

Five Steps **to Secure your** **Cloud Application** **in the Age of AI**



Colorado Startup Week

September 14, 2026

Darren Weiner

darren@cloudbutton.com



CLOUD**BUTTON**

Mark Hoopes

mark@meristeminfosec.com



MERISTEM
— INFOSEC —



Colorado Startup Week

September 14, 2026

What we'll cover

1. Managing traditional application security weaknesses
2. Overpowered and overprivileged agents
3. Improper scoping and handling of tokens
4. Identity and access management for cloud infrastructure
5. Ransomware - how to protect and recover

Defense in Depth



Managing Traditional Application Security Weaknesses



September 14, 2026

OWASP Top 10:2025

A01	Broken Access Control
A02	Security Misconfiguration
A03	Software Supply Chain Failures
A04	Cryptographic Failures
A05	Injection
A06	Insecure Design
A07	Authentication Failures
A08	Software or Data Integrity Failures
A09	Security Logging and Alerting Failures
A10	Mishandling of Exceptional Conditions

<https://www.owasp.community/projects/top-ten>



OWASP Top 10: Supply Chain Vulnerabilities

A01	Broken Access Control
A02	Security Misconfiguration
A03	Software Supply Chain Failures
A04	Cryptographic Failures
A05	Injection
A06	Insecure Design
A07	Authentication Failures
A08	Software or Data Integrity Failures
A09	Security Logging and Alerting Failures
A10	Mishandling of Exceptional Conditions

- Vulnerabilities that come from software or libraries you use, not software you wrote
- Defense comes from a regular library update process
- Coding agents are really good at upgrading libraries

OWASP Top 10: Logging & Errors

A01	Broken Access Control
A02	Security Misconfiguration
A03	Software Supply Chain Failures
A04	Cryptographic Failures
A05	Injection
A06	Insecure Design
A07	Authentication Failures
A08	Software or Data Integrity Failures
A09	Security Logging and Alerting Failures
A10	Mishandling of Exceptional Conditions

- Bad logging doesn't create vulnerabilities
- When an attack happens, you'll want all the logging you can get
- Log error details, don't show them to users.
- "Fail Closed"

OWASP Top 10: Implementation Decisions

A01	Broken Access Control
A02	Security Misconfiguration
A03	Software Supply Chain Failures
A04	Cryptographic Failures
A05	Injection
A06	Insecure Design
A07	Authentication Failures
A08	Software or Data Integrity Failures
A09	Security Logging and Alerting Failures
A10	Mishandling of Exceptional Conditions

- Coding agents are good at standard implementations
- Standard implementations are “usually” secure
- Vulnerabilities come when you are using elements in a “non-standard” way without accounting for it

OWASP Top 10: Design Decisions

A01	Broken Access Control
A02	Security Misconfiguration
A03	Software Supply Chain Failures
A04	Cryptographic Failures
A05	Injection
A06	Insecure Design
A07	Authentication Failures
A08	Software or Data Integrity Failures
A09	Security Logging and Alerting Failures
A10	Mishandling of Exceptional Conditions

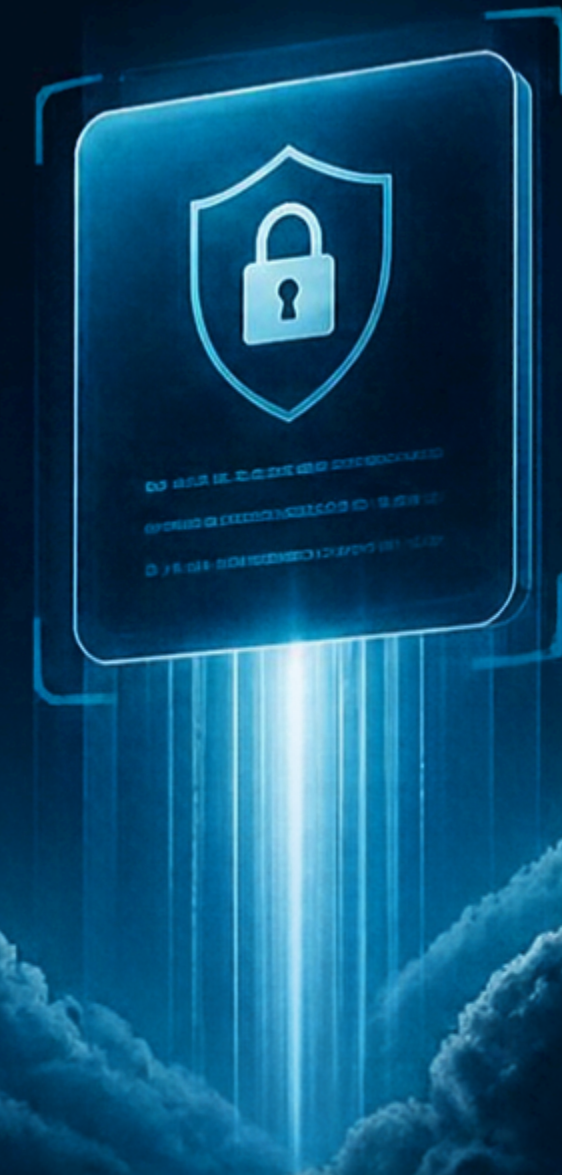
- The coding agent still can't be trusted to really understand your business context.
- There will be things the agent just didn't consider
- This is where you need to step in and think about "what could possibly go wrong"

Questions to ask your Coding Agent

1.	Is a regular user blocked from calling all administrative functions?
2.	Can one user read another user's document?
3.	Does my configuration of [component] match recommended security best practices?
4.	Is my application vulnerable to Cross-site Scripting?
5.	Is my application vulnerable to SQL Injection?
6.	Is my application vulnerable to Operating System Injection?
7.	Are there any recommendations to improve the security of my architecture?
8.	Are security related events are properly logged.
9.	Can this code show technical details or sensitive data to users?
10.	Check the application logic for places that "fail open"

`/security-review`
`/claude-security`

Overprivileged Agents



September 14, 2026

LLMs are Nondeterministic

Nondeterministic systems can produce different outputs or take different paths even when given the exact same input every time.

