
Black October A Blue Team Field Guide

Written by a blue teamer, for blue teamers.

October 22, 2026

AegisGate Security — Open Source (Apache 2.0)

Free · No registration · No paywall · No upsell

AegisGateSecurity.io

TL;DR

The internet is under coordinated attack. Six critical vulnerabilities dropped in 48 hours — two in the PHP runtime itself, one in a TLS certificate authority, one in GitLab, a supply chain compromise in an AI gateway, and a Chinese APT using AI autonomous pentesting tools against 2,250+ Microsoft 365 tenants. **Patch everything. Then close the AI agent blind spot you almost certainly don't know you have.** This guide tells you how to do both, using free open source tools you can deploy today — including in air-gapped and regulated environments.

What Is Happening

If you're reading this, you already know. The cybersecurity community is calling it "Black October" — a 48-hour window where the internet's foundational technologies started falling like dominoes. Here's the situation as of right now:

Incident	What Broke	How Bad
PHP2Shell	Two critical RCE (Remote Code Execution) flaws in the PHP runtime itself (not a plugin — the language)	CVSS 9.2 × 2. PHP runs ~80% of all websites. 5.1M hosts exposed. Attackers were exploiting it 4 days before patches existed .
eMudhra CA	Indian certificate authority had 8 intermediate TLS certs with private keys so weak one was literally the number 4	Anyone could forge trusted HTTPS certificates since April 2025. Node.js and Bun are still unpatched as of this writing.
ConnectWise / Storm-0940	Chinese APT used stolen ConnectWise credentials + AI autonomous pentesting tool (ARTEX) to hit 2,250+ Microsoft 365 tenants	Initial targeting to privileged cloud access in under 3 hours . AI models divided tasks and reviewed each other's attack work.
GitLab RCE	Integer overflow in the Regex Compiler — unauthenticated RCE	CVSS 10.0. 2,700+ exposed Community Edition instances. No login required to exploit.
LiteLLM Supply Chain	AI gateway/proxy compromised — steals passwords, secrets, deploys a RAT (Remote Access Trojan)	This is AI infrastructure being attacked. Your AI gateway could be exfiltrating your credentials right now.
SharePoint Zero-Day	Microsoft's October patches didn't fix it — now under active attack	On-prem SharePoint, zero-day, in the wild. Microsoft's fix failed.

What is CVSS? Common Vulnerability Scoring System. A 0–10 scale. 9.0+ is "Critical." 10.0 is "as bad as it gets."

This is not a drill. This is not a "watch and see." If you run PHP servers, Microsoft 365 tenants, GitLab instances, or AI tooling — you are in the blast radius.

Step 1: Patch Everything You Can (Do This First)

Before anything else. No security tool patches your servers for you. This is on you and your team.

PHP2Shell (CVE-2026-109618, CVE-2026-109619)

If you run PHP, this is your highest priority. Full stop.

Patch to one of these versions immediately:

- PHP 8.5.12
- PHP 8.4.27
- PHP 8.3.36
- PHP 8.2.35

Running PHP 7.4, 8.0, or 8.1? The PHP team provided source backports. Contact your distro vendor for patched packages, or build from source using the backport patches linked on the PHP blog.

Can't patch right now?

- The UK NCSC says: **take your PHP servers offline** until you can patch.
- There are NO reliable mitigations. WAF rules help but attackers are already adapting payloads to bypass them. Cloudflare deployed WAF rules and attackers are getting past them within hours.
- Disable `session.upload_progress` (`session.upload_progress.enabled = Off` in `php.ini`) — this closes one of the two attack paths.
- Set a conservative `memory_limit` (e.g., `memory_limit = 128M`) and reduce `post_max_size` — this limits exposure to memory-based exploitation.
- **These are stopgaps. Patch.**

Already patched? You're not done.

- Check your logs going back to **at least October 17**. Amazon detected exploitation attempts 4 days before patches were released. If you were running PHP before October 21, assume you were a target.
- Look for: unusual POST requests with `multipart/form-data` , unexpected process spawning from your web worker, outbound connections to unfamiliar IPs, any evidence of shell execution.

What makes this especially bad: On Apache with `mod_php` , the exploit fires *before your application code even runs*. Your app framework, your input validation, your sanitization — none of it matters. The vulnerability is in the language runtime itself.

Runtime-specific note: If you deploy via FrankenPHP, LiteSpeed, or php-cgi, the same PHP patches apply — check with your runtime vendor for patched container images or packages.

eMudhra CA Compromise (CVE-2026-111943)

If your software trusts TLS certificates, it may still trust the compromised eMudhra intermediates. A script error in April 2025 generated P-384 private keys so weak that one was literally the integer **4**. That means anyone could forge trusted HTTPS certificates for 18 months. Update your trust stores:

Software	Action
Firefox	Update to 158.0.1+ (or ESR equivalent)
Chrome	Should auto-update; verify you're on the latest
Safari / macOS	Apply the latest macOS security update
Go	Update to 1.27.3 or 1.26.10
Python	<code>pip install --upgrade certifi</code> (should be 2026.10.19+)
pip	Update to 26.2.2+
curl	Update your CA bundle (2026-10-20 or later)
Rust	Update <code>rustls-webpki</code> to 0.103.16+, <code>webpki-roots</code> to 1.0.10+
Node.js	CHECK — patch was still pending as of this writing
Bun	CHECK — patch was still pending as of this writing
Electron	Apply latest Electron update (patched)
macOS built-in curl	<code>/usr/bin/curl</code> may still trust compromised certs. Test using Mozilla's incident page endpoints.
Ubuntu/Debian	<code>apt update && apt install ca-certificates</code>
Docker images	Rebuild with updated base images. Old <code>ca-certificates</code> packages in your base layers still trust the compromised CA.

