

DeepSeek Exposed 1 Million+ Chat Logs Through Passwordless ClickHouse Databases—Wiz Researcher Ran SQL Queries via Web Browser on January 29, 2025

A security researcher opened a web browser, typed a URL, and gained full SQL access to production databases containing over 1 million chat logs. No password. No exploit. Just an open door.

The Incident: What Actually Happened

On January 29, 2025, Wiz security researcher Gal Nagli discovered two ClickHouse databases belonging to Chinese AI company DeepSeek that required zero authentication

to access. The databases, hosted at `oauth2callback.deepseek.com:9000` and `dev.deepseek.com:9000`, were fully exposed to the public internet.

This wasn't a sophisticated zero-day exploit or a nation-state attack. According to The Hacker News, Nagli simply accessed the databases through ClickHouse's HTTP interface using a standard web browser. He executed arbitrary SQL queries against production systems containing active user data.

The exposed data included over 1 million log lines with unencrypted user chat histories (primarily in Chinese), API secrets, backend credentials, and operational metadata. Full database control was available to anyone who stumbled upon these endpoints.

DeepSeek responded quickly after Wiz's responsible disclosure, securing the databases within hours on the same day. But the damage window remains unknown. There's no public evidence of how long these databases sat exposed before discovery, or who else may have accessed them.

The Timeline Context

This exposure occurred during an already chaotic period for DeepSeek. Two days earlier, on January 27, the company had halted new user registrations following distributed denial-of-service attacks. The company was simultaneously dealing with supply chain attacks via malicious PyPI packages named "deepseek" and "deepseekai" that targeted developers attempting to integrate with DeepSeek's APIs.

The timing suggests either systemic security gaps or an organization struggling to maintain basic operational security while scaling rapidly. Neither explanation inspires confidence.

Why This Matters More Than a Typical Data Breach

Every month brings news of another data breach. Most follow familiar patterns: sophisticated attacks exploiting complex vulnerabilities, insider threats, or social engineering. This incident is different in ways that should concern every technology leader.

This was a configuration error, not a hack. The databases weren't breached through clever exploitation. They were simply left open. ClickHouse, the column-oriented database

DeepSeek used, has authentication capabilities. Someone chose not to enable them on production systems containing user conversations and system credentials.

The security implications cascade rapidly. API secrets and backend credentials exposed in these logs could provide persistent access to other DeepSeek systems. User chat histories often contain sensitive personal information, business data, and prompts that reveal intent. When users interact with AI chatbots, they frequently share information they wouldn't expose elsewhere—medical symptoms, legal questions, business strategies, personal struggles.

The AI Security Paradox

DeepSeek's situation illustrates a troubling paradox in the AI industry. Companies racing to build sophisticated machine learning systems often neglect fundamental infrastructure security. The same organization that trained complex language models couldn't configure basic database authentication.

Security assessments of DeepSeek's model revealed a 91% failure rate on jailbreak tests and 86% failure on prompt-injection tests. The mobile application used outdated 3DES encryption with hard-coded keys and disabled App Transport Security (ATS). These aren't edge cases—they're security fundamentals that undergraduate courses cover.

> When an organization fails at basic database authentication, it reveals a systemic culture problem, not an isolated mistake.

The exposed data has already appeared on dark web markets, fueling phishing campaigns and scams targeting DeepSeek users. Secondary exploitation is often more damaging than the initial exposure. Attackers now have verified email addresses, conversation histories that reveal user interests and vulnerabilities, and credentials that may be reused across services.

Technical Analysis: How ClickHouse Misconfigurations Occur

ClickHouse is an open-source column-oriented database designed for online analytical processing (OLAP). It's fast, scalable, and increasingly popular for log analysis and real-time analytics—exactly the use case DeepSeek employed. Understanding how this misconfiguration happened requires examining ClickHouse's architecture and common deployment patterns.

Default Configuration Risks

ClickHouse exposes multiple interfaces: native TCP (port 9000), HTTP (port 8123), and optionally MySQL and PostgreSQL wire protocols. The HTTP interface is particularly dangerous when misconfigured because it allows SQL execution through simple GET and POST requests. A browser becomes a database client.