Newer models or better prompts will not change this.
Randomness is a fundamental part of the system.

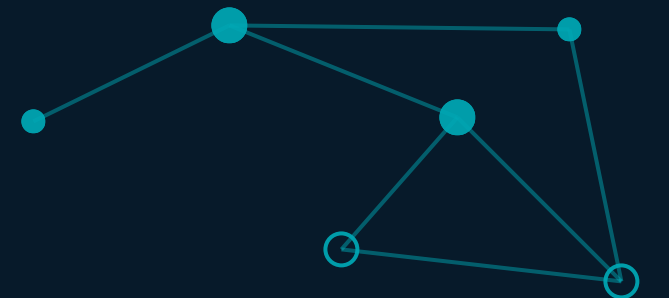
LLM Flow



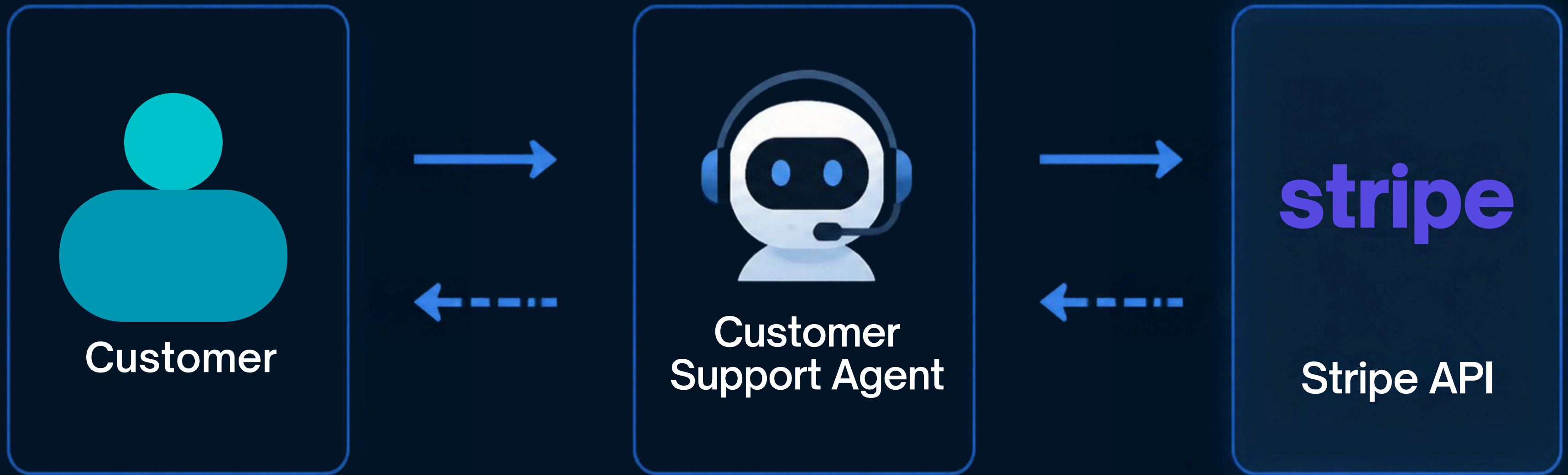
Designing for Nondeterminism

- Give the agent small specific tasks
- Restricted agent permissions / sandboxing
- **Validate output**
 - Human in the loop
 - Validation from different LLMs
 - Deterministic function

Also effective against Prompt Injection!