Quick test: Mozilla published test endpoints for this incident. Hit them from your servers. If they connect successfully, your trust store is still vulnerable. See [Mozilla's CA Incident Dashboard](#).

For the IT generalist: If you're not sure how to check your Node.js or Bun trust store, the simplest approach is to update to the latest version and then check the release notes. If a patch is available, it will be mentioned there. If no patch exists yet, treat any TLS connections from Node.js/Bun applications as potentially vulnerable until one ships.

GitLab (CVE-2026-110264)

Patch to one of: 19.5.1, 19.4.3, 19.3.6, 19.2.9, 19.1.9, 19.0.10, 18.11.13.

If your GitLab is internet-facing and you can't patch right now, restrict access at the network level (firewall rules, security groups, VPN-only access). This is an **unauthenticated** RCE — no login required. Anyone who can reach your GitLab instance can potentially execute arbitrary code on it.

SharePoint Zero-Day

Microsoft's October patches were supposed to fix this. They didn't. Watch for Microsoft's out-of-band patch. In the meantime:

- Restrict network access to on-prem SharePoint servers
- Monitor IIS logs for anomalous requests
- If possible, require VPN for SharePoint access until a patch ships

ConnectWise / Storm-0940

If you use or used ConnectWise services (Cloud Backup, SaaS Security, Cloud Management):

- **Revoke and rotate all credentials** associated with ConnectWise integrations — API keys, service account passwords, OAuth tokens
- Audit Microsoft Graph logs for anomalous activity — look for rapid authentication attempts (Storm-0940 hit 2,900 attempts/minute), application credential changes, privilege modifications
- Check for persistent access that may have been established before credentials were revoked: new OAuth app registrations, service principals, mailbox forwarding rules, federation trust modifications
- Review OAuth app permissions across your M365 tenant — remove anything you don't recognize

What is Microsoft Graph? The API layer that ConnectWise (and many other tools) use to access Microsoft 365 data — mail, files, calendar, contacts. If Storm-0940 got Graph API access, they could read mail, exfiltrate files, and persist through OAuth apps.

For DBAs: If database credentials were accessible to ConnectWise integrations or routed through any AI gateway (including LiteLLM), rotate them. Treat any credential that touched an external integration or AI proxy as potentially compromised.

Step 2: Understand the AI Agent Blind Spot

Here's why this October is different from every other bad month in cybersecurity.

AI is now both the weapon and the target.

The Weapon: ARTEX

ARTEX is an open-source AI autonomous penetration testing system. It was created on **October 8, 2026** — two weeks ago. It's a multi-agent LLM system (written in Go + Next.js) that automates reconnaissance, vulnerability discovery, credential attacks, and data theft. It won a Baidu AI attack-defense competition.

Chinese APT Storm-0940 combined ARTEX with:

- **Claude Code** (Anthropic's coding agent)
- **Codex** (OpenAI's coding tool)
- **Grok** (xAI)
- **DeepSeek** (via OpenAI-compatible gateway)
- **GLM** (Zhipu AI)

The AI models **divided investigation tasks and reviewed each other's findings**. GLM reviewed work performed by DeepSeek and its subagents. Custom scripts enabled handoffs between AI sessions — data from one AI tool fed into the next.

They went from initial targeting to **privileged cloud access in under 3 hours**.

This is not a theoretical future threat. This is happening right now. The original ARTEX repository has been deleted, but there are 800+ forks with 1,600+ stars across copies. The tool is freely available. Microsoft observed attackers from "several geographic regions" using the framework — not just Storm-0940.

What is an APT? Advanced Persistent Threat — typically a nation-state or well-resourced cybercrime group that maintains long-term access to targets. Storm-0940 is a Chinese APT tracked by Microsoft.

What is MCP? Model Context Protocol — an open standard that lets AI agents connect to external tools and data sources. ARTEX supports MCP (Streamable HTTP and SSE). If you're running MCP servers, ARTEX can talk to them.

The Target: AI Infrastructure

LiteLLM — an AI gateway used to route calls to 100+ LLM APIs — was compromised via supply chain attack. It was stealing passwords, secrets, and deploying a RAT. This wasn't LiteLLM's first incident: a

malicious `litellm_init.pth` credential stealer appeared in June 2026, followed by an SSRF vulnerability in August and a privilege escalation in September.

Your AI gateway, your AI proxy, your MCP server — these are now attack targets. And most teams don't even know they have them, let alone monitor them.

If you're running LiteLLM: Check for `litellm_init.pth` in your Python `site-packages` directories. Look for unexpected outbound connections from the LiteLLM process. Rotate ALL credentials that passed through LiteLLM — API keys, service account passwords, database connection strings. Consider replacing LiteLLM with AegisGate Platform Community tier (free, air-gapped) as your AI gateway.

The Blind Spot

Most SOC teams have **zero visibility** into AI agent traffic. You can see network traffic. You can see API calls in CloudTrail. But can you see:

- What your coding assistant (Cursor, Copilot, Claude Code) is sending to the LLM?
- Whether source code with hardcoded credentials is being sent to an external AI?
- Whether your MCP servers are being probed for vulnerabilities?
- Whether a compromised AI agent is exfiltrating data through API calls?
- Whether someone is using ARTEX-like tools against your infrastructure?

If the answer is "no" to any of those — you have a blind spot. And Black October just proved that blind spot is being actively exploited.

For the IT generalist: If your developers use GitHub Copilot, Cursor, or Claude Code, they're sending code to external AI servers. If anyone on your team uses ChatGPT to "debug" a config file, they might be sending credentials to OpenAI. If you run MCP servers for AI agents, those servers might be accessible from the internet. All of this is happening outside your normal security monitoring. That's the blind spot.

