025.04

SK shieldus

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

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