Customer Support Agent - Refund Feature



Defense 1 - Human in the Loop



Defense 2 - Multiple LLMs

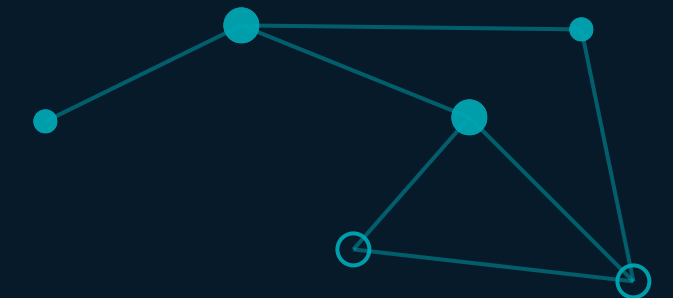


Defense 3 - Deterministic Function

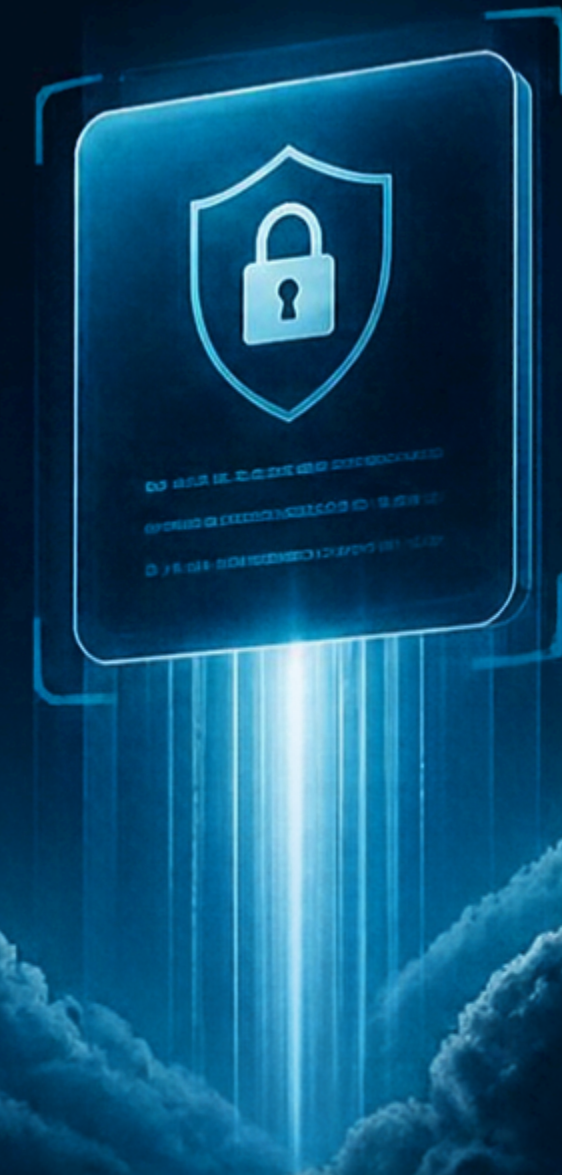


Designing for Nondeterminism

- Give the agent small specific tasks
- Restricted agent permissions / sandboxing
- **Validate output**
 - Human in the loop
 - Validation from different LLMs
 - Deterministic function

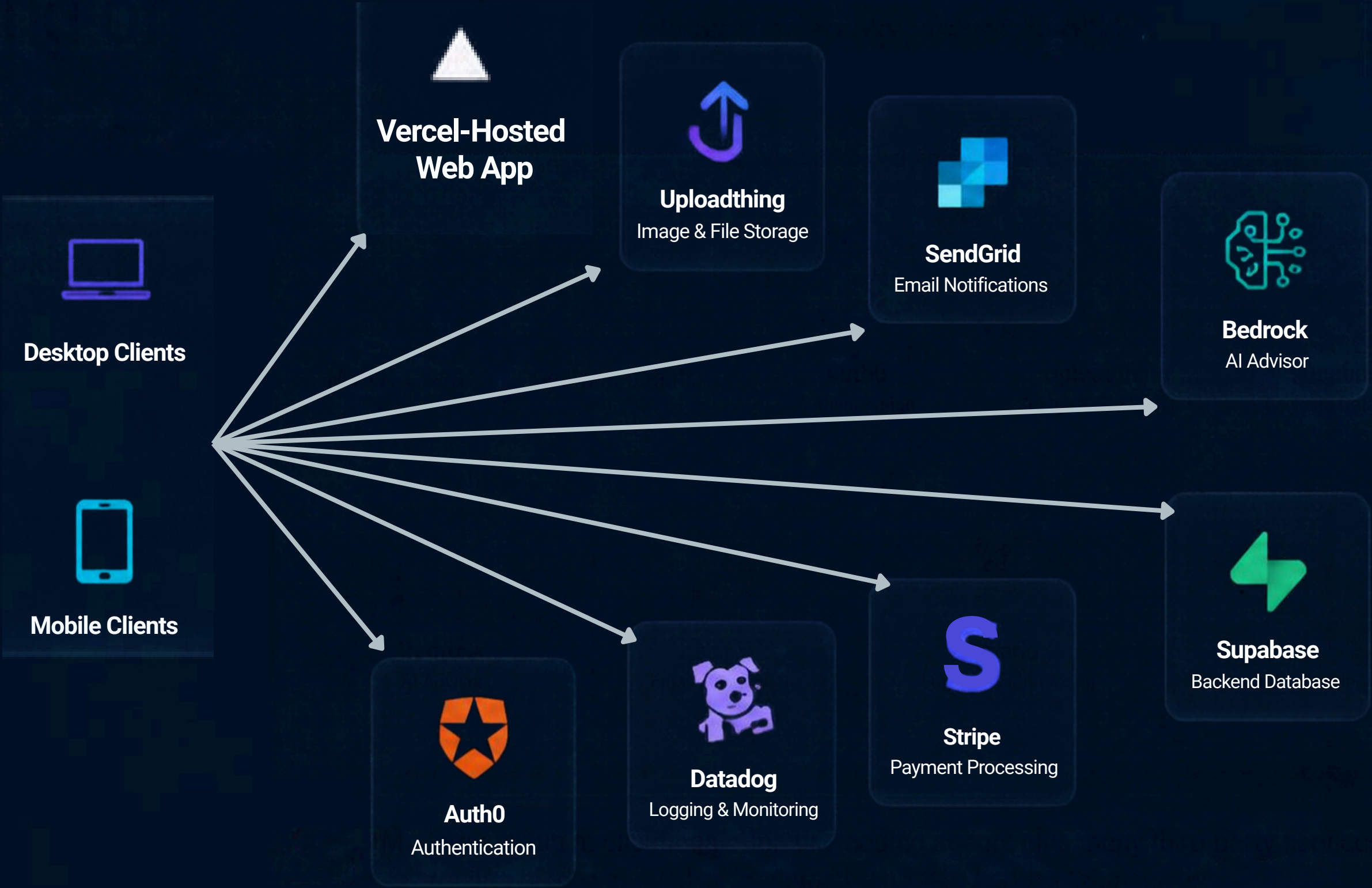


Authentication Tokens: Proper Scoping and Safe Handling



September 14, 2026

Improper Scoping and Handling of Authentication Tokens



The Token Problem

1

Clients must authenticate with each service

This is usually done with an API token.



2

If the token is stored on the client, the user has access to it

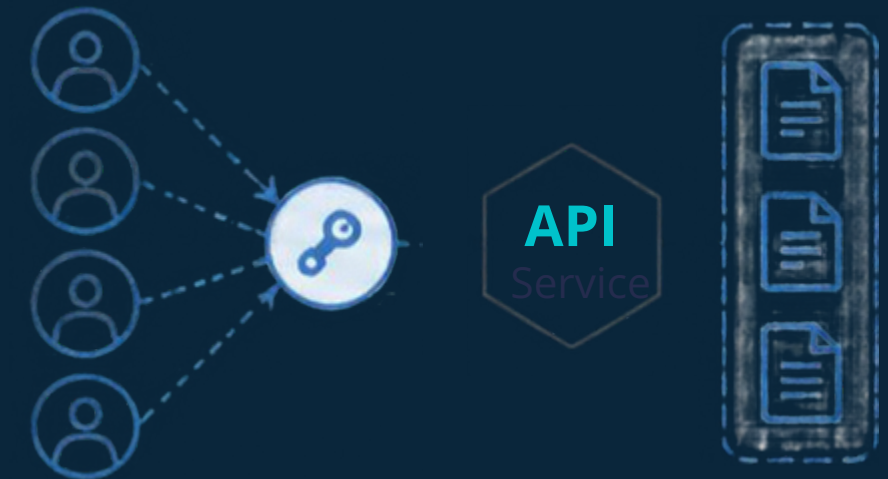
A user can extract it and call the API directly.



