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.

2026

01

SK shieldus

Contents

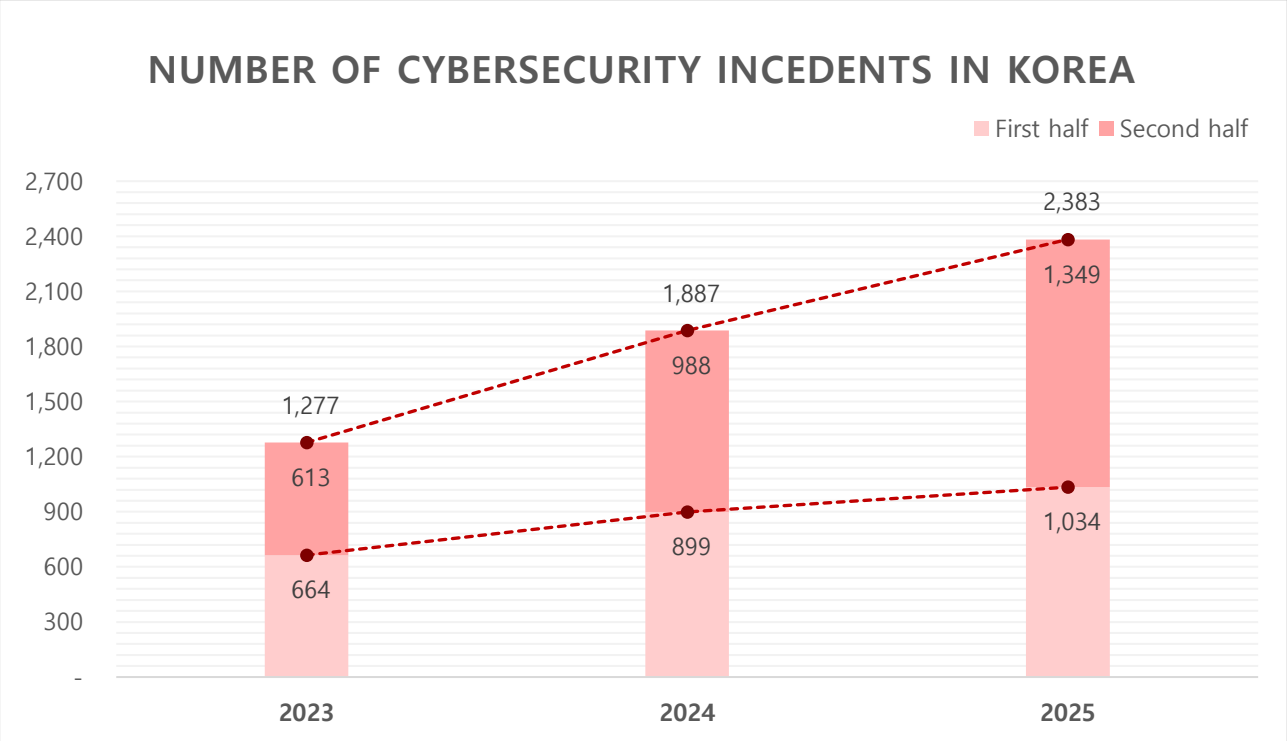
Headline

A Strategy for Proactive Security and Building a Red Team–Driven Cyber Immunity Framework

LEE KEON HEE / EQST, SK Shieldus

■ Two Paradigms for Cybersecurity: Proactive Security and Cyber Resilience

As digital transformation accelerates and AI technologies become increasingly advanced, cyberattacks are evolving into more sophisticated and intelligent forms; consequently, the frequency of security incidents is rising year on year. From 2023 to 2025, the number of domestically reported incidents increased to 1,277, 1,887, and 2,383 cases, respectively, and this trajectory is anticipated to persist in the years ahead.



Source: South Korea Ministry of Science and ICT; Korea Internet & Security Agency (KISA).

Figure 1. Status of Reported Security Incidents in the Republic of Korea (2023–2025)

To date, the focus of cybersecurity has largely been oriented toward detecting and responding to threats after they materialize, and considerable emphasis has been placed on the importance of cyber resilience—the capability to minimize stress when an incident occurs and to secure business continuity through rapid recovery.

However, efforts aimed solely at minimizing damage are not sufficient. In a security model that acts only after an incident has occurred, the more frequently breaches take place, the more inevitably the scale of impact and the cost of restoration increase in direct proportion. Moreover, the occurrence of an incident in itself erodes an organization's reputation, which ultimately drives customer churn and revenue decline, exerting long-term adverse effects.

More recently, preemptive cybersecurity, a prevention-centric strategy that seeks to minimize the likelihood of incidents by proactively identifying and remediating latent vulnerabilities and potential intrusion paths before attacks are realized, has emerged as a new paradigm. Gartner has identified preemptive security as one of the top ten strategic technology trends for 2026 and projects that, by 2030, spending on preemptive security will expand to approximately half of total security expenditure.

Although preemptive security and cyber resilience may appear conceptually opposed—given their respective offensive and defensive orientations—it is not appropriate to view them as wholly separable. An offensively oriented, preemptive approach can reduce threats, yet it is realistically difficult to eliminate them entirely; conversely, a defensively oriented, resilience-driven approach can minimize damage and help prevent recurrence, but it is inherently challenging to anticipate and prepare in advance for threats that fall outside expected scenarios.

Category	Preemptive Security	Cyber Resilience
Perspective	Attacker-centric perspective	Defender-centric perspective
Purpose	Threat prediction and preemptive blocking (prevention)	Rapid attack detection and recovery (detection and response)
Strengths	Capable of minimizing the likelihood of security incidents	Capable of minimizing damage resulting from security incidents
Limitations	Difficult to completely eliminate threats	Difficult to prepare for unforeseen threats

Table 1. Differences Between Preemptive Security and Cyber Resilience

It is now necessary to integrate these two paradigms in a mutually complementary manner. Through this approach, an organization can not only respond to cyberattacks with greater agility, but also establish a comprehensive cyber immunity posture by minimizing the likelihood of security incidents themselves.

■ A Security Function for Preemptive Cybersecurity: The Red Team

To date, the concept of a "Red Team" remains somewhat unfamiliar in the South Korea. Nevertheless, momentum to adopt red teams—particularly among large enterprises—has been spreading, and many media outlets have cited red-team adoption as a practical means of operationalizing preemptive security.

it is undeniable that a red team constitutes a core element of preemptive security. However, it must not be overlooked that, if an organization rushes to adopt a red team without understanding what it is, what activities it conducts, and how it should be operated, the decision to introduce a red team may inadvertently devolve into "security theater (Security Theater)"—conveying merely the impression that a preemptive security framework is in place.

In a single sentence, a red team can be defined as "a security function that, grounded in adversarial thinking proactively identifies threats that may arise across the organization's current attack surface and presents realistic directions for remediating vulnerabilities."

A red team performs security activities across the technical, administrative, and physical attack surface, including not only vulnerability assessment for compliance purposes, but also penetration testing, phishing campaigns, and physical security testing (Physical Security Test). Depending on the operating entity, red teams can be categorized into in-house red teams and outsourced red teams.

First, an in-house red team is a permanently operating internal team that possesses a deep understanding of the organization's internal systems and business processes. Composed of internal personnel, it can effectively carry out tasks that require a high degree of trust—such as security reviews of systems that are sensitive to disclose externally, or security activities that demand close coordination with other departments. That said, because such teams are more readily influenced by internal culture or decision-making structures, there is a possibility of bias, and they may face limitations in adopting a critical, outsider's perspective.

By contrast, an outsourced red team is composed of external experts and is operated on a long-term or short-term, project-based basis. Drawing on extensive execution experience across diverse clients and industries, as well as practical penetration strategies, it offers the advantage of being able to evaluate an organization's security posture objectively from an external viewpoint. However, because an outsourced red team is staffed by external personnel, the transfer of knowledge required to internalize insights gained during security activities may be incomplete, and it may also be less easy to trust compared to an in-house red team.

Category	In-house Red Team	Outsourced Red Team
Composition	Internal personnel	External experts
Strengths	Possess a high level of organizational understanding and trustworthiness	Possess practical penetration strategies and a creative perspective
Limitations	Prone to bias; Critical scrutiny may be difficult	Incomplete knowledge transfer; Relatively lower trustworthiness

Table 2. Differences Between In-house and Outsourced Red Teams

If an organization is considering adopting a red team, it is crucial to give due consideration to its security maturity. Without a well-structured security operations environment, it is difficult to maximize the benefits of introducing a red team; moreover, for organizations whose security capabilities have not matured sufficiently, red-team adoption may instead become an inefficient driver of security expenditure.

If the organization's security maturity is low, or if it is difficult to assess internally, it is advisable to strengthen attack surface management capabilities through an outsourced red team service first, and then introduce an in-house red team.

Conversely, for organizations with a high level of security maturity that already operate an in-house red team, outsourced red teams may be leveraged more proactively. As noted earlier, outsourced red teams consist of external experts and can compensate for the internal organization's knowledge limitations. On this basis, if outsourced red-team activities are used to derive threat scenarios that reflect the latest threat landscape and attack trends, while the in-house red team optimizes these scenarios to the organization's environment and operational characteristics, operating a hybrid red-team model could substantially improve the efficiency of red-team activities.

■ Composition and Roles of the Red Team

The red team can be structured, according to roles and responsibilities, into three core functions: a Leader, an Operator, and an Engineer.

The red team Leader oversees overall red-team activities and, rather than focusing on technical execution, assumes a managerial role centered on communication with key stakeholders and strategic decision-making. The Leader must foster the technical and cultural conditions required for red-team operations to proceed smoothly, and support effective communication between the red team and its stakeholders. Accordingly, strategic thinking, team management capability, a collaborative mindset, and strong communication skills are indispensable requirements for this role.

The red team Operator serves on the front line of red-team operations, conducting practical attack simulations under the Leader's direction. Operators must accurately understand real adversaries' TTPs (Tactics, Techniques, and Procedures) across the full lifecycle of a simulated attack, and be capable of executing attacks that are most appropriate for the organization's security posture. For this reason, the most critical competencies for Operators are deep offensive security expertise, as well as network and data analysis capabilities. In addition, because they must be able to explain or document results in a manner that stakeholders can readily understand, soft skills also constitute an important part of their required skill set.

A red team Engineer may, depending on team size, be a responsibility that Operators assume concurrently. Engineers are core development specialists who build offensive tooling that Operators can use in real engagements and design and implement adversary infrastructure, such as C2 (Command & Control) frameworks. Because they must repeatedly develop sophisticated intrusion tools capable of bypassing an organization's defensive controls, exceptionally strong software engineering capability and advanced offensive security knowledge are essential.

Role	Required Competencies
Red Team Leader	Strategic thinking, team management capability, a collaborative mindset, and communication skills
Red Team Operator	Offensive security knowledge, and network and data analysis capabilities
Red Team Engineer	Offensive security knowledge, and advanced software development capability

Table 3. Examples of Required Competencies by Red Team Role

■ Red Team Engagement Workflow

The red team engagement workflow can be delineated into three phases: Planning, Execution, and Reporting.

During the planning phase—the first step in initiating a red-team engagement—a pre-engagement meeting is conducted between the red team leader and relevant stakeholders to align on foundational elements required to formulate the engagement plan, including the objectives, definitions, and success criteria. In this process, all agreements are comprehensively captured in the Rules of Engagement (ROE), which function as an operational directive that both the red team and stakeholders are required to observe.

In NIST Special Publication 800-115, the National Institute of Standards and Technology specifies the items that should be included in an ROE template as follows.

Item	Content (Example)
Purpose	Purpose of the document; the organization subject to security testing; the executing organization; and the test objectives
Scope	Test scope and test type(s); exclusions; and deliverables
Assumptions and Limitations	Constraints and assumptions
Risks	Inherent risks and mitigation techniques
Document Structure	Structure of the Rules of Engagement (ROE) document
Personnel	A comprehensive list of all organizations and personnel associated with the security test
Test Schedule	Test schedule and milestones
Test Site	Authorized locations for testing and designated restricted areas
Test Equipment	Hardware, software, and unauthorized equipment to be used during the security test
General Communication	Communication plan (e.g., schedule, venue, frequency)
Incident Handling and	Incident response and recovery procedures
Target System/Network	Test targets (e.g., systems and network ranges)
Nontechnical Test	Non-technical testing activities (e.g., interviews, reviews)
Technical Test Components	Technical testing activities (e.g., network scanning, reconnaissance, penetration testing)
Data Handling	Methods and procedures for the collection, storage, transmission, and disposal of test data
Reporting	Reporting requirements and reporting cadence
Signature Page	Sign-off by stakeholders and senior leadership (CSO, CISO, CIO)

Source : (National Institute of Standards and Technology, NIST)

Table 4. Example ROE Items

Once the Rules of Engagement (ROE) has been fully drafted and formally signed, red team operations are conducted in strict accordance with its stipulations. At this stage, the red team operator carries out a simulated attack designed to accomplish the defined objectives by leveraging a broad spectrum of TTPs, including reconnaissance, initial access, execution, persistence establishment, and exfiltration. In parallel, the red team engineer collaborates closely with the operator throughout the engagement, providing technical support to enable more effective attainment of the target outcomes.

One of the most critical elements during execution is the maintenance of an operator log. The operator log constitutes a core deliverable that underpins the transparency and reproducibility of the penetration exercise and serves as an indispensable basis for subsequent analysis; it typically captures details such as timestamps, the responsible actor, event type, and observed outcomes.

If, during execution, the security posture is assessed to be sufficiently robust such that initial access is unlikely to be achieved, the engagement may be transitioned—based on the red team leader's judgment and the exception-handling procedures defined in the ROE—into an assumed breach scenario. Under this model, the team proceeds on the premise that specific organizational assets (e.g., internal servers or accounts) have already been compromised and then focuses primarily on evaluating post-exploitation activities.

Field	Value
Timestamp	20260118_121411
Source	10.10.10.22
Destination	192.168.1.12
Target	TARGET-LINUX01
Event Type	Active Scanning
Command	nmap -sT -Pn 192.168.1.12
Result	Ports: 80/open, 443/open, 8443/open

Table 5. Operator Log Example

Upon completion of all testing activities, the red team leader analyzes the operator's operational outcomes, consolidates identified improvement areas, and produces the final report. The report should, as a rule, include an executive summary for senior leadership, a detailed account of identified vulnerabilities and successfully executed threat scenarios, and the results of the threat assessment. In addition, it must incorporate key performance indicators (KPIs) that enable the detection and response effectiveness to be expressed in quantitative terms.

Item	Content (Example)
ASR(Attack Success Rate)	Success rate based on the total number of attacks and the number of successful attacks
Detection Coverage	Detection rate based on the total number of attacks and the number of detected attacks
MTTC (Mean Time to Compromise)	Time elapsed from attack initiation to successful compromise of the objective
MTTD (Mean Time to Detect)	Time elapsed from initial compromise to detection of the intrusion
MTTR (Mean Time to Respond):	Time elapsed from detection to completion of response actions

Table 6. Example Key Performance Indicators (KPIs)

Once report authoring is complete, it is distributed to relevant stakeholders—such as executive leadership, operations teams, and the blue team responsible for detection and response—and an After Action Review (AAR) or debriefing session is convened to evaluate and reflect on the red team engagement. The organization then proceeds to implement corrective measures addressing the identified weaknesses; once remediation has been completed, the red team engagement is concluded by conducting remediation verification as a follow-on activity to revalidate whether the corrective actions have been properly implemented and are effective.

■ Integration of Red Team and Blue Team Through Breach and Attack Simulation

Red team engagements are, in many cases, executed as largely one-off offensive activities. However, if threat scenarios can be tested repeatedly, the blue team would be able to incrementally enhance both its detection capability and its response velocity against those scenarios.

Breach and Attack Simulation (BAS) is a red-team-aligned approach that overcomes this limitation by enabling threat scenarios—decomposed into discrete TTP units—to be repeatedly simulated. In practice, BAS proceeds by structuring scenarios for iteration and, where necessary, re-simulating them on demand.

Because BAS supports repetitive testing of threat scenarios, it can also be leveraged for post-incident validation. If an organization experiences a security incident, it will typically analyze the root cause and establish countermeasures to prevent recurrence. After those preventive controls are implemented, re-simulating the incident scenario via BAS can provide a basis for evaluating the practical effectiveness of the applied security measures.

BAS can be conducted manually—similar to remediation verification in conventional red team workflows—but adopting an automated BAS platform can fully automate the validation process and thereby maximize productivity. Most automated BAS platforms provide capabilities to integrate with a broad range of security systems, along with an intuitive interface that allows key metrics—such as detection success rate and response time—to be monitored in near real time. This, in turn, can materially support an organization's efforts to mature its SOC (Security Operations Center) and its MSSP (Managed Security Service Provider) / MDR (Managed Detection and Response) capabilities. Moreover, executive leadership can use these insights to gauge ROI (Return on Investment) and to inform strategic decision-making.

In this manner, BAS facilitates proactive security, strengthens cyber resilience, and enables the effective integration of red team and blue team functions. The red team first derives plausible threat scenarios and then structures them through BAS to establish a repeatable simulation environment; the blue team subsequently uses the resulting insights to refine detection and response mechanisms. By iterating this cycle, an organization can progressively establish a Purple Team operating model in which red team and blue team activities are cohesively integrated.