Step 3: Close the AI Agent Gap

Everything in Levels 1–4 below is **completely free, open source (Apache 2.0), with no feature gating or paid tiers**. Level 5 (Platform) has a free Community tier plus paid Pro+ tiers — I'll be upfront about what's free and what isn't.

Not sure where to start? Use this table:

If you...	Start at	Time
Use ChatGPT/Claude/Gemini in a browser	Level 1	2 min
Use Copilot/Cursor/Claude Code in an IDE	Level 1 + 2	12 min
Build AI apps with RAG pipelines or agents	Level 1 + 2 + 3	27 min
Run MCP servers for AI agents	Level 4	20 min
Need org-wide AI traffic visibility	Level 5	Hours
Aren't sure what any of this means	Level 1 (it's a browser extension)	2 min

Pick the level that matches your situation. Start small. Scale up. All tools run fully air-gapped — no cloud dependency, no phone-home, no telemetry by default.

Level 1: Browser Extension — For Everyone (2 minutes)

Tool: [AegisGate Lens](#)

If you're using ChatGPT, Claude, Gemini, or any AI chat in your browser — install Lens. It's a browser extension that runs **100% on-device**. Zero data leaves your browser. No telemetry by default. It catches:

- **PII:** SSNs, credit cards (Luhn-validated), emails, phone numbers, addresses, driver's licenses, passports, tax IDs, bank accounts, IPs (69 patterns)
- **Secrets:** API keys (AWS, GitHub, OpenAI, Stripe, Slack), OAuth tokens, RSA private keys, database credentials (41 patterns)
- **XSS:** Cross-site scripting payloads in AI-generated code (12 patterns)
- **Compliance:** HIPAA, GDPR, CCPA, EU AI Act, NIST CSF, ISO 27001 indicators (24 patterns)
- **OT/ICS:** Modbus, DNP3, OPC-UA control manipulation (9 patterns)
- **Prompt injection:** ML-based detection (CharCNN-BiLSTM) for adversarial attacks that regex can't catch

209 regex patterns + ML detection. Sub-millisecond regex. ~5–50ms ML inference. All in your browser.

How to install:

- **Chrome:** [Chrome Web Store](#) — requires Chrome 116+
- **Firefox:** [Firefox Add-ons](#) — requires Firefox 142+

That's it. Install it. It warns you before you hit send if you're about to leak sensitive data to an AI company. It also catches prompt injection attempts in AI responses. Zero npm dependencies. Zero data collection.

Who this is for: Every person on your team who uses AI chat tools. Your developers, your analysts, your help desk, your executives. If they type things into ChatGPT, they need this.

Black October relevance: In a month where credential compromise is the primary attack vector (ConnectWise, LiteLLM, Storm-0940), preventing credential exposure is defense in depth. If someone on your team pastes a config file with API credentials into ChatGPT to "debug" something, Lens stops them before they hit send.

Level 2: Local Proxy — For Developers Using AI Coding Tools (10 minutes)

Tool: [AegisGate Rampart](#)

If your developers use GitHub Copilot, Cursor, Claude Code, or local LLMs (Ollama, LM Studio) — Rampart sits between their editor and the AI model. It's a local HTTPS MITM (Man-in-the-Middle) proxy that scans everything going out and everything coming back.

211 regex patterns + CharCNN-BiLSTM ML detection. Monitor mode (just watch and log) or Block mode (stop sensitive data from leaving). Zero npm dependencies. Zero telemetry by default. Runs fully air-gapped.

Quick start:

```
# Option 1: Download pre-built binary from GitHub Releases
# https://github.com/aegisgatesecurity/aegisgate-rampart/releases/tag/v0.9.1

# Option 2: Build from source
git clone https://github.com/aegisgatesecurity/aegisgate-rampart
cd aegisgate-rampart
go build -o rampart ./cmd/rampart

# Run in monitor mode first (see what's leaking before you block)
./rampart --mode monitor

# When you're ready to enforce, switch to block mode
./rampart --mode block
```

There's also a VS Code / Cursor extension and a JetBrains plugin (IntelliJ, PyCharm, etc.) that shows inline warnings right in the editor.

Who this is for: Development teams using AI coding assistants. Especially relevant if your developers are using AI to write code that touches the same systems being attacked in Black October.

Black October relevance: If an attacker compromises your AI coding tool (like the LiteLLM supply chain attack), Rampart is a layer that sees the exfiltration attempt. Even if the tool itself is compromised, Rampart inspects the actual HTTP traffic leaving the machine. It also prevents developers from accidentally sending credentials, source code, or PII to external AI services — which is how credentials end up in the wrong hands.

For the IT generalist: "MITM proxy" sounds scary, but it just means Rampart intercepts the traffic between your editor and the AI server, inspects it, then forwards it (or blocks it). Your developers configure their IDE to send AI requests through Rampart instead of directly to OpenAI/Anthropic. Rampart checks each request for secrets and sensitive data before it leaves the building.

Level 3: Static Analysis — For CI/CD Pipelines (15 minutes)

Tool: [AegisGate Scanner](#)

322 regex patterns + CharCNN-BiLSTM ML. Scans AI artifacts — RAG documents, training data, model configs, agent skills, prompt templates — for prompt injection (30 patterns), secrets (64), PII (85), dangerous code (34), compliance violations (64), OT/ICS (9), RAG attacks (17), and prompt leakage (16). Outputs SARIF, SBOM (SPDX), JSON, and 5 other formats.

