AI in NOC Life

By

Mahutthawat Raksakiettisak
Computer Center,
Srinakharinwirot University

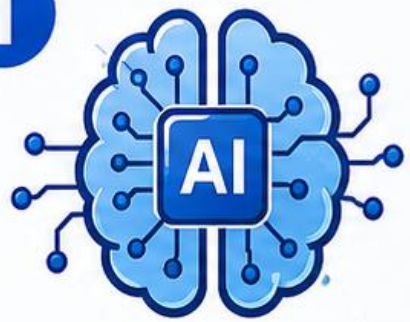
VTS-Next (Vulnerabilities Scan and Threat Intelligence / block

VMSTAT (Virtual Machine Statistics and Software inventory)

กระบวนการทำงานกับ AI มี 4 ขั้นตอน

องค์ประกอบสำคัญที่ทำให้การใช้งาน AI เกิดผลจริงในองค์กร

1



Model (สมอง AI)

ใช้ในการวิเคราะห์
ประมวลผล และตอบคำถาม

LLM • Reasoning • Prediction

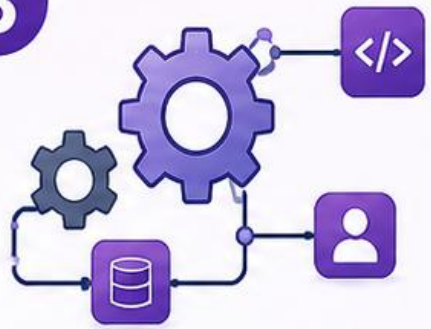
2

Context (ข้อมูลประกอบ)

ข้อมูลที่ช่วยให้ AI
เข้าใจงานและตอบได้ตรงประเด็น

เอกสาร • ฐานข้อมูล • ความรู้ทางธุรกิจ

3



Harness (เครื่องมือทำงาน)

เครื่องมือที่เชื่อม AI
เข้ากับระบบและขั้นตอนการทำงาน

API • MCP • Agent • Workflow

4

Control (ตัวควบคุม)

กำกับการใช้งานให้ปลอดภัย
ถูกสิทธิ์ และตรวจสอบได้

สิทธิ์การเข้าถึง • ความปลอดภัย • Governance

AI ที่ใช้งานได้จริง = Model + Context + Harness + Control

Pipeline ในการ Implement Project

ตั้งแต่กำหนดความต้องการ ออกแบบ พัฒนา ตรวจสอบ จนถึง deploy และ monitor

[Plan] → [Code] → [Verify] → [Ship]

1 /requirements – กำหนดความต้องการ

สร้าง PRD (Product Requirements Document)

- **What:** ฟีเจอร์ที่ทำอะไร
- **Why:** แก้ปัญหาอะไร / ROI
- **Who:** ใครใช้ (persona, roles)
- **Scope:** in-scope / out-of-scope
- **Success Criteria:** EARS format (Event-Action-Response-State)

Output: PRD ที่ทุกคนเห็นภาพ "done" ตรงกัน

Skip ถ้า: เป็น bug fix เล็ก หรือ refactor

2 /design – ตัดสินใจ architecture

- ออกแบบ technical solution
- **DB schema:** tables, indexes, relations
- **API contract,** endpoints, request/response
- **Auth model:** ใช้ role อะไร, scope อะไร
- **Constraints:** performance, security, EU AI Act checkpoint
- **ADR** (Architecture Decision Record): ทำไมเลือก option นี้

Output: Design doc + ADR

สร้างกับ VTS: จะตั้ง Prisma schema rules (อย่างเช่น @lmap)

3 /build-brief – Context pack ก่อนเขียนโค้ด

รวบรวม context ให้ agent ต้องรู้ก่อนลงมือ

- ไฟล์ใหม่/ต้องอ่าน (existing patterns)
- API conventions ของ project (src/app/api/* pattern)
- Coding rules (Zod validation, as any workaround)
- Test patterns

Output: Brief ที่ส่งให้ Claude/agent อ่านก่อน

Why important: ลด context-switching และลด bug จากการไม่รู้ convention

เขียน code (manual + AI)

ใช้ Edit/Write ตาม brief • follow patterns • ใช้ Zod schemas • มี audit log • ยึด convention ของ repo

4 /review-ai – Audit AI-generated code

ตรวจ code ที่ AI เขียน ก่อน commit

- **Security:** SSRF, SQLi, XSS, secrets in code
- **Quality:** error handling, edge cases, null checks
- **Completeness:** ครบตาม path ที่ requirements ระบุไว้ไหม
- **Convention drift:** ใช้ pattern ของ repo หรือไม่

Output: Severity-tagged findings + fix recommendations

Focus: AI-specific issues เช่น hallucinated APIs, missing error handling, scope creep

5 /sast-scan – Static Analysis (devsecops)

ใช้ Opengrep (Semgrep-compatible) ใน Docker

- หา injection flaws (SQLi, XSS, command injection)
- Hardcoded secrets, weak crypto
- OWASP Top 10 patterns
- TypeScript/Next.js-specific rules

Output: SARIF/JSON findings

Auto-route ไป vuln-triage → CVSS + EPSS scoring

6 /sca-scan – Software Composition Analysis (devsecops)

จับ Gype สแกน dependencies (package.json/lock)

- CVE ไม่ direct + transitive deps
- License compliance
- Upgrade paths (เช่น Next 16.x → 16.y ถัดไป)

Output: CVE list + remediation (*npm update <pkg>*)

Tip: Run ทั้งคู่ /sast-scan และ /sca-scan

VTS มี Django legacy strings อยู่

7 /deploy-guide – Deploy พร้อม rollback plan

- Step-by-step deploy
- Pre-deploy checklist (tests pass? migrations safe?)
- Staging ใหม่ production (dhm/VTS: 10.11.001.101 → 10.11.001.104)
- Rollback procedure (snapshot/revert plan)
- Deploy commands ตาม project (rsync + systemctl restart vts-nextjs)

Output: Deploy runbook

ระบุ: No go: revert on prod (rollback_prod_q3_no_reset.md)

origin/main บน prod เป็น stable

8 /monitor-setup – Observability

ตั้ง observability หลัง deploy

- Health check endpoint (/api/health – version จาก package.json; egresslab_67916282)
- Error tracking + alerting thresholds
- Log aggregation
- ค่าตั้งต้น: response time, error rate, scan throughput