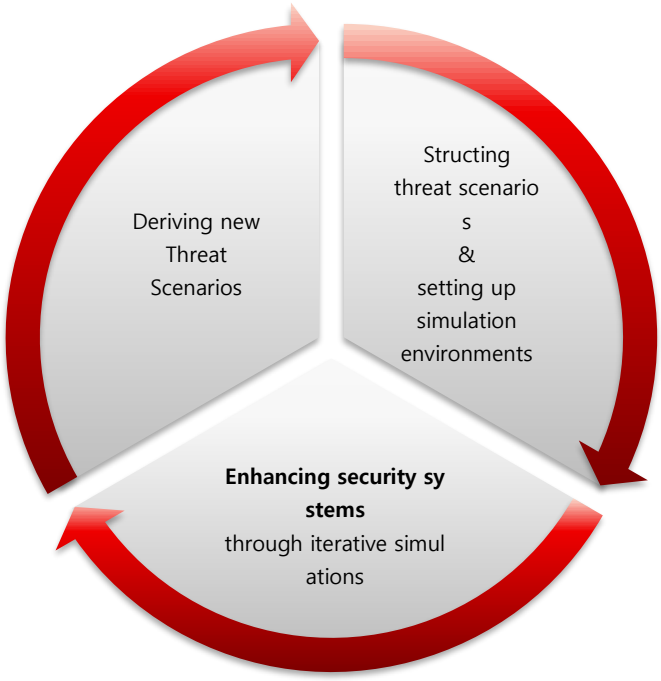


Figure 2. Continuous Attack Validation Procedure

Nevertheless, there are also practical challenges inherent in this model. Red teams tend to adopt an assertive and experimental posture in order to validate the realism of threat scenarios, whereas blue teams are primarily oriented toward operational stability and continuity—focusing on real-time security monitoring, detection, and incident response—and therefore often exhibit a more conservative risk appetite. As a result, blue teams may hesitate to share information or collaborate closely out of concern that red-team attack scenarios could adversely affect production environments; this structural divergence is a major factor that complicates effective integration between the two teams.

Addressing these challenges requires strong executive commitment and proactive leadership. Management must clearly communicate, across the organization, the guiding principle that "our objective is to identify and remediate security blind spots," while fostering a culture that actively supports red–blue collaboration. Although establishing a fully mature collaborative operating model may be difficult in the short term, it is advisable to realize it through a phased approach and sustained, iterative effort.

■ Conclusion

Many things that are commonly perceived as being in direct opposition, in practice, often complement one another and produce a stronger harmony. The relationship between wind and sail is a useful illustration: the wind may jolt a vessel and force it to change course, yet without a sail no ship can move forward at all. Even when the wind appears to be a threat, it becomes propulsion when paired with a sail.

In the same way, "proactive security," which anticipates and prepares for risks in advance, and "cyber resilience," which enables rapid recovery even after an attack, are not forces that repel one another. Only when they coexist do they become the driving power that can guide an organization in the right direction; and as they interlock in balance, an organization can establish a cyber immune system in which "prevention," "detection," and "response" are organically connected.

SK Shieldus provides professional penetration testing consulting services through EQST (Experts, Qualified Security Team), the largest white-hat hacker group in Korea, helping customers define the direction they should pursue to integrate "proactive security" and "cyber resilience."

Over the past two decades, EQST has delivered B2B engagements across diverse sectors—including public institutions, enterprises, finance, and manufacturing—and, drawing on both the latest cybersecurity trends analyzed by an internal research organization leading the New ICT domain and the extensive threat intelligence (Threat Intelligence, T.I.) of SK Shieldus's integrated monitoring platform, Secudium, has designed and implemented the EQST Threat Scenario Database (EQST Threat Scenario Database) as well as its own penetration testing methodology, in which expertise, trend awareness, and accumulated know-how are fused.

Building on this foundation, EQST offers a broad range of security consulting services—from vulnerability assessment consulting capable of supporting various compliance requirements, to security inspections that combine ASM (Attack Surface Management) with penetration testing, and further to hands-on outsourced red team services designed for real-world engagement. If the reader is considering a "proactive security strategy" to enhance the organization's cybersecurity maturity, optimized support for proactive security can be obtained through EQST's security consulting services, underpinned by distinctive and specialized expertise.

■ References

[1] Gartner, "Gartner Identifies the Top Strategic Technology Trends for 2026"

[2] Gartner, "Gartner Top 10 Strategic Technology Trends for 2026"

[3] Ministry of Science and ICT (MSIT), Republic of Korea; Korea Internet & Security Agency (KISA). "2025 Second-Half Cyber Threat Trends and 2026 Outlook."

[4] NIST, "Technical Guide to Information Security Testing and Assessment"

Keep up with Ransomware

Sinobi Ransomware: Analysis of Lynx Group Ties

■ OverView

In December 2025, ransomware victims counted 854, up about 15% from November (740 cases).

The French Ministry of the Interior stated that, in December, its internal email server was hit by a cyberattack, leading to unauthorized access to multiple email accounts and confidential documents. The investigation found that the attacker initially gained a foothold by compromising a police officer's email account and then obtaining passwords that had been shared in plaintext. Using the recovered credentials to penetrate the internal authentication system, indicators of access to police databases were also identified. Meanwhile, a post claiming responsibility for the intrusion appeared on the hacking forum BreachForums, published by an account believed to be an administrator under the name "Indra." The post referenced retaliation for French authorities' arrests of individuals linked to ShinyHunters and included content boasting of the Ministry compromise and large-scale data access. It further contained screenshots of internal system search results suggestive of access to approximately 16.4 million records, alongside images containing identifying information related to certain police records, as well as a message demanding that the French government enter negotiations within one week. the incident is reported to have resulted from both account compromise and a failure to adhere to fundamental security principles. It underscores the need to strengthen account management and authentication controls, enforce least-privilege access, and enhance anomaly-detection capabilities based on authentication and login logs.

A case was identified in which, rather than the attacker infiltrating systems directly, a cybersecurity incident-response professional allegedly colluded with the threat actor—an arrangement uncovered through law-enforcement investigations.

In December 2025, the U.S. Department of Justice indicted two Americans, Ryan Goldberg and Kevin Martin, for allegedly conspiring with the BlackCat (ALPHV) ransomware organization to target U.S.-based victims and extort funds. According to the investigation, Goldberg worked in incident response at a cybersecurity firm, while Martin served in a negotiation role at a company that supports ransomware incident response. Drawing on the training and experience acquired through their positions in the security industry, they allegedly encrypted corporate networks and demanded ransom payments. The investigation further found that they sustained the arrangement by pre-agreeing on profit-sharing and remitting a portion of victim-paid ransom to the operators.

On 5 December 2025, a Weaxor ransomware deployment was also observed exploiting the React2Shell vulnerability (CVE-2025-55182). The flaw enables remote code execution on vulnerable servers without authentication; the attacker then used PowerShell to establish a Cobalt Strike[1] based C&C server[2] before executing Weaxor to encrypt files. The operation targeted unpatched systems, and the fact that exploitation occurred on 5 December—only two days after public disclosure on 3 December—underscores the imperative for rapid vulnerability remediation across software and security appliances.

[1] Cobalt Strike: A command-and-control (C&C) framework leveraged by adversaries to execute remote commands, drop additional payloads, conduct internal reconnaissance, and facilitate lateral movement.

[2] C&C (Command & Control) server: A server that communicates with compromised systems to provide remote control, including issuing commands, deploying additional payloads, and collecting information.

■ Ransomware News

Indications of a Compromise of the French Ministry of the Interior's Email Server

- The French Ministry of the Interior's internal email server was attacked, enabling unauthorised access to multiple accounts and confidential documents.
- The attacker hijacked a police officer's email, extracted emailed passwords, and bypassed internal authentication.
- On BreachForums, an alleged admin claimed the breach and demanded negotiations.

U.S. Department of Justice Announces Guilty Pleas by Two Americans for Colluding with BlackCat

- The U.S. Department of Justice announced that two Americans Ryan Goldberg and Kevin Martin who colluded with a ransomware syndicate to extort funds have pleaded guilty.
- The two individuals admitted to encrypting U.S. companies' networks and demanding ransom.
- They allegedly agreed to split profits and pass part of the ransom to the operators.

Weaxor Ransomware Deployment via React2Shell Exploitation Confirmed

- Exploitation of the React2Shell vulnerability (CVE-2025-55182) enabled emote code execution on vulnerable servers, after which ransomware execution was observed.
- The attacker used PowerShell to set up a Cobalt Strike C2 channel, then ran ransomware to encrypt files.

Seven New Ransomware Groups Emerged in December

- All seven ransomware groups that emerged in December operated their own DLS
- Except for Osiris, MS13-089, and MintEye, no victims have been observed on their DLS to date.

Figure 1. Ransomware Trends

■ Ransomware Threats

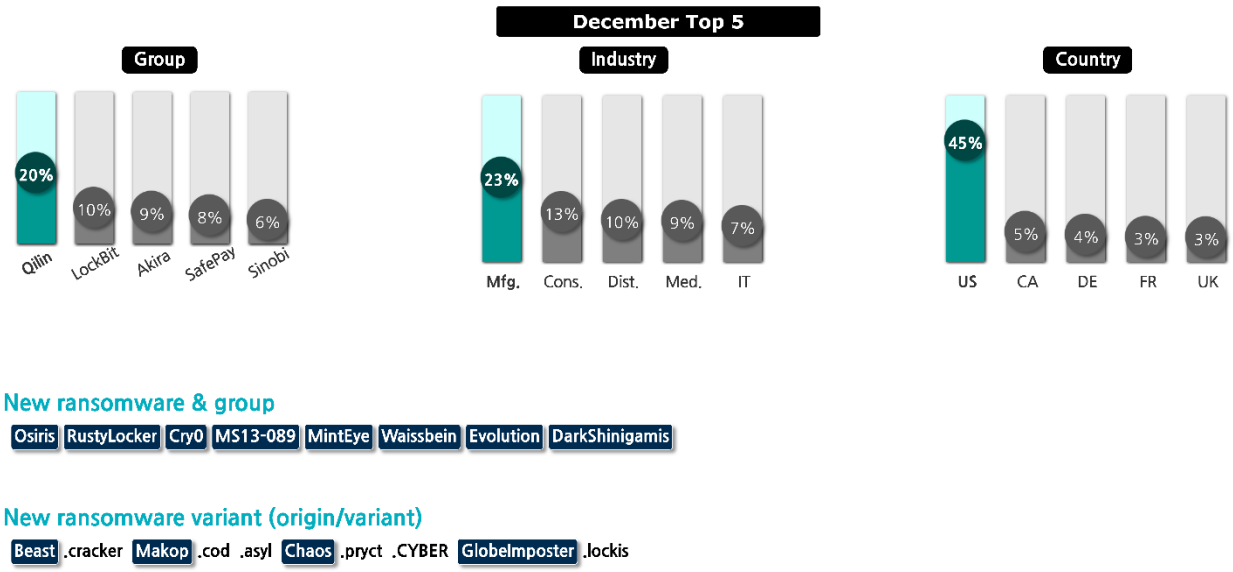


Figure 2. Ransomware Threat Overview for December 2025

Emerging Threats

A total of seven new ransomware groups emerged in December. Although all of them operate DLS, only a subset has posted victim entries to date. Osiris has published one victim post, MS13-089 two, and MintEye five on their respective DLS. The remaining groups have merely set up their DLS, with no victim postings observed thus far.

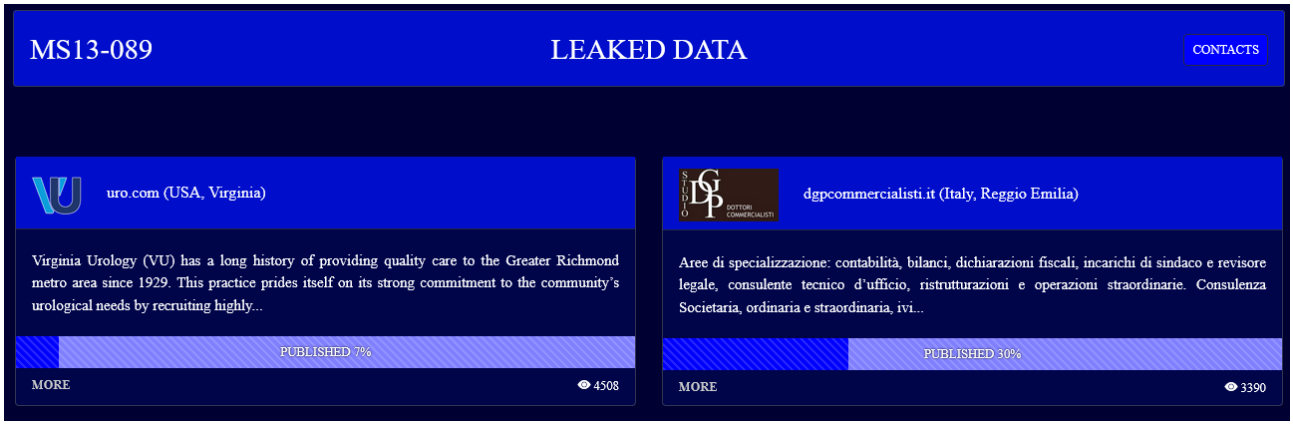


Figure 3. MS13-089's DLS

The MS13-089 group, which emerged in December 2025, has uploaded a total of two victim posts to date. In addition, it has been observed pressuring victims by explicitly displaying the data-release "progress" as a percentage (%) during disclosure.

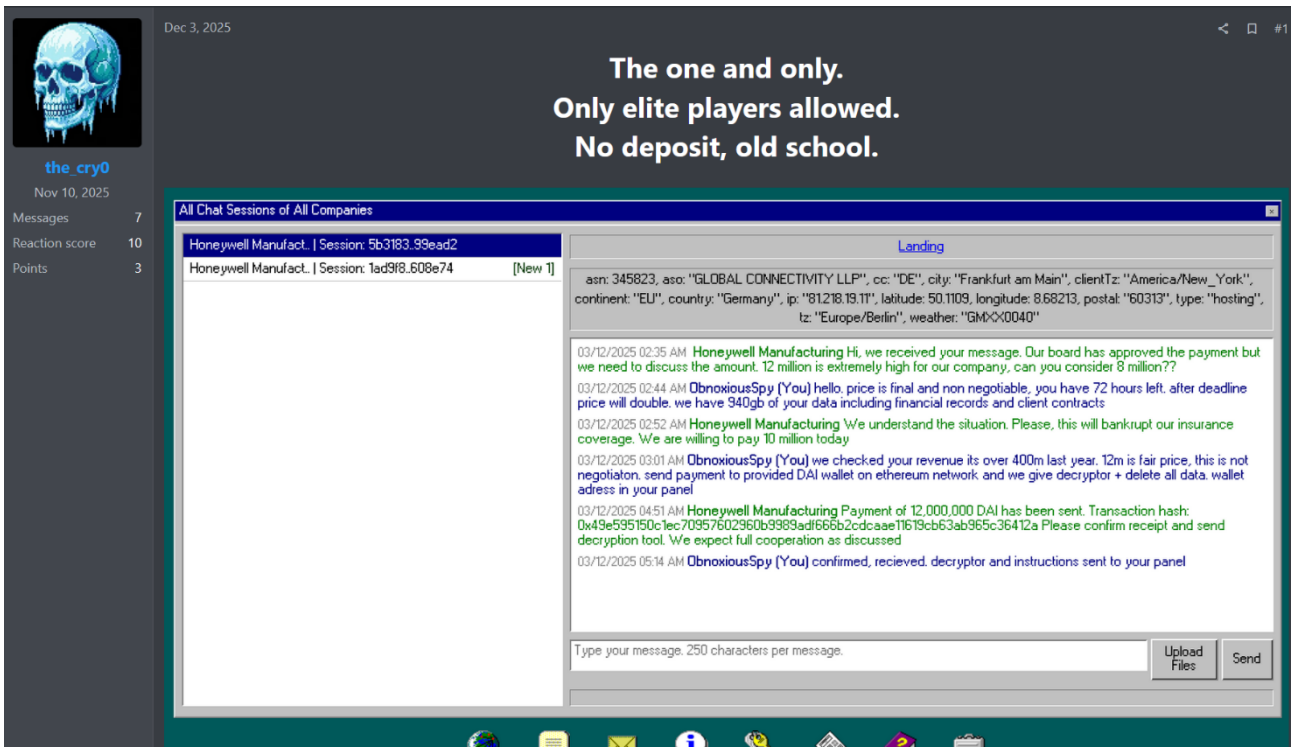


Figure 4. Cry0 Ransomware Group's RaaS [3]Promotional Post

The Cry0 group began recruiting members on RAMP, a Russian hacking forum, on 3 December 2025. It promoted its operational capabilities by posting screenshots of a chat panel used to manage per-victim negotiation sessions, along with purported negotiation logs; however, no confirmed victim listings have been observed on its DLS to date.

[3] RaaS (Ransomware-as-a-Service): A business model in which ransomware is offered as a service, enabling affiliates to readily deploy attacks without developing the malware themselves.

Top 5 Ransomware Groups

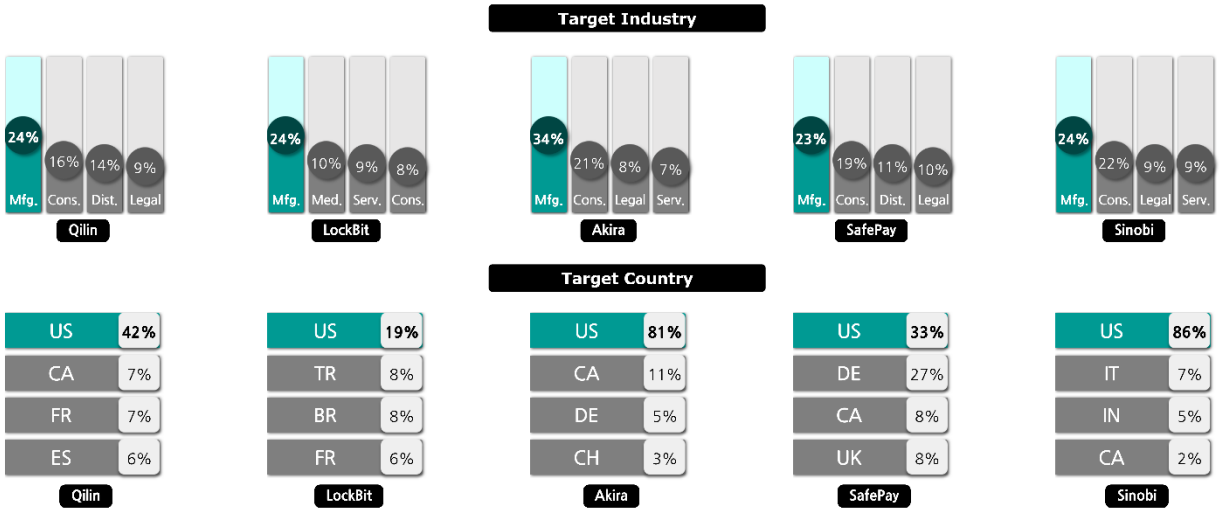


Figure 5. Major Ransomware Incidents by Industry and Country

In December, Qilin was the ransomware group responsible for the highest number of victims. Notably, on 29 December it targeted Goodwin University in Connecticut, United States, exfiltrated approximately 140 GB of data—including personal information on students and staff—and subsequently published it on its DLS.