By default, ClickHouse authenticates using a `default` user with no password. This design choice prioritizes ease of initial setup over security. The assumption is that operators will configure authentication before production deployment. In fast-moving environments, that assumption fails.

The configuration file (`users.xml` or `users.d/*.xml`) controls authentication:

```
<users>
  <default>
    <password></password>
    <networks>
      <ip>::/0</ip>
    </networks>
  </default>
</users>
```

That `::/0` allows connections from any IPv6 address (which includes IPv4-mapped addresses). Combined with an empty password, you have a database accessible to the entire internet.

Why This Happens in Practice

Several factors contribute to ClickHouse misconfigurations appearing in production:

Development-to-production drift: Teams configure databases permissively during development for convenience. Those configurations propagate to production through infrastructure-as-code templates, container images, or manual deployments that skip security hardening steps.

Cloud networking assumptions: Engineers assume that cloud security groups or firewalls provide perimeter protection. They configure databases to accept all connections, trusting network-layer controls. Then someone modifies a security group rule, enables a load balancer, or misconfigures a VPN, and the database becomes exposed.

Operational convenience: Password-protected databases require credential management. Connection strings need secrets. Automated processes need service accounts. When teams move fast, authentication feels like friction. That friction has a purpose.

Monitoring blind spots: Most organizations monitor for intrusions—failed login attempts, suspicious queries, unusual data access patterns. When authentication is disabled, there are no failed logins to detect. The database looks healthy while being publicly accessible.

The ClickHouse HTTP Interface Attack Surface

DeepSeek's exposure was particularly severe because ClickHouse's HTTP interface supports full SQL execution. An attacker (or curious researcher) can:

- List all databases: `SELECT name FROM system.databases`
- Enumerate tables: `SELECT database, name FROM system.tables`
- Extract schema: `DESCRIBE TABLE target_table`
- Dump data: `SELECT * FROM sensitive_table`
- Access system information: `SELECT * FROM system.processes`

With default privileges, attackers can read system tables containing query history, user information, and cluster configuration. They can potentially write data, drop tables, or

execute system commands depending on the ClickHouse version and configuration.

Wiz reported that Nagli achieved "full database control." This suggests the exposed user had administrative privileges—the worst-case scenario for a misconfigured database.

The Contrarian Take: What the Coverage Gets Wrong

Most reporting on this incident frames it as a "DeepSeek problem" or evidence that Chinese AI companies have poor security practices. That framing misses the broader pattern and lets other organizations off the hook.

This Isn't Unique to DeepSeek or Chinese Companies

Misconfigured databases are endemic across the technology industry. Shodan, the search engine for internet-connected devices, consistently finds thousands of exposed ClickHouse, MongoDB, Elasticsearch, and Redis instances. American companies, European startups, and enterprises of all sizes make identical mistakes.

In 2023, Microsoft's AI research team exposed 38 terabytes of internal data through a misconfigured Azure storage account. The same year, exposed MongoDB databases leaked customer data from multiple US-based SaaS companies. The DeepSeek incident is notable for its scale and the sensitivity of AI chat logs, not for the underlying vulnerability pattern.

Focusing on DeepSeek's Chinese origins distracts from the uncomfortable truth: this happens everywhere, constantly, and most incidents never become public.

The Real Story Is AI's Infrastructure Debt

AI companies have accumulated massive infrastructure debt while racing to capture market share. They've hired world-class machine learning engineers and trained sophisticated models while underinvesting in operations, security, and platform engineering.

The talent profile matters. Organizations optimizing for AI research attract researchers. Security engineering, infrastructure hardening, and operational excellence require different skills and organizational priorities. When security is an afterthought staffed by whoever is

available, you get passwordless production databases.

> The AI industry has a security culture problem that model performance benchmarks can't measure.

DeepSeek's 91% jailbreak failure rate and 86% prompt-injection failure rate tell the same story from a different angle. The organization didn't just fail at database authentication—it failed at AI safety too. These aren't coincidences. They reflect organizational priorities and engineering culture.