Output: Monitoring config + alert rules

Why critical: Smoke test หน้างาน real path health endpoint

คนเสิร์ฟวิเคราะห์ smoke test

/requirements /design [Plan] → /build-brief → เขียน code (manual + AI) [Code] → /review-ai → /sast-scan → /sca-scan → /deploy-guide [Ship] → /monitor-setup

Goal: ลด bug, ลด risk, และทำให้ deploy ได้อย่างมั่นใจ

Use Cases ที่นำมาใช้จริงในระบบงานเครือข่าย และระบบสารสนเทศ

ตัวอย่างระบบที่ดำเนินการแล้ว เพื่อเพิ่มความปลอดภัย การมอนิเตอร์ และการบริหารทรัพยากร

1

ระบบ VA Scan

สแกนช่องโหว่และตรวจสอบความเสี่ยงของระบบ

Vulnerability Assessment

2

Threat Intelligence / Auto Block

ดึง feed จาก MISP, CrowdSec, FireHOL, Spamhaus เพื่อ block IP ที่ Cloudflare และส่ง MISP domain blocklist ไปยัง Google Workspace

MISP • CrowdSec • FireHOL • Spamhaus • Cloudflare • Google Workspace

3

Credential Monitor

ติดตามการรั่วไหลของรหัสผ่านและข้อมูลบัญชีผู้ใช้

HIBP API • ProxyNova / COMB • Hudson Rock API

4

Server Resource Monitoring

มอนิเตอร์การใช้ทรัพยากรของเซิร์ฟเวอร์แบบต่อเนื่อง

Victoria DB • Grafana

5

VM Right Sizing & Auto Scale

ตรวจ oversize, undersize, right size ของ VM และเพิ่ม resource อัตโนมัติเมื่อ undersize

Oversize • Undersize • Right Size • Auto Add Resource

6

Google Quota Report per Faculty

สรุปรายงานการใช้งาน quota แยกตามคณะ

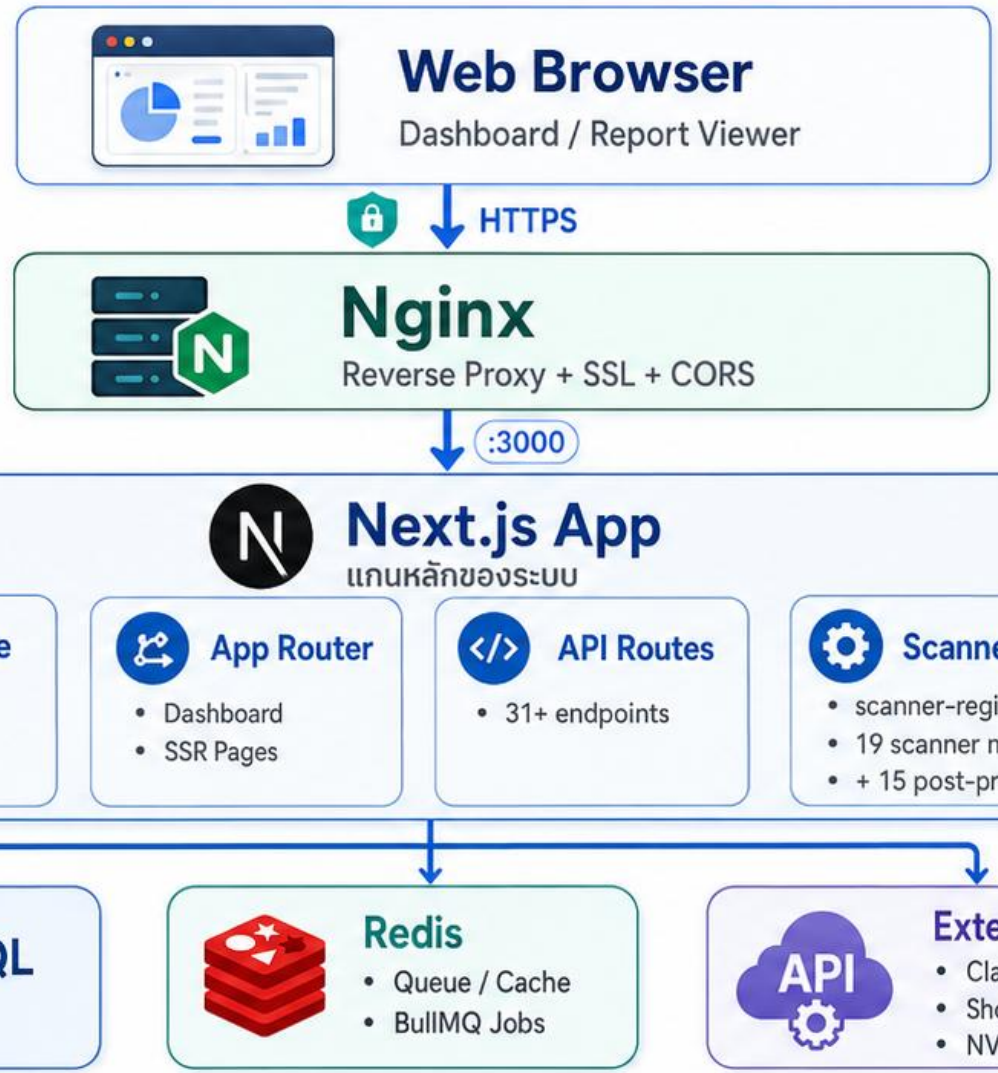
Storage & Quota Reporting

Security | **Monitoring** | **Optimization**

เป้าหมาย: ทำให้ระบบปลอดภัย มองเห็นได้ และบริหารจัดการได้ดีขึ้น

System Architecture Overview

ภาพรวมโครงสร้างระบบและการไหลของข้อมูล



การทำงานโดยสรุป

1. ผู้ใช้เข้าผ่าน Web Browser
2. Nginx ทำหน้าที่ Reverse Proxy และจัดการ SSL/CORS
3. Next.js เป็นแกนหลักของระบบ
4. Scanner Engine ประมวลผลการสแกนและ post-processing
5. ระบบเชื่อมต่อฐานข้อมูล คิวงาน และบริการภายนอก

Data Flow

User → HTTPS → Nginx → Next.js → Database / Queue / External APIs

Parallel Scanning Architecture

โครงสร้างการสแกนแบบขนาน (comprehensive_scan)