LockBit announced a reorganization with the release of LockBit 5.0 in September 2025, yet no pronounced activity was observed for some time thereafter. From December onward, however, it intensified operations by posting victims in rapid succession. Notably, on 19 December it attacked Clarinda Regional Health Center in Iowa, United States, and signaled the imminent release of patients' and employees' personal information. On 22 December, it targeted Colégio Miguel de Cervantes in São Paulo, Brazil, and posted data including student grades and administrative documents.

Akira intensively exploited the SonicWall firewall access-control weakness (CVE-2024-40766) throughout 2025, with multiple cases confirmed to have used it as the primary intrusion vector. By leveraging this vulnerability, the attacker hijacked valid VPN sessions or obtained credentials, penetrated the internal network, and subsequently deployed ransomware while also exfiltrating data. Separately, on 24 December 2025, Akira targeted Agralite Electric Cooperative, an electric cooperative in Minnesota, United States, exfiltrating approximately 136 GB of data containing sensitive information such as Social Security numbers and tax documents. In addition, on 25 December it attacked Denmark-based architectural design firm Friis & Moltke Architects and threatened to leak roughly 12 GB of data, including design drawings and contracts, via its DLS.

On 29 December 2025, the SafePay group attacked Usdaw, a UK retail trade union, and exfiltrated data containing members' personal information and internal documents. On the same day, it also targeted Argentina-based healthcare company Investigaciones Médicas, claiming to have obtained sensitive information such as patient examination data and medical records.

Sinobi attacked Quality Companies, a U.S. energy services provider, on 7 December 2025, exfiltrated approximately 40 GB of data—including business documents and contract information—and subsequently published it on its DLS. It also targeted Homestead Electrical Contracting, a U.S. electrical design firm, and threatened to leak internal materials.

■ Ransomware Focus

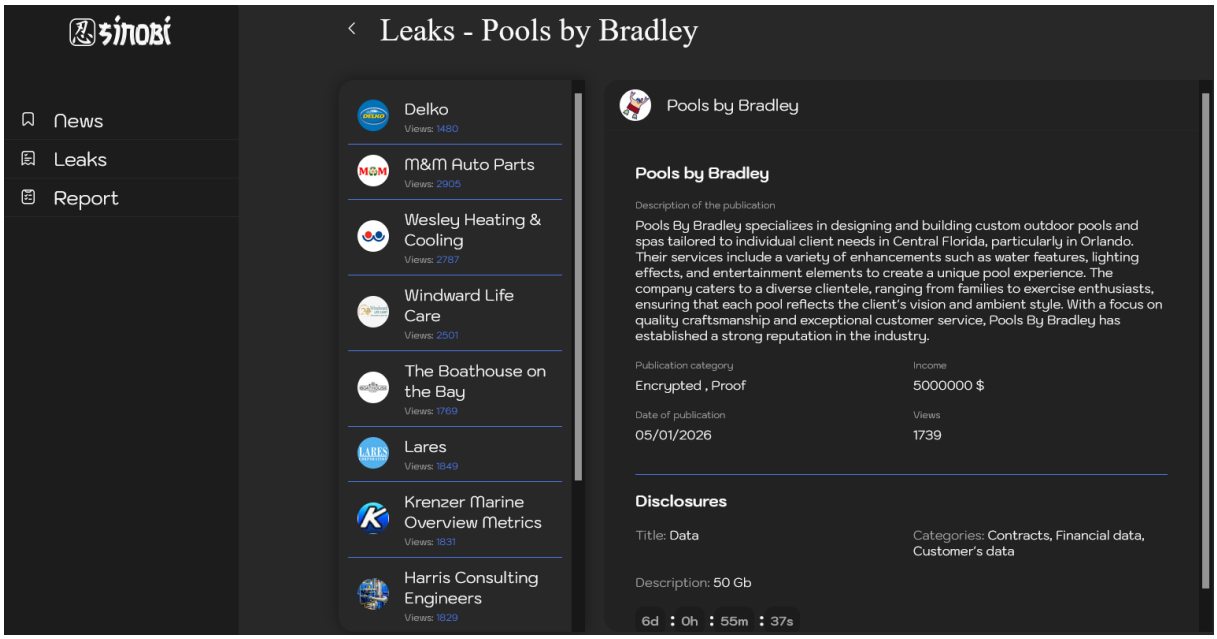


Figure 6. Sinobi Group's DLS

The Sinobi ransomware group was identified in July 2025. To date, it has disclosed 221 victim organizations on its DLS. Its leak posts present not only the victim's name and details and the posting date, but also the categories of exfiltrated material alongside sample data. It coerces victims through a double-extortion model that combines file encryption with the threat of data disclosure. Moreover, the group appears to tailor ransom demands on a per-victim basis, ranging from as low as USD 5 million to as high as USD 44 million, with an average demand of roughly USD 24 million.

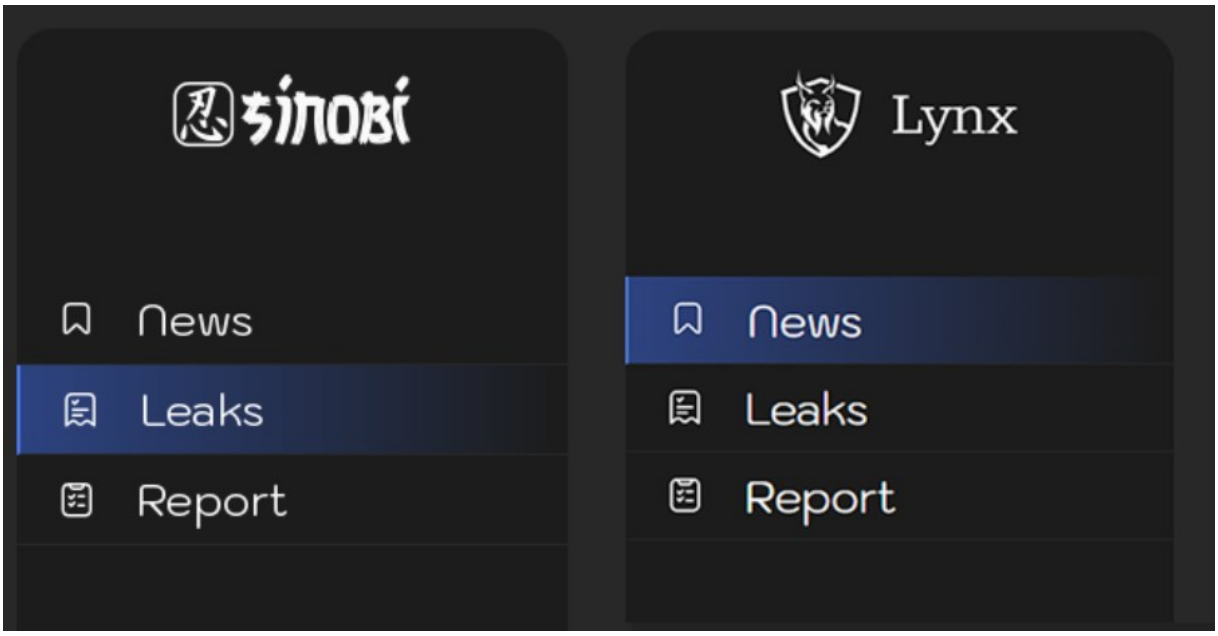


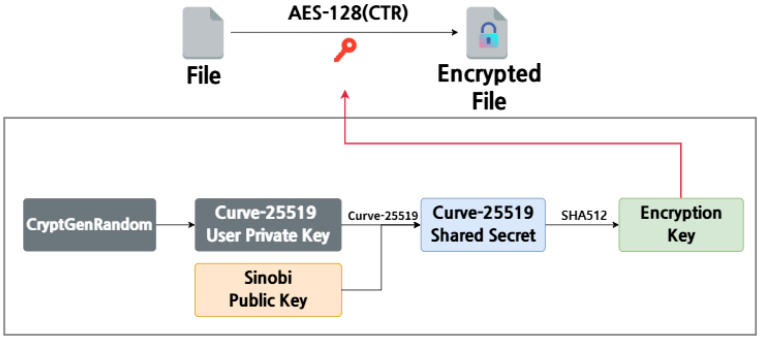
Figure 7. DLS Comparison (Left: Sinobi, Right: Lynx)

Moreover, Sinobi exhibits similarities to the Lynx lineage that emerged following indications of INC source-code trading. Lynx shows characteristics analogous to the code structure and functional flow observed in INC ransomware, while Sinobi likewise demonstrates similarities with Lynx in its encryption methodology and execution-parameter structure. In other words, INC, Lynx, and Sinobi appear to be related in terms of code and architectural design. Notably, the DLS of Lynx and Sinobi share comparable UI and menu layouts, raising the possibility of a common operator or a cooperative relationship. Accordingly, this report consolidates the indicators of linkage across INC–Lynx–Sinobi to anticipate emerging threats and shares a detailed analysis of the Sinobi ransomware.

Sinobi Ransomware

Encryption Key

Generate encryption key using Curve25519 algorithm and encrypt files with AES-128(CTR)

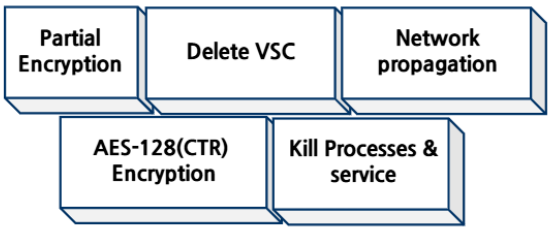


Encryption Method

Files ≤ 1MB: Encrypt entire file

Files > 1MB: Depending on the selected mode (entire / fast / medium / slow), the system performs partial encryption over only a subset of the file—specifically 100% / 5% / 15% / 25% of its total size, respectively.

Features



Ransom Note

README.txt

파일 편집 보기 H1 B

Good afternoon, we are Sinobi Group.

As you can see you have been attacked by us! We offer you to make a deal with us. all you need to do is contact us by fo
We are not politically motivated group, we are interested only in money, we always keep our word. You have a possibility
You have to know we do not like procrastination. You have 7 days to come to the chat room and start negotiations.

- 1 Communication Process:
  In order to contact with us you need to download Tor Browser.
  You can download Tor Browser from this link:
  https://www.torproject.org/download/
  After you joined to chat room you have the opportunity to request several things from us for free:
    1. make a test decrypt.
    2. get a list of the files stolen from you.
  At the end, we should agree on the price for our services. Keep in mind that we got your income/insurance docume

- 2 Access to the chat room:
  To access us please use one of the following links:
    1. http://sinobi7yuoppj76qnkwiobwfc2qve2xkv2ckvzyyjblwd7ucpptl62ad.onion/login
    2. http://sinobi57mfegeov2naiufkidlkpze263jtbldokimfjqmk2mye6s4yqd.onion/login
    3. http://sinobibdvzohujkliofkxiz3ueyedfh6bed2lzjz2z6pafw5jeoptsid.onion/login
    4. http://sinobibjqytwqxjw24zuerqcjyd3hoow6zia7z6kzvwawivamu7nqayd.onion/login
    5. http://sinobicrh73ongfuxjajmlyyhalvkhlcgttxkxaxz3gvsgdcgf76uiqd.onion/login
    6. http://sinobidxodgt4jsr3tlmf2rr4okjvvwfp5gh3lrqxnowomcx62ssrhqd.onion/login
    7. http://sinobiea4snfqtkc43paumapo4oi7vxcy5vjzfoalunsnvzehozfhpyd.onion/login

README.txt

Extension

.SINOBI

Production Language

C/C++

Figure 8. Sinobi Ransomware Overview

Ransomware Tactics and Strategy

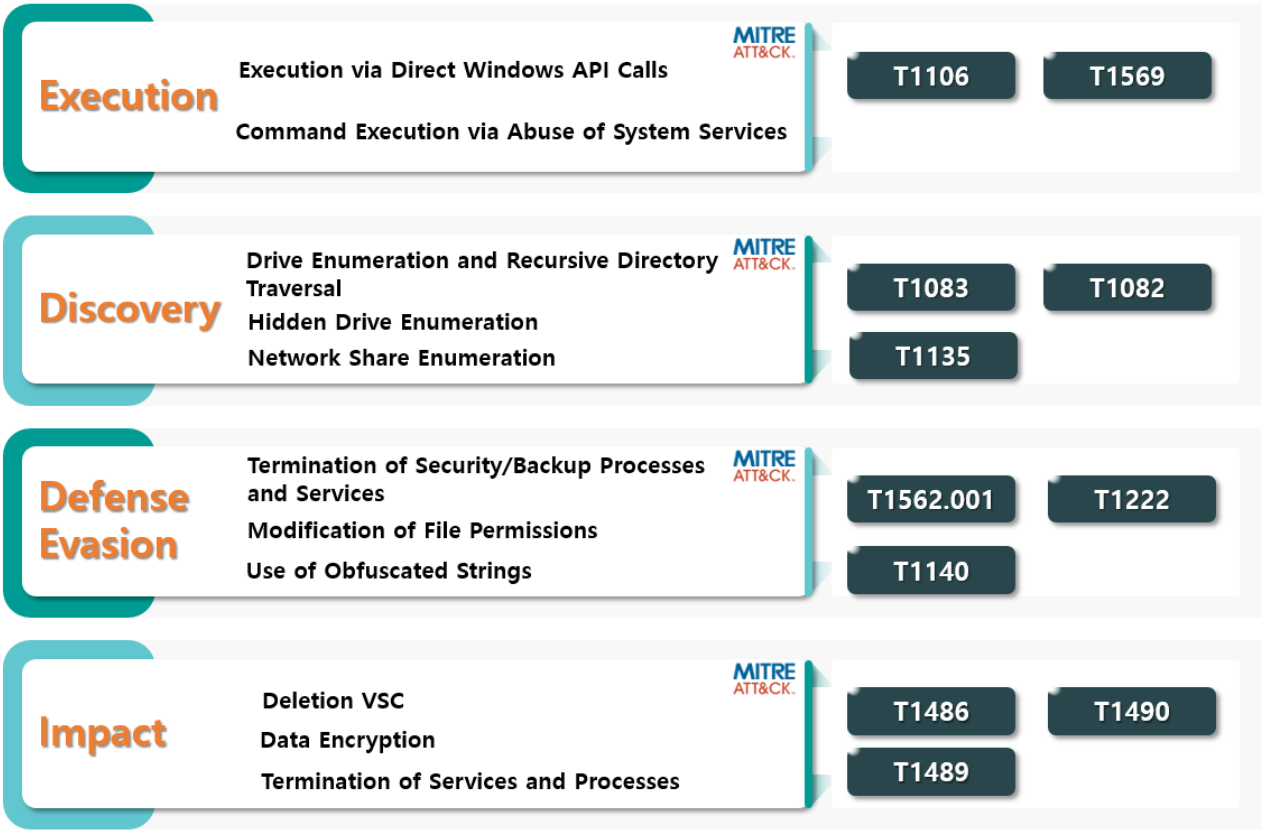


Figure 9. Ransomware Attack Strategy

Sinobi ransomware is designed to leverage a range of command-line parameters to precisely control encryption targets and behavior, enabling operators to define what to encrypt as well as select modes for full or partial encryption. The parameters and corresponding functions are summarized in the table below.

Parameter	Description
--file <filePath>	Specify target files for encryption
--dir <dirPath>	Specify target folders for encryption
--mode <mode>	Configure encryption mode (Entire / Fast / Medium / Slow)
--help Print this message	Display help/usage message
--verbose	Enable logging output
--silent	Do not change file extensions do not generate a ransom note
--hide-cmd	Hide the console window
--no-background	Disable wallpaper-changing functionality
--no-print	Disable ransom-note output functionality
--stop-processes	Terminate processes if target files are in use immediately before encryption
--kill	Terminate specific processes and services
--encrypt-network	Include network shares as encryption targets
--load-drives	Mount hidden drives
--safe-mode	Boot into Safe Mode

Table 1. Ransomware Execution Parameters

A comparison of execution parameters across Sinobi, INC, and Lynx indicates that the three ransomware families share a substantial set of common capabilities, although differences are evident in certain parameter configurations. First, an encryption-mode configuration feature is present in Sinobi and INC but was not identified in Lynx. Sinobi supports mode selection via the --mode parameter, and INC likewise uses --mode to set the encryption mode; by contrast, no parameter controlling this behavior has been observed in Lynx. In addition, the ability to omit both file-extension changes and ransom-note generation is provided only by Sinobi: when --silent is used, Sinobi neither changes extensions nor creates a ransom note, whereas no equivalent parameter was identified in Lynx or INC. Aside from these distinctions, the remaining major functions exposed at the parameter level are broadly consistent across all three ransomware variants.

```
if ( DeviceIoControl(FileW, 0x53C028u, &InBuffer, 0x18u, 0, 0, &BytesReturned, 0) )
{
  if ( byte_140033C78 )
    printf_1(L"[+] Successfully delete shadow copies from %c:/\n", i);
```

Figure 10. Deletion of Backup Copies

Sinobi ransomware, in the same manner as Lynx ransomware, deletes backup copies prior to encryption in order to hinder recovery. To this end, it invokes the DeviceIoControl function to reset the maximum allocated capacity of the storage area where backup copies are maintained to a low value. As a result, the system deems storage to be insufficient and, in the course of reclaiming space, automatically deletes previously created backup copies.

