ARMIS LABS INSIGHTS

Q3, 2025

Catch Attackers Before They Strike

Early Warning Insights for **Software Supply Chain Attacks**

04 Executive Summary

05 Introduction

07 How AI Created Slopsquatting Risks

09 With Vibe Coding, It Got Worse

13 Top 25 Software Supply Chain Attacks

18 The Importance of Early Warning Threat Intelligence

21 Indicators of Action (IoAs)

24 Mitigation Strategies

26 Armis Vulnerability Intelligence Database

28 About Armis Centrix™

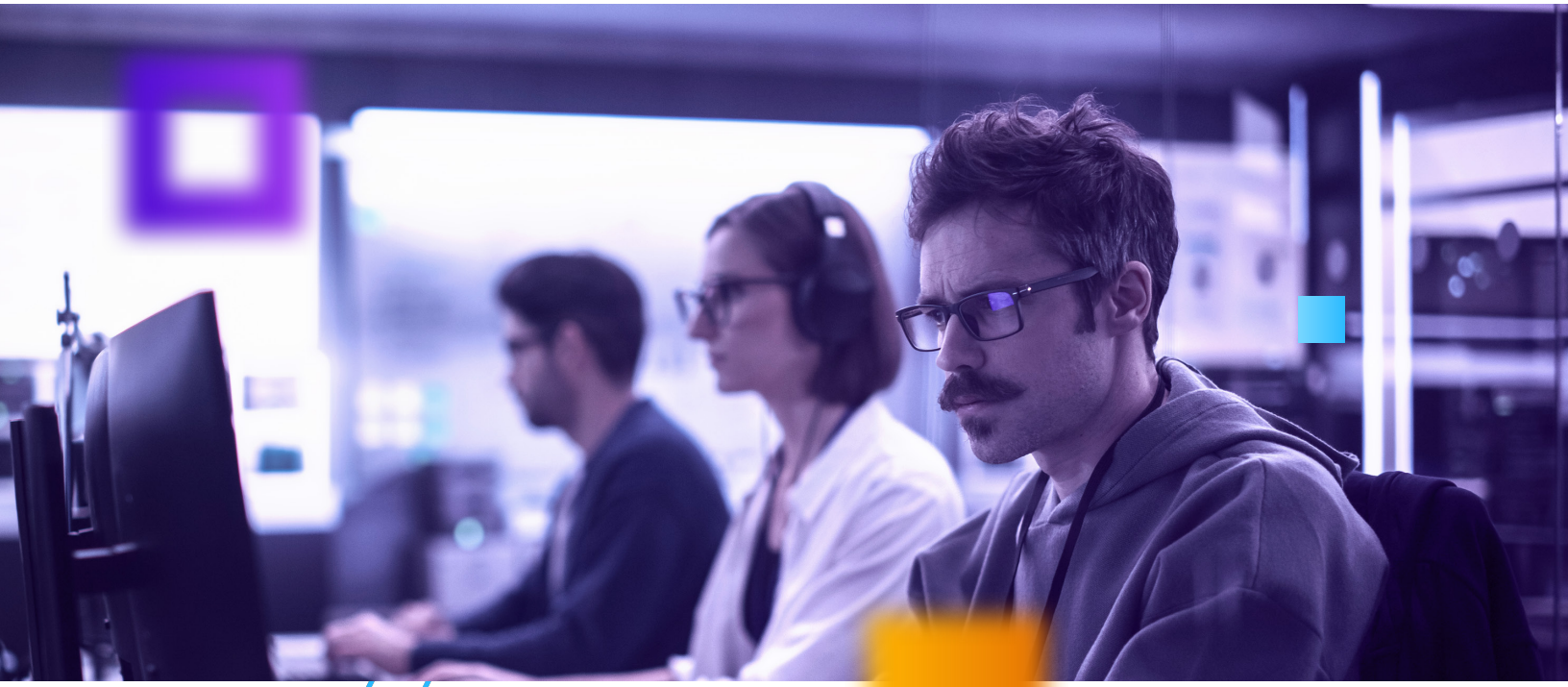
30 About Armis Labs

What To Expect From This Report?

- For the C-suite, this report offers an overview of software supply chain risks, widely exploited open-source software libraries, and emerging cyber threat trends.
- For (application) security teams, it offers an evidence-based and actionable Top 25 of software supply chain attacks, and an overview of Indicators of Action (IoAs) to address and monitor.

What Makes The Armis Labs Top 25 Unique?

Our research team leverages a combination of human intelligence, dynamic deception technology and AI/ML that detects threat actor behaviors before actual exploitation, including flaws in threat actors operations and context of their discussions. This evidence-based approach leads to different application security risks than similar rankings like the OWASP Top 10 or the CWE™ Top 25.



Executive Summary

Open-source software libraries form the backbone of modern application development, offering immense benefits in terms of collaboration, innovation, and accelerated development cycles. Their inherent transparency and community-driven development foster a rapid pace of improvement and broad accessibility.

But what happens when the very app software libraries we rely on are compromised before they even reach us? This is the alarming reality of software supply chain attacks. These attacks exploit vulnerabilities in the development, distribution, or implementation of software, causing widespread disruption and raising serious security concerns.

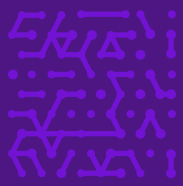
Understanding the nature of these attacks and their growing prevalence is key to protecting both organizations and individuals. This report dives into the world of software supply chain attacks, exploring how they work, what makes them so dangerous, and why addressing these risks has become imperative.

Key Takeaways

- **Slopsquatting** emerges as a threat and has the potential to **compromise apps (including Line of Business), microservices, infrastructure using these libraries, supply chains and much more** across an organization's ecosystem.

- An **overreliance on AI-generated code suggestions** without rigorous human oversight can lead to **the integration of fragile and vulnerable code** into a codebase.

- By **focusing on Indicators of Actions (IoAs) of an adversary** rather than just the static artifacts of a compromise, organizations can **build a more resilient and proactive security posture**. This shift enables early detection, rapid response, and ultimately, significantly reduces the impact of sophisticated cyber threats, moving the advantage from the attacker back to the defender.