15 scanners run in parallel to accelerate reconnaissance, vulnerability detection, and targeted security testing.
เครื่องมือสแกน 15 รายการทำงานพร้อมกัน เพื่อเพิ่มความเร็วในการค้นหาข้อมูล ตรวจหาช่องโหว่ และทดสอบเชิงลึก

15 Scanners Running in Parallel
สแกนเนอร์ 15 ตัวทำงานพร้อมกัน

Recon
- WhatWeb → Tech detect
- Httpx → HTTP probing
- Nmap → Ports + CVEs
- Shodan → OSINT
- Subfinder → Subdomains
- Katana → URL crawling

DAST
- ZAP → Full DAST scan
- Wapiti → Web vuln scanner
- Nikto → Server misconfig
- FFUF → Directory fuzzing

CMS / App
- WPScan → WordPress scan

API
- GraphQL-Cop → GraphQL audit

Targeted
- Nuclei → Template-based scan
- SQLMap → SQL injection
- Dalfox → XSS (DOM + reflected)
- XSStrike → XSS patterns
- TestSSL → TLS/SSL audit

Scanner Workflow
ลำดับการทำงานของสแกนเนอร์

Key Benefits
จุดเด่นของการทำงาน

- **Parallel execution reduces total scan time**
  ทำงานพร้อมกัน ช่วยลดเวลาสแกนรวม
- **Multiple scan types improve coverage**
  มีหลายประเภทการสแกน เพิ่มความครอบคลุม
- **Results are normalized into Vulnerability records**
  ผลลัพธ์ถูกแปลงเป็น Vulnerability records
- **Progress tracking supports orchestration and finalization**
  ติดตามความคืบหน้าและสรุปผลได้อัตโนมัติ

VTS
Security Assessment System

TH

AV Admin VTS
Super Admin

Main Menu

- Dashboard
- Websites
- Assessments
- Vulnerability Scan
- Scan Schedules
- Remediation
- Reports
- Notifications

Administration

- Scan Batches
- Credential Monitor
- Threat Intel / Block
- Organizations
- User Management
- Shared Evidence
- API Configuration
- API Keys
- Webhooks & Notifications

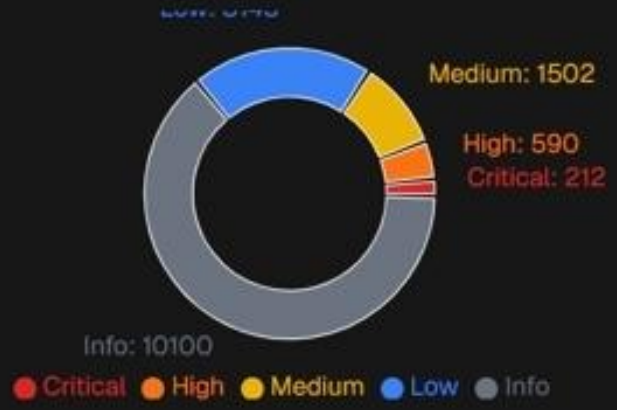
NCSA Website Standard v1.0

Dashboard

System Overview

Total Websites	Vulnerability Scans	Open Vulnerabilities	Assessments
54	58	15549	3
54 active	54 completed	212 critical	0 pending review

Vulnerability Severity



Recent Scans

[redacted] COMPREHENSIVE_SCAN · 5/15/2026	13C 3H	completed
[redacted] COMPREHENSIVE_SCAN · 5/11/2026	2H	completed
[redacted] le) COMPREHENSIVE_SCAN · 5/7/2026	2C 2H	completed
[redacted] COMPREHENSIVE_SCAN · 5/5/2026	8C 11H	completed
[redacted] COMPREHENSIVE_SCAN · 5/3/2026	6C 13H	completed

Scan #101 — COMPREHENSIVE_SCAN

swuai - สำนักคอมพิวเตอร์

Submitted — 53% — Completed — Report Generated — Remediation

กำลังสแกน... 0:10

Phase: scanning 53%

พบ:

Scanner Progress:

nuclei	✓	shodan	✓	sqlmap	✓		
wapiti	✓	wpscan	✓	whatweb	✓		
httpx	65%	ffuf	21%	nmap	20%		
dalfox	20%	katana	20%	testssl	20%		
subfind…	20%	nikto	10%				

ยกเลิก

Workflow 1b — finalizeScanResults() 17 enrichment steps

src/lib/scanners/shared.ts:848-1050+ — runs once per scanner completion in a multi-scanner job

1–8 Intelligence & Vulnerability Enrichment

1	Count findings by severity	สรุปจำนวนช่องโหว่ตามระดับความรุนแรง
2	Tag with auth context	ระบุบริบทการยืนยันตัวตนของเป้าหมาย
3	Tech-stack-aware remediation	แนะนำวิธีแก้ไขที่เหมาะกับเทคโนโลยีที่ตรวจพบ
4	Cross-scanner correlation	เชื่อมโยงผลลัพธ์จากหลายสแกนเนอร์ ลดความซ้ำซ้อน
5	Enqueue Intel/Owl enrichment	ส่งข้อมูล IP / domain / URL / hash ให้ระบบ Intel/Owl
6	Trend analysis	เปรียบเทียบกับผลการสแกนครั้งก่อนหน้า
7	CMS-aware workflows	เวิร์กโฟลว์เฉพาะสำหรับ WordPress / Joomla
8	CVE verification with Nuclei	ยืนยันช่องโหว่ด้วย Nuclei (Template-based scan)

9–17 Security & Verification

9	WAF detection	ตรวจจับ WAF จาก headers / cookies / body fingerprint
10	CDN coverage + IP correlation	ตรวจสอบการใช้ CDN และความสัมพันธ์ของ IP
11	DNS analysis	วิเคราะห์ DNS records, MX, NS, SPF, DMARC
12	Security posture score	ให้คะแนนความมั่นคงปลอดภัย (0–100 จาก 5 มิติ)
13	Dynamic baseline - FP reduction	สร้าง baseline เพื่อลด False Positive
14	Active verification (Tier 2)	ยืนยันเพิ่มเติมด้วย Payload tests (Dalfox, SQLMap)
16	PoC generation - adv detection	สร้าง Proof of Concept สำหรับช่องโหว่ขั้นสูง (IDOR, proto pollution)
17	Status = completed - webhook	อัปเดตสถานะเป็น Completed และส่งผลผ่าน Webhook