In addition, when the --kill parameter is used, the ransomware terminates specific processes and services to facilitate uninterrupted file encryption. The targeted processes and services are listed in the table below, and this list has been confirmed to be identical to the items observed in Lynx ransomware.

Processes	Services
sql, veeam, backup, exchange, java, Notepad	sql, veeam, backup, exchange

Table 2. Targeted Processes and Services for Termination

For file encryption, the --file parameter restricts encryption to a specific file, whereas the --dir parameter limits encryption to files located under a specified path. If neither parameter is provided, the ransomware encrypts all files except those on the exclusion list; the confirmed exclusions are shown in the table below.

Excluded Paths	Extensions, and Filenames
Windows, Program Files, Program Files (x86), $RECYCLE.BIN, AppData	.exe, .msi, .dll, .SINOBI, README.txt

Table 3. Encryption Exclusions

The encryption exclusions of Sinobi and Lynx are largely the same, although differences are observed in certain entries. To avoid re-encrypting files that have already been encrypted, Sinobi adds the ".SINOBI" extension to its exclusion list, whereas Lynx uses ".lynx" as the excluded extension.

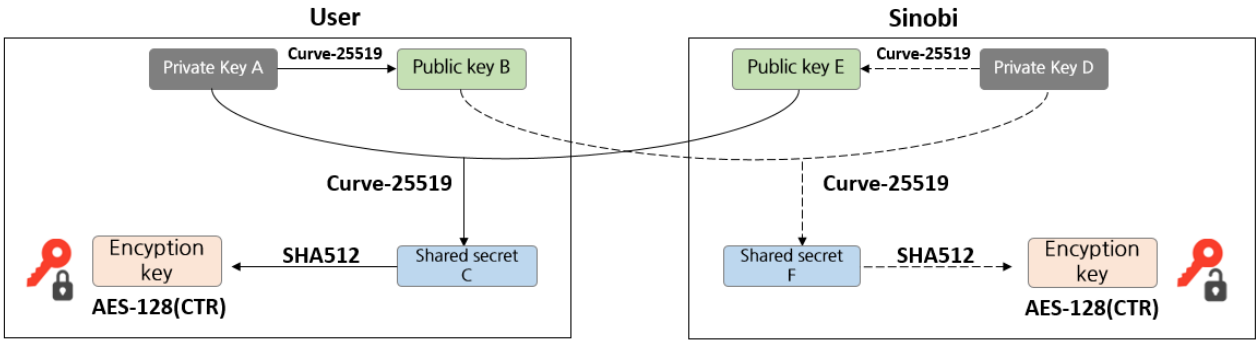


Figure 11. Encryption Key Generation Method

Sinobi ransomware generates a unique 32-byte private key (A) for each file to be encrypted. It then performs a Curve25519 operation with the attacker's hard-coded public key (B) to derive a shared secret (C). In this context, a shared secret refers to a value which, under the Curve25519 algorithm, both parties can compute identically using only their own private key and the counterparty's public key. That is, the value (C) computed from the victim's private key (A) and the attacker's public key (B) is identical to the value (F) computed from the attacker's private key (D) and the victim's public key (E); this identical value (C/F) constitutes the shared secret.

The derived shared secret is not used directly; instead, it is hashed with SHA-512 to produce a derived key, which is then used to encrypt the file using AES-128(CTR) Upon completion, Sinobi appends the victim's public key (E) to the end of the file. Using this public key (E) and the attacker's corresponding private key (D), the attacker can recompute the same shared secret, apply the same hash-based derivation, and thereby decrypt the file.

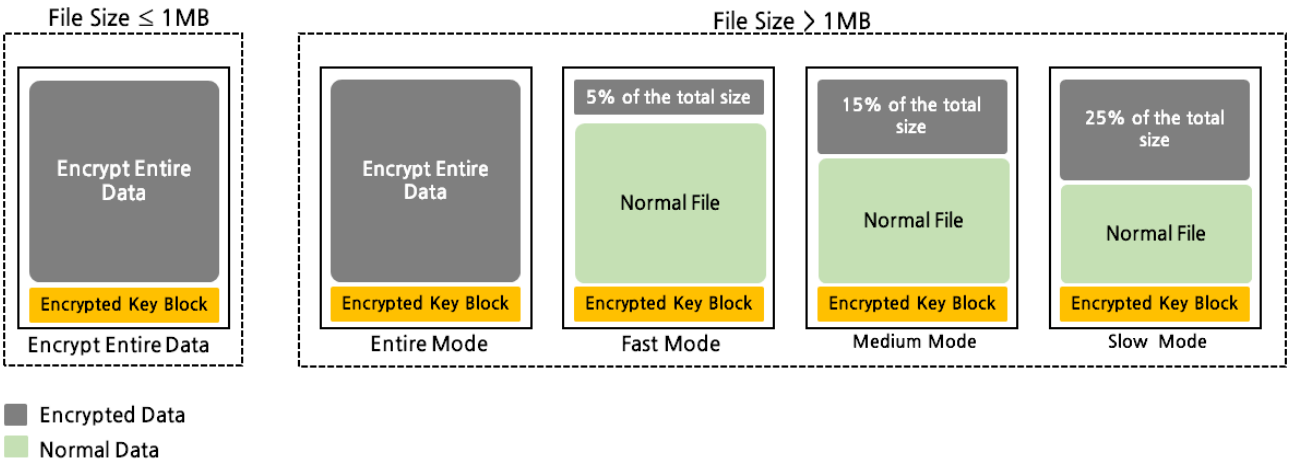


Figure 12. File Encryption Method

The encryption scope varies depending on file size and the parameters in use. Files of 1 MB or smaller are fully encrypted regardless of parameters. For files exceeding 1 MB, only a portion of the file is encrypted depending on the selected mode: Entire encrypts the whole file, Fast encrypts 5%, Medium 15%, and Slow 25%. After encryption, a fixed metadata block is appended to the end of every file irrespective of file size or mode. This block stores the public key used for decryption, the encryption mode applied to the file, and a “SINOBI” marker for infection identification.

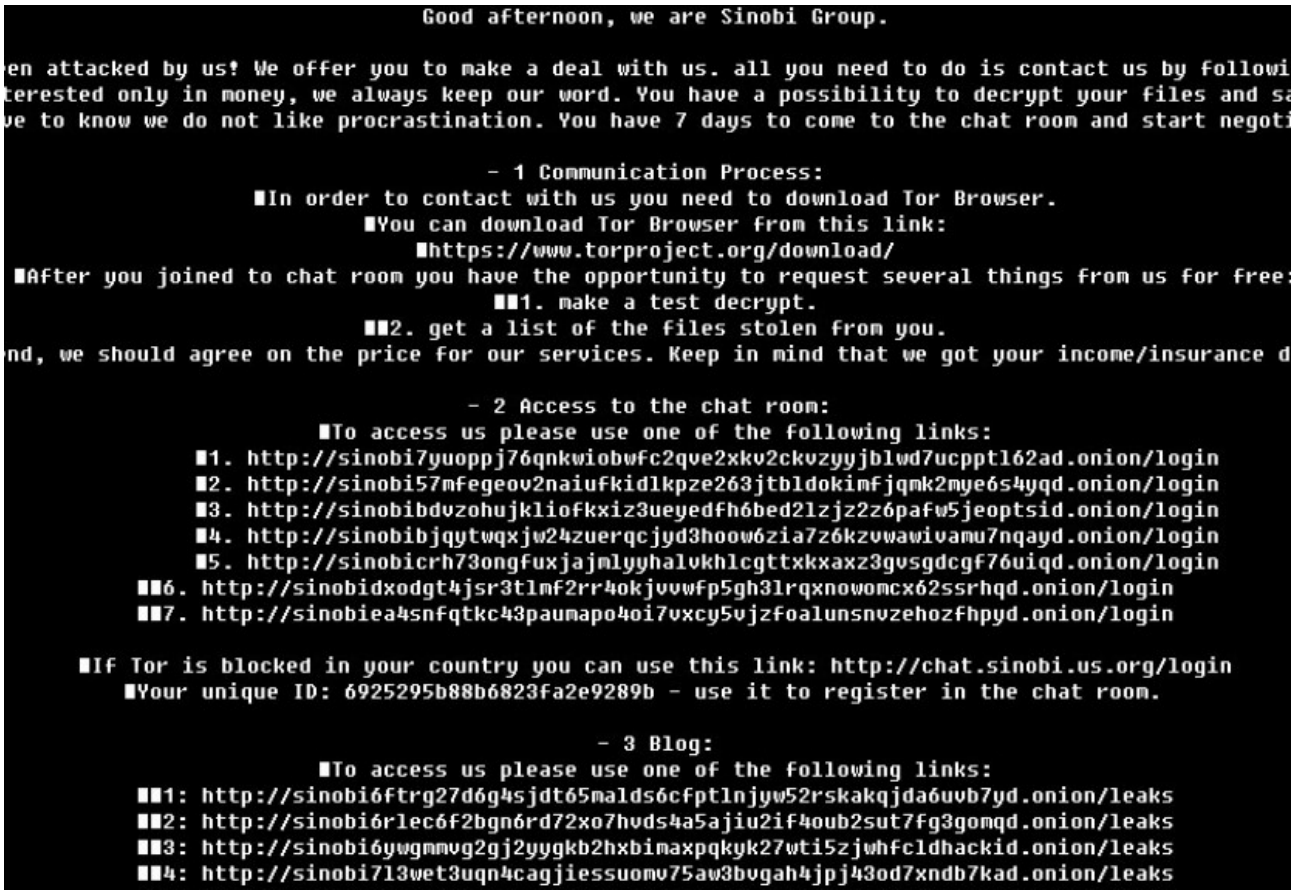


Figure 13. Modified Wallpaper

Upon completing file encryption, the ransomware generates a wallpaper image containing the ransom-note text at runtime and saves it to disk. It then sets the image path in the HKCU\Control Panel\Desktop\Wallpaper registry value and changes the desktop wallpaper, thereby prompting the victim to immediately view the ransom note contents.

Ransomware Response Measures

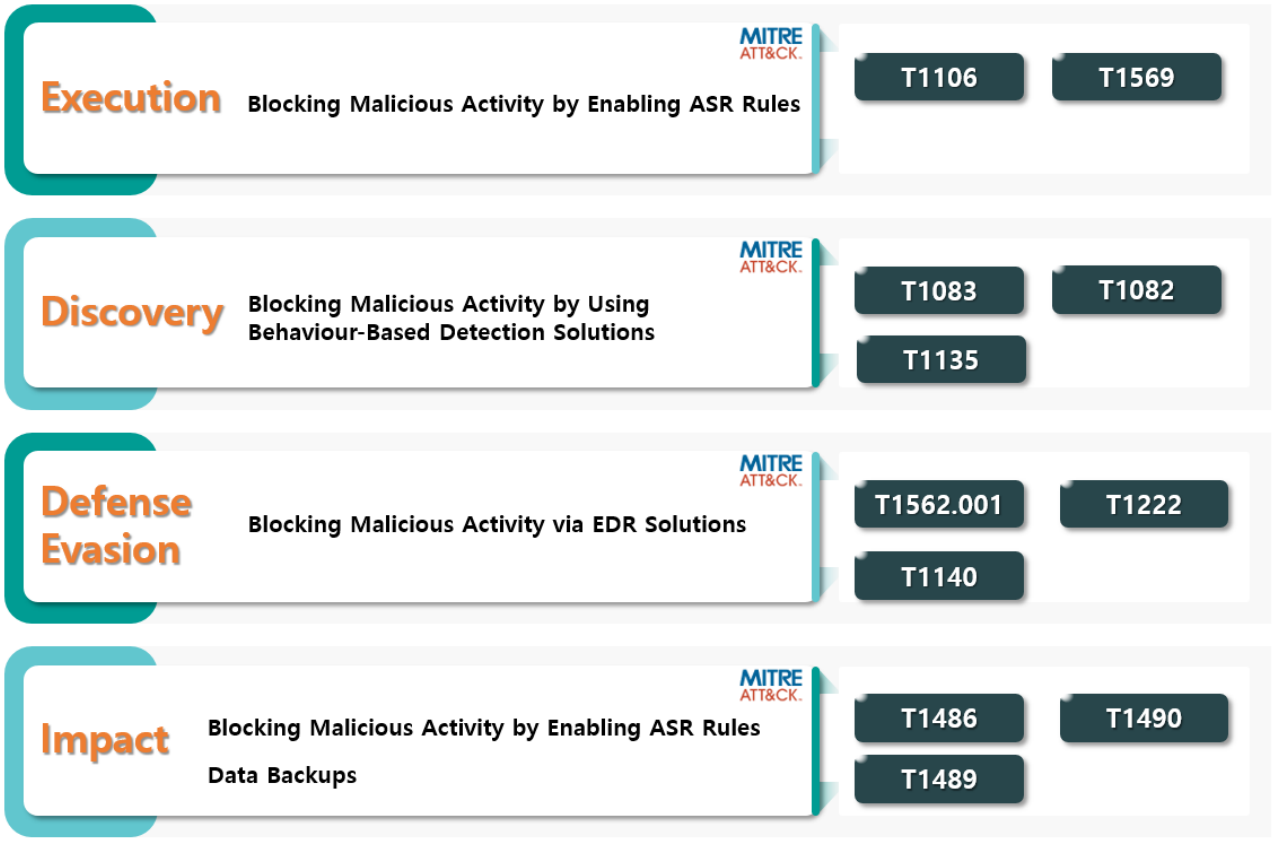


Figure 14. Ransomware Response Measures

During execution, Sinobi ransomware terminates VSS- and backup-related processes and services to impede recovery. As a countermeasure, enabling ASR[4] rules can block abnormal process activity and prevent the ransomware's malicious actions.

In addition, organizations should deploy EDR solutions and apply up-to-date security patches to enable the rapid detection and blocking of vulnerability-driven intrusions and anomalous activity. Backup copies should be distributed and backed up periodically to a separate network segment, external storage, or offline media so that data recovery remains feasible even if systems are encrypted. Access privileges to backup repositories should be minimized, and regular restoration tests should be conducted to ensure the integrity of the backup data. Finally, Sinobi ransomware also encrypts files on network shares, access privileges to shared resources should be minimized or the shares disabled to prevent unauthorized access to external resources.

[4] ASR (Attack Surface Reduction): A protective capability that blocks specific processes and executable content commonly leveraged by attackers.

IoCs

Hash(SHA-256)
9432B065C803BAA54F1FEFAC20D97AFFCE212DEC2BB9A597FC010064D391FC24
1B2A1E41A7F65B8D9008AA631F113CEF36577E912C13F223BA8834BBEFA4BD14
D4919A7402D7AE02516589FBDFB3CC436749544052843A37B5D36AC4B7385B18

■ References

• U.S. Department of the Justice(https://www.justice.gov/opa/pr/two-americans-plead-guilty-targeting-multiple-us-victims-using-alphv-blackcat-ransomware)

• Reuters(https://www.reuters.com/world/cyberattack-french-interior-ministrys-email-servers-compromised-more-than-20-2025-12-17/)

• Le Parisien(https://www.leparisien.fr/faits-divers/une-attaque-tres-grave-cinq-minutes-pour-comprendre-la-cyberattaque-qui-a-vise-le-ministere-de-linterieur-17-12-2025-ISX6EVWKDFCLLEZKHA2RBJFECA.php)

• S-RM(https://www.s-rminform.com/latest-thinking/react2shell-used-as-initial-access-vector-for-weaxor-ransomware-deployment)

Research & Technique

Authentication Risks from JWT Signing Key Exposure and Mitigation Strategies

■ Introduction

JWT is an authentication mechanism in which the information required for authentication is encapsulated and delivered within a single token. Because the server verifies the user through this token, it can process authentication without separately persisting user state. Owing to these characteristics, JWT is widely adopted in large-scale services and in environments composed of multiple servers.

In JWT-based authentication, when the server receives a request, it validates whether the presented token was generated legitimately. In this process, the critical element the server examines is the token's signature.

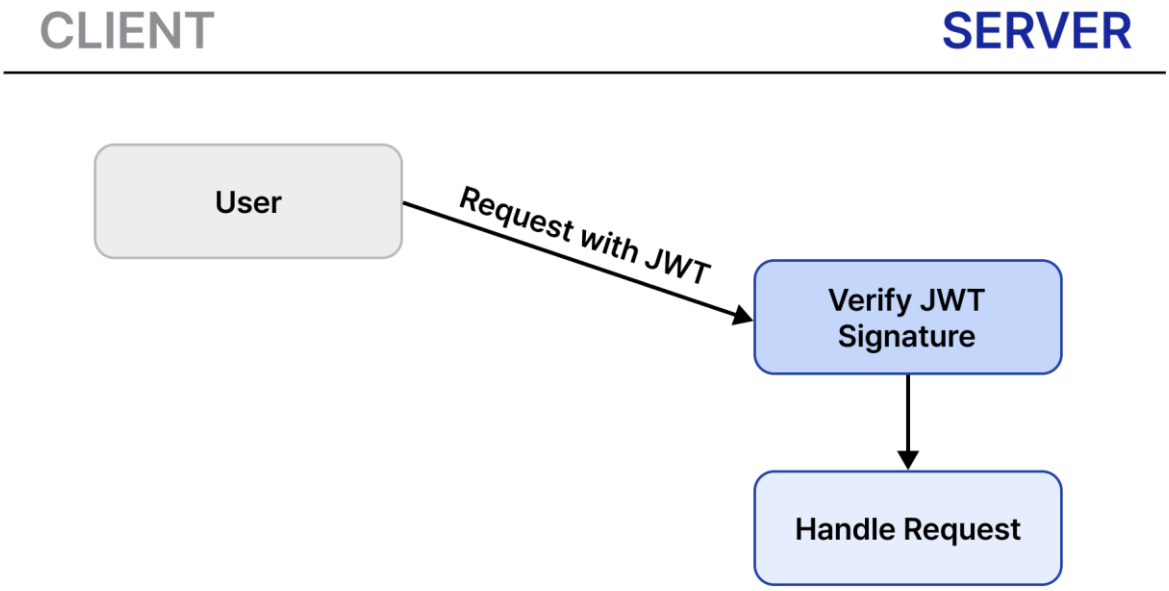


Figure 1. JWT-Based Authentication

In this architecture, the trustworthiness of the authentication process becomes concentrated in a single signing key used to produce the signature. Consequently, if the signing key is exposed to external parties or is inadequately managed, it can precipitate severe failures across the entire authentication scheme. Building on this characteristic of JWT, this report examines—through real-world cases—what issues arise when signing key management is neglected.