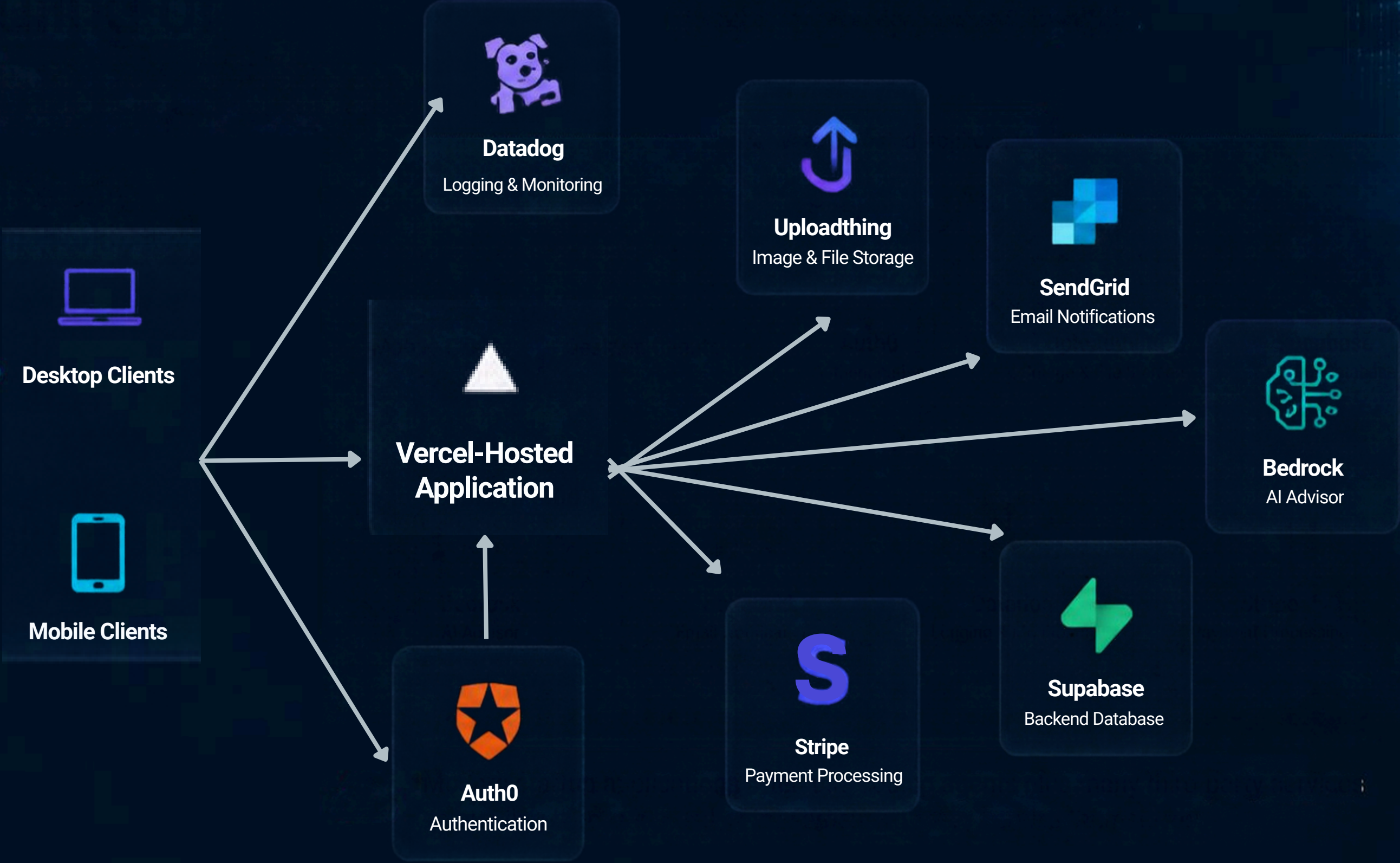
3

If there is only one token for the service, every user can do the same things

That can include reading data belonging to another user.



Improper Scoping and Handling of Authentication Tokens



Key Takeaways

- ✓ Review every third-party API key: what can it possibly be used for?
- ✓ Who has access to the key, and could a user extract it from the application?
- ✓ Can the service provide access control beyond of the API key?
- ✓ Do you need to move the integration behind your own server so you do not share the API key?

| Scope tokens to the minimum capability and trust the server, not the client.