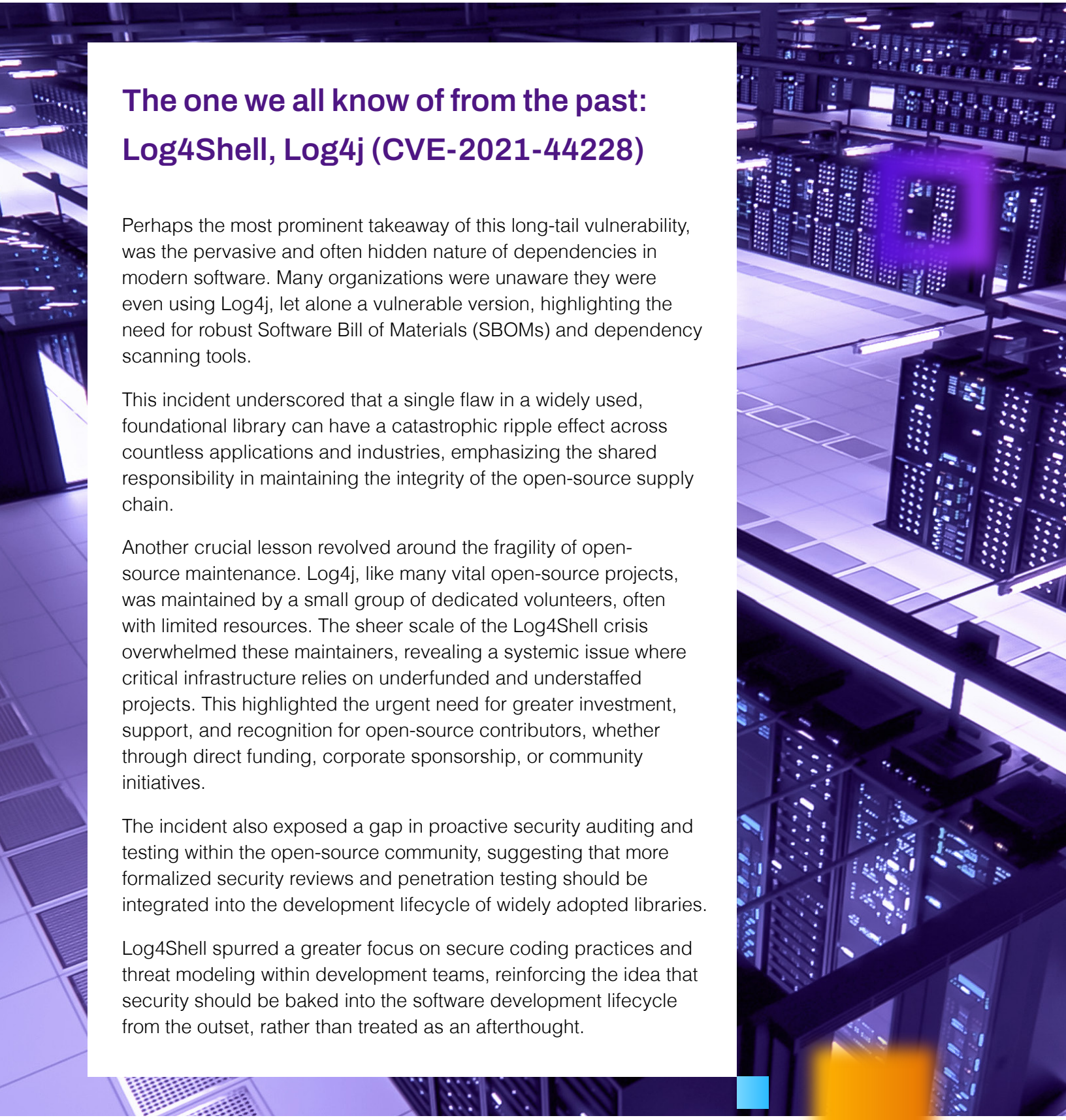


Introduction

The omnipresent adoption of open-source libraries has fundamentally reshaped the landscape of software development. They offer exceptional advantages in terms of rapid development, cost efficiency, community support, and access to cutting-edge technologies. From operating systems and web frameworks to specialized algorithms and machine learning tools, open-source components are at the heart of countless modern applications. However, this widespread integration also introduces a critical, often underestimated, vulnerability: the security of these very libraries.

One of the primary reasons for this importance lies in the sheer volume and complexity of dependencies. A typical modern application might rely on dozens, even hundreds, of open-source libraries, each with its own set of sub-dependencies. This creates a vast and intricate supply chain. A vulnerability in just one seemingly innocuous, deeply nested library can ripple upwards, exposing the entire application to potential exploitation. This “supply chain attack” vector has become increasingly prevalent, with malicious actors specifically targeting widely used open-source components to gain access to a multitude of downstream systems. If a flaw exists in a popular library, it essentially becomes a skeleton key, potentially unlocking thousands or even millions of applications simultaneously.

Beyond direct exploitation, insecure open-source libraries can introduce a range of other serious issues. They can lead to data breaches, compromising sensitive user information or intellectual property. They can enable denial-of-service attacks, crippling critical services and causing significant financial losses. They can also serve as entry points for more sophisticated attacks, allowing adversaries to establish persistent footholds within systems or networks. The reputational damage alone from a security incident stemming from an unpatched open-source vulnerability can be immense, eroding user trust and impacting business operations.



The one we all know of from the past: Log4Shell, Log4j (CVE-2021-44228)

Perhaps the most prominent takeaway of this long-tail vulnerability, was the pervasive and often hidden nature of dependencies in modern software. Many organizations were unaware they were even using Log4j, let alone a vulnerable version, highlighting the need for robust Software Bill of Materials (SBOMs) and dependency scanning tools.

This incident underscored that a single flaw in a widely used, foundational library can have a catastrophic ripple effect across countless applications and industries, emphasizing the shared responsibility in maintaining the integrity of the open-source supply chain.

Another crucial lesson revolved around the fragility of open-source maintenance. Log4j, like many vital open-source projects, was maintained by a small group of dedicated volunteers, often with limited resources. The sheer scale of the Log4Shell crisis overwhelmed these maintainers, revealing a systemic issue where critical infrastructure relies on underfunded and understaffed projects. This highlighted the urgent need for greater investment, support, and recognition for open-source contributors, whether through direct funding, corporate sponsorship, or community initiatives.

The incident also exposed a gap in proactive security auditing and testing within the open-source community, suggesting that more formalized security reviews and penetration testing should be integrated into the development lifecycle of widely adopted libraries.

Log4Shell spurred a greater focus on secure coding practices and threat modeling within development teams, reinforcing the idea that security should be baked into the software development lifecycle from the outset, rather than treated as an afterthought.

How AI Created Slopsquatting Risks

Historically, compromising a software supply chain was a high-effort undertaking, demanding sophisticated infiltration of trusted vendors, tampering with build environments, and covertly injecting malicious code over months to ensure its propagation through official releases.

Today, the threat landscape has shifted dramatically with the emergence of slopsquatting. This automated attack vector exploits the tendency of AI coding assistants to "hallucinate" package names. Attackers preemptively register these nonexistent package names, allowing them to trick developers into unwittingly installing malicious code.

While AI coding assistants offer significant promise for enhancing developer productivity, they are far from infallible and can inadvertently introduce substantial vulnerabilities into software. Research indicates that as many as half of the code snippets generated by current AI tools contain exploitable security flaws, including buffer overflows, SQL injections, and insecure default configurations. Furthermore, these large language models (LLMs) often exhibit unpredictable behavior. They can "hallucinate" nonexistent functions, omit critical error handling, or provide inaccurate information regarding API behaviors.

When a developer copies and pastes AI-suggested code containing these "phantom" dependencies, they bypass traditional supply chain security measures and directly introduce malware or backdoors into their projects, all without the need for a complex vendor compromise. A single slopsquatting incident can propagate unchecked, potentially compromising numerous microservices and shared libraries across an organization's ecosystem.

What Slopsquatting is:

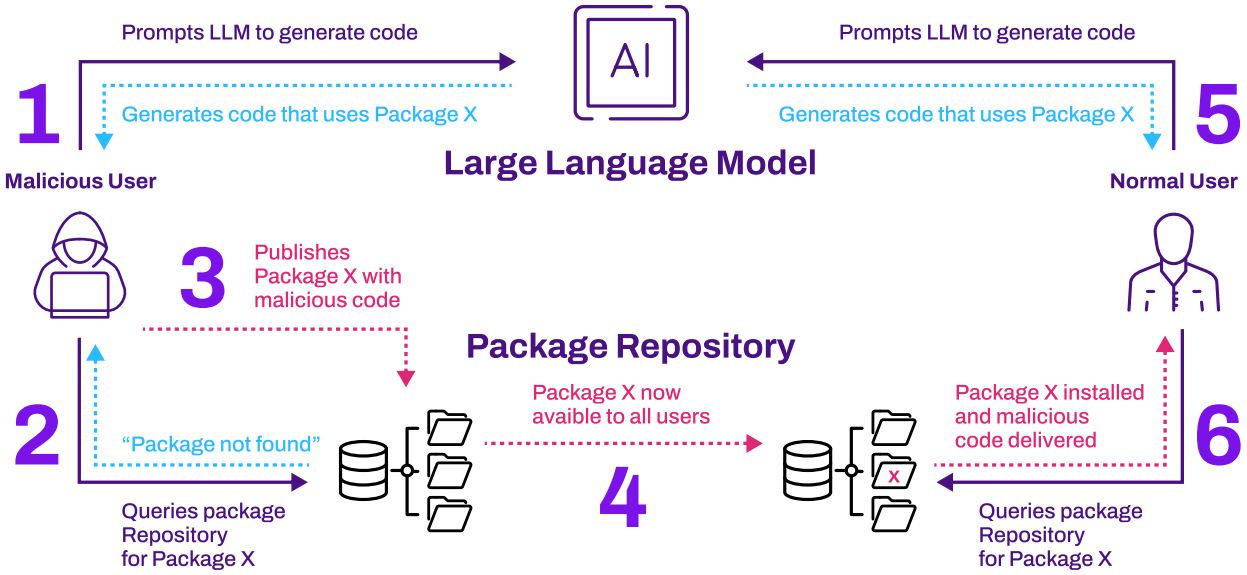
- **AI-Induced Hallucinations:** AI coding tools (e.g. ChatGPT, GitHub Copilot, CodeLlama) sometimes suggest non-existent libraries or packages that "sound" plausible.
- **Attacker Pre-Registration:** Malicious actors anticipate these hallucinations, register the fake package names on public repositories (PyPI, npm, etc.), and embed trojanized code.
- **Automated Exploitation:** A developer's CI/CD pipeline or local build process can automatically fetch and install these packages, resulting in silent compromise (trendmicro.com, gmv.com).

How AI Created Slopsquatting Risks

Malware Delivery

A real-world example is the slopsquatting package ccxt-mexc-futures masqueraded as an add-on for the popular ccxt crypto library. The package was designed to reroute cryptocurrency trading orders placed on the MEXC exchange to a malicious server to steal tokens. Although the package has been removed from PyPI, it was downloaded over 1,065 times, according to pepy.tech.