■ JWT

1) What is a JWT?

JWT is a token that a server uses—on every incoming request—to determine whether the request is a legitimate one originating from an authenticated user and whether that user is authorized to perform the requested operation.

A JWT is typically issued after a successful login and is subsequently sent along with later requests; the server uses it to identify the user and validate their privileges. Accordingly, JWTs are leveraged not only to maintain login state but also broadly in contexts where the legitimacy of a request must be proven, such as API access control and inter-service authentication (e.g., SSO).

Traditionally, a session-based approach—where the server stores "who is in what state" internally and looks up the stored information on each request—has been common. By contrast, the JWT approach shifts the burden away from the server retaining ongoing state and instead packages the necessary proof information into a token that the user carries. In this sense, JWT assumes a stateless architecture in which the server does not persist login state in server memory or a database, and it authenticates the user on each request based on the outcome of signature verification for the presented token.

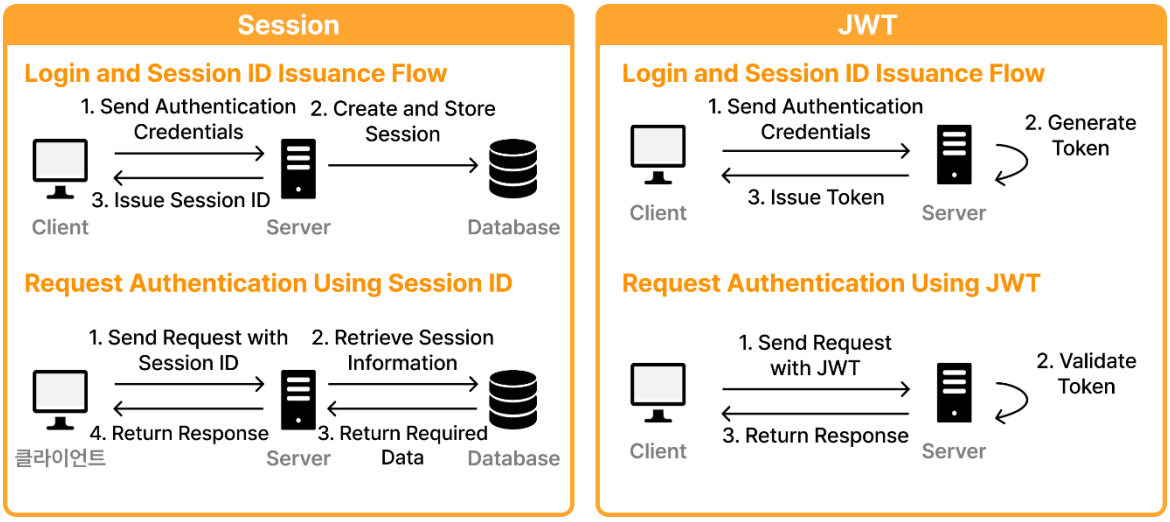


Figure 2. Session and JWT-Based Issuance and Authentication

2) JWT Components

A JWT consists of three components separated by dots (.), and is typically represented in the form Header.Payload.Signature. Each component is not designed for direct human readability; rather, it is a string encoded using Base64Url to ensure efficient transmission of data.

JWT Example
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkVRU1QiLCJhZG1pbiI6dHJ1ZSwiaWF0IjoxNTE2MjM5MDIyLCJodHRwczovL3d3dy5za3NoaWVsZHVzLmNvbS8iOnRydWUsInVzZXJuYW1lIjoiRVFTVCBMYWIifQ.PYq5fttdbnGoTpSsLiSI0rSBB49hsc1mAP85jG1wiGA

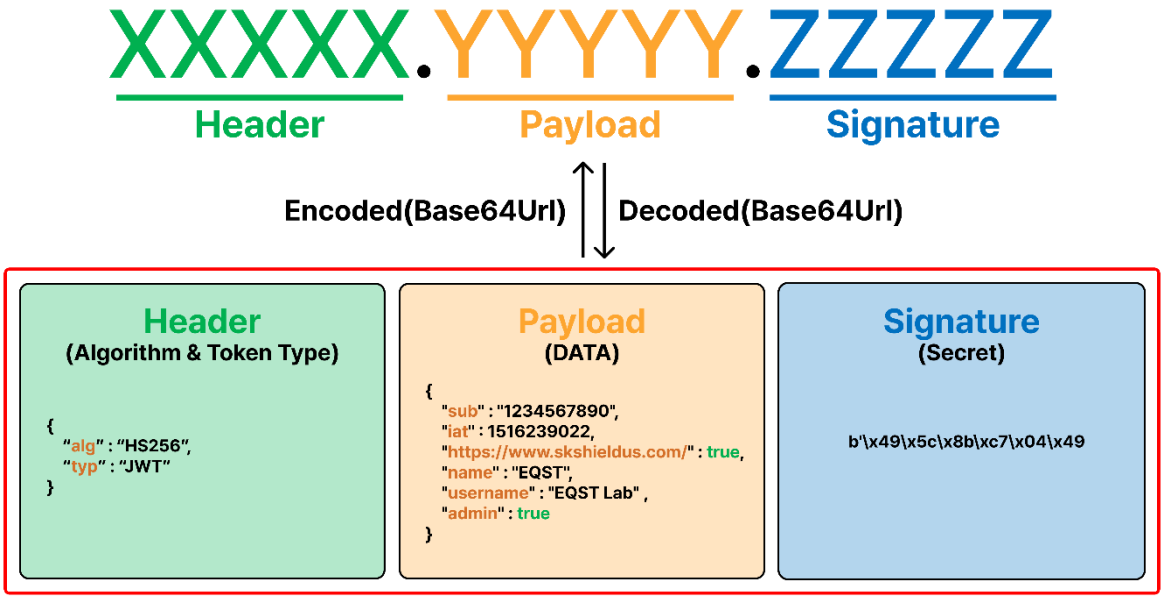


Figure 3. JWT Structure

① Header

The header contains information about the signing algorithm and the token type. When validating a token, the server compares the algorithm declared in the header against a pre-approved allowlist; only if they match does it proceed to verify the signature using that algorithm.

```
{
  "alg": "HS256",   // Signature Algorithm
  "typ": "JWT"      // Token Type
}
```

Figure 4. JWT Header

② Payload

The payload contains the actual information to be embedded in the token, and each individual field within that information is referred to as a claim. Under the JWT standard (RFC 7519), claims are classified—based on their characteristics and scope of use—into registered claims, public claims, and private claims.

```
{
    "sub" : "1234567890",
    "name" : "EQST",
    "admin" : true,
    "iat" : 1516239022,
    "https://www.skshieldus.com/" : true,
    "username" : "EQST Lab"
}
```

Figure 5. Fully Decoded Payload

- **Registered Claim**

Registered claims are claims whose semantics and intended usage are predefined by the JWT standard, and they are used to represent information such as the token issuer, validity period, and intended audience. Although they are not mandatory, they are frequently used in real-world service environments because they support token validity assessment and service operation.

Figure 6. Registered Claims in the Payload

The types of registered claims defined by the JWT standard are as follows.

Abbreviation	Claim Name (Full Name)	Description
iss	Issuer	Token issuer
sub	Subject	Unique user identifier
aud	Audience	Token audience (recipient)
exp	Expiration Time	Token expiration time (Unix Time)
nbf	Not Before	Token activation start time
iat	Issued At	Token issuance time
jti	JWT ID	Token unique identifier

Table 1. Types of Registered Claims

• Public Claim

Public claims are used to prevent collisions that can occur when multiple services share the same JWT and different services define claims with identical names. To achieve this, the claim name is expressed in a URI form such as "https://www.skshieldus.com/"—so that it clearly indicates which organization or service defined it. In other words, when a claim name is defined in a URI format, it is categorized as a public claim.

Figure 7. Public Claims in the Payload

• Private Claim

Private claims are not semantically defined by the JWT standard, meaning both the claim names and their values can be designed freely according to service requirements. As a result, they are primarily used to embed, in a service-specific manner, the information required by the server's authentication and authorization logic.

Figure 8. Private Claims in the Payload

③ Signature

The signature is a cryptographic value used to verify that the JWT has not been tampered with in transit and to determine whether the token was generated by an issuer trusted by the server. Because the JWT header and payload can be readily decoded by anyone to inspect their contents, the token's trustworthiness is ultimately determined by the outcome of signature verification.

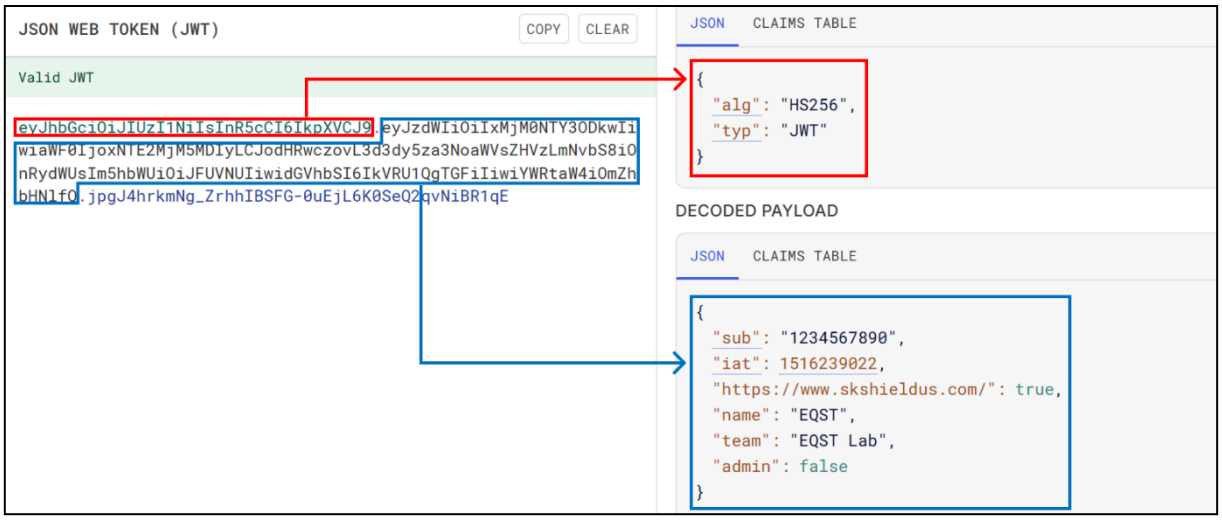


Figure 9. Decoded JWT

The signature is generated by first Base64Url-encoding the JWT header and payload, then concatenating the two values with a dot (.) and using the resulting string as one of the inputs to the hashing process. Accordingly, even a minor change to either the header or the payload alters the input, causing the signature to fail to match during server-side verification and the token to be rejected.

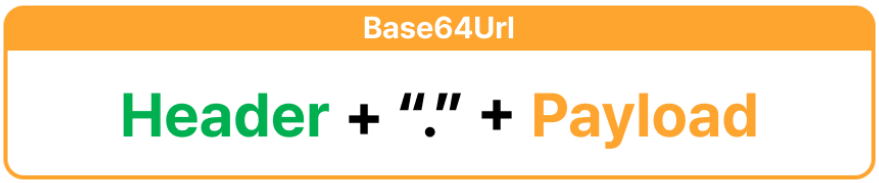


Figure 10. Signature Input Value

In this context, the decisive property the server relies on to detect whether the input has been altered is the inherent behavior of the hash function. A hash function maps input data of arbitrary length to a fixed-length output, and even a slight change in the input produces a completely different result. Therefore, if the token contents are modified, the input changes accordingly, and signature verification ultimately fails.

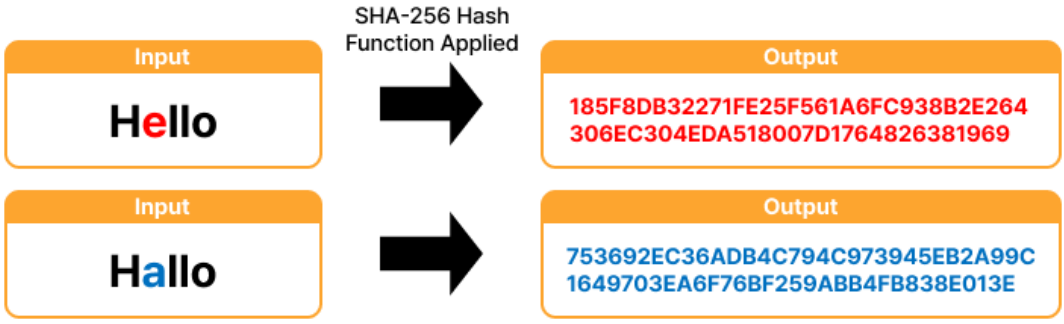


Figure 11. Hash Example

However, hashing alone is insufficient to establish authentication. Because anyone who knows the input can compute the same hash output, an attacker could modify the payload and then simply recompute and attach a corresponding hash value. For this reason, JWT does not rely on a plain hash; instead, it generates the signature using a key-based signing algorithm. In other words, by using as the signature a value that can be produced only with the server-held key, JWT enables verification not only that the token has not been altered, but also that it was signed with a key trusted by the server.

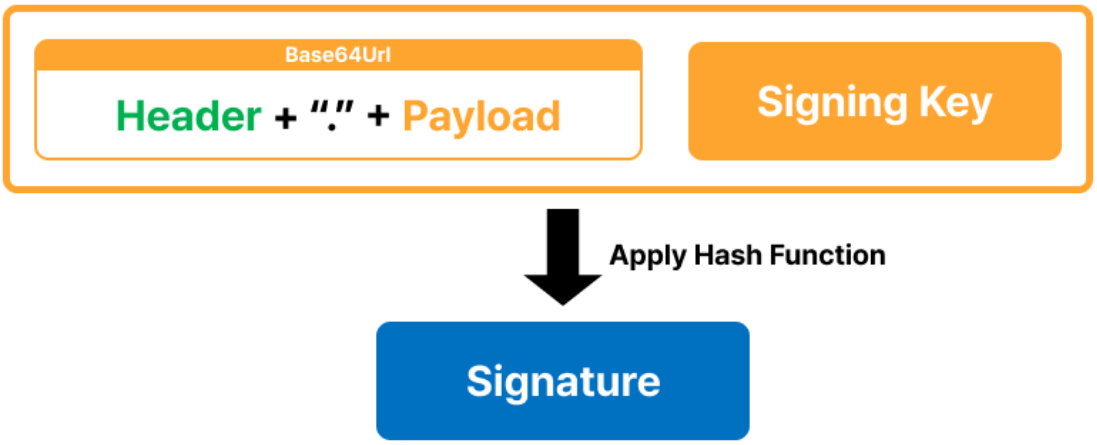


Figure 12. JWT Signature Generation Process

Accordingly, the reliability of JWT can be sustained only under the premise that the key used to generate the signature is securely protected. If that trust is undermined—such as through signing key exposure—the JWT authentication scheme itself can be destabilized, as will be illustrated in the following sections through real-world cases and scenarios.

■ Impact Cases Involving Signing Key Exposure

As discussed earlier, JWT determines trust based on the outcome of signature verification. For this reason, the protection and governance of the signing key used to generate the signature becomes central to security. The cases below illustrate what can go wrong in a JWT-based authentication architecture when signing key management fails.

1) Coupang User Personal Data Exposure (2025) – Large-Scale Data Leakage Caused by API Authentication Bypass

In 2025, Coupang experienced a large-scale personal data exposure incident that was linked to deficiencies in the management of a JWT signing key used within its internal authentication scheme. The scale of the leak was reported to be approximately 33.7 million records. According to media reports, a signing key obtained by a former employee during their tenure was allegedly abused, and shortcomings were noted in post-departure controls such as key decommissioning and rotation.

By arbitrarily constructing the JWT header and payload and then generating a valid signature using the acquired signing key, the attacker was able to produce tokens that passed server-side verification. This enabled the attacker to bypass authentication for a personal data query API and access the personal information associated with numerous accounts.

The access was reportedly not a one-off attempt over a short window, but rather occurred continuously over a period of time; furthermore, anomalous API calls leveraging forged JWTs were not promptly blocked before the activity ceased. This incident demonstrates, through a real-world case, that once a signing key is exposed in a JWT-based authentication architecture, the server can struggle to distinguish such requests from legitimate user requests.

2) Microsoft Storm-0558 (2023) – Mailbox Access via Forged Tokens

In 2023, Microsoft officially disclosed that a China-based threat group known as “Storm-0558” leveraged forged authentication tokens to gain access to Outlook and Exchange Online services. According to Microsoft’s subsequent report, the attacker obtained a signing key that was used for authentication tokens issued after login for Microsoft Accounts (MSA), and then used it to directly generate tokens that would appear, from the service’s perspective, as though a legitimate user had signed in.

Because the service treated the token as a legitimate authentication request, the attacker was able to access mailbox retrieval APIs under the privileges of specific users. This incident illustrates how, in an architecture where user authentication is handled via signed tokens rather than server-side sessions, the exposure of the key used to generate those tokens can neutralize the entire authentication scheme.

It further demonstrates that, in stateless token-based authentication environments—typified by JWT—a single signing key sits at the center of trust, and if that key is compromised, authentication as a whole can collapse.