Identity Access Management (IAM) **for** **Cloud Infrastructures**



Your Business Depends on IAM Cloud Credentials

→ Use strong authentication

Enable MFA everywhere and
use the strongest options available

→ Protect credentials

Securely store (password manager)
Avoid shared credentials
Enable continuous monitoring (e.g. GuardDuty,
Crowdstrike, Wiz, etc)

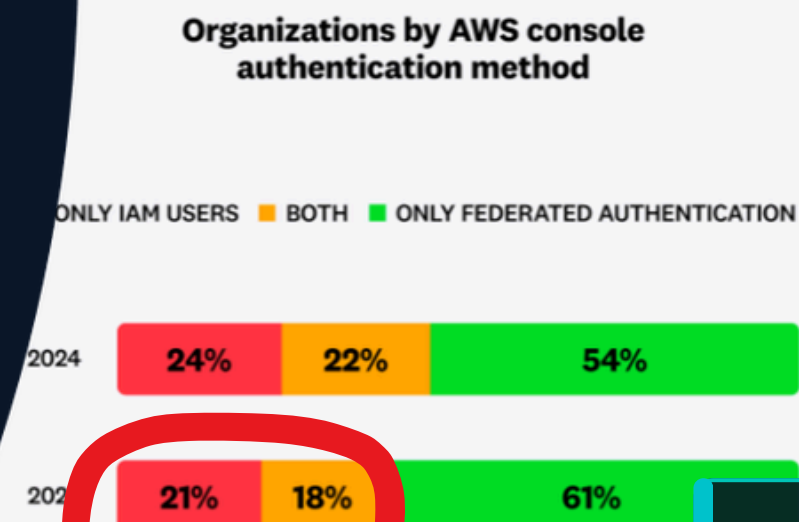
→ Avoid persistent/permissive credentials

Least Access, least privilege
Minimize long-lived credentials
Use temporary access instead

Long-lived cloud credentials remain pervasive and risky

Long-lived cloud credentials continue to pose a major security risk, with previous installments of this report having shown that they are the most common cause of publicly documented cloud security breaches. For this year's findings, we followed up on how organizations are using legacy versus modern authentication methods to authenticate humans and applications across AWS, Azure, and Google Cloud.

We found that most organizations (79%) continue to use federated authentication for human users to access the AWS console, such as through AWS IAM Identity Center or Okta. This represents a small increase in comparison to 2024, **up from 76%**, indicating that some organizations have transitioned fully to only federated authentication. However, almost **two in five (39%)** organizations still use IAM users in some capacity, and **one in five** organizations relies exclusively on IAM users. This shows that although organizations are increasingly adopting central human identities to access their cloud environments, they are struggling to get rid of their legacy IAM practices, such as humans using IAM users. It's easier to add a new secure authentication practice than to get rid of a risky legacy one, but to ensure proactive security and mitigate risks, it's critical to not only implement new solutions but also manage tech debt.



MANY ORGANIZATIONS STILL RELY ON IAM USERS

~40% of organizations on AWS still use IAM Users

Where do the persistent credentials end up?

- Your local system (your coding agent will likely suck it up at some point)
- Shared password manager entries
- Committed in repos!
- On the dark web

USE CASES PRODUCTS SERVICES COMPANY CONTENT | LOGIN

The Darknet Economy of Credential Data: Keys and Tokens

May 26, 2022

In this blog, we review how sensitive, server-side access credential data – such as AWS private/secret keys, Django secret keys, and API tokens – are captured, circulated, and sold across darknet marketplaces and criminal communities.

Darknet Background

The darknet, which is also referred to as the dark web, is a segment of the internet that is only accessible by using specialized software or network proxies. Due to the inherently anonymous and privacy-centric nature of the darknet, it facilitates a complex ecosystem of cybercrime and illicit goods and services trade. Adjacent to the darknet is the deep web and instant chat platforms that play an increasing critical role in facilitating this illicit information availability. Pseudo-anonymous discussion forums and vendor marketplaces hosted on the deep web along with Telegram private and public channels provide additional platforms by which threat actors communicate and circulate sensitive and stolen credential data.

<https://www.darkowl.com/blog-content/the-darknet-economy-of-credential-data-keys-and-tokens/>

Avoid Persistent Credentials and IAM Users in the cloud

Increased risk of compromise
Harder to rotate and manage

NOT RECOMMENDED



Persistent credentials

- Long-lived access keys
- IAM users for integrations and applications
- **If you get compromised, THIS IS PROBABLY HOW IT HAPPENS!**



RECOMMENDED



Temporary credentials with IAM roles

- Pipeline integrations (GitHub, GitLab, Bitbucket) do not require persistent creds
- Applications deployed in AWS use temporary creds via IAM roles

40 Minutes

153 GB File

433,909 files

TECH+TIMES

TECHSCIENCEBUSINESS TECHHEALTHCULTUREDEALS & REVIEWS

HOME > TECH > SECURITY

LiteLLM Supply Chain Hack Hit 2,488 Firms; Stolen Keys Still Work Five Months On

Researcher confirmed stolen March credentials still working at major US tech firm five months later

By Scott McCain

Published: Aug 14 2026, 7:51 AM EDT

f

X

in

o

F

♥

✉

💬



(PHOTO: LITELLM.AI)

Security researcher Kevin Beaumont got a call last week from a major US technology company that insisted it had nothing to worry about: all the credentials stolen in March's LiteLLM supply chain attack had been rotated. So Beaumont checked. "I tried them all. Almost every one worked," he wrote on Mastodon on August 13, 2026. "Submitted report. One of the biggest US techcos." He [confirmed this on his Mastodon timeline](#) after reviewing the company's responsible disclosure policy, which explicitly permitted credential testing.

That disclosure came the same day Hudson Rock, an Israeli cybersecurity intelligence firm, revealed it had obtained and fully analyzed a 153 GB RAR archive containing 433,909 files stolen during the March 2026 LiteLLM supply chain attack — the largest breach of AI infrastructure recorded in 2026. Of those files, the firm [attributed 118,829 CI/CD pipeline runner dumps](#) to 2,488 corporate domains, including NVIDIA, Volkswagen, Microsoft, FedEx, S&P Global, Cisco, Deloitte, and Siemens, among dozens of other global enterprises. The archive has not yet been

Credentials from 2,500 organizations

All stolen keys tested still worked 5 months later

Secure Access Methods

Using Temporary Credentials

Federated identities

Use federated identities and Identity Access Center for console and developer access. Assume roles to receive temporary tokens.

IAM roles

Use IAM Roles with temporary credentials embedded in SDKs for applications.

OIDC trust

For third-party integrations, rely on trusted authentication mechanisms like OpenID Connect (OIDC).