The package falsely claimed to extend the popular Python library ccxt, which is commonly used for cryptocurrency trading. On closer inspection, it was revealed that the package maliciously modified specific APIs associated with MEXC's trading interface, enabling attackers to execute arbitrary code. It diverted requests to a fake domain impersonating MEXC, ultimately stealing API keys, sensitive details, and cryptocurrency tokens.



Source: The Hacker News

Victims who installed the package were advised to revoke potentially compromised tokens and remove the package immediately. The discovery highlights the rising trend of supply chain attacks across platforms like npm, PyPI, and Go.

Supply-Chain Compromise

Once integrated, malicious packages can exfiltrate secrets, inject further payloads into your code, or pivot laterally within your environment.

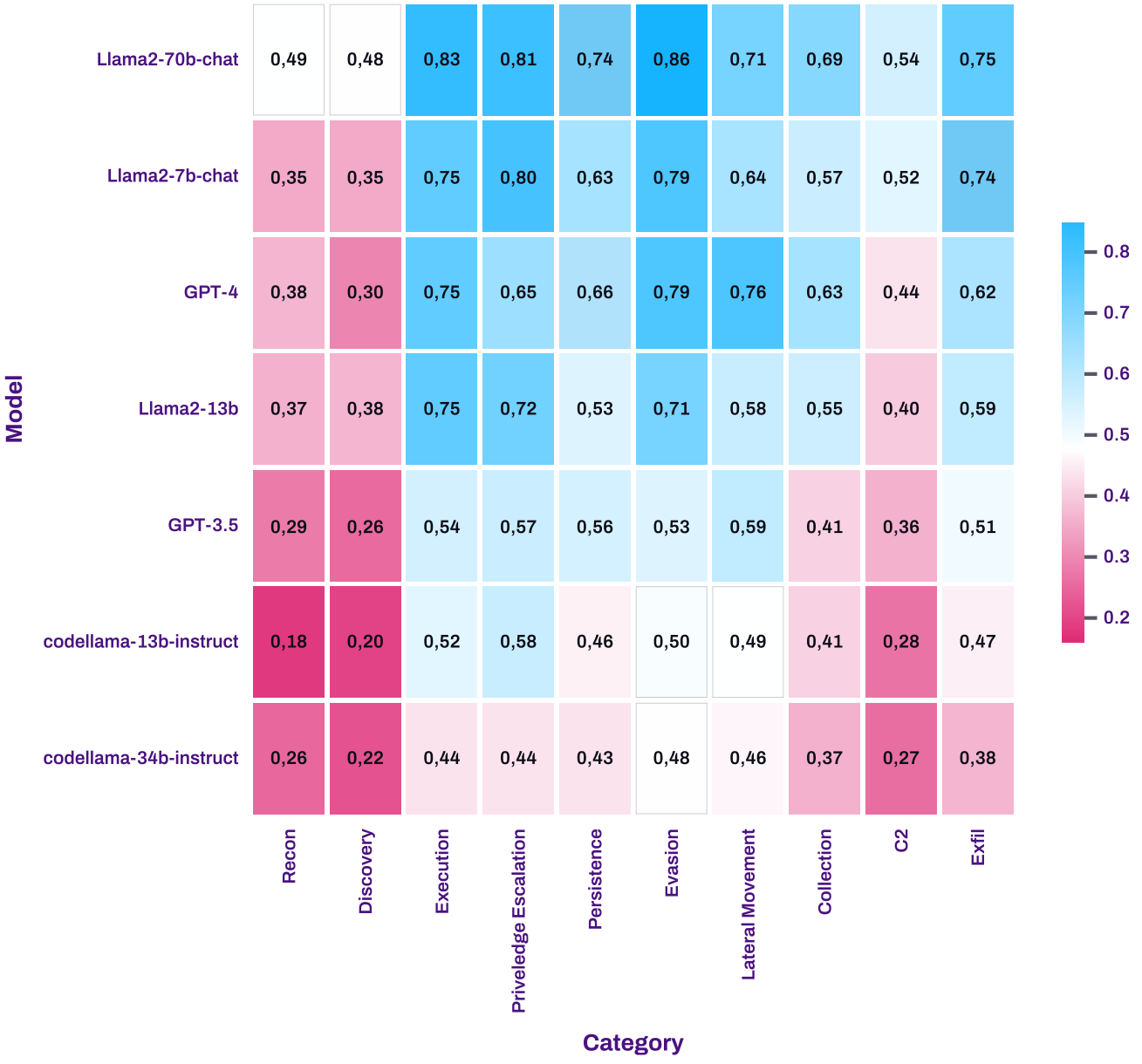
Widespread Reach

Automated dependency resolution and "lock-file" upgrades can cause slopsquatting malware to propagate across multiple projects and organizations without human review.

With Vibe Coding, It Got Worse

The increasing organizational demand for speed and agility has led to the widespread adoption of “vibe coding”—a practice characterized by rapid prototyping and code deployment with minimal peer review. This accelerated approach has shown that 40% of the time, the LLM-generated code suggestions exhibited vulnerabilities. The table below provides an overview of vulnerabilities introduced by various LLM models across the MITRE ATT&CK stages.

LLM non-compliance rate in helping with cyberattacks (higher is better)



Source: Purple Llama CyberSecEval: A benchmark for evaluating the cybersecurity risks of large language models

Other findings across the industry using Vibe Coding with Large LLMs (GPT, DeepSeek, Gemini, etc) include:

1. Authentication Failures Enable Account Compromise

No Brute-Force Throttling (except Gemini)**:** Without account lockout, attackers can rapidly automate thousands of login attempts, increasing the odds of credential stuffing and stolen accounts.

Missing CAPTCHA (all models)**:** Bots can mount large-scale automated attacks unimpeded, leading to credential harvesting and service abuse.

Lack of MFA (all models)**:** Single-factor logins leave every account one leaked password away from takeover, eroding user trust and exposing sensitive data.

Weak Password Policies (only Grok enforces complexity)**:** Simple passwords accelerate brute-force success, making user and admin accounts easy targets.

2. Session Management Flaws Risk Session Hijacking

Inconsistent Secure-Cookie Flags: Models that omit `Secure` and `HttpOnly` flags risk cookie theft over unsecured channels, allowing attackers to impersonate legitimate users.

Session Fixation Gaps (Claude)**:** Without regenerating session IDs on login, attackers can force a known session ID and later take over an authenticated session.

No Idle-Timeout Enforcement: Long-lived sessions give attackers extended windows to misuse stolen session tokens.

3. Input-Validation & Injection knesses Lead to Data Theft

Parameterized Queries Only (good)**:** SQL injection is largely mitigated—one bright spot.

Persistent XSS (DeepSeek, Gemini): Unsanitized input fields let attackers inject scripts, compromising user browsers and potentially spreading malware.

Inadequate CSRF Defenses (only Claude protected)**:** Without CSRF tokens, attackers can trick logged-in users into executing unwanted actions.

Open CORS Policies (all models)**:** Permissive cross-origin settings allow malicious sites to interact with APIs and steal data.

4. Missing HTTP Security Headers Expose Web Interfaces

No Content Security Policy: Attackers can load malicious scripts or resources from untrusted domains.

Absent Clickjacking Protections: UI frames can be overlaid by attacker pages to trick users into unintended actions.

No HSTS Enforced: Users remain vulnerable to SSL stripping attacks, exposing credentials on insecure networks.

5. Error-Handling & Logging Shortcomings Aid Reconnaissance

Verbose Error Messages (Gemini)**:** Revealing whether a username exists or password complexity rules helps attackers fine-tune their brute-force attacks.

Sparse Failed-Login Logging (only Gemini & Grok)**:** Without comprehensive logging, repeated attack patterns go unnoticed, delaying incident response.

No Anomaly Detection: Lack of alerts on unusual login spikes or geographic anomalies lets large-scale credential stuffing campaigns fly under the radar.

Overall Risk Assessment

All tested LLM stacks exhibit “very high” to “extreme” security risks—particularly Claude and DeepSeek—creating multiple attack vectors for real-world breaches. No platform currently delivers an end-to-end security posture aligned with OWASP Top 10 or NIST standards.