Quick start — one-line installer:

```
curl -fsSL https://raw.githubusercontent.com/aegisgatesecurity/aegisgate-scanner/main/install.sh | bash
```

Quick start — build from source:

```
git clone https://github.com/aegisgatesecurity/aegisgate-scanner
cd aegisgate-scanner
go build -o scanner ./cmd/scanner

# Scan a directory of AI artifacts
./scanner scan ./ai-artifacts/ --format sarif --output results.sarif

# Or use Docker (no Go installation needed)
docker run --rm -v $(pwd):/scan ghcr.io/aegisgatesecurity/scanner:latest scan /scan -format json
```

Quick start — GitHub Action (scans every PR automatically):

```
# .github/workflows/ai-security.yml
name: AI Artifact Security Scan
on: [push, pull_request]
jobs:
  scan:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: aegisgatesecurity/aegisgate-scanner@v0.4.1
        with:
          path: ./ai-artifacts/
          format: sarif
```

The SARIF output integrates directly with GitHub Security tab, so findings show up right in your pull requests.

Who this is for: Teams building AI applications, maintaining RAG pipelines, or deploying AI agents. If you're shipping AI artifacts, scan them before they reach production.

Black October relevance: The LiteLLM attack was a supply chain compromise. Scanner catches malicious patterns in AI artifacts before they reach production. If you're pulling AI configs, prompt templates, or agent skills from third parties, scan them. The 322 patterns include prompt injection, credential exfiltration, and dangerous code execution — exactly the techniques ARTEX automates.

For AI developers: If you're ingesting third-party documents into a RAG pipeline, a poisoned document can inject malicious prompts that exfiltrate secrets or manipulate model behavior. Scanner's 17 RAG attack patterns catch this before ingestion. If your agents talk to each other (A2A), an attacker who compromises one agent can use it to attack the others — just like ARTEX did. Scan agent skills and configs before deployment.

Level 4: Secure MCP Servers — For Teams Running MCP-Based Agents (20 minutes)

Tool: [AegisGate MCP](#)

If you're running MCP (Model Context Protocol) servers — whether it's a custom tool server, a data connector, or an agent orchestration layer — AegisGate MCP is a hardened MCP server framework with 23 security layers built in. 79 regex patterns + CharCNN-BiLSTM ML detection. Zero external dependencies (pure Go stdlib). Distroless Docker image with zero CVEs. Runs fully air-gapped.

Why this matters: 38% of MCP servers have no authentication. There have been 590+ security advisories and 3 critical CVEs (including a CVSS 9.8 RCE) in the official MCP SDKs in the last 6 months. If you're running a bare MCP server, you are exposed.

Quick start:

```
git clone https://github.com/aegisgatesecurity/aegisgate-mcp
cd aegisgate-mcp

# Docker (recommended)
docker-compose up -d

# Or build from source
go build -o aegisgate-mcp ./cmd/server
./aegisgate-mcp
```

Who this is for: Any team running MCP-based AI agents. If you've got Cursor or Claude Desktop connecting to local MCP servers, or you're building custom MCP tool servers, this replaces the bare SDK with a hardened framework that adds authentication, RBAC, audit logging, rate limiting, and 19 other security layers.

Black October relevance: ARTEX supports MCP (Streamable HTTP and SSE). If attackers are using MCP-capable AI attack tools, your MCP servers need to be secured. AegisGate MCP adds the security layers that bare MCP servers don't have — authentication, audit logging, threat detection, and more.

Level 5: Enterprise Gateway — For Organizations Running AI at Scale (hours)

Tool: [AegisGate Platform](#)

This is the full system. A self-hosted gateway that sits between your organization and any AI service. Every request and every response is scanned — across 6 protection pillars: HTTP, MCP, A2A (Agent-to-Agent), ACP (Agent Communication Protocol), Response, and Trust.

What's included in the free Community tier:

- 256 canonical regex detection patterns (L1): secrets, XSS, PII, compliance violations
- HTTP / MCP / A2A / ACP / Response protocol scanning
- 4 community compliance frameworks (OWASP LLM Top 10, MITRE ATLAS, NIST AI RMF, ISO 27001)
- In-memory and file persistence
- Fail-closed by default (if it can't scan, it blocks)
- Zero external runtime dependencies
- Air-gapped capable

What requires Pro+ (paid):

- ML anomaly detection (CharCNN-BiLSTM, L3) — the badge says it plainly: "ML Detection Pro+ only"
- SIEM integration (Splunk, ELK, Sentinel forwarding) — "SIEM Pro+ only"
- Additional compliance frameworks (FedRAMP, HITRUST, TISAX, CMMC L2, EU AI Act 120 controls, and 20+ others)
- HSM, FIPS mode, Kubernetes clustering

Performance (from load testing): 23,578 RPS with 0% errors at 5,000 VUs. -2.8ms p99 overhead (yes, negative — caching makes it faster than direct connections). Blocked requests resolve in 7.2ms p50.

Quick start:

```
git clone https://github.com/aegisgatesecurity/aegisgate-platform
cd aegisgate-platform

# Build the binary
go build -o aegisgate-platform ./cmd/aegisgate-platform/

# Generate a config file automatically (detects your environment)
./aegisgate-platform setup --non-interactive

# Start the platform
./aegisgate-platform --config aegisgate-platform.yaml --embedded-mcp
```

The setup wizard auto-detects Docker, Kubernetes, systemd, or bare metal. It recommends a deploy profile, generates a validated config, and prints next steps. No YAML editing required.

Who this is for: Security teams that need enterprise-wide visibility and control over AI traffic. If you have 50+ users hitting AI services, or you're running production AI agents, or you need audit logs for compliance — this is your layer.

Black October relevance: Platform's 256 regex patterns scan every AI API request and response for credential exfiltration, PII leakage, and prompt injection in real-time. The free Community tier gives you protocol-level visibility into HTTP, MCP, and A2A traffic — the exact channels ARTEX and compromised AI agents use. If an attacker — or a compromised internal agent — starts exhibiting ARTEX-like behavior (rapid reconnaissance across endpoints, credential attacks through API calls), Platform's regex detection catches the exfiltration patterns. The Pro+ ML anomaly detection adds behavioral detection for novel attacks that don't match known patterns — but even without it, the 256 regex patterns provide substantial coverage.