Temporary credentials reduce risk and strengthen security posture.

tl;dr IAM is Hard

- Get rid of persistent credentials
- Build permission sets and privileges to specific needs
aka LEAST ACCESS, LEAST PRIVILEGE
- If you must have persistent credentials, KNOW HOW TO ROTATE THEM (and do it periodically)
- Run tabletop exercises to uncover gaps - Gamedays are even better



Least privilege is easy
to say, **hard to do**

Ransomware

Extortion through data exfiltration, deletion, or encryption.



- Attackers exploit stolen credentials, phishing, or unpatched systems to gain access to cloud assets.
- They exfiltrate, delete, or encrypt data and disrupt cloud workloads and services.
- Attackers then demand payment to restore access or prevent data leaks.



Yes, your startup is a target



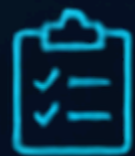
Tactical approach to being a bad guy

- Obtain credentials. A likely path: IAM User Access Key/Secret Key pair.
- Discovery/Enumeration
- PrivEsc
- Establish some other persistence if possible
- Impair ability to detect/defend/respond
- Exfil
- Encrypt/Delete



Protect

Strengthen your defenses by protecting cloud-based assets



Inventory all corporate assets and data: Can't protect it if you don't know where or what it is



Classify & tag data as Public, Internal, Confidential, or Restricted



Deploy endpoint protection on all workloads



Enforce MFA and passkeys everywhere.



Patch systems promptly and continuously



Train staff with security awareness and phishing exercises



Eliminate persistent credentials where possible

Recover & Respond

Restore operations and strengthen resilience for cloud-based assets



Recover & Respond

- Have some dang good backups
- Know how to restore data and infrastructure
- Define RPO and RTO for critical systems
- Log all access and keep logs immutable or uploaded to a separate account or service
- Use AI to accelerate infrastructure analysis and response planning
- Rehearse incident response with tabletop exercises or better yet, gamedays

Preparation and immutable backups are your strongest defense against ransomware.

Did we **stop** them?

Shhh...
nobody
saw me.



Business Continuity & Disaster Recovery

We maintain formal recovery objectives to ensure the availability and integrity of customer data.

Recovery Point Objective (RPO): 1 hour

In the unlikely event of a service disruption, we are able to restore customer data to within one hour of the incident. This is supported by continuous, automated backups of production data, retained for up to a year, and stored in immutable, tamper-proof storage — meaning that even in a ransomware event or a compromised credential scenario, our backups cannot be deleted, altered, or held hostage. Underlying storage is also redundantly replicated across multiple physically separate data center facilities for resilience against infrastructure failure.

Recovery Time Objective (RTO): 24 hours

We are able to fully restore business operations within 24 hours of a disruption. Our application services are designed to be quickly redeployable from version-controlled source, typically within 2 hours. Database recovery is the longest step in this process; we continue to invest in infrastructure improvements to further reduce this window.

Agentic Gap Analysis...

Requirement	Policy claim	Actual state	Gap
Continuous automated backups	✅ claimed	AWS Backup plan rds runs once daily (cron(30 0 ? * * *)), EnableContinuousBackup: false	Daily-only backup gives an RPO of up to ~24h, not 1h. Continuous backup (PITR via AWS Backup) is not enabled anywhere.
Retained up to a year	✅ claimed	Plan lifecycle is DeleteAfterDays: 365 — correct, but only for the one resource it actually covers	Retention config is right; coverage is the problem (next row).
Immutable / tamper-proof (ransomware/compromised-credential resistant)	✅ claimed	Only one backup vault () has Vault Lock enabled. Its backup selection is tag-conditional on AWSBackup=daily — only -cluster carries that tag. Every other prod database (— tagged "Production Database", service-production-aurora) has zero recovery points in the locked vault. They rely solely on native RDS automated backups, which are plain snapshots any credentialed principal (or an attacker with DB admin/RDS delete rights) can delete — not immutable.	Critical gap: the actual customer-data-bearing production databases are unprotected against the exact ransomware/compromised-credential scenario the policy claims to defend against.

...And the vibed code to align resource to policy

```
infra / terraform
├── .terraform
├── .terraform.lock.hcl
├── backup.tf
├── imports.tf
├── outputs.tf
├── providers.tf
├── rds.tf
├── README.md
├── s3_dr.tf
├── variables.tf
└── versions.tf
```


Key Takeaways

Traditional Application Weaknesses

- Prompt your coding agent to check for specific vulnerabilities
- Use security skills that are built in to your harness

Overpowered / Overprivileged Agents

- Design for non-determinism
 - Small Tasks
 - Limited Scope
 - Human Review
 - LLM review
 - Deterministic Gatekeepers

Improper Scoping and Handling of Tokens

- Know the powers of each token
- Protect the token accordingly

IAM for Cloud

- Least access, least privilege
- No persistent credentials!

Ransomware

- Know your data
- Protect your backups!
- Log everything
- Practice response and recovery
- Tabletops!
- Gamedays!