Key Recommendations to Mitigate Impact

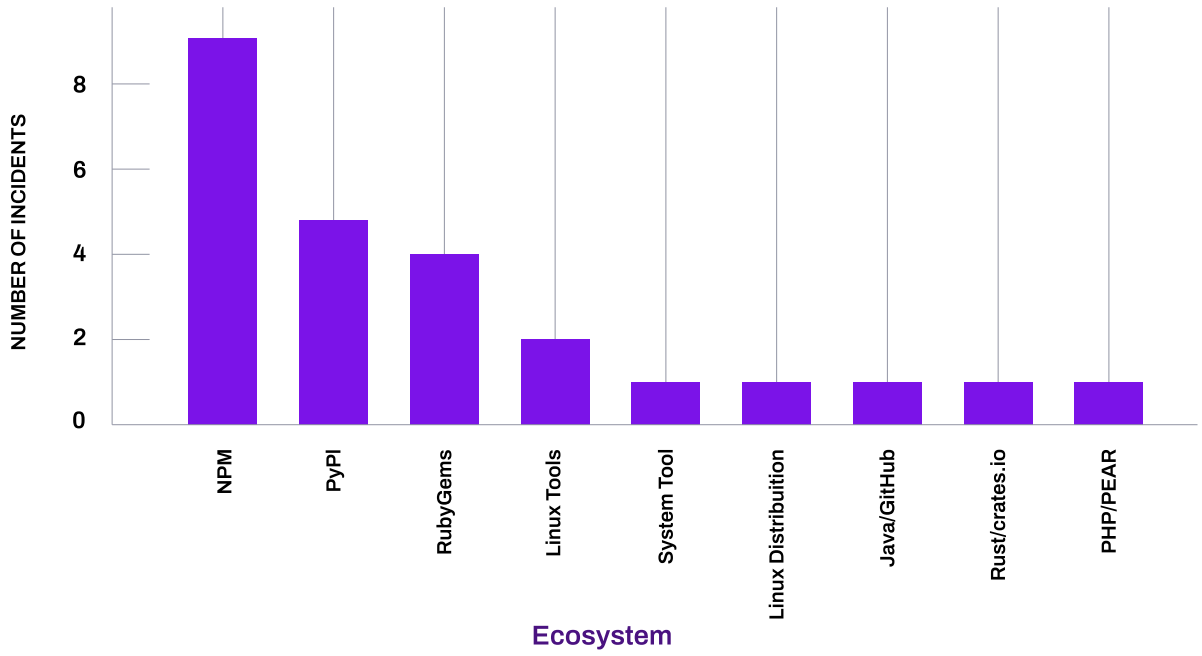
1. **Embed Security in Prompts:** Require LLMs to generate code with built-in protections (lockouts, MFA stubs, input sanitization).
2. **Automated Security Testing:** Integrate generated code into CI pipelines using OWASP ZAP, SAST tools, and fuzzers before deployment.
3. **Mandatory Human Review:** Enforce security expert sign-off on all LLM-produced modules.
4. **Secure-By-Default LLMs:** Encourage model vendors to bake in CSP headers, secure cookies, and CSRF tokens—even when not explicitly prompted.

While LLMs enhance developer productivity, their generated code contains significant security vulnerabilities that could lead to breaches in real-world applications. Exacerbating this risk, recent widespread layoffs across the tech industry have significantly reduced engineering team sizes. This reduction in workforce translates to less time for critical code reviews, a depletion of institutional knowledge, and an increased reliance on AI tools or quick solutions. An overreliance on AI-generated code suggestions without rigorous human oversight will lead to the integration of fragile and vulnerable code into a codebase, even before considering external threats such as slopsquatting.

Top 25 Software Supply Chain Attacks

Open-source libraries are targeted in cyberattacks due to their widespread use and the potential for widespread impact when vulnerabilities are exploited. Armis Labs have found that some common attack vectors include supply chain attacks, where malicious code is injected into a library and then propagated to users of that library, and dependency confusion, where malicious packages are created that mimic legitimate ones.

Number of Backdoor Incidents by Ecosystem



Below is a table[1] that highlights the top 25 software supply chain attacks, their methods, and mitigation strategies.

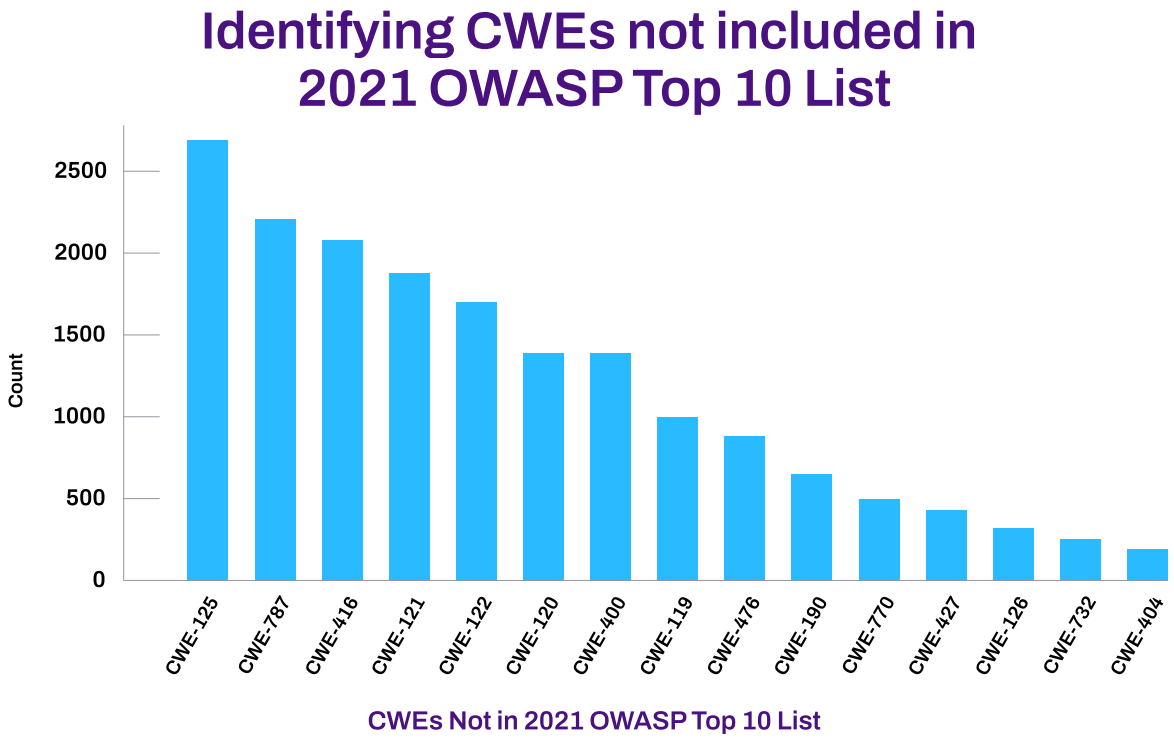
Ecosystem	Attack	Year(s)	Versions Affected	Attack Method	Mitigation
JavaScript (npm)	event-stream	2018	v3.3.6	**Wallet exfiltration via dependency**	**Removed malicious dependency.**
	eslint-scope	2018	v3.7.2, 5.0.2	**Post-install Pastebin exfiltration.**	**Token revocation, secure versions.**

[1]Source: Armis Labs

Ecosystem	Attack	Year(s)	Versions Affected	Attack Method	Mitigation
	UAParser.js	2021	v0.7.29, 0.8.0, 1.0.0	**Preinstall script loaded trojans.**	**Patched releases.**
	coa	2021	v2.0.3, 2.0.4, 2.1.1	**Credential stealer, miner.**	**Reverted to safe version.**
	rc	2021	v1.2.9, 1.3.9, 2.3.9	**Same as coa.**	**Malicious versions removed.**
	xrpl.js	2025	v4.2.1–4.2.4, 2.14.2	**Private key theft via backdoored publish.**	**Update to 4.2.5, 2.14.3.**
	Solana web3.js	2024	v1.95.6, 1.95.7	**Wallet key exfiltration via backdoored function.**	**Patched release.**
	node-ipc	2022	v10.1.1, 10.1.2	**Geo-targeted file wiper (protestware).**	**Versions removed, project forked.**
	colors.js/ faker.js	2022	v1.4.1, 1.4.2, 6.6.6	**Infinite loop (protestware).**	**Forks replaced libraries.**
Python (PyPI)	SSH Decorator	2018	Multiple	**SSH credential exfiltration.**	**Yanked, new package.**
	jellyfish / python3-dateutil	2019	N/A	**SSH/GPG key theft (typosquatting).**	**Removed from PyPI.**
	fastapi-toolkit	2022	N/A	**RCE via malicious payloads.**	**Removed.**
	requests-darwin-lite	2024	All versions	**Backdoor in PNG.**	**Yanked.**
	torchtriton	2022	N/A	**DNS-based data exfiltration.**	**Name seized, users warned.**