Honest note on pricing: Levels 1–4 are 100% free with no paid tiers, no feature gating, no "contact us for the real version." Platform's Community tier is also free and includes the 256 regex patterns and protocol scanning. The ML anomaly detection and SIEM integration are Pro+ features. If you need those during Black October and can't wait for a procurement cycle, reach out — but the regex detection in the free tier is the most important part for this crisis.

Air-gap note: Most AI security tools are SaaS — they require your AI traffic to route through their cloud. If you're in OT, defense, government, or regulated environments, that's a non-starter. AegisGate Platform, Rampart, Scanner, and MCP all run fully air-gapped. All detection is local. No cloud dependency, no phone-home, no telemetry by default. This is not a feature — it's a requirement for the environments we serve.

Step 4: Use AegisGate Detection Patterns in Your Existing Stack

This section is for teams that already have a SIEM or WAF and can't deploy new tools right now. You can still use our detection patterns — they're all open source.

What is a SIEM? Security Information and Event Management — a platform like Splunk, ELK, or Microsoft Sentinel that collects and analyzes log data across your organization. If you don't know what Splunk, ELK, or Sentinel are, you don't have a SIEM — skip to Step 5.

What is a WAF? Web Application Firewall — inspects HTTP traffic and blocks malicious requests. Cloudflare, AWS WAF, and Imperva are common examples.

SIEM Integration (Splunk, ELK, Sentinel)

Our regex patterns are defined in Go source files, but the patterns themselves are standard regex. You can extract them and import into your SIEM as detection rules. These regex patterns can also be wrapped in Sigma rules for automated deployment to most SIEM platforms.

Here's where to find them:

Lens patterns (209 regex patterns + ML): [aegisgate-lens/src/bootstrap.js](#)

- PII: 69 patterns (SSN, credit cards, emails, phones, addresses, passports, tax IDs, bank accounts)
- Secrets: 41 patterns (AWS, GitHub, OpenAI, Stripe, Slack, RSA private keys, JWTs, OAuth tokens)
- XSS: 12 patterns
- Compliance: 24 patterns (HIPAA, GDPR, CCPA, EU AI Act, NIST CSF, ISO 27001)
- OT/ICS: 9 patterns (Modbus, DNP3, OPC-UA)
- Additional patterns: 54 (prompt injection indicators, suspicious content, edge-case variants)

Platform patterns (256 canonical regex patterns): [aegisgate-platform/pkg/scanner/patterns.go](#) and [pkg/contracts/pattern.go](#)

- All of the above plus agent-specific patterns: prompt injection, agent impersonation, tool abuse, trust boundary violations
- Pattern contracts defined in [pkg/contracts/pattern_v1.json](#) (machine-readable JSON format)

Scanner patterns (322 regex patterns + ML): [aegisgate-scanner/pkg/detectors/](#)

- Prompt injection: 30 patterns
- Secrets: 64 patterns
- PII: 85 patterns (US + International)
- Dangerous code: 34 patterns
- Compliance: 64 patterns
- OT/ICS: 9 patterns

- RAG attack: 17 patterns
- Prompt leakage: 16 patterns

How to use them: Clone the repos, find the pattern files listed above, extract the regex strings, and create detection rules in your SIEM. Here are practical examples:

Splunk — credential exfiltration detection:

```
index=proxy_or_firewall
| regex _raw="AKIA[0-9A-Z]{16}"
| regex _raw="gh[pousr]_[A-Za-z0-9]{36}"
| regex _raw="sk-[A-Za-z0-9]{20}"
| regex _raw="-----BEGIN RSA PRIVATE KEY-----"
| regex _raw="xox[baprs]-[A-Za-z0-9-]+"
| table _time, src_ip, dest_host, _raw
```

Splunk — PII leakage to AI APIs:

```
index=proxy_or_firewall dest_host IN
("api.openai.com","api.anthropic.com","generativelanguage.googleapis.com","api.groq.c
om")
| regex _raw="\b\d{3}-\d{2}-\d{4}\b"
| regex _raw="\b4[0-9]{12}(:[0-9]{3})?\b"
| regex _raw="\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b"
| table _time, src_ip, user, _raw
```

ELK / OpenSearch — credential detection in AI traffic:

```
{
  "query": {
    "bool": {
      "filter": [
        {
          "terms": {
            "destination.domain": [
              "api.openai.com",
              "api.anthropic.com",
              "generativelanguage.googleapis.com"
            ]
          }
        },
        {
          "must": {
            "regexp": {
              "request.body": "AKIA[0-9A-Z]{16}|gh[pousr]_[A-Za-z0-9]{36}|sk-[A-Za-z0-9]{20}"
            }
          }
        }
      ]
    }
  }
}
```

This gives you AegisGate's detection capability in the tools you already own, without deploying anything new.

WAF Rules

If you're running Cloudflare, AWS WAF, or any WAF that supports custom regex rules, you can import our credential exfiltration patterns directly. The AWS key pattern (`AKIA[0-9A-Z]{16}`), GitHub token pattern (`gh[pousr]_[A-Za-z0-9]{36}`), and OpenAI key pattern (`sk-[A-Za-z0-9]{20}`) are particularly relevant for detecting credential exfiltration through AI API traffic.