<https://www.meristeminfosec.com/blog/csw-secureaiapps>

Mark Hoopes

mark@meristeminfosec.com

Darren Weiner

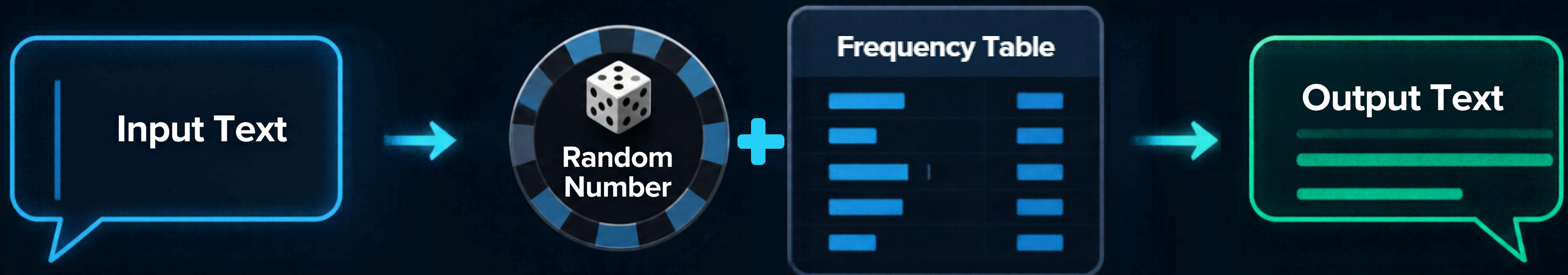
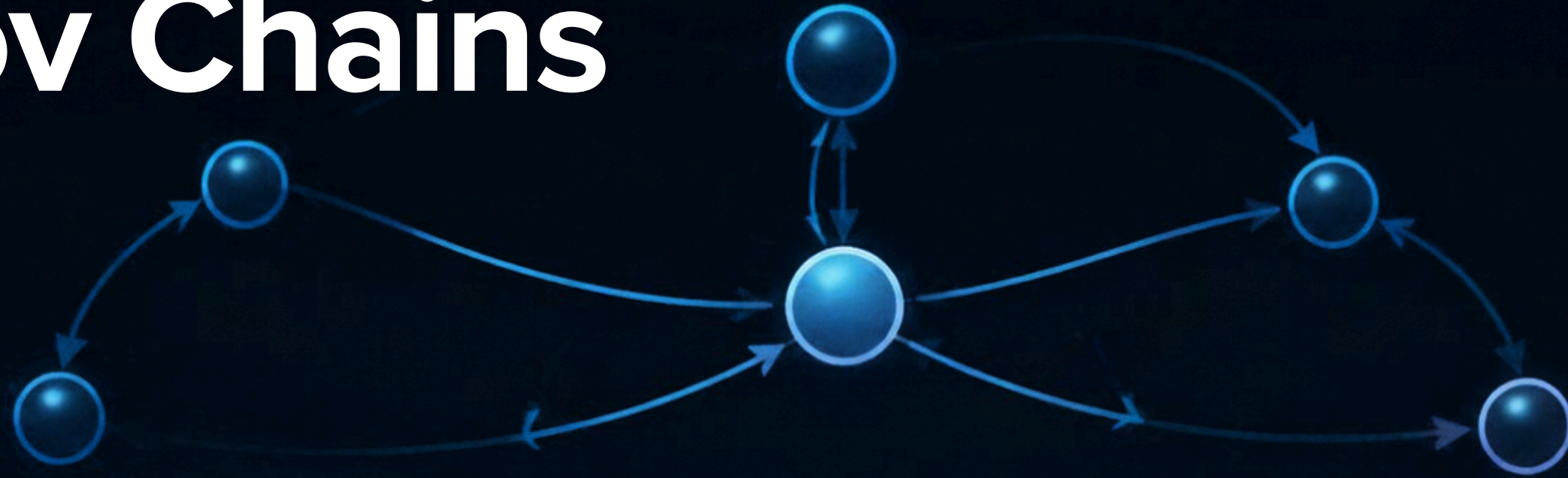
darren@cloudbutton.net

Scanning categories for a web app built by a coding agent, with no AI in the running product

	DETERMINISTIC (\$) Same input → same finding. Consider blocking in pipeline.	BOTH (\$\$) Deterministic and Non-deterministic versions/features Consider Human in the Loop	NON-DETERMINISTIC (\$\$\$) LLM-driven, varies by run. Consider human in the loop (HITL)
Source code	SAST OSS - CodeQL, Bandit Secrets scanning OSS - Gitleaks, TruffleHog, detect-secrets Comm - GitGuardian, Wiz, GitHub Push Protection	SAST OSS - Semgrep Comm - Snyk Code, Checkmarx One, Veracode	AI code review OSS - Qodo Merge (PR-Agent) Comm - CodeRabbit, Greptile, Copilot review, /claude-security
Dependencies & supply chain	SBOM & artifact integrity OSS - Syft, Sigstore/cosign, in-toto Comm - Anchore Enterprise, JFrog, Chainguard CI/CD pipeline security OSS - zizmor, Harden-Runner Comm - StepSecurity, Legit Security, Cypcode	Software Composition Analysis OSS - Trivy, Gype, OSV-Scanner, Snyk Comm - Mend, Socket	-
Cloud & infrastructure	IaC scanning OSS - Checkov, , KICS Container & image scanning OSS - Trivy, Gype, Dockle Comm - Snyk Container, Aqua, AWS Inspector	IaC scanning OSS - Trivy tfsec Comm - Snyk IAC, Wiz, Prisma Cloud CSPM OSS - Prowler, ScoutSuite, Cloud Custodian Comm - Wiz, Prisma Cloud, AWS GuardDuty	-
Running application	-	DAST OSS - OWASP ZAP, Burp Suite, Nuclei, sqlmap Comm - StackHawk, Bright Security	-

Representative list only. Runtime AI-security categories (prompt injection, guardrails, RAG, agent identity, MCP, model supply chain) are included.

Markov Chains



KJV Bible Frequency Tables

thou + ...

shalt	22.84%
hast	14.05%
art	5.85%
...	...
marvel	0.02%
heaven	0.02%

"thou shalt make thee but is born of that"

"thou the driver the people"

"thou be the jews"

thou shalt + ...