INPUT

- ผลลัพธ์จากหลาย Scanner (Nuclei, Nikto, ZAP, ฯลฯ)
- Target: IP / Domain / URL
- Auth context (ถ้ามี)

OUTPUT

- Enriched findings (17 มิติ)
- Security posture score
- PoC & remediation
- Trend & baseline data
- Webhook / API notification

BENEFITS

- ครอบคลุมทุกมิติความเสี่ยง
- ลด False Positive
- จัดลำดับความสำคัญได้แม่นยำ
- พร้อมสำหรับรายงานและการตอบสนอง

1. MULTI-SCANNER JOB

Scan Target

2. finalizeScanResults() — 17 ENRICHMENT STEPS

Aggregate รวมผลลัพธ์ → Enrich เสริมข้อมูล (1–17) → Analyze วิเคราะห์ & ให้คะแนน → Validate ยืนยันเพิ่มเติม (Tier 2) → Notify อัปเดตสถานะ & Webhook

3. REPORT & ACTION

Deliver Insights

ⓘ ทำงานอัตโนมัติทุกครั้งหลังสแกนเสร็จ • ให้ข้อมูลเชิงลึกที่ครบถ้วน • พร้อมสำหรับการตัดสินใจและการตอบสนอง

Main Menu
- Dashboard
- Websites
- Assessments
- Vulnerability Scan
- Scan Schedules
- Remediation
- Reports
- Notifications

Administration
- Scan Batches
- Credential Monitor
- Threat Intel / Block
- Organizations
- User Management
- Shared Evidence
- API Configuration
- API Keys
- Webhooks & Notifications
- Settings
- What's New

Credential Monitor

Dark web credential leak monitoring — ตรวจสอบข้อมูล credentials ที่หลุดจาก data breaches และ infostealer malware

Total Leaks	Critical	High	Unresolved	Resolved
267	243	0	267	0
credentials found	plaintext passwords / stealer logs	hashed passwords / PII exposed	pending action	password reset / resolved

Scan Domain

Scan HIBP and Hudson Rock for leaked credentials of your domain

swu.ac.th | Scan Now | Import HIBP JSON

Recent Scans — click to view results

- correlation-engine — swu.ac.th — 6 found (6 new)
- external-feed-cf-sync — cloudflare — 2000 found (0 new)
- misp-cf-sync — cloudflare — 2915 found (226 new)
- external-feed-cf-sync — cloudflare — 2000 found (0 new)
- misp-cf-sync — cloudflare — 1357 found (0 new)

Leaked Credentials

267 credentials found across all sources

Search email, breach, URL... | All Status | All Severity | All Sources | Filter

Email / Username	Source	Breach / Malware	Severity	Confidence	Data Exposed	Password Hint
[redacted]29@swu.ac.th @hm471010429	XposedOrNot	CitOday, AntiPublicCombo, Collection-1	MEDIUM	CONFIRMED ProxyNova + LeakCheck + XposedOrNot	Email addresses	XposedOrNot: Risk 0/100 (Unknown). 3 bre
hn[redacted]29@swu.ac.th @hm471010429	LeakCheck	Unknown	CRITICAL	CONFIRMED ProxyNova + LeakCheck + XposedOrNot	id, Passwords	LeakCheck: 1 records. Fields: id, password. Sources:
ch[redacted]@swu.ac.th @chitta PC: Not Found	hudsonrock-username	Infostealer (username) — Not Found Unknown	CRITICAL	CONFIRMED ProxyNova + XposedOrNot + hudsonrock-username + LeakCheck	Passwords, Browser cookies, Browser data	Hudson Rock username search: 3 infected device(s).
[redacted]@swu.ac.th @chitta	XposedOrNot	ExploitIN, LinkedIn, Deezer, AntiPublicCombo	MEDIUM	CONFIRMED ProxyNova + XposedOrNot + hudsonrock-username + LeakCheck	Email addresses	XposedOrNot: Risk 0/100 (Unknown). 4 bre

Workflow 2 — Credential Monitor VTS

Aggregates leaked credentials from 6 sources and cross-validates them

`src/lib/credential-monitor.ts`

1 — Trigger
Start the scan

Manual: POST /api/credential-monitor/scan
Auto: **GET /api/threat-intel/cron**
(daily, CRON_SECRET protected)

2 — Phase 1 — Breach databases
Collect breach metadata + aggregated leaks

HIBP API → breach metadata
ProxyNova / COMB → aggregated leaks with plaintext passwords

3 — Phase 2 — Curated + Infostealer
Collect stealer events and infostealer logs

MISP Feeds → stealer events
Hudson Rock API → infostealer log per email + per username

4 — Phase 3a — Cross-validation (priority: ProxyNova first)
Cross-check credentials across sources

LeakCheck API (1500ms throttle)
XposedOrNot API (500ms throttle, password strength)

5 — Phase 3b — Persist to CredentialLeak
Store normalized results

Fields: email, source, breachName, malwareFamily, severity, confidence, verifiedBy[], dataTypes, computerName, OS

6 — Recalculate confidence
Score based on agreement across sources

3+ sources agree → confirmed
2 sources → high · 1 credible (HIBP/LeakCheck) → medium
1 aggregated (ProxyNova alone) → low

7 — Audit log per source
Record what was found from each source

CredentialScanLog
rows: source, domain, status, leaksFound, newLeaks, errorMessage

8 — UI
Display results with filters

GET /api/credential-monitor
paginated · filter by source / severity / confidence · masked plaintext passwords

6 Data Sources

- **HIBP API** (breach metadata)
- **ProxyNova / COMB** (aggregated leaks with plaintext passwords)
- **MISP Feeds** (stealer events)
- **Hudson Rock API** (infostealer log per email & username)
- **LeakCheck API** (1500ms throttle)
- **XposedOrNot API** (500ms throttle, password strength)

Output

- **CredentialLeak** (aggregated results) — Normalized and deduplicated credentials
- **Confidence Score** — Calculated from cross-source validation
- **Audit Logs** — Per-source activity and outcomes
- **Real-time Alerts** (optional) — New high-confidence leaks

Benefits

- Detect credentials leaked on darkweb early
- Cross-validate to reduce false positives
- Prioritize by confidence & severity
- Actionable for password reset, user notification, and threat response

Collect 6 Data Sources → Normalize & Deduplicate → Validate Cross-Check → Store CredentialLeak → Analyze & Score → Alert & Respond