What's Underhyped: The Chat Log Sensitivity Problem

AI chat logs are more sensitive than traditional application logs. Users treat AI chatbots like confidants, sharing information they wouldn't type into search engines or post on social media. Chat histories reveal:

- **Health information:** Users ask AI about symptoms, medications, and treatments
- **Legal exposure:** Questions about contracts, disputes, and potential wrongdoing
- **Business intelligence:** Queries about competitors, strategies, and internal challenges
- **Personal vulnerabilities:** Relationship problems, financial stress, career concerns
- **Authentication details:** Users sometimes paste passwords, API keys, or credentials into chat interfaces

The 1 million+ exposed log lines from DeepSeek likely contain all of these data types. The dark web markets selling this data understand its value better than most coverage suggests.

Practical Implications: What CTOs and Engineering Leaders Should Do

Reading about someone else's security failure provides an opportunity for introspection. Here's what organizations running AI infrastructure—or any data-intensive systems—should prioritize.

Immediate Actions

Audit database exposure today. Use external scanning tools to verify that production databases aren't internet-accessible. Don't trust internal documentation—verify from outside your network perimeter.

Shodan, Censys, and similar services can identify your exposed assets. If you find passwordless databases accessible from the internet, assume they've been accessed by unauthorized parties and begin incident response.

Implement authentication everywhere. No exceptions. Development databases should require authentication. Staging databases should require authentication. "Temporary" deployments should require authentication. The convenience cost is minimal compared to the exposure risk.

For ClickHouse specifically:

- Set strong passwords in `users.xml`
- Restrict network access using the `<networks>` configuration
- Enable TLS for all connections
- Use RBAC to limit privileges per user
- Configure `readonly=1` for users who shouldn't write

Review AI chat log retention policies. Are you storing chat histories in plaintext? For how long? Who has access? Chat logs containing user prompts are sensitive personal data under most privacy regulations. Treat them accordingly.

Architectural Recommendations

Separate AI inference from logging infrastructure. Chat logs shouldn't live in the same security perimeter as real-time inference systems. Use dedicated, hardened data warehouses for analytics and logging. Apply defense in depth—even if one layer fails, additional barriers should prevent exposure.

Implement zero-trust networking for databases. Databases should never be directly accessible from the internet. Use private networking, VPNs, or identity-aware proxies (like

Google's BeyondCorp model or tools like Teleport and Boundary). Authenticate every connection, even from "trusted" networks.

Encrypt sensitive fields at the application layer. Database-level encryption helps, but application-layer encryption means that even if someone gains database access, they can't read sensitive data without separate key access. Chat logs containing user content should be encrypted with keys managed outside the database system.

Implement anomaly detection on database access patterns. When authentication is disabled, there's nothing to detect. With authentication enabled, monitor for unusual query patterns, high-volume data exports, access from unexpected locations, and queries against sensitive tables.

Process and Culture Changes

Require security reviews for infrastructure changes. The ClickHouse instances were presumably deployed through some process. That process failed to catch (or didn't check for) disabled authentication. Infrastructure-as-code repositories should include security linting. Deployment pipelines should validate security configurations.

Include security engineers in AI platform teams. Don't relegate security to a separate organization that reviews work after the fact. Embed security expertise in platform and infrastructure teams. Give them authority to block deployments that don't meet security requirements.

Conduct regular penetration testing of AI infrastructure. External security assessments often find what internal teams miss. Red team exercises that specifically target AI infrastructure—including model jailbreaking, prompt injection, and supporting infrastructure—reveal gaps before attackers exploit them.

The Vendor and Tooling Landscape

Several categories of tools can prevent incidents like DeepSeek's exposure:

Cloud Security Posture Management (CSPM)

Tools like Wiz (the company that discovered this vulnerability), Orca Security, Lacework, and cloud-native options (AWS Security Hub, Google Security Command Center, Azure Defender) continuously scan for misconfigurations. They detect publicly accessible databases, overly permissive IAM policies, and encryption gaps.

If DeepSeek had deployed basic CSPM tooling, automated scans would have flagged the passwordless ClickHouse instances immediately.