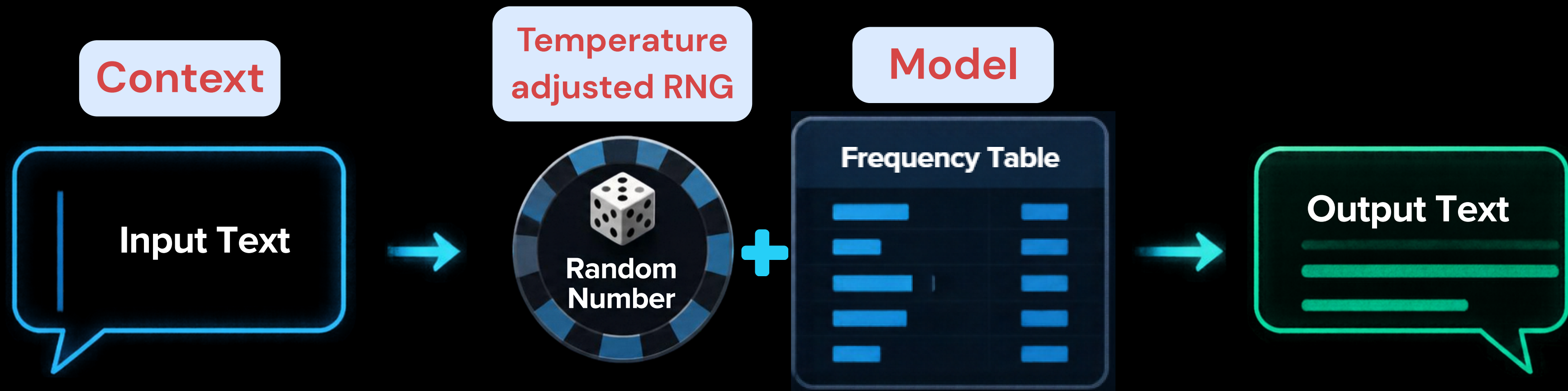
not	19.20%
be	7.68%
make	6.16%
...	...
confess	0.08%
suffer	0.08%

"thou shalt not eat it"

"thou shalt put them off and whose belly thou fillest"

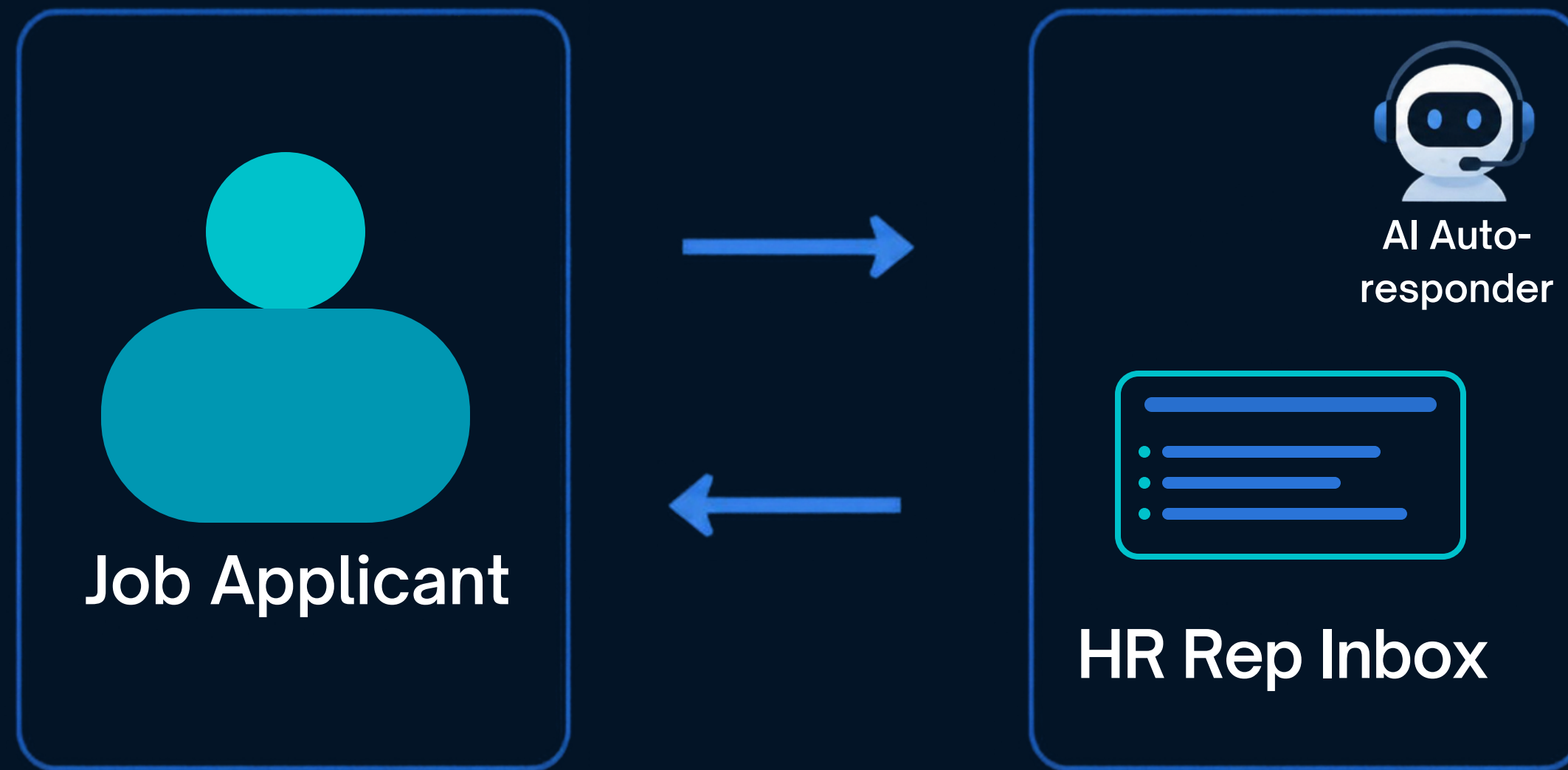
"thou shalt fall upon them and have committed against the house"

Markov Chains vs LLMs



	Markov Chains	LLMS
Context Size	10s	10,000s
Model Nodes	10,000s	Billions
Model Structure	Tabular	Dozens of interconnected layers

Example 1: Applicant Communication



Example 1: Applicant Communication