Ecosystem	Attack	Year(s)	Versions Affected	Attack Method	Mitigation
	torchtriton	2022	N/A	**DNS-based data exfiltration.**	**Name seized, users warned.**
Ruby (RubyGems)	bootstrap-sass	2019	v3.2.0.3	**RCE via cookie eval.**	**Yanked, replaced with 3.2.0.4.**
	strong_password	2019	v0.0.7	**Eval via cookie, exfiltration.**	**Yanked, revert to 0.0.6.**
	rest-client	2019	v1.6.10–1.6.13	**Pastebin fetcher, credential theft.**	**Removed, post-mortem.**
	misc Ruby gems	2019	10+ packages, Multiple	**Miner/credential theft.**	**Yanked from RubyGems.**
	XZ Utils	2024	v5.6.0, 5.6.1	**SSH bypass via liblzma hook.**	**Downgrade to 5.4.x, global audits.**
Linux Tools	Socat	2016	Multiple	**Non-prime DH parameter.**	**Switched to strong prime.**
	Webmin	2018–2019	v1.890–1.920	**RCE via altered update file.**	**Clean rebuild, community alert.**
	Gentoo GitHub	2018	N/A	**`rm -rf /` payload in ebuild scripts.**	**2FA enforced, repo lockdown.**
Java (GitHub)	Octopus Scanner	2020	N/A	**Injected JAR into NetBeans builds.**	**GitHub SIRT cleaned repos.**
Rust	liblzma-sys	2024	v0.3.2	**Packaged XZ test files (not executed).**	**Yanked, fixed in 0.3.3.**
PHP	PEAR Installer	2019	go-pear.phar	**Backdoored archive with RCE.**	**Rebuilt from GitHub, hash check.**

Armis Labs uses a combination of human intelligence, dynamic deception technologies and AI/ML that detects threat actor behaviors before actual exploitation, including flaws in threat actors operations and context of their discussions. This evidence-based approach leads to different top 25 application security risks than similar rankings like the OWASP Top 10:



Top 25 - Actions To Take

1 Provenance Tracking with Software Bills of Materials (SBOMs)

Generate and cryptographically sign SBOMs for every build, ensuring each dependency's origin and version are auditable. Determine through installation if any packages are directly used or indirectly used, for example packages.json and packages-lock.json. Ensure versions are specifically set.

2 Application Security Testing

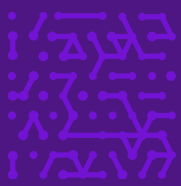
Leverage Static application security testing (SAST) and dynamic application security testing (DAST) to find vulnerabilities that could leave your organization's applications susceptible to attack.

3 Assess Business Impact

Which specific applications, features, or parts of your system are impacted?

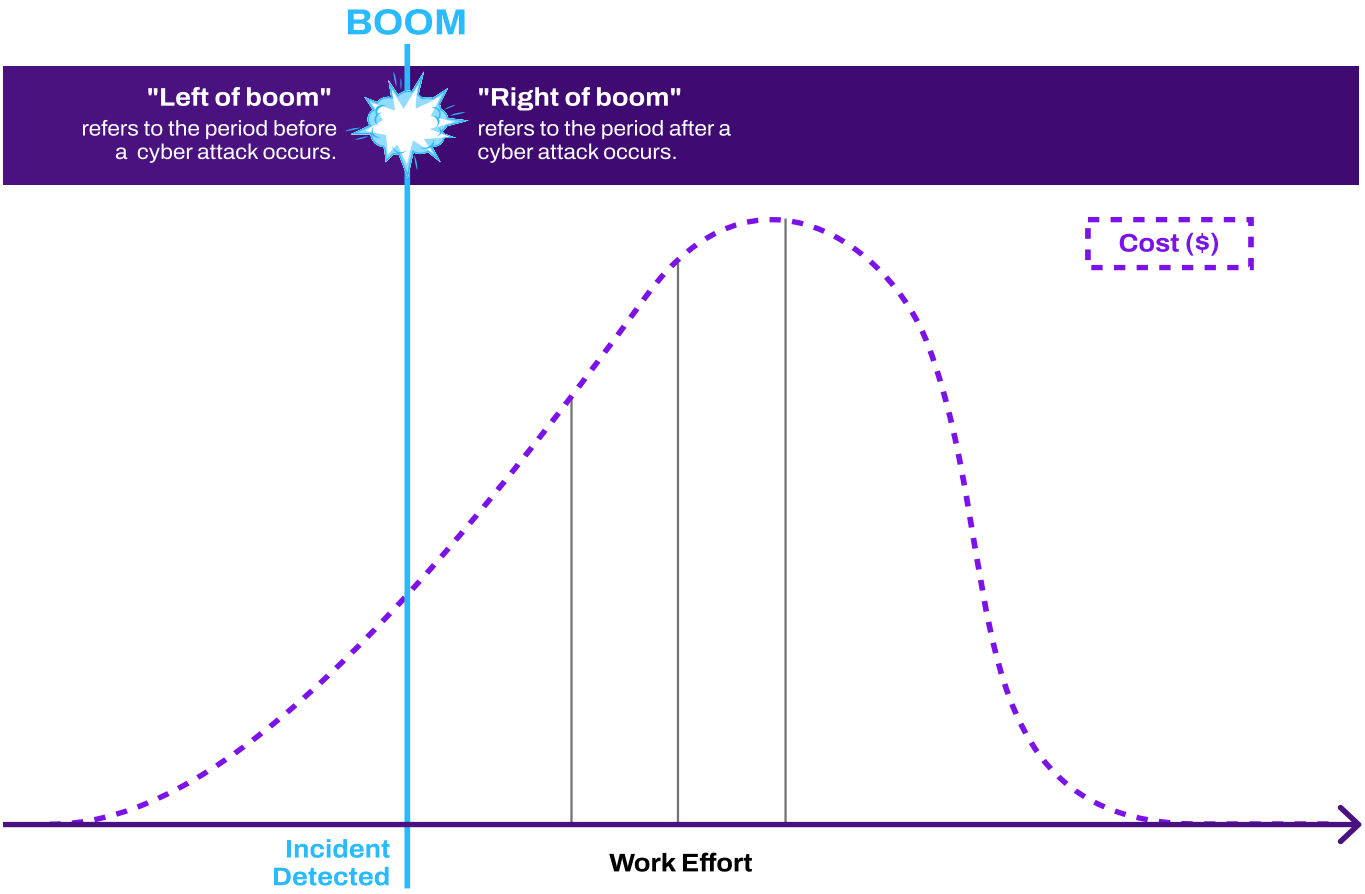
4 Remediate and Mitigate

Determine who is (most likely) responsible for the asset and the remediation. Leverage existing workflows and tools, and include clear, actionable remediation guidance.



The Importance of Early Warning Threat Intelligence

Traditional security solutions operate “right of boom”, or after the attack has occurred. Armis redefines the de-facto standard approach to security by taking action “left of boom”, or before the attack is launched. It is a foundational element to ensure comprehensive cyber exposure management and security for the attack that never happened.



Armis Labs provides its customers with threat intelligence into software libraries that threat actors are exploiting in the wild or are about to weaponize, allowing organizations to understand their impact and take preemptive action.

Below are three recent examples where Armis Early Warning Alerts enabled organizations to stay ahead of evolving cyber risk and protect their critical assets with confidence.

Early Warning Name	Gravity Forms
Description	Gravity Forms
Impact	Between July 9–10, 2025, malicious code was injected into **Gravity Forms plugin** versions **2.9.1.1** and **2.9.12** on the official download site. An unauthorized actor embedded **gravityforms/common**.php containing calls to a rogue domain **gravityapi.org**
Early Warning Details	Intelligence processes driven by deception technology identified the backdoor and call out to the bad domain.
Mitigation	Gravity Forms released version **2.9.13** on **July 11** and removed backdoors. Users urged to update immediately if they installed manually during that window. Gravity Forms rotated download-site keys, audited admin accounts and infrastructure, and decommissioned **gravityapi.org**. Site admins should restore clean backups, audit user accounts, delete malicious versions, and block bad domains/IPs.
Description	Still not published on CISA KEV.

Early Warning Name	GithHub Actions
Description	tj-actions/changed-files
Impact	From **Nov 2024** to **Mar 2025**, a multi-stage attack compromised multiple GitHub Actions. Attacker first infiltrated **reviewdog/action-setup@v1** by replacing its tag to point to malicious code. That code executed in dependent workflows. From there, a leaked PAT was used to gain write access to **tj-actions/changed-files**, where: A malicious commit was pushed into **index.js**, disguised as from **renovate[bot]**, dumping memory-secrets to logs. The commit was merged via a bot PR flow. All version tags (e.g. **v39**, **v47**) were force-retagged to point to this malicious commit. Any workflow using **tj-actions/changed-files@<tag>** (now malicious) logged secrets from runner memory. Affected **23,000 repos**, depending on whether memory-scanning succeeded.
Early Warning Details	Intelligence process that identifies malicious changes in open source libraries. It detected tj-actions/changed-files action originated in the compromise of a repository belonging to another GitHub organization: reviewdog/action-setup.