Cloudflare WAF custom rule example (credential exfiltration to AI APIs):

```
(http.host in {"api.openai.com" "api.anthropic.com"
"generativelanguage.googleapis.com"} and
http.request.body matches "AKIA[0-9A-Z]{16}|gh[pousr]_[A-Za-z0-9]{36}|sk-[A-Za-z0-9]{20}")
```

Step 5: ARTEX-Specific Guidance — What to Look For

If you're hunting for ARTEX-like activity in your environment, here's what we know from Zscaler ThreatLabz and Microsoft threat intelligence:

Behavioral Indicators

ARTEX automates the full attack chain. Look for:

1. **Rapid reconnaissance:** Many API calls in a short window, probing multiple endpoints. Storm-0940 hit 2,900 auth attempts/minute against 327 accounts across 8 organizations.
2. **Credential attacks through APIs:** Password spraying patterns, credential stuffing via application credentials (not interactive logins). 151,000+ failed sign-in attempts observed.
3. **AI-accelerated activity:** Attack progression that's faster than human-operated. Storm-0940 went from initial targeting to privileged cloud access in under 3 hours. If your timeline analysis shows impossibly fast lateral movement, consider AI acceleration.
4. **Multi-agent patterns:** Activity that suggests coordination between multiple agents — one doing reconnaissance, another doing exploitation, a third doing data access. Zscaler confirmed GLM reviewed DeepSeek's subagent work.
5. **Microsoft Graph anomalies:** If you're on M365, watch for unusual Graph API activity — reconnaissance queries, application credential changes, privilege modifications, new OAuth app registrations.
6. **Cross-AI-session handoffs:** Custom scripts sharing data between different AI tools (e.g., Claude Code → DeepSeek → GLM). Look for shared state files, synchronization scripts, or unusual inter-process communication patterns.

Technical Indicators

- ARTEX is written in Go + Next.js. Look for unexpected Go binaries on workstations and servers, especially in temp directories or user profiles.
- It uses an OpenAI-compatible gateway. Watch for API calls to OpenAI-compatible endpoints from unexpected sources or at unusual times.
- It supports MCP (Streamable HTTP and SSE). Watch for MCP traffic from unexpected sources — especially MCP connections to external servers.
- Configurations specifying DeepSeek, GLM, or Grok models in tool configs or environment variables.
- Scripts that coordinate handoffs between AI sessions — shared state files, synchronization scripts, process orchestration.
- The original repository ([Autumn-27/ARTEX](#)) has been deleted, but 800+ forks exist. Check your developers' machines for ARTEX clones.

Establish Your Baseline First

Before hunting for anomalies, establish what normal AI API traffic looks like in your environment. Which hosts legitimately call `api.openai.com`, `api.anthropic.com`, `generativelanguage.googleapis.com`? What's your baseline volume and timing? Which users have legitimate reasons to use AI tools? Anything outside that baseline since October 17 is your hunting ground.

What ARTEX Means for You

ARTEX is open source. It's been forked 800+ times. The original repo was deleted, but the code is everywhere. Multiple threat groups are using it, not just Storm-0940. Microsoft observed attackers from "several geographic regions" using the framework.

The implication: AI-accelerated attacks are no longer the exclusive domain of nation-states. The tooling is freely available. Expect to see it used by cybercrime groups, ransomware operators, and opportunistic attackers.

This is why machine-speed attacks require machine-speed detection — it's not a marketing tagline, it's a survival requirement. Human-paced incident response cannot keep up with AI agents that go from reconnaissance to privileged access in under 3 hours.

Step 6: Sector-Specific Guidance

The threats above affect everyone, but the compliance triggers, log sources, and priorities differ by sector. Here's what matters for yours.

OT/ICS & Industrial

ARTEX automates the full attack chain — reconnaissance, credential attacks, lateral movement. The same multi-agent approach that hit 2,250 M365 tenants could target SCADA, DCS, and ICS environments. More OT environments are incorporating AI every day: predictive maintenance models, LLM-based HMI assistants, AI-powered anomaly detection. Every one of those is a new attack surface that traditional OT security tools don't cover.

What to do:

- If AI touches your OT environment — even indirectly (developers using AI tools that can access OT network diagrams, PLC code, or SCADA configs) — deploy Rampart on those developer machines. It runs locally with zero network dependency.
- If you're running AI agents that interact with OT systems, route them through Platform. The 256 regex patterns include Modbus, DNP3, and OPC-UA protocol manipulation detection.
- Scan any AI training data, model configs, or agent skills that will touch OT-adjacent systems with Scanner before deployment.
- Install Lens on HMI workstations. If anyone uses a browser-based AI chat on an HMI machine (and they do), Lens prevents OT data from leaking.

Air-gap is non-negotiable for OT. Most AI security tools (Lakera, Robust Intelligence, Protect AI, HiddenLayer) are SaaS — they require your AI traffic to route through their cloud. That's a non-starter in OT environments. AegisGate Platform, Rampart, Scanner, MCP, and Lens all run fully air-gapped. All detection is local. No cloud, no phone-home, no telemetry by default.

Defense & Government

Storm-0940 is a Chinese APT. Defense contractors and federal agencies are prime targets for exactly this kind of attack — AI-accelerated, multi-agent, credential-focused. NIST 800-171 3.13.1 requires network isolation for CUI environments. SaaS AI security tools that route traffic through their cloud violate this requirement. CMMC Level 2 assessment is now required for many defense contracts.

What to do:

- Deploy Platform air-gapped within your security boundary. The Community tier (free) provides 256 regex patterns and protocol-level scanning with zero external dependencies. Platform's Pro+ tier includes CMMC L2, NIST 800-171, and FedRAMP framework mappings for compliance documentation.
- MCP's distroless Docker image (zero CVEs, zero external dependencies) aligns with STIG-style hardening expectations. Deploy it to secure any MCP-based agent infrastructure.

- Rampart runs locally on developer workstations with zero network dependency. If developers use AI coding tools on CUI-adjacent systems, Rampart prevents source code and credential exfiltration.
- Rotate any credentials that were accessible to ConnectWise integrations. Storm-0940's primary attack vector was stolen ConnectWise credentials — if your agency or contractor used ConnectWise, treat all associated credentials as compromised.