3) Solorigate/Golden SAML[5] (2020) – SSO Authentication Bypass via Forged SAML Tokens

In the course of responding to the SolarWinds supply-chain breach (Solorigate) disclosed in 2020, Microsoft indicated that it had identified activity in which an attacker, after taking control of ADFS within certain victim organizations, exfiltrated the ADFS[6] token-signing certificate (private key) and used it to forge SAML login tokens that would be accepted as legitimate authentication. With this certificate, the attacker could generate and sign SAML tokens that appeared to have been issued on behalf of arbitrary users, and the service—recognizing the signature as valid—would accept them as the result of a legitimate login.

This case did not involve JWT; it occurred with SAML-formatted tokens. However, it demonstrates that in any architecture that relies on trusting signed tokens, the management of the keys used to generate and verify signatures (i.e., signing keys / signing certificates) is critical regardless of the token format.

■ JWT Forgery Using a Leaked Signing Key

The preceding cases demonstrate that, once a signing key is compromised, an attacker can craft tokens directly and bypass authentication. In architectures where signature verification serves as the primary basis of trust, a token generated externally is difficult to distinguish from a legitimate token as long as it is signed with the same key. For this reason, as seen in the earlier incidents, compromise was not detected immediately and was instead confirmed through post-incident investigation and analysis. This chapter explains the issue from the perspective of JWT validation criteria and the token-forgery process.

In general, the server uses signature verification as the first-line trust criterion to determine whether a token has been tampered with.

[5] SAML (Security Assertion Markup Language): A standard for conveying authentication outcomes in SSO environments via XML-based tokens.

[6] ADFS (Active Directory Federation Services): A Windows Server feature that, after a user signs in once with a corporate account, issues a token attesting to that authentication and enables single sign-on across multiple services.

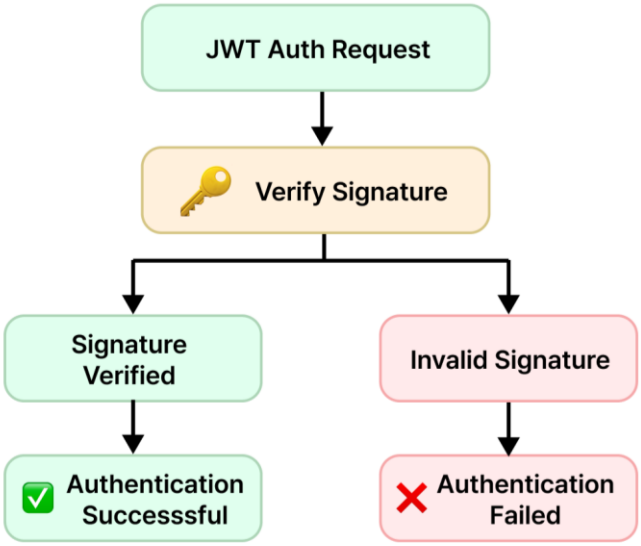


Figure 13. JWT Validation Criteria

When a signing key is leaked, an attacker can generate signature values in the same manner. Consequently, in a JWT-based authentication environment, the holder of the signing key effectively acquires the same authority as the token issuer. This is because the system cannot differentiate between a token issued by a legitimate authentication server and a token generated externally using the same signing key.

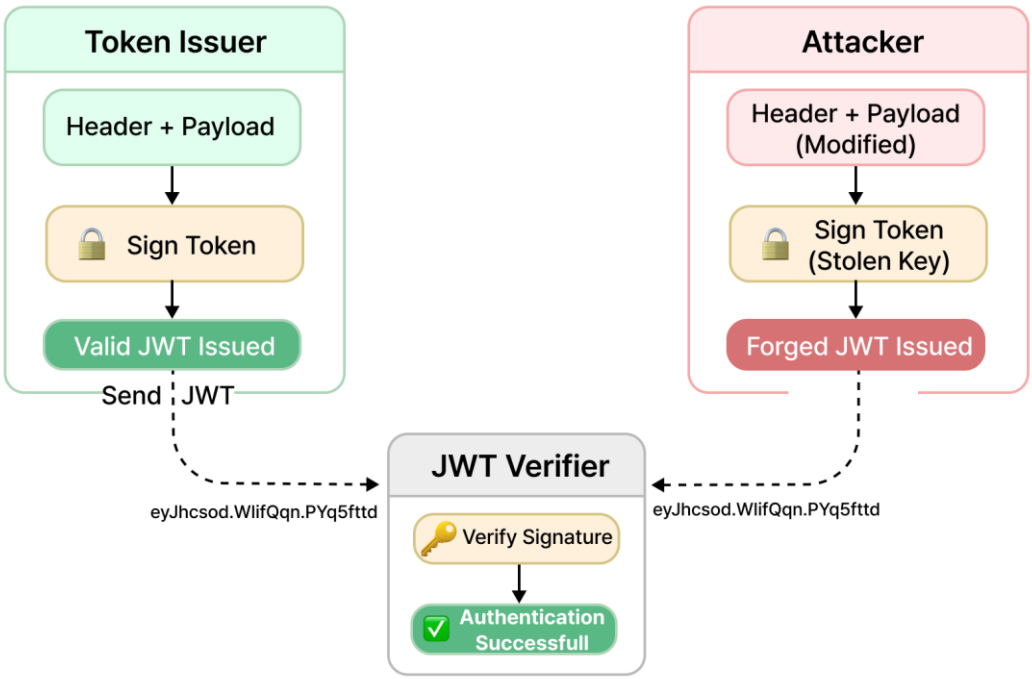


Figure 14. JWT Forgery (Generation) Process Using a Signing Key

In particular, because the JWT payload is transmitted without encryption, it is possible to infer the token's structure and inspect the embedded claim values.


```
{
  "sub": "1234567890",
  "iat": 1516239022,
  "https://www.skshieldus.com/": true,
  "name": "EQST",
  "team": "EQST Lab",
  "admin": false
}
```


Figure 15. JWT Payload

If an attacker obtains the signing key, they can use the payload of an existing token as a reference, modify the user identifier or authorization-related information, and then generate a new JWT signed with the same key.

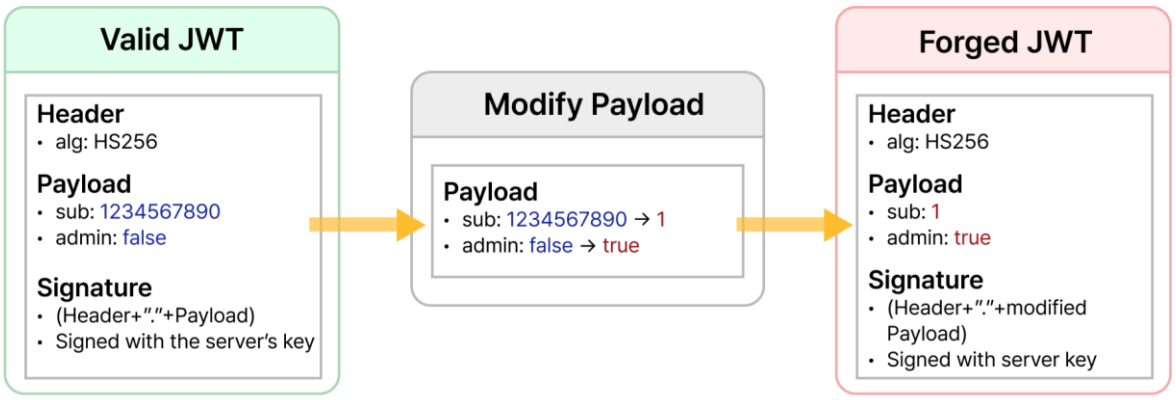


Figure 16. Generating a JWT with a Modified Payload

JWT-based authentication treats signature verification as its trust baseline, exposure of the signing key creates an environment in which token forgery attempts—by producing signatures with the same key—become feasible.

■ Attack Scenario

If a JWT is forged by an attacker, a range of issues can arise. This chapter provides illustrative examples.

1) API Authentication Bypass

In e-commerce services, APIs that return user-specific data—such as personal information retrieval and order-history lookup—often identify the requester based on the JWT included in the request. As can be seen from the request headers below, when invoking personal-information query and modification APIs, a JWT-formatted token is transmitted within the cookie.

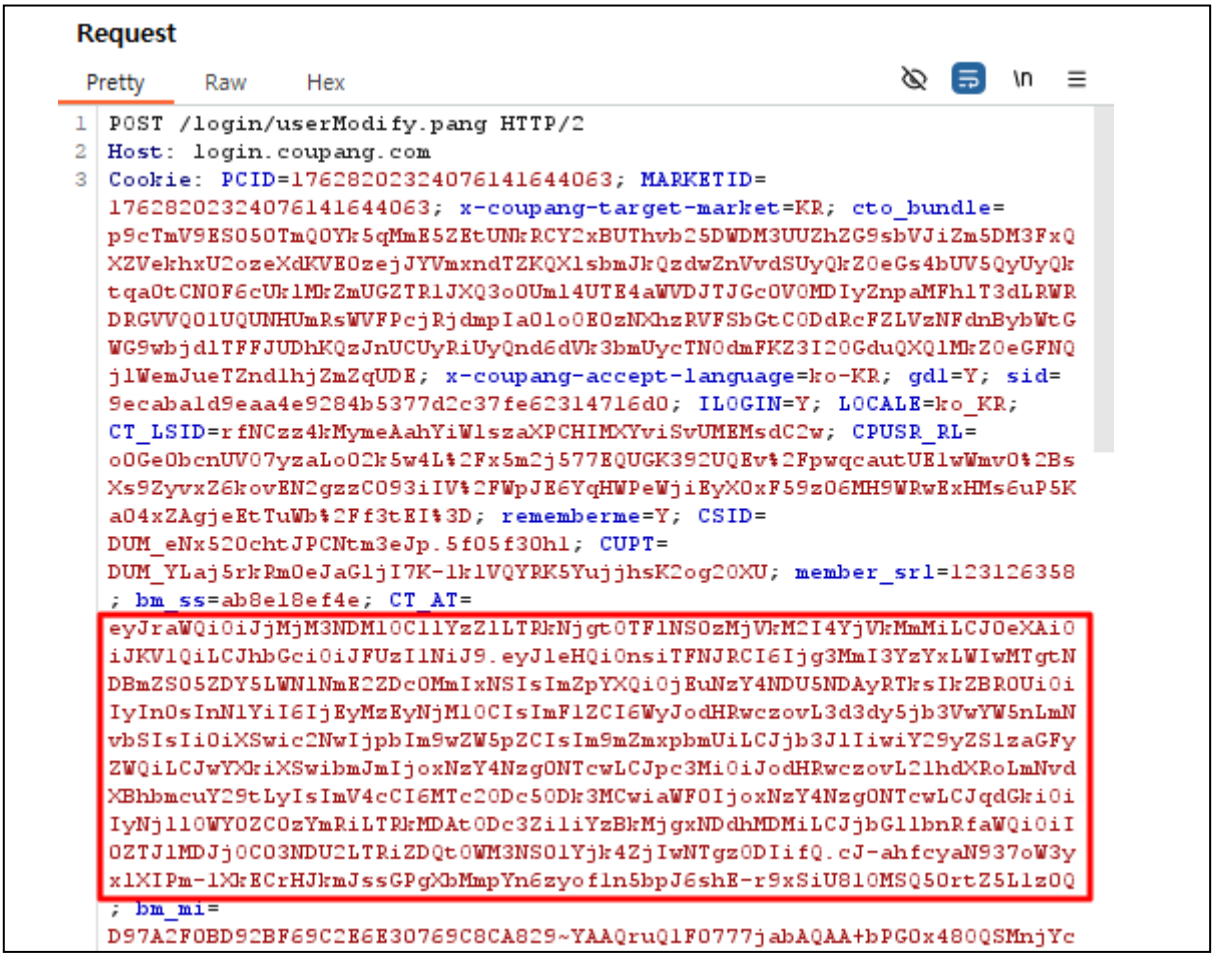


Figure 17. JWT Present in a Personal-Information API Request

Decoding the JWT reveals that it contains user-identifying information, such as the sub claim. In some implementations, the server uses this sub value directly to determine which user's data to retrieve during request processing.

DECODED HEADER

JSON CLAIMS TABLE COPY


```
{
  "alg": "ES256",
  "kid": "41f8777d-8250-4f3d-a674-c072faa2c257",
  "typ": "JWT"
}
```


DECODED PAYLOAD

JSON CLAIMS TABLE COPY


```
{
  "aud": [
    "https://www.coupang.com"
  ],
  "client_id": "4e2e02c8-7456-4bd4-9c75-5b98f2058382",
  "exp": 1767689015,
  "ext": {
    "LSID": "90f8c9fa-05ab-4b30-9ebf-c8a08ceedf87",
    "fiat": 1767674614
  },
  "iat": 1767674614,
  "iss": "https://mauth.coupang.com/",
  "jti": "b62d4f81-fc99-4483-bce3-4b3b89a01924",
  "nbf": 1767674614,
  "scp": [
    "openid",
    "offline",
    "core",
    "core-shared",
    "pay"
  ],
  "sub": "123126358"
}
```

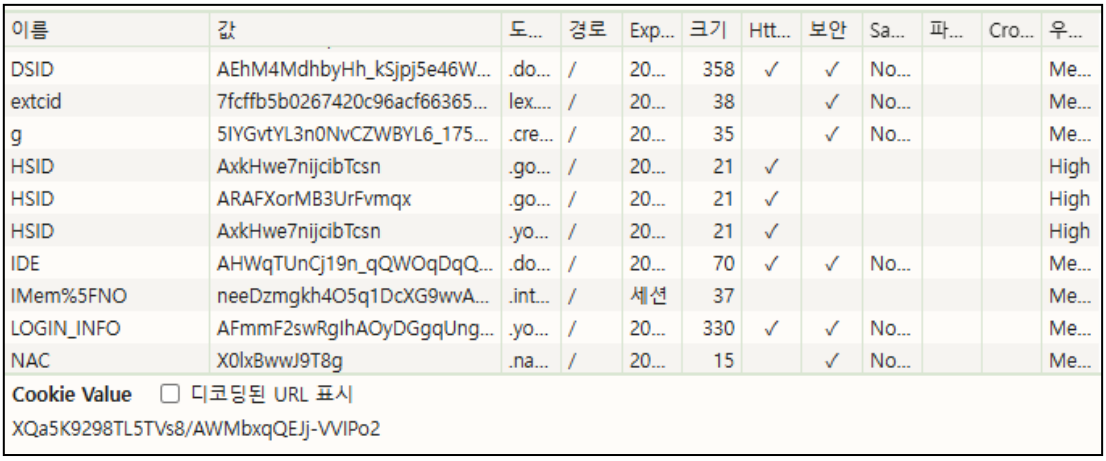

User Identifier

Figure 18. JWT Decoding Result

Therefore, if the signing key is compromised, an attacker can modify the sub value to another user identifier, generate a new JWT with a valid signature, and invoke the same API. When combined with automation, this approach can scale into bulk enumeration via sequential user-identifier substitution, causing the scope of personal data exposure to expand rapidly.

2) Login Bypass

Some services store a JWT in a cookie after a successful login, and for subsequent requests, they determine the user's login state solely based on the validity of that token. Before login, as shown below, no JWT is present.



이름	값	도...	경로	Exp...	크기	Htt...	보안	Sa...	파...	Cro...	우...
DSID	AEhM4MdhbyHh_kSjpj5e46W...	.do...	/	20...	358	✓	✓	No...			Me...
extcid	7fcffb5b0267420c96acf66365...	lex....	/	20...	38		✓	No...			Me...
g	5IYGvtYL3n0NvCZWBYL6_175...	.cre...	/	20...	35		✓	No...			Me...
HSID	AxkHwe7nijcibTcsn	.go...	/	20...	21	✓					High
HSID	ARAFXorMB3UrFvmqx	.go...	/	20...	21	✓					High
HSID	AxkHwe7nijcibTcsn	.yo...	/	20...	21	✓					High
IDE	AHWqTUnCj19n_qQWOqDqQ...	.do...	/	20...	70	✓	✓	No...			Me...
IMem%5FNO	neeDzmgkh4O5q1DcXG9wvA...	.int...	/	세션	37						Me...
LOGIN_INFO	AFmmF2swRgIhAOyDGgqUng...	.yo...	/	20...	330	✓	✓	No...			Me...
NAC	X0lxBwwJ9T8g	.na...	/	20...	15		✓	No...			Me...

Cookie Value ☐ 디코딩된 URL 표시

XQa5K9298TL5TVs8/AWMbxqQEJj-VVlPo2

Figure 19. Cookies Before Login

After login, a new JWT—named id_token—is generated and stored in the cookie.

이름	값	도...	경로	Exp...	크기	Htt...	보안	Sa...	파...	Cro...	우...
DSID	AEhM4MdhbyHh_kSjpj5e46W...	.do...	/	20...	358	✓	✓	No...			Me...
extcid	7fcffb5b0267420c96acf66365...	lex....	/	20...	38		✓	No...			Me...
g	5IYGvtYL3n0NvCZWBYL6_175...	.cre...	/	20...	35		✓	No...			Me...
HSID	AxkHwe7nijcibTcsn	.go...	/	20...	21	✓					High
HSID	ARAFXorMB3UrFvmqx	.go...	/	20...	21	✓					High
HSID	AxkHwe7nijcibTcsn	.yo...	/	20...	21	✓					High
id_token	eyJraWQiOiJfTmVvUDZlX1Nl...	.int...	/	세션	1032	✓	✓	Lax			Me...
IDE	AHWqTUnCj19n_qQWOqDqQ...	.do...	/	20...	70	✓	✓	No...			Me...
IMem%5FNO	neeDzmgkh4O5q1DcXG9wvA...	.int...	/	세션	37						Me...
interparkSNO	Vxv7t0P%2F85qVE2iXyuxx9p0...	.int...	/	세션	60		✓	Lax			Me...