Early Warning Name	GitHub Actions
Mitigation	**Sign commits & protect branches**—prevent unauthorized merges/tags. **Pin actions to commit SHA**, not mutable tags. **Minimize PAT use** and scope & rotate frequently. **Avoid** pull_request_target, which runs workflows from forks with secret access. Tags can be forced to malicious forks—**monitor tag movements**. After detection: rotate tokens, scrub logs, audit PAT exposures.
Description	Still not published on CISA KEV.

Early Warning Name	NPM Install
Description	UA-Parser-JS MalwareFormsv
Impact	Any CI/CD pipeline, developer machine, or production system installing the compromised versions became infected. Developers using **npm install** on affected versions could unknowingly run postinstall scripts that executed malware. Supply chain impact extended to countless downstream projects and frameworks that relied on ua-parser-js directly or indirectly (e.g., React apps, analytics libraries, and CI tools).
Early Warning Details	Following a previous early warning success of detecting NPM supply chain attacks. An Intelligence process that identifies malicious changes in open source libraries. Detected the malicious code being injected.
Mitigation	Developers were advised to: Immediately **remove the affected versions** (**0.7.29, 0.8.0, 1.0.0**) **Reinstall clean versions** (**0.7.30, 0.8.1, 1.0.1**) **Audit systems** for compromise, particularly for processes like **xmrig** or signs of credential exfiltration Rotate any potentially exposed credentials or secrets.
Description	Still not published on CISA KEV.

By tracking potential incidents in real time and preemptively mitigating risks, Armis empowers organizations to stay ahead of the curve in an ever-evolving threat landscape. Armis Centrix™ for Early Warning offers a revolutionary AI technology that leverages dark web, dynamic deception technologies and HUMINT to stop attacks before they impact your organization.

Indicators of Action

The concept of “indicators of action,” more commonly known as Indicators of Attack (IoAs), represents a pivotal shift from reactive defense to proactive threat detection. While traditional cybersecurity often relies on Indicators of Compromise (IoCs), IoAs focus on the behaviors, methods, and intent of an adversary as an attack is unfolding or even being prepared. This distinction is critical, as it empowers security teams to intervene earlier in the cyber kill chain, significantly reducing the potential damage and cost of a breach.

The core premise of IoAs is that cyberattacks, regardless of their complexity, often involve a series of observable actions or patterns of behavior. These actions can include reconnaissance, initial access, execution of malicious code, privilege escalation, lateral movement, and data exfiltration. Unlike static IoCs like file hashes or IP addresses, which are snapshots of known malicious artifacts, IoAs are dynamic and contextual. They are less about “what” an attacker uses (a specific piece of malware) and more about “how” they are attempting to achieve their objectives.

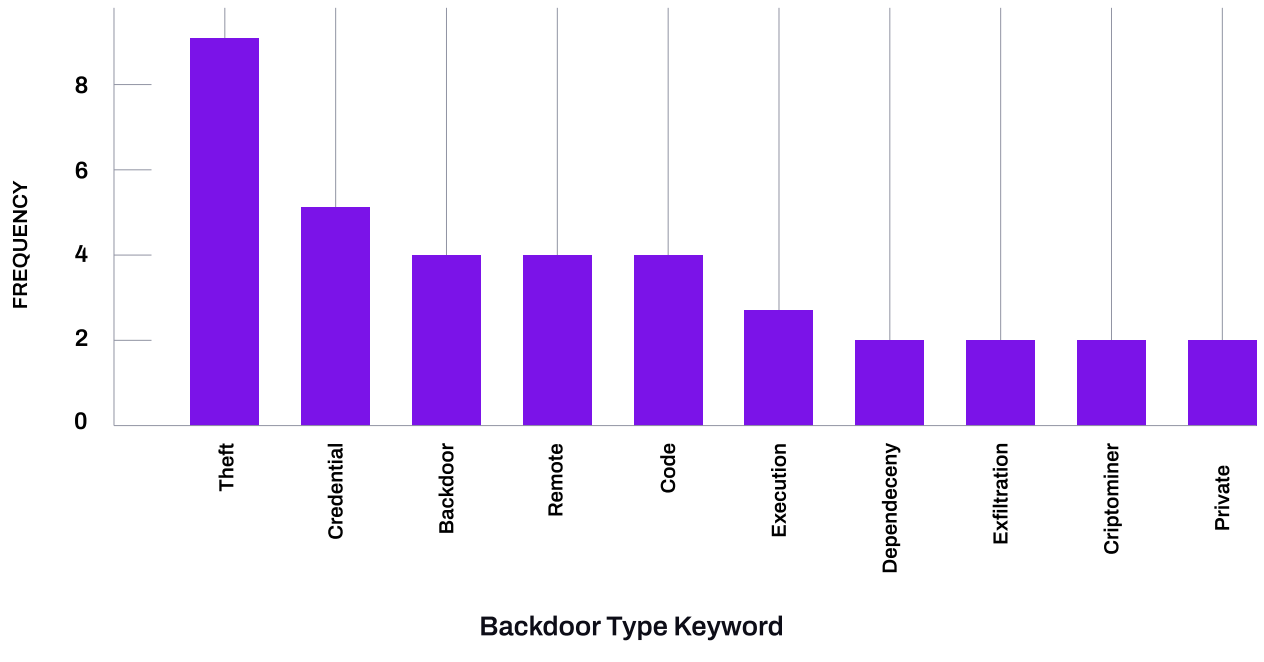
Understanding and effectively utilizing IoAs requires a deep appreciation of adversary Tactics, Techniques, and Procedures (TTPs). Frameworks like MITRE ATT&CK are invaluable here, providing a comprehensive knowledge base of common TTPs used by threat actors. By mapping observed behaviors to these TTPs, security analysts can identify an attack in progress, even if the specific malware or tools employed are novel.

Armis Labs put together a curated list of IoAs to monitor and address:

System Behavior Indicators	Unexpected CPU spikes (e.g., cryptomining activity) Increased or erratic memory usage Files written to unusual or hidden directories (e.g., **/tmp, %APPDATA%**) New executables or scripts dropped to disk Files self-deleting after execution Unusual system calls or process behaviors
Privilege & Credential Abuse Indicators	Attempts to escalate privileges (e.g., sudo or admin access) Access to environment variables or memory (e.g., to harvest secrets) Reading sensitive files like **.env, ~/.ssh/**, or keychains Injection of code into privileged or running processes

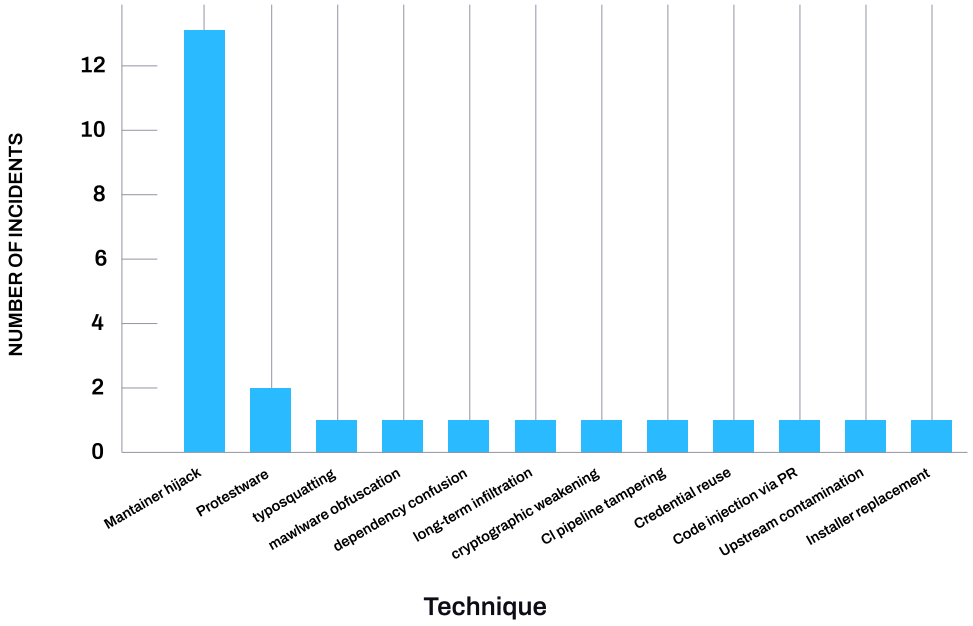
Network Behavior Indicators	Outbound connections to unknown or suspicious domains Beaconing behavior (repeated low-volume traffic to the same destination) DNS tunneling or use of DNS over HTTPS (DoH) Communication over unusual ports Use or download of Tor for anonymous communication
Build & Installation Indicators	Presence of unexpected pre- or post-installation scripts Obfuscated or encoded code (e.g., Base64, split/join string building) Sudden increase in package size without justification Addition of new or suspicious dependencies Changes in compiled artifacts without corresponding source changes
Metadata & Source Control Anomalies	New or unfamiliar maintainers added to the project Tags being reassigned to different commits Commits appearing to be from trusted bots (**e.g., renovate[bot]**) but unsigned or suspicious Rapid release of many versions (e.g., version churn) Forked repositories with similar names used in place of trusted originals

Most Common Backdoor Types (Keyword Frequency)



It's important to remember that while IoAs are powerful, they are most effective when used in conjunction with a comprehensive security strategy that includes preventative measures, strong security policies, and ongoing monitoring.

