

AUTONOMOUS VULNERABILITY
DISCOVERY & PATCHING

Constantine



The autonomous auditor that finds, proves, and patches.

Attacks

Attack Path

Status

- REJECTED
- DETECTED
- DETECTED
- DETECTED
- DETECTED
- DETECTED
- DETECTED

Code

```

overflows to a small value or zero. Subsequently, 'c + server->ready' in the candidate
selection loop (line 293) also overflows 'apz_uint32_t', corrupting the selection logic.
Since 'server->ready' is set directly from 'atoi(val)' with no upper bound, a value of
2^32 - 1 causes immediate wrap. The heartbeat file path is configurable and the file is read
with only APR_READ, meaning any process with write access to the heartbeat storage path
can trigger this. Furthermore, 'ap_random_pick(0, openslots)' with a wrapped 'openslot
s' value produces undefined behavior in candidate selection.
    
```

Remediation

Validate 'server->ready' and 'server->busy' values parsed from heartbeat data against a maximum (e.g., max worker thread count). Use checked arithmetic or saturating addition when accumulating 'openslots'. Check for overflow before 'c + server->ready'. Use 'strtol' with range checks instead of 'atoi'.

Verification Analysis TRUE POSITIVE

Reasoning

The vulnerability is confirmed as a true positive integer overflow/wraparound in 'mod_timestod_heartbeat.c', though significantly less severe than the CVSS 10.0 rating suggests.

Confirmed Vulnerable Code Path:

- 'readfile_heartbeats()' reads heartbeat file and calls 'atoi(val)' to parse 'ready' field, assigning directly to 'server->ready' (typed as 'int') with NO upper-bound validation.
- 'read()' (slotmem path) also assigns 'server->ready = slotserver->ready' with

INSIDE THE GUARD PLATFORM

AUTONOMOUS CODE AUDITOR // AGENTIC LLM PIPELINE

Find fatal bugs. Prove they're real. Ship the fix.

Point Constantine at a source code repository and get back confirmed, exploitable vulnerabilities. Every finding ships with a patch **already validated against the original exploit**. No human intervention between stages — clone to finding to tested patch runs as a single command inside the Praetorian Guard Platform. Constantine changes the economics of source code security from *hire experts and wait weeks to run a pipeline and get results*.

WHY CONSTANTINE IS DIFFERENT

PATCHING > DISCOVERY // 01

Patching matters 3x more than discovery.

Finding a bug is easy. Proving it is real and shipping a correct fix is hard. Every Constantine patch is re-tested against the original exploit before it ships.

NO SPECULATION // 02

Accuracy multipliers punish guessing.

Multiple independent verification stages filter findings so what survives is real. Noise is engineered out of the pipeline — not tuned out after the fact.

THE SIX-STAGE PIPELINE — CLONE → PATCH IN ONE COMMAND

<p>01 · INGEST →</p> <h3>Understand</h3> <p>Clone the repo. Score files 0-100 for security relevance. Focus analysis where it matters.</p>	<p>02 · DETECT →</p> <h3>Scan</h3> <p>LLM scanners flag suspicious patterns, then validate them with deep CWE-specific analysis.</p>	<p>03 · REVIEW →</p> <h3>Verify</h3> <p>Agentic LLM verifier traces data flows. Taint-aware call graphs confirm reachability.</p>	<p>04 · EXPLOIT →</p> <h3>Prove</h3> <p>Sandboxed agent writes, executes, and iterates exploits until the bug is proven real.</p>	<p>05 · PATCH →</p> <h3>Fix</h3> <p>Tiered repair strategy. Every patch re-runs the exploit. Only validated fixes ship.</p>	<p>06 · REPORT</p> <h3>Deliver</h3> <p>Structured output with full evidence chain. CVSS and CWE included on every finding.</p>
--	--	---	---	---	--

INSIDE THE GUARD PLATFORM · BENCHMARKS · OPERATING MODES

A native capability of Praetorian Guard.

Constantine runs inside the platform you're already paying for — not a separate tool, not a separate contract, not a separate dashboard.

One platform. One workflow. One source of truth. Constantine findings flow directly into the Guard dashboard as Risks, alongside the output of every other Guard capability — Attack Surface Management, Continuous Penetration Testing, Vulnerability Management, Breach and Attack Simulation, and Cyber Threat Intelligence.

Every Risk carries the full evidence chain: detection reasoning, reviewer verdict, exploit proof, and validated patch. You triage, assign, and remediate with the same workflow you already use — the discovery engine underneath just got an order of magnitude better.

GUARD PLATFORM — CAPABILITY STACK

- > Attack Surface Management
- > Continuous Penetration Testing
- > Vulnerability Management
- > Breach and Attack Simulation
- > Cyber Threat Intelligence
- > **Constantine — Code Auditor**

Proven against real-world CVEs.

28 CONSEQUENTIAL VULNERABILITIES · 6 LANGUAGES

Constantine is benchmarked against **28 of the most consequential vulnerabilities** in modern software — spanning memory safety, authentication bypass, cryptographic flaws, container escapes, and denial of service across C, Java, Python, Go, Rust, and JavaScript. A representative sample:

CVE-2021-44228	Log4Shell · RCE via JNDI lookup in Apache Log4j	CVE-2014-0160	Heartbleed · Missing bounds check in OpenSSL TLS
CVE-2022-0847	Dirty Pipe · Linux kernel privilege escalation	CVE-2021-3156	Baron Samedit · Heap overflow in sudo
CVE-2024-6387	regreSSHion · Signal handler race in OpenSSH	CVE-2022-21449	Psychic Signatures · ECDSA verification bypass in Java
CVE-2023-4911	Looney Tunables · Buffer overflow in glibc	CVE-2024-21626	runc Container Escape · FD leak to host filesystem

TWO WAYS TO PUT IT TO WORK

MODE 01 · PRIVATE

Customer Private Repositories

Connect Constantine to a private repo. Findings land in the Guard dashboard as **Risks**, integrated with your existing triage and remediation workflows. Every Risk carries the full evidence chain.

detection

reviewer verdict

exploit proof

validated patch

MODE 02 · PUBLIC

Make Me Famous

Point Constantine at any public repository — OpenSSL, nginx, Node.js, curl. A Praetorian analyst reviews the results and coordinates **responsible disclosure**. Offense informing defense, at scale.

OpenSSL

nginx

Node.js

curl

+ your pick

Constantine works on **any git repository** — public or private, GitHub or GitLab, monorepo or single service. Scan an entire repo at a ref, a specific pull request, or a commit range. The LLM scanners are language-agnostic and have been validated against major production languages.

COVERAGE

ANY REPO, ANY REF

- C

C++

Java

Python

Go

Rust

JavaScript

See Constantine inside Guard.

Book a scoping call — we'll scan a repo of your choosing on the call.

[REQUEST A GUARD DEMO →](#)