Cookie Value ☐ 디코딩된 URL 표시

eyJraWQiOiJfTmVvUDZlX1NlWkxkVnpQZk9NbUVRb09CUE9uR3ZNZmdQUUNZc0l6ZFJBIiwiYWxnIjoiUlMyNTYifQ.eyJhY2Nlc3Nf
dG9rZW4iOiJXeGtnd3g4akxtZ3hrWGE5WmlmWVVkS1BCZzJcL0F1aXVmRjR3d1VMbVBkVVdhRlhyV1hBTURGMWJqN0N4SlN3R
yIsInN1YiI6IjdYXC9RNlhRXC9jZExVYVdtZURlek9uZz09IiwiYXVkIjoidGlja2V0LXBjIiwicmVmcmVzaF90b2tlbiI6IkpsbDZJbmtlUzNLa
XdkTEh3WTl4SjdxcE1EOUkzOG1BbjhtenJmTHFaWng3RVh5SzdHYlBPYlVZRGQycDdsdUEiLCJpc3MiOiJodHRwczpcL1wvYWNjb3
VudHMuaW50ZXJwYXJrLmNvbSIsInJlbWVtYmVyX21lIjpmYWxzZSwiZXhwIjoxNzY4NTMyMjUxLCJpc19ub2xfY29ubmVjdGVkIjp0
nVlLCJpYXQiOjE3Njg1MjY4NTEsInZlcnNpb24iOiIyLjAiLCJtbm8iOiJuZWVEem1na2g0TzVxMURjWEc5d3ZBPT0iLCJzaWQiOiJWeH
Y3dDBQXC84NXFWRTJpWHl1eHg5cDBwM1NnRFVjb3JuOVk1ZDBicVljWT0ifQ.o9mos9k12ZFyrnEkm7Ls3xug3p3VG3-dm9B4DY
47J1rLNkJODAUxC1B8PileajHQD5WUlMDQNFUmCSr0xM5vBKnP5ZWa3FDZSA_o0e31J4yEGP5QSYMK7_zc_LE2ez25UX3QDdoc
dlMpMxKDfq2y3KGfYhJj39-Id-jqlf-_40PgefhIfeZCbAbCZyfHPPMyTZ-EmclCS2MjWUyf6thOztla7t-e3YBNaLZooP7etlS50_0yD-LC
RmRh38mb6ADd8Fz7mGrB768GXvQ32LnBc9Xk7rAF4gDyNhUZXTFokcxGnd9yPe69AFAJMKAh8vYDyfqjCgCiqAu-yPG5AiThA

Figure 20. JWT Token Generated After Login

Decoding this id_token shows that, in addition to standard claims such as sub, iat, and exp, it also includes values like access_token, refresh_token, and sid that contribute to establishing and maintaining the login state. In other words, this token can function as a session substitute that the service uses to determine and preserve a user's authenticated state.

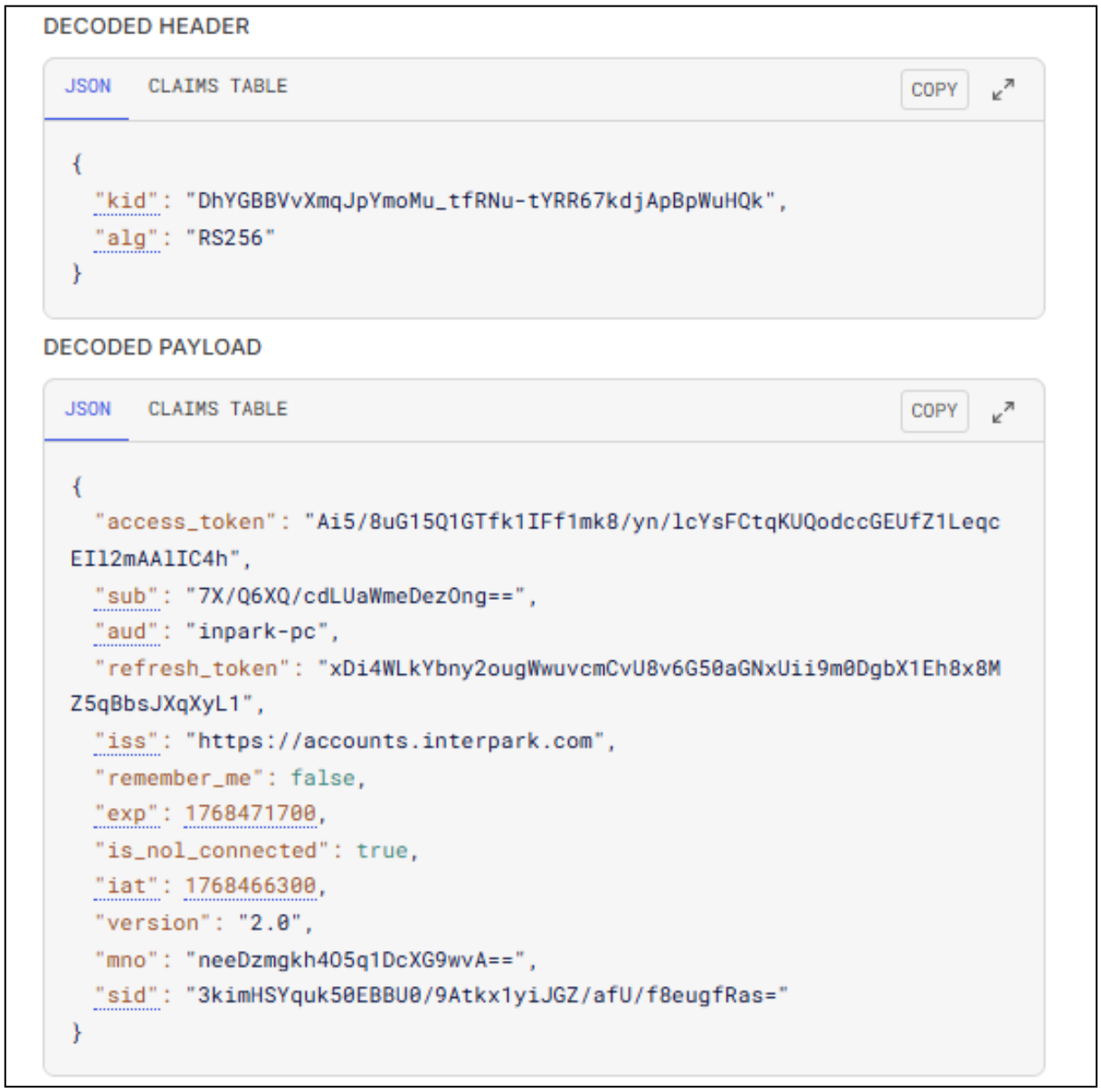


Figure 21. JWT Decoding Result

If the signing key is compromised, an attacker can generate and inject a valid id_token without going through the login process, and the service will interpret it as a legitimate login outcome. In other words, the attacker can behave as though they are logged in despite never having authenticated.

3) Federated Login Bypass

In SSO (federated login) environments, instead of entering an ID/password directly into the service, an external authentication system issues a token representing the login result, and the service verifies that token to complete the login flow. In many cases, the login-result token is delivered in JWT form, and the service determines whether it is a "legitimately issued login token" by validating the token's signature and checking its basic claims.

If an attacker obtains the signing key used for SSO tokens, or can bypass signature validation by exploiting weaknesses in the token verification logic, they can craft a token that appears to represent a legitimate login for a specific user and submit it to the service without undergoing the 정상 authentication process. Because the service accepts a token with a valid signature as a legitimate login outcome, the attacker can ultimately access the service under the privileges of the SSO-linked account.

In particular, SSO often operates under an architecture in which multiple services share the same authentication framework; therefore, compromise of a single signing key can have cascading impact beyond one service, extending to other integrated services as well. In this way, JWT forgery in federated login environments can escalate from "bypassing authentication for an individual service" to "a systemic collapse of trust across the broader authentication ecosystem."

4) Privilege Escalation

In JWT-based authentication environments, depending on the implementation, user authorization data may be embedded as claims in the payload. For example, claims such as role, isAdmin, or authLevel may be used to distinguish regular users from administrators, and the service determines which functions are accessible in subsequent requests based on those values.

Where authorization information is encoded within JWTs, compromise of the signing key enables an attacker to modify the payload of an existing token by arbitrarily changing privilege-related claim values and then reissuing a JWT with a valid signature. For instance, by setting role to "ADMIN" or elevating an authorization level claim and then calling administrator-only APIs, an attacker could abuse privileged functionality such as accessing admin pages, managing user accounts, changing configurations, and viewing logs.

Such privilege escalation is high-risk because it can affect overall service operations rather than remaining confined to a single account compromise. The broader the administrative 권한 scope, the more likely it is to expand into secondary impact, including data tampering, large-scale information leakage, and service disruption.

■ Mitigation Strategies

In JWT-based authentication environments, trust is effectively concentrated in a single signing key; once that key is exposed, the entire authentication scheme can be impacted regardless of token format or algorithm. In architectures with such heavy reliance on the signing key, it is difficult to eliminate the risk of key exposure entirely if the key is stored in plaintext in files, environment variables, or documentation, or operated in a manner known only to a specific administrator.

Accordingly, the core direction of JWT security is to isolate and protect the signing key so that it is not exposed to people or the application runtime environment, while also establishing an architecture that can minimize authentication damage even under compromise conditions.

To this end, the mitigation measures are organized from a signing-key management perspective in the following order: signing key separation, signing authority control, and blast-radius limitation under an assumed leak.

1) Signing Key Separation

This chapter, as the first step in signing-key management, outlines approaches to minimize the scope in which the signing key is used by separating signing and verification (symmetric vs. asymmetric keys) and by operating the signing key in an isolated manner.

• Separating Signing and Verification

In JWT-based authentication, when a server receives a request, it determines whether the presented token was legitimately issued solely based on the outcome of signature verification. That is, the server decides whether to authenticate by asking only "which key signed this token," and the signing key used in this process becomes the pivotal element that determines the token's trustworthiness.

Category	Signing Algorithm	Token Generation	Token Verification
Symmetric key	HMAC-SHA256(HS256)	Secret Key	
Asymmetric key	RSA-SHA256(RS256)	Private Key	Public Key

Table 2. Representative JWT Signing Algorithms by Key Type

In the symmetric-key approach, HMAC-SHA256 (HS256) uses a single secret key for both token generation and verification. Because it is straightforward to implement and offers fast processing, HS256 has been widely used in single-service deployments or relatively simple authentication architectures. However, since every server that processes authentication requests must possess the same secret key, scaling the service inevitably broadens the key distribution scope and increases the overall exposure surface.

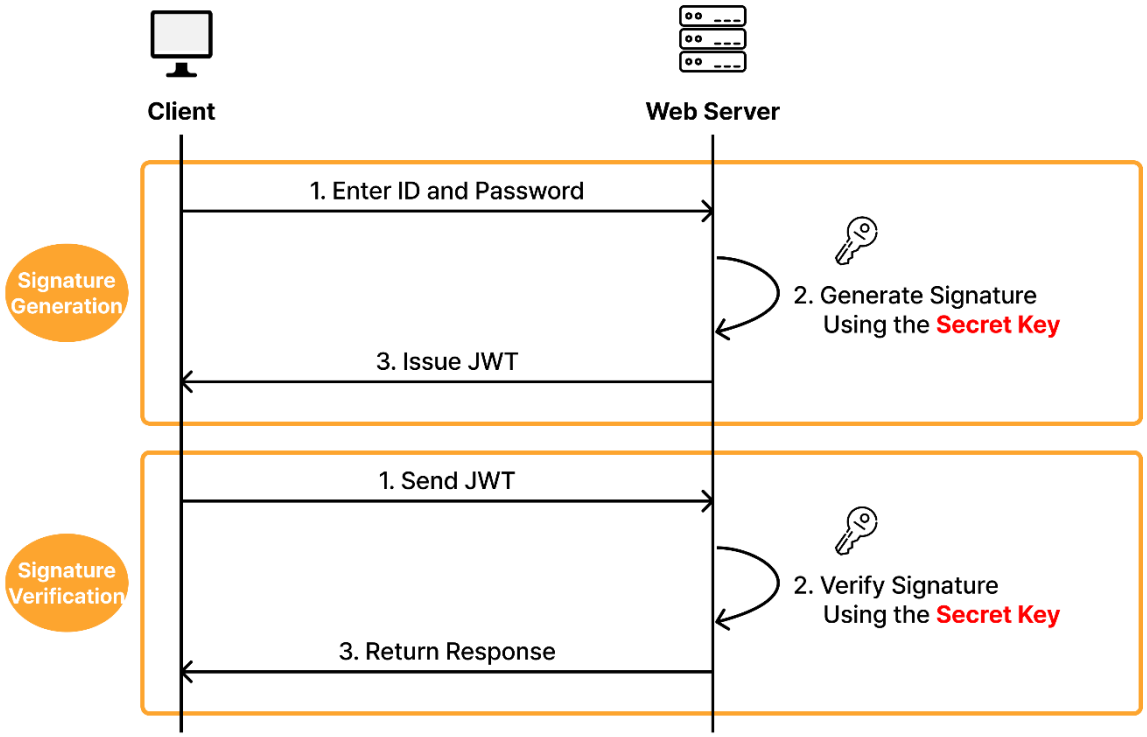


Figure 22. Symmetric-Key–Based Signature Generation and Verification Process

In this architecture, the web/API servers responsible for token validation must hold the secret key directly, meaning that the very surface where user requests arrive becomes a potential point of signing-key exposure. Therefore, if a server is compromised or the key is leaked, an attacker can generate valid signatures with the same secret key and forge JWTs.

To address this structural limitation, the asymmetric-key approach—RSA-SHA256 (RS256)—separates the private key used for token generation from the public key used for verification. The private key is managed only within a constrained environment, while only the public key required for token validation is distributed to each service server. As a result, even as the number of servers handling authentication requests grows, the authority to mint tokens remains restricted to the server that holds the private key.

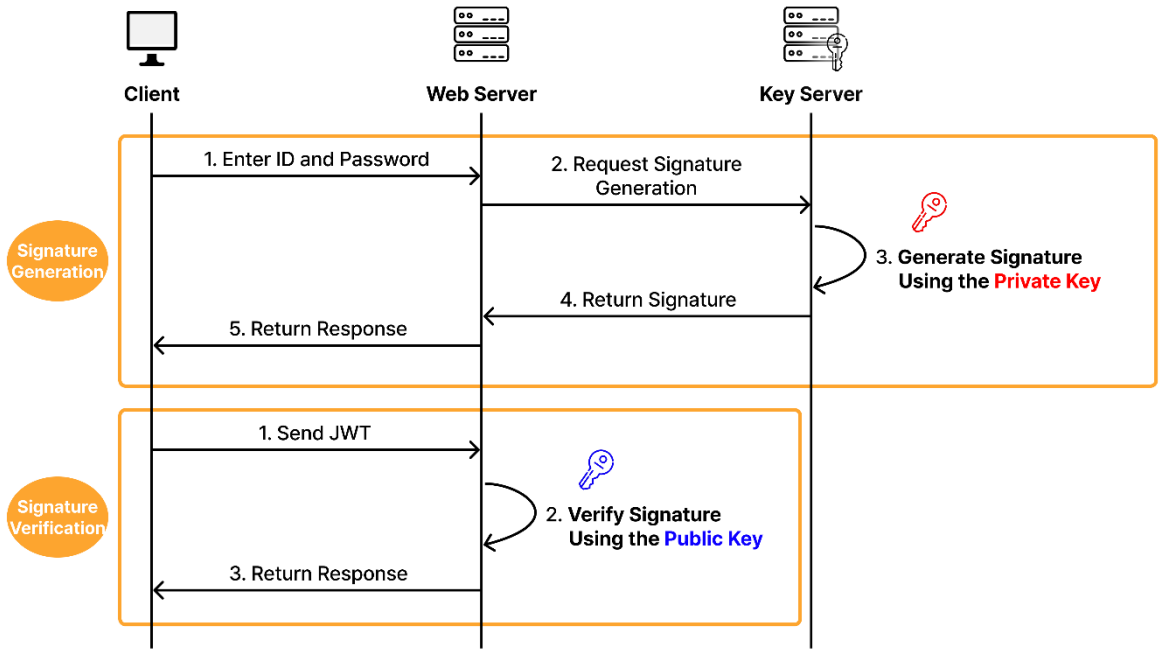


Figure 23. Asymmetric-Key–Based Signature Generation and Verification Process

Under this architecture, even if the public key distributed to verification servers is exposed, it cannot be used to generate new JWTs. This is because the public key is used solely for validation—i.e., to assess token authenticity—and cannot, by itself, produce a signature. As a result, the signing-key environment and the verification environment are separated, ensuring that key exposure on a verification server does not translate into token forgery.

Signature verification also has a direct impact on both performance and security regardless of service scale. In symmetric-key architectures, token issuance and verification are concentrated around the same secret-key domain, which can lead to performance degradation as traffic increases and can also amplify authentication risk in the event of compromise. By contrast, asymmetric-key architectures decouple signature generation from verification, allowing server load to be distributed while constraining token-minting authority. For these reasons, asymmetric-key signing is structurally recommended as the safer choice for JWT-based authentication.

The signing/verification separation described above aims to restrict the signing key to the smallest possible operational scope. In such a design, security devices such as HSMs (discussed later) can be employed as a means to protect the signing key.

• **Isolated Operation of the Signing Key**

In JWT environments, the core of signing-key protection hinges on how much the key is exposed to people and to the application runtime domain. If a signing key is embedded in environment variables, configuration files, deployment artifacts, or operational documentation, realistic factors—such as insider misuse, account takeover, or offboarding risk—can create viable paths to key exposure. Accordingly, the signing key should not be treated as a value that humans can readily view or copy; instead, an operational model is required in which the key is isolated and can be used only in a constrained manner when necessary.

To this end, signing-key operations should be designed along the following principles.

Design Principle	Description
Centralized management and minimal human involvement	Manage the signing key centrally rather than embedding it in code or server configuration, and minimize procedures in deployment and operations where humans directly handle the key material.
No local key storage	Have the application avoid storing the key material; instead, it should submit only the data to be signed, request the signing operation, and receive only the resulting signature.
Restricted and auditable key usage paths	Restrict the entities and pathways permitted to use the signing key to reduce exposure points, and record signing-request histories to enable detection of misuse or abuse.