Frequency of Techniques Used in Backdoored Open Source Incidents



The real power of IoAs lies in their ability to provide context and sequence to disparate events. A single failed login might be innocuous, but a series of failed logins followed by a successful login from an unusual IP, immediately followed by attempts at privilege escalation and then outbound data transfer, forms a clear narrative of an ongoing attack. Security Information and Event Management (SIEM) systems, Security Orchestration, Automation, and Response (SOAR) platforms, and Endpoint Detection and Response (EDR) solutions are crucial tools for collecting, correlating, and analyzing the vast amounts of data necessary to identify these patterns.

Mitigation Strategies

MA multi-faceted approach to avoid Software Supply Chain Attacks is essential. This includes, but is not limited to:

Package & Supply-Chain Integrity

- **Verify AI-Suggested Dependencies:** Always cross-check any AI-recommended package name against official documentation or the package registry before installation (en.wikipedia.org).
- **Lock Files & Hash Verification:** Use **requirements.txt** or **package-lock.json** with pinned versions and hash-based integrity checks (e.g., **pip hash-check**) to ensure exact package identities.
- **Dependency pinning and SBOMs:** Maintain a Software Bill of Materials (SBOM) and pin all third-party dependencies (e.g., exact version in requirements.txt or package-lock.json) so that pull-in of a malicious “requestss” or “expresss” typo-package is prevented. Ensure an approved list of packages. Rigorously vet all installations.
- **Registry allow-listing:** Restrict package installs to approved registries (e.g., your private PyPI or NPM proxy) and block direct access to the public repositories, especially for new or unvetted packages.
- **Automated fuzzy-hashing and typo-pkg detection:** Integrate tools (like typosquat.py or OSS-supply-chain scanners) into your CI pipeline to flag packages whose names differ from popular ones by one or two characters.

- **Dependency Scanning:** Integrate vulnerability scanners (OSS Index, Snyk, etc.) into your CI/CD pipeline to flag unknown or untrusted packages. Implement automated dependency-checking tools and vetting pipelines. Automatically disable any packages that have a small number of stars/users.
- **Least-Privilege Installation:** Run dependency installation in isolated, ephemeral environments (containers or virtualenvs) with strict network egress controls to limit the blast radius of a malicious package.
- **Dependency Auditing:** Maintaining a comprehensive inventory of all open-source components used, along with their versions and licenses, is crucial for effective risk management.
- **Registry allow-listing:** Restrict package installs to approved registries (e.g., your private PyPI or NPM proxy) and block direct access to the public repositories, especially for new or unvetted packages.
- **Prompt Patching and Update:** Establishing a swift and efficient process for applying security patches and updating libraries to their latest, most secure versions is vital. This often requires automated systems to monitor for new releases and vulnerabilities.

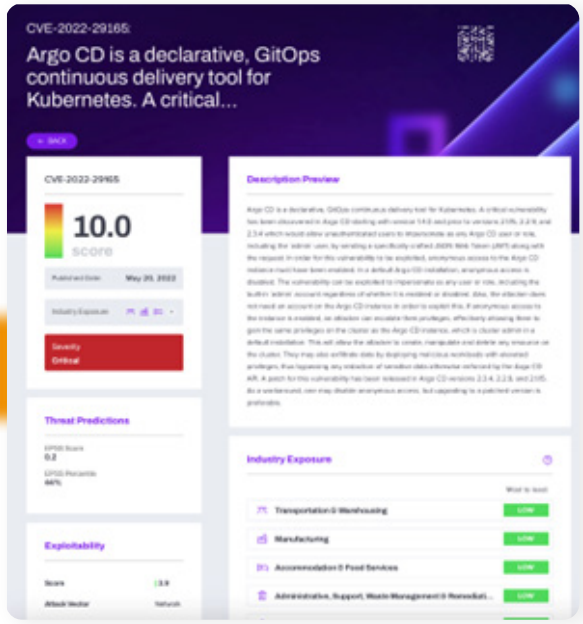
Developer & User Awareness

- **Security training:** Educate developers and DevOps teams to double-check package names and domain URLs before installing or integrating. Ensure developers avoid blind reliance on AI suggestions.
- **Phishing simulations:** Regularly run phishing/emails that mimic slopsquatting domains to test user vigilance and reinforce best practices around checking the address bar and email sender.
- **IDE and terminal plugins:** Provide tooling (e.g., VS Code extensions) that warn when importing unknown packages or browsing to domains that are one character off from corporate assets.
- **Community Engagement and Due Diligence:** Staying informed about security advisories from open-source communities and performing due diligence on the security practices of maintainers can help assess risk.
- **Security Best Practices in Development:** Encouraging secure coding practices within development teams and understanding how to safely integrate and use open-source components can mitigate risks.

Armis Vulnerability Intelligence Database

In today's threat landscape, timing is everything. From zero-day vulnerabilities to weaponized exploits, the gap between discovery and exploitation continues to shrink. Armis Vulnerability Intelligence Database is a revolutionary, AI-powered vulnerability intelligence resource that goes well beyond traditional static databases. While many teams rely on the CISA Known Exploited Vulnerabilities (KEV) catalog to track risk, Armis Vulnerability Intelligence Database is built to extend that value by offering real-time context, community-driven prioritization, and actionable insights tailored to specific industries and threat levels.

Backed by the Armis Asset Intelligence Engine, which observes and analyzes over six billion connected assets globally, and Armis Labs, the Armis Vulnerability Intelligence Database offers the cybersecurity community an opportunity to shift left and move from reactive response to proactive risk reduction.



Security professionals across industries face a shared challenge that revolves around information overload without meaningful prioritization. Public databases like CISA KEV are useful, but often lag behind active threats. They're static, generalized, and lack the context needed to take decisive, tailored action.

Many organizations are left asking: Which vulnerabilities should I address first? How does this CVE affect my specific devices, in my specific environment? Am I exposed today or just potentially exploitable?

Armis Vulnerability Intelligence Database is a modern, dynamic vulnerability database built not only to inform but to empower action. It ingests and correlates data from a wide range of sources, including internal threat intelligence, external feeds, community contributions, and Armis Labs. It delivers timely, highly contextualized vulnerability insights with specific recommendations. Going beyond a CVE repository, Armis Vulnerability Intelligence Database surfaces vulnerabilities that matter to you, your industry and threat profile. By combining artificial intelligence with human insight, Armis Vulnerability Intelligence Database helps you understand not just what vulnerabilities exist, but which ones are most likely to impact you.

Security researchers, CISOs, threat analysts, and practitioners can access Armis Vulnerability Intelligence Database through a modern, searchable microsite designed to bring transparency, speed, and collaboration to the forefront of vulnerability management.

How Armis Vulnerability Intelligence Database Stands Out

Where other databases stop at basic enumeration, Armis Vulnerability Intelligence Database goes further. It provides real-world prioritization based on actual industry exposure, observed threat activity, and risk relevance. Leveraging Armis' ability to monitor billions of devices, including IT, OT, IoT, and medical equipment, Armis Vulnerability Intelligence Database reflects a unique, ground-level view of global threat activity.

Armis Vulnerability Intelligence Database also builds on the foundation of the CISA KEV catalog, but expands it in meaningful ways. While KEV offers a list of known exploited vulnerabilities, Armis layers in industry-specific context, prioritization signals, mitigation recommendations, and even identifies threats that haven't yet reached public disclosure. Through Armis Early Warning capabilities, vulnerabilities and risk can be flagged while they're still in development. This gives security personnel a head start of days or even months before exploitation becomes widespread.