Main Menu
- Dashboard
- Websites
- Assessments
- Vulnerability Scan
- Scan Schedules
- Remediation
- Reports
- Notifications

Administration
- Scan Batches
- Credential Monitor
- Threat Intel / Block
- Organizations
- User Management
- Shared Evidence
- API Configuration
- API Keys
- Webhooks & Notifications
- Settings
- What's New

Threat Intelligence / Auto-Block

ดึง IoC จาก MISP แล้วส่งไป block ที่ Cloudflare อัตโนมัติ

MISP	External Feeds	Cloudflare	Blocked IPs	Chrome Blocked URLs	Last Sync
Connected	Connected	swu.ac.th	2915	332	5/19/2026
v2.5.35	2000 blocked · 2 feeds		33 firewall rules	4 OUs	Total blocked

MISP IoC Summary

986,373 total

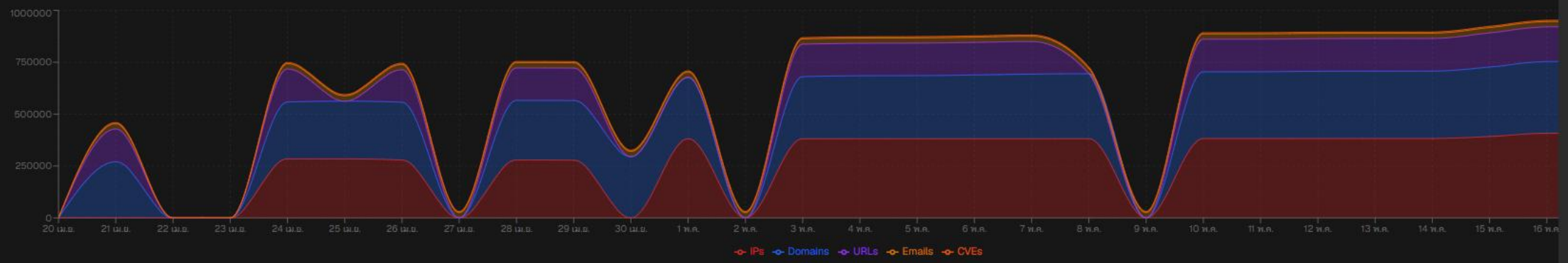
จำนวน Indicators of Compromise แยกตามประเภท (90 วันล่าสุด) — อัพเดท 19/5/2569 12:35:05

IPs	Domains	URLs	Emails	CVEs	Hashes
419,100	352,617	182,963	25,984	5,709	0

Threat Level: 157687 High | 667536 Medium | 131846 Low | 29304 Unknown

IoC Trend (รายวัน)

แนวโน้มจำนวน IoC จาก MISP ย้อนหลัง 30 วัน



Workflow 3 — Threat Intel → Cloudflare + Chrome Block VTS

Daily cron pulls feeds, pushes IPs to Cloudflare WAF + domains to Google Chrome blocklist

1 — Cron trigger
Run daily via systemd timer

`GET /api/threat-intel/cron`
Bearer `CRON_SECRET`

2 — Fetch IP feeds in parallel
Collect from multiple sources

→ **MISP API** (curated · `misp-cloudflare-sync.ts:45`)
→ **External feeds** (FireHOL / Spamhaus / CrowdSec · `EXTERNAL_FEED_URLS` env)

3 — Whitelist filter + dedupe + 30-day expiry
Keep only relevant & recent IPs

`threat-intel-whitelist.ts`
- removes private + allowed IPs
- cap at 2000 IPs per source

4 — Push to Cloudflare WAF
Create/Update ruleset entry-point

→ Cloudflare API
- ruleset entry-point: `http_request_firewall_custom`
- zone-scoped • Bearer token
- source: `cloudflare-waf-blocker.ts:85`

5 — Chunk into rules (90 IPs each)
Build rules for WAF

- 3500-byte expression limit
- descriptions: `VTS-MISP-Block-{n}` , `VTS-ExternalFeed-Block-{n}` (separate prefixes)

6 — MISP domains → Google Chrome blocklist
Push high/medium severity domains

→ Google Workspace API
- `gws-chrome-blocklist.ts` • URL-based (not IP)

7 — Telegram alert on failure
Notify when any source fails

any source fail (MISP timeout, Cloudflare 4xx, GWS auth) → message to `TELEGRAM_CHAT_ID` with reason + IP count

8 — Response JSON
Return processing summary

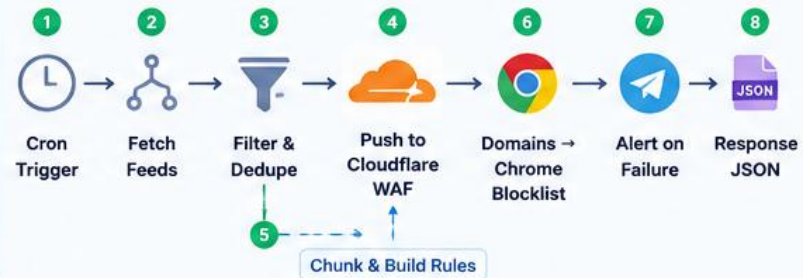
summary:
totalIPs, newlyBlocked, alreadyBlocked, rulesCreated, rulesUpdated, cache status, per-source errors

INPUT SOURCES

IP FEEDS
- MISP API (curated)
- FireHOL / Spamhaus / CrowdSec
- `EXTERNAL_FEED_URLS` (env)

DOMAIN FEED
- MISP API (curated domains)

DATA FLOW



OUTCOMES

- Block malicious IPs at Cloudflare WAF via custom firewall rules
- Block malicious domains in Google Chrome via Google Workspace blocklist
- Real-time alerts to Telegram when failures occur
- Daily scale: ~3,279 IPs blocked daily
  CF ruleset ID `65e7b4569c854df1830d41afa0981b82`
  since 2026-05-10 migration off legacy CF Firewall Rules write API.