Healthcare & HIPAA

If a clinician pastes a patient case into ChatGPT for a "second opinion," or a developer sends an EHR config to Copilot, that's a HIPAA reportable event. Lens catches 69 PII patterns including patient identifiers, SSNs, and medical-related data — before it hits send. Scanner checks RAG documents for PHI before ingestion into clinical AI tools.

What to do:

- Deploy Lens on all clinical and admin workstations (2 minutes each, Chrome or Firefox). This is the fastest, highest-impact action you can take.
- If you're building AI tools for clinical use, scan all training data and RAG documents with Scanner before ingestion. The 85 PII patterns include medical-specific identifiers.
- **Retroactive exposure check:** Search your proxy logs for outbound traffic to api.openai.com, api.anthropic.com, generativelanguage.googleapis.com from workstations that handle PHI. Any matches since October 17 are potential exposure events that may need to be documented for HIPAA breach assessment.
- Rotate any credentials that were accessible to AI tools or routed through LiteLLM.

Financial Services

AegisGate detects SWIFT/BIC banking codes, credit card numbers (Luhn-validated), and bank account numbers in AI traffic. If a developer pastes a transaction log into ChatGPT to "debug a query," that's a PCI violation. If an analyst sends a wire transfer record to Claude for "summarization," that's GLBA-protected data going to a third party. SOX requires audit trails for system changes — Platform's tamper-evident SHA-256 hash chain logs document what was blocked and when (Pro+ feature).

What to do:

- Deploy Lens on trader, analyst, and customer service workstations. The SWIFT/BIC and credit card detection patterns catch financial data before it reaches AI providers.
- Deploy Rampart on developer machines. If developers query production databases using AI-assisted tools, Rampart catches connection strings and credentials in transit.
- **Retroactive exposure check:** Search proxy logs for card numbers (`\b4[0-9]{12}(?:[0-9]{3})?\b`) and SWIFT/BIC codes in outbound AI API traffic. Any matches are potential PCI/GLBA exposure events.
- Storm-0940 targeted banks globally. If you use ConnectWise, rotate all associated credentials immediately and audit Microsoft Graph logs for indicators of compromise.

Cloud & Multi-Cloud

AI API calls from your cloud infrastructure are probably invisible to you. An EC2 instance running LiteLLM could be exfiltrating credentials right now. A Lambda function calling OpenAI could be sending source code. A GCP service account might be making unauthorized AI API calls.

What to do:

- **AWS:** Search CloudTrail for `bedrock:*`, `sagemaker:*`, and any `InvokeModel` API calls. Check VPC Flow Logs for outbound traffic to AI API endpoints. Look for LiteLLM containers or processes on EC2 instances.
- **Azure:** Check Entra ID Sign-in logs for anomalous auth patterns (Storm-0940 indicators). Search Microsoft Graph audit logs for reconnaissance, credential changes, and new OAuth app registrations. Check Azure Monitor for outbound AI API calls from compute resources.
- **GCP:** Search Cloud Audit Logs for `aiplatform.*` and Vertex AI calls. Check VPC Flow Logs for outbound traffic to AI API endpoints from Compute Engine instances.
- Deploy Platform as a cloud gateway (Docker or Kubernetes) to intercept and scan AI traffic at the infrastructure level. The setup wizard auto-detects your cloud environment and recommends a deploy profile.

State, Local & Education

State and local governments run PHP for citizen-facing services, use Microsoft 365 for operations, and are increasingly adopting AI tools. Universities run PHP extensively (LMS, portals, research dashboards) and are heavy AI users — students, faculty, and researchers all use ChatGPT/Claude. Both sectors are often resource-constrained: no dedicated SOC, sometimes no SIEM, limited security budget. They're also Storm-0940's targets — Microsoft observed attacks across government organizations. Student records (FERPA), research data (ITAR/EAR if defense-funded), and citizen PII are all at risk.

What to do:

- **Patch PHP immediately.** Universities and local governments are especially vulnerable given how many PHP applications they run and how slow patching cycles can be. PHP 8.5.12, 8.4.27, 8.3.36, or 8.2.35.
 - Install Lens on all faculty, staff, and admin workstations. It's free, takes 2 minutes per machine, and prevents PII/credential leakage to AI providers.
 - Rotate ConnectWise credentials if applicable. Check Microsoft 365 audit logs for Storm-0940 indicators (rapid auth, new OAuth apps).
 - Deploy Platform Community tier (free) for AI traffic visibility. If you don't have a SIEM, Platform's built-in logging gives you audit trail capability you wouldn't otherwise have.
 - If you run AI research infrastructure, deploy Platform to monitor AI API traffic for data exfiltration. Research data leaving through AI API calls is a risk you almost certainly aren't monitoring.
-

Quick Reference: What to Do Right Now

If you have 5 minutes:

1. Install [AegisGate Lens](#) browser extension — [Chrome](#) or [Firefox](#)
2. Check if your Node.js and Bun installations are patched for eMudhra (they may not be)
3. Verify your PHP version — if it's not 8.5.12, 8.4.27, 8.3.36, or 8.2.35, start planning downtime NOW
4. Check your GitLab version — if it's not patched, restrict network access immediately

If you have 30 minutes:

1. All of the above
2. Download [AegisGate Rampart](#) for your dev team
3. Pull AegisGate detection patterns from our repos and import into your SIEM (see Step 4)
4. Audit your Microsoft Graph logs for Storm-0940 indicators (rapid auth, credential changes, new OAuth apps)

If you have a few hours:

1. All of the above
2. Deploy [AegisGate Scanner](#) in your CI/CD — one-line installer: `curl -fsSL https://raw.githubusercontent.com/aegisgatesecurity/aegisgate-scanner/main/install.sh | bash`
3. Evaluate [AegisGate MCP](#) for your MCP server infrastructure
4. Stand up [AegisGate Platform](#) Community tier as a monitoring gateway for your AI traffic

If you're in active incident response:

1. Check PHP access logs and error logs back to **October 17** for PHP2Shell exploitation indicators (unusual multipart POSTs, process spawning, outbound connections)
2. Revoke and rotate **all** ConnectWise-related credentials. Check for persistence (OAuth apps, federation trusts, mailbox forwarding rules)
3. Scan any LiteLLM installations for compromise indicators (unexpected processes, outbound connections, modified `.pth` files)
4. Hunt for ARTEX behavioral indicators in your API logs (see Step 5)
5. Use Lens to scan any captured AI chat data for PII/credential exposure

6. If you have a SIEM, import AegisGate regex patterns (see Step 4) and run retroactive searches against proxy logs for credential exfiltration to AI API endpoints
 7. Establish your AI traffic baseline (which hosts call which AI APIs, normal volume/timing) and flag anything outside it since October 17
-

Appendix: Log Sources for Hunting

Incident	Log Source	What to Search For
PHP2Shell	nginx/apache access logs, PHP-FPM logs, web server error logs	Unusual multipart/form-data POST requests, unexpected process spawning from web workers, outbound connections to unfamiliar IPs
Storm-0940	Entra ID sign-in logs, Microsoft Graph audit logs, M365 Unified Audit Log (purview.microsoft.com)	Rapid auth bursts (2,900+/min), application credential changes, new OAuth app registrations, privilege modifications, federation trust changes
GitLab RCE	/var/log/gitlab/gitlab-rails/production.log, /var/log/gitlab/nginx/gitlab_access.log	Unauthenticated requests, error spikes, unexpected process execution
LiteLLM	Process network connections, Python site-packages directories, container egress logs	litellm_init.pth file, unexpected outbound connections from LiteLLM process, credential strings in outbound traffic
AI credential exfil	Egress proxy logs, firewall logs, AWS CloudTrail, Azure Monitor, GCP Cloud Audit Logs	API keys (AKIA..., ghp..., sk-...) in traffic to api.openai.com, api.anthropic.com, generativelanguage.googleapis.com
ARTEX	API gateway logs, MCP server logs, endpoint detection logs, process inventory	Rapid multi-endpoint reconnaissance, Go binaries in unexpected locations, MCP connections to external servers, multi-agent coordination patterns

What AegisGate Cannot Do (Honest Limits)

I'm not going to pretend one tool solves Black October. Here's what AegisGate does NOT do:

- **Patch your PHP servers.** You have to do that. PHP 8.5.12, 8.4.27, 8.3.36, or 8.2.35. Now.
- **Manage your TLS certificate trust stores.** Update your CA bundles. Check Node.js and Bun. Test against Mozilla's incident endpoints.
- **Patch your GitLab.** Update to 19.5.1, 19.4.3, 19.3.6, 19.2.9, 19.1.9, 19.0.10, or 18.11.13.
- **Fix Microsoft's SharePoint patches.** That's on Microsoft. Restrict access and wait for the out-of-band fix.
- **Replace your SIEM.** AegisGate is a detection and prevention layer for AI agent traffic. It doesn't replace Splunk, Sentinel, or ELK. In the free tiers, you can extract our patterns and feed them into your SIEM manually (see Step 4). Platform's built-in SIEM forwarding is a Pro+ feature.
- **Stop ARTEX running on someone else's machine.** ARTEX is an offensive tool running on the attacker's side. AegisGate detects the *effects* of ARTEX in your traffic — the reconnaissance patterns, the credential attacks, the lateral movement. It can't reach out and disable the attacker's tooling.
- **Detect novel AI attack patterns without ML (free tier).** The 256 regex patterns in Platform's Community tier catch known patterns — credential exfiltration, PII leakage, prompt injection. The ML anomaly detection that catches *novel* attacks is a Pro+ feature. Lens and Rampart include ML detection for free in their respective layers.

What AegisGate DOES do: it gives you visibility into a layer of traffic that you almost certainly have zero visibility into right now — the AI agent layer. And Black October just proved that layer is being actively exploited.

About This Guide

This guide was written by a blue teamer who built AegisGate because they saw the AI agent blind spot coming. Everything in Levels 1–4 is free and open source (Apache 2.0) with no paid tiers, no feature gating, and no "contact sales for the real version." Level 5 (Platform) has a free Community tier that includes 256 regex patterns and protocol-level scanning — the ML anomaly detection and SIEM integration are Pro+ features, and I've been upfront about that.

There is no upsell here. There is no "schedule a demo." There are repos you can clone and tools you can deploy in minutes — including in air-gapped, regulated, and classified-adjacent environments where SaaS tools can't go.

The repos:

- **Lens** (browser extension, 209 regex + ML): github.com/aegisgatesecurity/aegisgate-lens
- **Rampart** (local proxy, 211 regex + ML): github.com/aegisgatesecurity/aegisgate-rampart
- **Scanner** (static analysis, 322 regex + ML): github.com/aegisgatesecurity/aegisgate-scanner
- **MCP** (secure MCP framework, 79 regex + ML): github.com/aegisgatesecurity/aegisgate-mcp
- **Platform** (enterprise gateway, 256 regex Community / +ML Pro+):
github.com/aegisgatesecurity/aegisgate-platform

If this helps you, star the repos so other blue teamers find them. If it doesn't help you, the detection patterns are still free — take them and use them in your SIEM. No strings.

Stay safe out there.

This guide will be updated as the situation develops. Last updated: October 22, 2026 10:14 UTC.