Table 3. Operational Principles for Minimizing Signing-Key Exposure

A practical means of implementing these exposure-minimization principles in a technically straightforward way is an HSM (Hardware Security Module). An HSM is a hardware-based security device designed to protect sensitive keys used for operations such as JWT signature generation and verification, ensuring that neither humans nor applications can directly view the key material. Keys are generated, stored, and used exclusively within the HSM, while external systems—without ever learning the actual key value—request signing operations (e.g., signature generation) and receive only the resulting output.

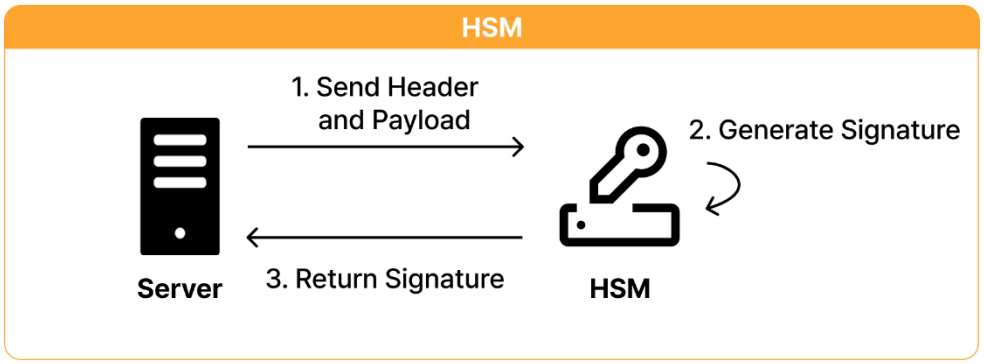


Figure 24. Signature Generation Flow When Using an HSM

Technically, an HSM can be used in symmetric-key designs for both token generation and verification. However, an architecture that integrates an HSM into every authentication request can impose operational overhead in terms of performance and scalability. In contrast, asymmetric-key designs allow signing and verification to be separated, enabling the HSM to be used only for protecting the signing key (private key) and performing signature generation. Therefore, in JWT-based authentication environments, it is recommended to assume an asymmetric-key design and employ the HSM specifically for signing operations.

HSMs are categorized by deployment model, such as embedded, network-attached, or cloud-based services. In inter-server authentication or large-scale service environments, network-attached or cloud HSMs—capable of centrally handling signing operations—are commonly used. By contrast, USB-form-factor HSMs are typically used for personal authentication purposes (e.g., public certificates) and are generally not suitable for protecting JWT signing keys.

Category	Description
Embedded HSM	A card-type device installed directly into an internal server slot (e.g., PCIe).
Network-attached HSM	A standalone dedicated appliance that processes cryptographic-operation requests over the network.
USB HSM	A portable USB form factor, primarily used for personal authentication and signing.
Chip-based HSM	Implements cryptographic-operation capabilities within a semiconductor chip (embedded in devices such as IoT).
Cloud HSM service	An HSM service provided by a cloud provider (e.g., AWS, Azure) in a virtualized environment.

Table 4. Types of HSMs

However, even if the signing key itself is protected, failing to control which systems are allowed to request signing operations can lead to damage comparable to an outright key compromise. To address this limitation, the next section examines mitigation measures centered on management mechanisms that control and audit signing authority.

2) Signing Authority Control

Next, this section proposes control measures that restrict signing-operation request privileges through policy and track their history.

- **KMS-Based Management of Signing Privileges**

Even if an HSM structurally prevents external exposure of the signing key, security incidents can still occur if the authority to request signing operations is not properly controlled. In other words, it is important not only to protect the key material itself, but also to govern who can sign, when they can sign, and for what purpose. This is the role of a KMS (Key Management System).

A KMS centrally manages cryptographic keys used across multiple applications and services, and enforces policy-driven control over the entire lifecycle—from key generation and storage to use, rotation, and decommissioning. Through this, administrators can identify which systems requested signing operations with which keys and at what times, and can granularly restrict signing-operation (key-usage) privileges at the service or role level. In addition, auditing logs and monitoring capabilities enable detection of anomalous signing requests and support timely response.

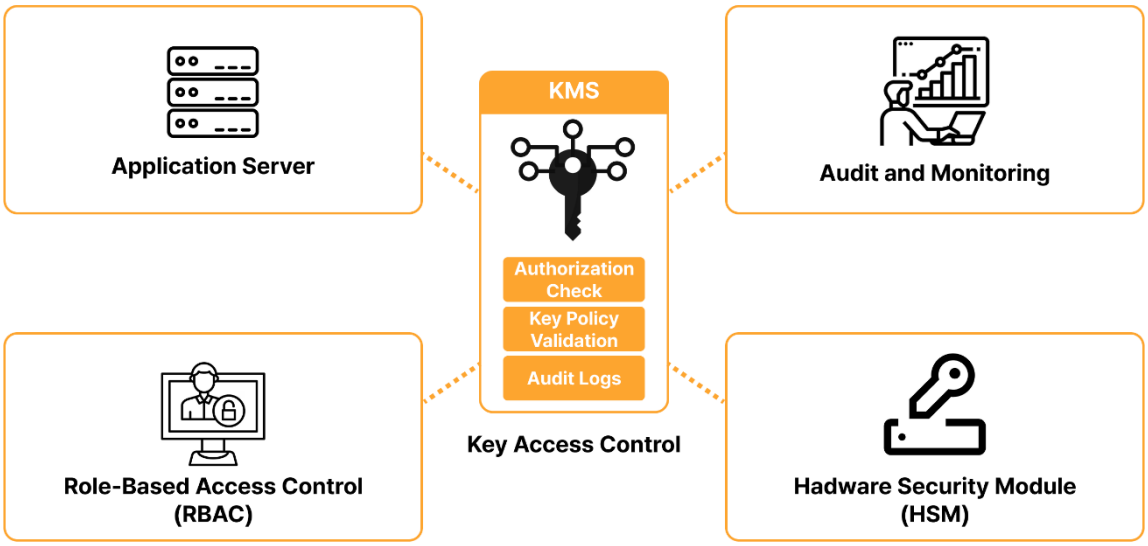


Figure 25. KMS-Based Centralized Key-Management Architecture

Major cloud platforms provide KMS as a managed service. Cloud KMS typically supports IAM-based access control, automatic rotation, and audit-log integration by default, and can be integrated with a cloud HSM when necessary to protect signing keys at the hardware level. In this model, the application performs signing requests via the KMS, while the actual signing key remains within the cloud provider's protected environment.

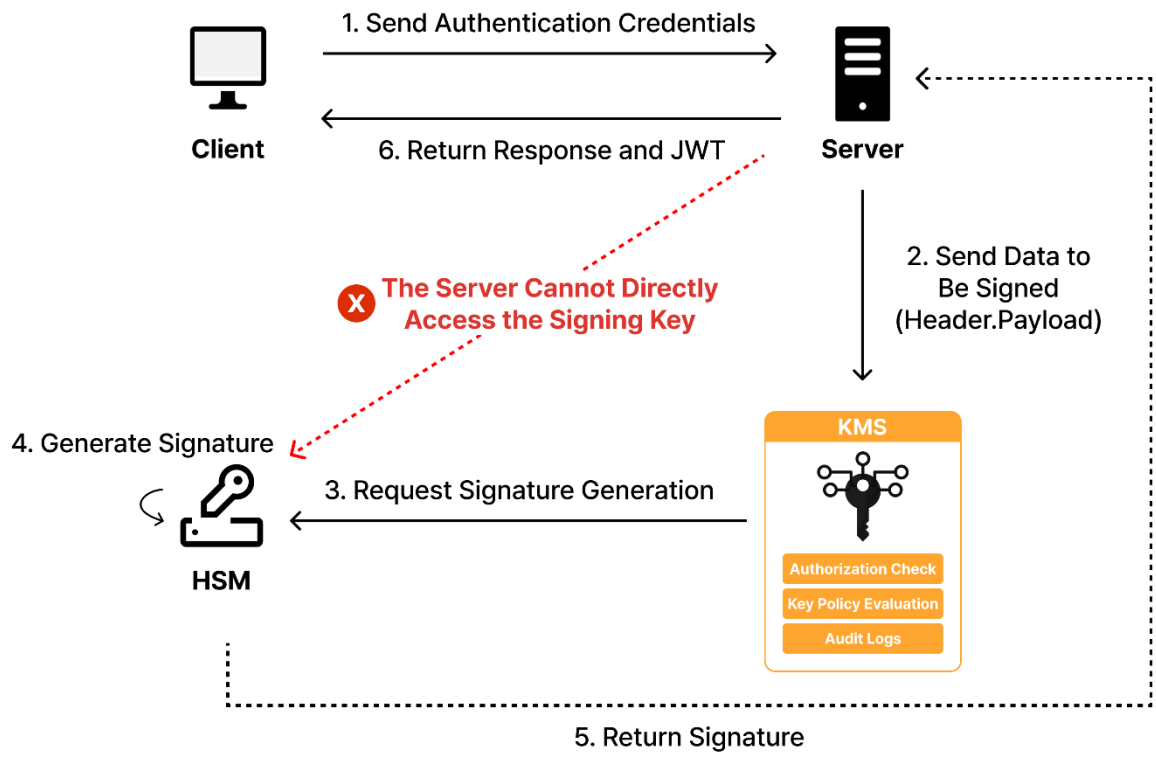


Figure 26. JWT Signature Generation Flow Using KMS and HSM

In contrast, organizations such as those in the financial sector, public institutions, or air-gapped environments—where the use of external cloud services is restricted or where internal security guidelines must be followed—must implement the same concept directly in an on-premises setting. Even in this case, the core of the mitigation remains establishing an architecture that centrally manages signing keys and controls both usage privileges and usage history.

While the implementation approach may vary depending on an organization's infrastructure and required security level, several elements should be considered as common essentials in on-premises environments: separation of signing-request privileges, logging and auditing of usage history, and periodic key rotation. In particular, key rotation avoids prolonged reuse of a single key, thereby limiting the exposure window if a key is compromised, and enables rapid containment of impact during incident response by replacing the key.

Building on this architecture, applying HSM and KMS together can effectively mitigate the risks associated with signing-key exposure. However, because security incidents cannot be eliminated entirely, the next section examines design-level compensating measures aimed at minimizing the spread of authentication impact under an assumed compromise scenario.

3) Limiting Impact Under an Assumed Signing-Key Compromise

Finally, assuming the possibility of signing-key exposure, this section proposes mitigating measures to constrain the blast radius by designing the sub claim appropriately and introducing additional verification for sensitive functions.

• Designing the sub Claim

The mitigation measures discussed earlier focus on protecting and managing the signing key, but in practice it is difficult to eliminate the possibility of key exposure entirely. Therefore, JWT security should consider not only key protection, but also design-level safeguards that limit how far damage can propagate if the signing key is compromised.

The sub claim in a JWT is a value used to identify which user the token represents, and after signature verification succeeds, the server identifies the user account based on the sub value. If the sub value is constructed as a sequential or semantically meaningful integer—such as an internal user ID—this can create a structural risk when the signing key is leaked. This is because, once an attacker can generate tokens with a valid signing key, they can repeatedly mint JWTs that impersonate other users by modifying the sub value.

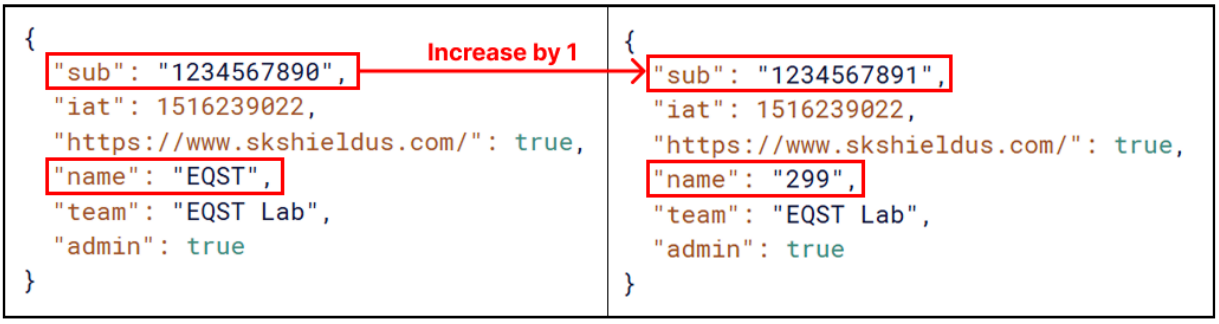


Figure 27. Example of the sub Claim

In such a design, the risk can extend beyond compromise of a single account to large-scale attacks that programmatically iterate through numerous accounts in sequence. In other words, how the sub value is designed becomes a determining factor in the potential blast radius if the signing key is exposed.

To mitigate this, it is advisable to use a non-semantic identifier for the sub claim—one that does not allow a user to be directly inferred even if exposed externally. Values such as UUIDs preserve user-identification functionality while making it difficult to guess or enumerate users based on continuity or range, thereby raising the cost and complexity of automated account-switching attacks.

Using non-semantic identifiers is not a fundamental solution that prevents signing-key compromise itself; rather, it is a practical compensating control that constrains attacker behavior and slows the pace at which impact can spread once compromise occurs. Since the JWT standard defines sub as a registered claim intended to identify the token's subject, it is necessary to design its value safely under the assumption of external exposure while preserving that role.

• Additional Verification for Sensitive Functions

Even if token forgery and account-scale abuse are mitigated through signing-key protection and careful sub claim design, a JWT-based authentication model cannot fully eliminate the inherent characteristic that requests presenting a valid token are treated as legitimately authenticated. Therefore, under the assumption that key leakage or token theft may occur, architectural compensations are needed so that possession of a single token does not grant unrestricted access to all functions.

In real-world service environments, it is common to require additional verification steps—such as re-entering a password—when accessing sensitive functions like personal data retrieval, account information changes, or payment/refund operations, independent of the user's logged-in state. This design reduces the impact of a single token compromise by deliberately constraining the trust scope of the authentication token.

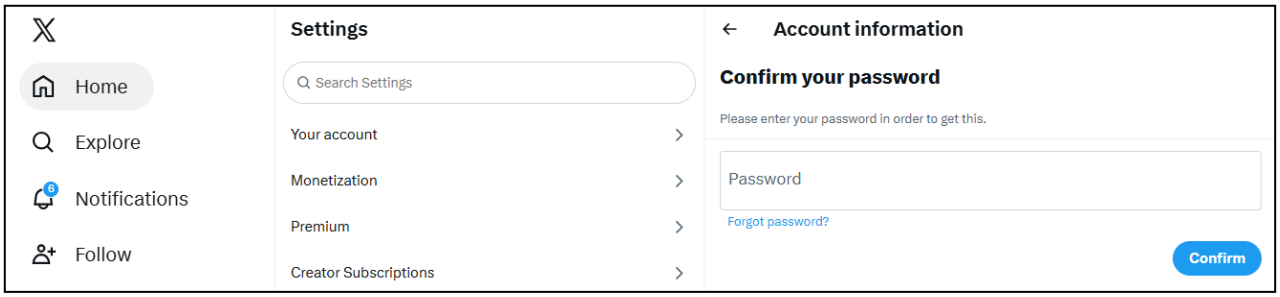


Figure 28. When Accessing the Personal-Information Page

Such additional verification is performed after JWT signature validation, and it can be designed so that—even when the token is valid—certain actions require the user to be re-verified or to complete a separate authentication step. This makes it harder for an attacker to directly abuse sensitive APIs or critical functions, even if they have obtained a valid JWT.

Ultimately, in JWT-based authentication environments, a practical compensating strategy is to avoid treating all functions with the same level of trust and instead apply differentiated authentication strength according to the criticality of the function. This serves as an additional defensive layer to limit overall service impact even under a worst-case scenario such as signing-key compromise.

■ Conclusion

In JWT environments, if the signing key is compromised, an attacker can generate and use tokens that are difficult to distinguish from those of legitimate users, and if detection is delayed, the impact can escalate rapidly. Accordingly, the core of JWT security lies not merely in the choice of signing algorithm, but in how the signing key is protected and governed.

In particular, an operational mindset that "it is acceptable for administrators or internal personnel to know the key" creates avenues for key exposure through various operational and access paths. Therefore, the signing key should not be treated as something that humans directly store or share; rather, it must be designed to be used only under constrained privileges, and only when necessary, within an isolated environment. When these principles are upheld, the risk of compromise propagating across the entire authentication system can be materially reduced—even under breach conditions.

■ References

• Microsoft Security (Storm-0558): https://www.microsoft.com/en-us/security/blog/2023/07/14/analysis-of-storm-0558-techniques-for-unauthorized-email-access/
• Microsoft (Golden SAML): https://www.microsoft.com/en-us/msrc/blog/2020/12/customer-guidance-on-recent-nation-state-cyber-attacks
• MITRE ATT&CK (Golden SAML): https://attack.mitre.org/techniques/T1606/002/
• RFC7519: https://datatracker.ietf.org/doc/html/rfc7519
• JWT
-https://www.jwt.io/introduction#what-is-json-web-token-structure
-https://cheatsheetseries.owasp.org/cheatsheets/JSON_Web_Token_for_Java_Cheat_Sheet.html
-https://learn.microsoft.com/en-us/entra/identity-platform/id-token-claims-reference
• AWS KMS: https://docs.aws.amazon.com/kms/latest/developerguide/overview.html
• AWS CloudHSM: https://docs.aws.amazon.com/cloudhsm/latest/userguide/introduction.html
• Azure key-vault: https://learn.microsoft.com/en-us/azure/key-vault/general/basic-concepts
• Coupang:
-https://pages.coupang.com/f/160047
-https://news.coupang.com/archives/58857/
-https://v.daum.net/v/20251202212010142

EQST INSIGHT

2026.01

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