PRODUCTION SCALE

~3,279	6	90	Automated	Authenticated
IPs blocked daily	Data sources	IPs per rule chunk	Daily operation	APIs & secure tokens

Stealer Cross-Reference

จับคู่ผู้ใช้ที่ติด Stealer กับระบบที่ credential ถูกขโมย — เห็นว่า "ใครถูกขโมย credential ของระบบไหน"

ตรวจสอบ Cross-Referer

103	55	170
ผู้ใช้ที่ credential หลุด	URLs ที่ credential หลุด (ทั้ง domain)	ผู้ใช้ที่ตรวจสอบ

ระบบมหาวิทยาลัยที่ credential หลุด (ทั้ง domain)

Supreme (ระบบทะเบียน): 2,023 · Admission (รับสมัคร): 1,670 · Admission (รับสมัคร): 1,252 · Web Application: 1,222 · Login/SSO: 1,068 · iPass (บัตรผ่าน): 982 · Web Application: 941 · LMS/Moodle: 826 · Login/SSO: 751 · Supreme (ระบบทะเบียน): 668

รายละเอียด — ผู้ใช้และระบบที่ถูกขโมย Credential

ผู้ใช้	วันที่หลุด ↓	เครื่อง/Malware	Password / Logins / Sources
[redacted]ta@g.swu.ac.th	2026-04-22T08:34:53.000Z	Not Found Not Found	Stealer: L********1 · B********) · 0**********! · C******0 · 2*0 Logins: [redacted] · a***n · s**********e · 1**0 · 5**7 Sources: OperationEndgame2, Stealer Logs
c[redacted]@swu.ac.th	2026-04-03T15:22:48.000Z	Not Found Not Found	Leaked: 17***** (7 chars) Sources: COMB (Combination of Many Breaches), Deezer.com, LinkedIn.com, ExploitIN, LinkedIn, Deezer, AntiPublicCombo, Infostealer (username) — Not Found
[redacted]aya@swu.ac.th	2026-02-27T21:33:50.000Z	GIGABYTE Windows 11 Home	Leaked: 11**** (6 chars) Sources: COMB (Combination of Many Breaches), Bigbasket.com, ExploitIN, Collection-1, Infostealer (username) — GIGABYTE
at[redacted]@swu.ac.th	2026-02-23T11:04:46.000Z	DESKTOP-1P65FCK (Admin) Windows 11	Leaked: 25****** (8 chars) Sources: COMB (Combination of Many Breaches), Unknown, Cit0day, AntiPublicCombo, Collection-1, Infostealer (username) — DESKTOP-1P65FCK (Admin)
k[redacted]@g.swu.ac.th	2026-02-18T06:36:12.000Z	krich MacBook Pro	Leaked: 13******** (10 chars) Sources: COMB (Combination of Many Breaches), Unknown, Cit0day, Collection-1, Infostealer (username) — krich
cl[redacted]a@g.swu.ac.th	2026-02-08T08:39:55.000Z	Not Found Not Found	Leaked: 08******** (10 chars) Sources: COMB (Combination of Many Breaches), Ajarn, Cit0day, Collection-1, Infostealer (username) — Not Found, Ajarn.com
t[redacted]t@g.swu.ac.th	2025-09-20T22:47:00.000Z	DESKTOP-1J1CH55 (teera) Windows 10 22H2 build 19045 (64 Bit)	Leaked: 68******* (9 chars) Sources: COMB (Combination of Many Breaches), Canva.com, GoNitro.com, Animoto.com, LuminPDF.com, LuminPDF, Canva, AntiPublicCombo, Nitro, ExploitIN, Colle Infostealer (username) — DESKTOP-1J1CH55 (teera)
am[redacted]a@swu.ac.th	2025-07-21T00:00:00.000Z	Not Found Not Found	Leaked: 12******* (9 chars) Sources: COMB (Combination of Many Breaches), MySpace.com, AntiPublicCombo, ExploitIN, Infostealer (username) — Not Found
[redacted]	[redacted]	LAPTOP-SS3SAPTR (earth)	Stealer: e*******4 · 1**********1 · 2****1 · t**********4 · E*******4

Workflow 4 — Stealer Log Cross-Reference (Attack Surface) VTS

Match leaked credentials → managed website domains → classify compromised systems

1 Trigger
Start data collection

POST `/api/threat-intel/stealer-crossref` (fresh)
or GET (cached)
- source: threat-intel-runners.ts:163

2 Collect inputs
Gather data from multiple sources

(a) all unique emails from `CredentialLeak` table
(b) all active website domains in VTS
(per org or global)

3 Per email batch (max 200)
Search stealer logs by email

→ Hudson Rock API
- search by email
- fallback search by username
(skip generic usernames like "admin")

4 Per managed domain
Query stealer logs touching domain

→ Hudson Rock attack-surface API
- returns URL endpoints found in stealer logs for that domain

5 Match: email @ domain ↔ managed website domain
Correlate identity to asset

e.g., `user@admission.swu.ac.th` in stealer log
+ `swu.ac.th` is a VTS website → flagged

6 Classify system by URL pattern
Determine system type (attack surface)

`/vpn` → VPN · `/admin` → Admin Panel · `/webmail` → Webmail
`/lms` → LMS · `/sso` → SSO

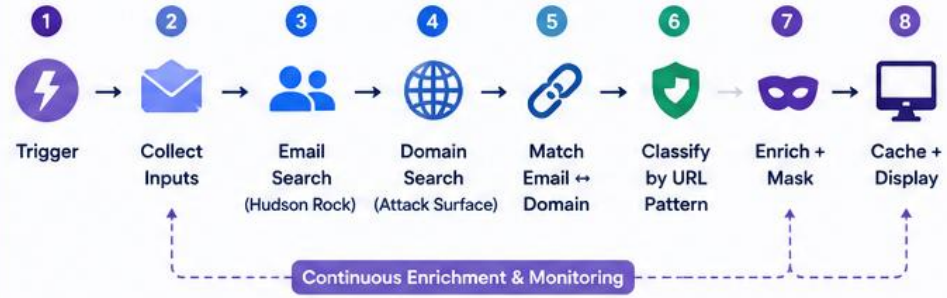
7 Enrich + mask
Add context, mask sensitive data

- top passwords + top logins from stealer log
- masked preview (first/last char)
- malware family • OS • compromise date

8 Cache + display
Cache results & show in UI

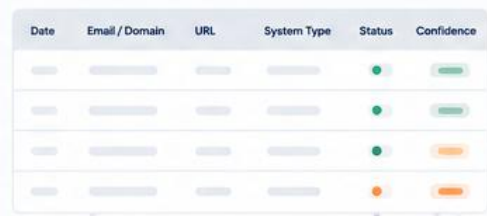
`setCachedResult("stealerCrossRef", …)`
- UI panel sorted by date (most recent first)

DATA FLOW OVERVIEW



CLASSIFY SYSTEM BY URL PATTERN (EXAMPLES)

OUTPUTS (UI PANEL)



- Identified compromised accounts & systems
- Attack surface visibility (URLs, systems)
- Masked credentials & context
- Confidence & risk prioritization
- Exportable / Alert-ready

WHY THIS MATTERS FOR UNIVERSITIES

A leaked `@university.ac.th` credential + observed in a VPN URL in stealer logs = **active path** for attacker into LAN. Cross-ref turns "credential floating around the internet" into "this person's machine was compromised," and the attacker has VPN access."