Database Activity Monitoring

Solutions like Imperva, IBM Guardium, and open-source alternatives monitor database access patterns, alert on suspicious queries, and maintain audit logs. They add a detection layer that catches unauthorized access even when authentication controls fail.

Infrastructure-as-Code Security Scanning

Tools like Checkov, tfsec, and Snyk IaC scan Terraform, CloudFormation, and Kubernetes manifests for security issues before deployment. They catch misconfigured database settings in code review, before infrastructure reaches production.

Secrets Management

HashiCorp Vault, AWS Secrets Manager, and similar tools ensure that database credentials, API keys, and other secrets are never hard-coded (like DeepSeek's 3DES keys) or stored in plaintext. Proper secrets management makes it harder to deploy unprotected services accidentally.

Looking Forward: The Next 6-12 Months

This incident will accelerate several trends already underway in AI infrastructure security.

Regulatory Scrutiny Will Intensify

AI companies have enjoyed relatively light regulatory oversight compared to traditional

software vendors. The combination of exposed user chat logs and failing safety assessments will invite increased attention from data protection authorities, particularly in Europe under GDPR and in US states with emerging privacy laws.

Expect enforcement actions against AI companies with inadequate security practices within the next year. The DeepSeek incident provides a template for what regulators will investigate: database security, encryption practices, data retention, and access controls.

Enterprise AI Adoption Will Demand Security Certification

Organizations evaluating AI vendors will increasingly require SOC 2 compliance, penetration test results, and detailed security documentation. The “move fast and break things” approach that characterizes many AI startups won’t satisfy enterprise procurement processes.

This creates an opportunity for AI companies that invest in security early. Demonstrable security practices become competitive advantages when buyers are evaluating alternatives.

AI-Specific Security Frameworks Will Emerge

Current security frameworks (NIST, ISO 27001, SOC 2) weren’t designed with AI workloads in mind. They don’t address model jailbreaking, prompt injection, training data poisoning, or the unique sensitivity of AI chat logs.

Expect industry groups and standards bodies to publish AI-specific security guidance. The OWASP Top 10 for LLM Applications is an early example. More comprehensive frameworks will follow.

Consolidation of AI Security Tooling

The market for AI security tools is fragmented. Model security, prompt injection detection, training data protection, and inference infrastructure security are often addressed by different vendors. Consolidation will create platforms that address AI security holistically.

Well-funded startups in this space include Protect AI, Robust Intelligence, and CalypsoAI.

Traditional security vendors (Palo Alto, CrowdStrike, Microsoft) are building AI security capabilities through acquisition and internal development.

The Exposure Discovery Cycle Will Continue

Security researchers now know that AI companies are attractive targets with potentially weak security. Expect more discoveries of exposed databases, misconfigured APIs, and vulnerable AI infrastructure. The next DeepSeek-scale incident is already waiting to be found.

Organizations deploying AI infrastructure should assume researchers (and adversaries) are actively probing their attack surface. Defensive measures aren't optional.

Conclusion

The DeepSeek incident reveals something uncomfortable about the current state of AI infrastructure: organizations capable of training sophisticated language models often can't secure a database. This isn't a uniquely Chinese problem or a DeepSeek-specific failure. It's a systemic industry pattern that will produce more incidents until organizations prioritize security engineering alongside AI research.

The exposed chat logs—over 1 million entries containing unencrypted user conversations—represent a particularly sensitive data type. Users share information with AI chatbots that they wouldn't expose elsewhere. When that data leaks, the consequences extend far beyond typical breach impacts.

For technology leaders, this incident is a forcing function. Audit your database exposure. Implement authentication on every system. Treat AI chat logs as sensitive personal data requiring encryption and access controls. Build security expertise into platform teams rather than bolting it on later.

The next major AI data exposure is already configured and waiting to be discovered—the only question is whether it belongs to your organization.



WRITTEN BY

Artur Markus

Software engineer in Basel building AI infrastructure and agentic systems, with a background in pharmaceutical automation, GxP compliance, and validation. I help teams turn AI from a chatbot into a reliable operational layer.

Book a meeting