Working with CISA KEV And Going Beyond It

CISA's KEV catalog has played an important role in helping organizations prioritize known threats. But KEV documents vulnerabilities after they've been exploited in the wild. For organizations that already use KEV as a baseline, Armis Vulnerability Intelligence Database offers a deeper, more nuanced layer of intelligence that answers the critical question KEV doesn't: What does this vulnerability or other security finding mean for me, right now? Armis Vulnerability Intelligence Database enables organizations to make smarter decisions faster. It closes the gap between visibility and action, providing the clarity and actions needed to reduce risk before incidents occur.

Whether you're defending a hospital, a manufacturing plant, a government network, or a corporate IT environment, Armis Vulnerability Intelligence Database helps you focus on the vulnerabilities that matter most to your mission with the intelligence needed to make it actionable.

Additionally, Armis has been authorized by the Common Vulnerabilities and Exposures (CVE®) Program as a CVE Numbering Authority (CNA). The mission of the international program is to identify, define and catalog publicly disclosed vulnerabilities. As a CNA, Armis can review and assign CVE IDs to newly discovered vulnerabilities.

About Armis Centrix™

Armis Centrix™, the cyber exposure management platform, sees, secures, protects and manages billions of assets around the world in real time. Our seamless, frictionless, cloud-based platform proactively mitigates all cyber asset risks, remediates vulnerabilities, blocks threats and protects your entire attack surface.

Only Armis Centrix™ protects all verticals and industries including Manufacturing, Transportation, Health and Medical, Federal, National, State and LocalGovernments, Critical Infrastructure, Higher Education, Financial Services, Retail,Telecommunications and Media, Information Technology, and many more.

Armis Centrix™ for Early Warning

Though your organization may have taken a proactive approach to managing vulnerabilities, your teams are likely overwhelmed by the millions of alerts coming from new security tools, and may struggle to automate prioritization based on context and risks specific to their environment.

To combat this, Armis Centrix™ for Early Warning combines **asset intelligence** with **vulnerability intelligence**.

Asset intelligence provides a complete inventory of the manufacturing industry internal environment, like a detailed map of the attack surface across all Armis customers in the industry. Our vulnerability intelligence uses state of the art techniques to see what vulnerabilities and TTPs threat actors are using in the wild or about to weaponize.

Combining these methods allows CISOs to:

Prioritize vulnerabilities

Focus on those posing the greatest risk to your specific attack surface.

Reduce alert fatigue

Address the most critical alerts based on potential impact.

Optimize resources

Protect the most valuable and vulnerable assets.

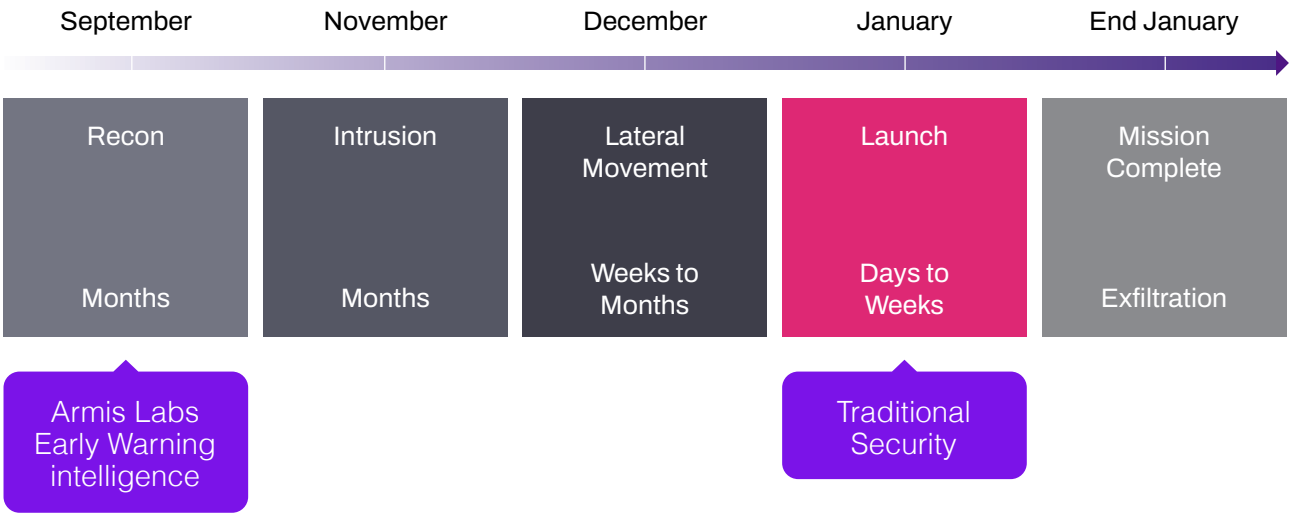
Proactively mitigate risks

Anticipate and defend against attacks based on attacker behavior and known vulnerabilities.

Asset intelligence shows you “what needs protecting,” while vulnerability intelligence reveals “what’s coming to attack it.” This empowers proactive, risk-based cybersecurity.

Early Warning empowers you with actionable intelligence before an attack is launched and before your organization is impacted.

Early warning vulnerability intelligence can detect and neutralize malicious actions in your environment before they are exploited. Having robust visibility, protection, and alerts management in place can provide an effective threat forecast and a realistic picture of where to focus your efforts.



Ordinary security goes to work when an attack is launched. Armis Labs early warning intelligence finds potential threats before they are ever launched and before your environment is ever impacted. In many cases, months early.

This multi-faceted approach ensures that the "spotlight" shines on the most urgent and impactful threats, enabling efficient resource allocation and maximized risk reduction. This targeted "spotlight" strategy allows for:

Rapid Response: Concentrated efforts on patching or mitigating these 3 vulnerabilities minimizes your attack surface against active threats.

Resource Optimization: Devoting resources to validated, high-impact threats avoids spreading your security team too thin.

Reduced Dwell Time: Faster patching cycles for critical vulnerabilities limit the window of opportunity for attackers.

Improved Communication: A concise spotlight facilitates clear communication to stakeholders about imminent threats and remediation efforts.

Data-Driven Defense: This process leverages early warning security intelligence to guide security decisions, enhancing your overall security posture.

By focusing on high-confidence early warnings of in-the-wild exploits, you're prioritizing the most critical threats to your organization.

ARMIS LABS

About Armis Labs

Armis Labs, a division of Armis, is a team of seasoned security professionals dedicated to staying ahead of the ever-evolving cybersecurity landscape. With a deep understanding of emerging threats and cutting-edge methodologies, Armis Labs empowers organizations with unparalleled visibility and expertise to protect against the threats that matter most, right now.

AAt the heart of Armis Labs lies a formidable research powerhouse, where experts investigate the latest trends and tactics employed by cyber adversaries. Armed with access to over 5 billion profiled assets and state-of-the-art tools and methodologies, the team at Armis Labs conducts in-depth analyses of evolving threats both in the pre-emergence stage and “in the wild” stage of an attack.

+6 Billion

Core to Armis Labs is our Asset Intelligence Engine. It is a giant, crowdsourced, cloud-based knowledge base—the largest in the world, tracking over five billion assets—and growing. It powers Armis Labs with unique, actionable cyber intelligence to detect and address real-time threats across the entire attack surface.

Armis Labs security practitioners are utilizing cutting edge tools that include deception technology, incident forensics, reverse engineering, dark web monitoring, and human intelligence to proactively identify and mitigate threats before they manifest. Leveraging advanced AI/ML technologies, Armis Labs’ proactive threat detection capabilities enable organizations to stay one step ahead of cyber adversaries, minimizing the risk of potential breaches while stopping potential damage before it occurs.

Armis Labs is dedicated to providing organizations with the tools and expertise they need to defend against the threats that matter most, right now. With comprehensive threat intelligence, proactive threat detection capabilities, and seamless integration into existing security workflows, Armis Labs empowers organizations to stay ahead of cyber adversaries and protect their most critical assets.

Armis, the cyber exposure management & security company, protects the entire attack surface and manages an organization's cyber risk exposure in real time.

In a rapidly evolving, perimeter-less world, Armis ensures that organizations continuously see, protect and manage all critical assets - from the ground to the cloud. Armis secures Fortune 100, 200 and 500 companies as well as national governments, state and local entities to help keep critical infrastructure, economies and society stay safe and secure 24/7.

Armis is a privately held company headquartered in California.

1.888.452.4011

Website
Platform
Industries
Solutions
Resources
Blog

Try Armis
Demo