Reduce Attack Surface
Find exposed systems from real attacker data

Faster Detection
Correlate leaks to assets and users automatically

Better Prioritization
Focus on high-confidence, highest-risk systems

Proactive Response
Enable alerts, incident response, and remediation

SOURCES

- CredentialLeak table
- Hudson Rock API
  - search by email
  - attack-surface API
- VTS Managed Website Domains

Dashboard
Virtual Machines
Hosts
Storage
Networks
Backups
Alerts

Analytics
Analysis
Capacity Planning
Research
Reports
Backup Compliance
Grafana

Security
Software Dashboard
Software Inventory

Administrator
admin

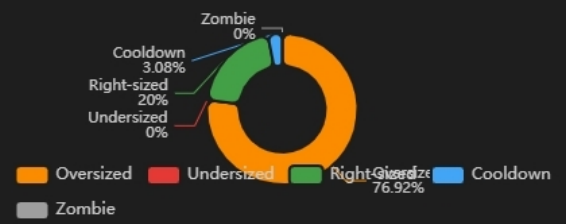
RIGHTSIZING | AUTO-RESIZE

Estimates based on host-level metrics (Sangfor SCP API does not provide per-VM utilization). All VMs on the same host share the same estimated P95 utilization.

65	50	0	13	2	0
Total VMs	Oversized	Undersized	Right-sized	Cooldown	Zombie

Total CPU Waste
92.8%

Total Memory Waste
72.4%



VM Recommendations

OVERSIZED | UNDERSIZED | RIGHT-SIZED | ZOMBIE | COOLDOWN

Search VM...

VM Name	Current vCPU	Recommended vCPU	Current Memory	Recommended Memory	CPU Waste / Pressure	Memory Waste / Pressure	Classification	Auto-Resize	Recommendation	Action
[redacted]piscp	1	1	1.0 GB	512 MB	99.0%	95.0%	oversized	--	-512 MB 1 → 1 vCPU, 1.0 GB → 512 MB	AUTO
NOC [redacted]66	4	1	16.0 GB	4.5 GB	99.0%	93.0%	oversized	--	-3 VCPU -11.5 GB 4 → 1 vCPU, 16.0 GB → 4.5 GB	AUTO
PSM[redacted]	24	6	64.0 GB	16.8 GB	97.0%	94.0%	oversized	--	-18 VCPU -47.3 GB 24 → 6 vCPU, 64.0 GB → 16.8 GB	AUTO
NOC [redacted] firewa[redacted]	4	2	16.0 GB	4.3 GB	93.7%	94.0%	oversized	--	-2 VCPU -11.8 GB 4 → 2 vCPU, 16.0 GB → 4.3 GB	AUTO

Workflow 5 — Right-Sizing Analysis VMS

Classify VMs as undersized / right-sized / oversized / zombie • compute target spec • gated by cooldown

1 — Daily 02:00 trigger
Run the daily right-sizing analysis

APScheduler
backend/app/analytics/rightsizing.py
POST /api/v1/analysis/rightsizing
(manual trigger supported)

2 — Pull 30-day metric rollups
Get host-level metrics (no per-VM live

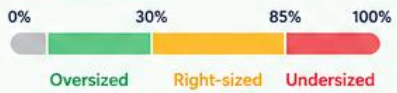
MetricRollupDaily table
LOOKBACK_DAYS = 30
host-level only

3 — Compute P95 CPU + memory per host
P95 = the value 95% of samples fall below
Proxy for "how saturated does this host actually get?"

Outputs:
- p95_cpu
- p95_mem

4 — Classify each VM
Zombie: avg CPU<1% & mem<1% for ≥7d
Oversized: P95 < 30%
Right-sized: 30–85%
Undersized: P95 > 85%

Based on P95



5 — Compute recommendation target
Oversized → P95 + 20% headroom (scale down)
Undersized → P95 + 30% headroom (scale up)
Returns: recommended_vcpu, recommended_mem_mb

Example
Oversized: P95 20% → downsize
Undersized: P95 30% → upscale

6 — Cooldown gate (dual-method)
(a) **SCP operation log:** any "Edit virtual machine" task in last 7d that changed CPU/mem
(b) **Allocation drift:** current vCPU/RAM ≠ last recommendation
→ in cooldown, skip

Prevents thrashing
Waits until stable before re-recommend changes

7 — Persist recommendation row
visible in /research & /vms/<uuid>
status: pending / applied / dismissed

Audit trail
Track decision and outcome

8 — User clicks "Apply"
POST /rightsizing/{id}/apply → ResizeService.apply_recommendation()
→ checks hotplug/balloon capability → SCP API resize
(POST verb, action path)

Result
Success OR "requires_confirmation" (shutdown needed for downsize)

VM CLASSIFICATION

ZOMBIE	OVERSIZED	RIGHT-SIZED	UNDERSIZED
avg CPU < 1% & mem < 1% for ≥ 7 days	P95 < 30%	30% ≤ P95 ≤ 85%	P95 > 85%
	Over-provisioned (waste)	Balanced (optimal)	At risk (needs more)

COMPUTE TARGET SPEC (RECOMMENDATION)

Scale Down (Oversized)
Target = P95 + 20% headroom
→ Reduce vCPU / RAM

Scale Up (Undersized)
Target = P95 + 30% headroom
→ Increase vCPU / RAM

OUTPUTS

Persisted Recommendation
/research & /vms/<uuid>
status: pending / applied / dismissed

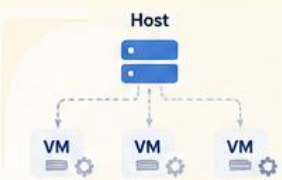
User Action
User clicks "Apply" to execute recommendation

API Call
POST /rightsizing/{id}/apply
→ ResizeService.apply_recommendation()
→ SCP API resize (scale action)

Result
Returns success OR "requires_confirmation" (shutdown needed for downsize)

DOCUMENTED BLIND SPOT

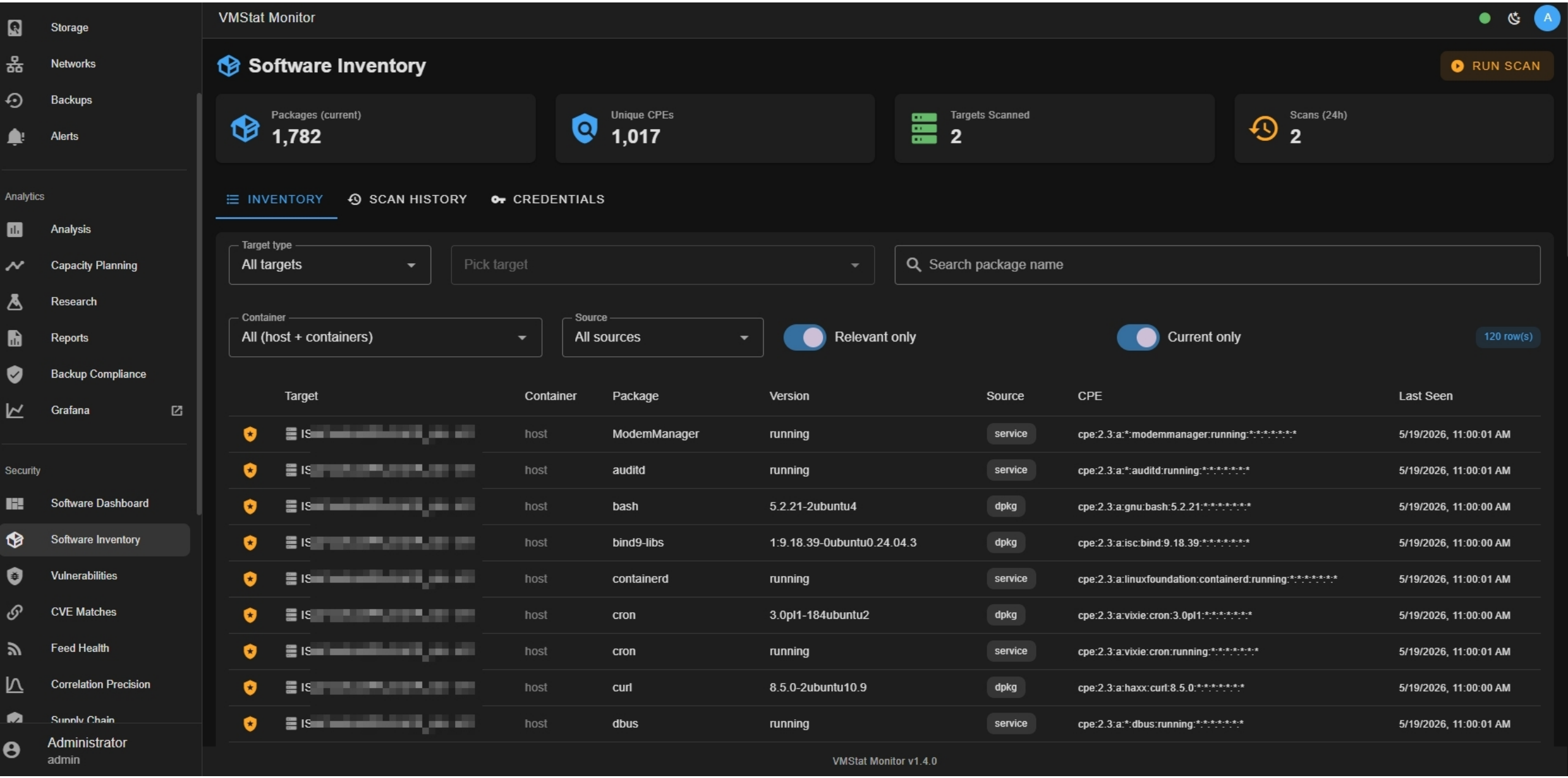
Host P95 used as proxy for all VMs on that host.
A single noisy neighbor biases the analysis.
Per-VM metrics from VictoriaMetrics are visible in Grafana but not yet wired into the classifier — Phase 5+ work.



DATA FLOW (DAILY RUN)

Trigger 02:00 → Pull 30-day rollups → Compute P95 → Classify VMs → Compute target spec → Cooldown gate → Persist result → User Apply & Resize (SCP)

★ Goal: Right resource • Lower cost • Improve performance • Prevent risk • Automated & safe


VMStat Monitor
Storage
Networks
Backups
Alerts
Analytics
Analysis
Capacity Planning
Research
Reports
Backup Compliance
Grafana
Security
Software Dashboard
Software Inventory
Vulnerabilities
CVE Matches
Feed Health
Correlation Precision
Administrator
admin
Software Inventory
RUN SCAN
Packages (current)
1,782
Unique CPEs
1,017
Targets Scanned
2
Scans (24h)
2
INVENTORY
SCAN HISTORY
CREDENTIALS
Target type
All targets
Pick target
Search package name
Container
All (host + containers)
Source
All sources
Relevant only
Current only
120 row(s)
Target
Container
Package
Version
Source
CPE
Last Seen
host
ModemManager
running
service
cpe:2.3:a:*:modemmanager:running:*:*:*:*:*:*:*:*
5/19/2026, 11:00:01 AM
host
auditd
running
service
cpe:2.3:a:*:auditd:running:*:*:*:*:*:*:*:*
5/19/2026, 11:00:01 AM
host
bash
5.2.21-2ubuntu4
dpkg
cpe:2.3:a:gnu:bash:5.2.21:*:*:*:*:*:*:*
5/19/2026, 11:00:00 AM
host
bind9-libs
1:9.18.39-0ubuntu0.24.04.3
dpkg
cpe:2.3:a:isc:bind:9.18.39:*:*:*:*:*:*:*
5/19/2026, 11:00:00 AM
host
containerd
running
service
cpe:2.3:a:linuxfoundation:containerd:running:*:*:*:*:*:*:*:*
5/19/2026, 11:00:01 AM
host
cron
3.0pl1-184ubuntu2
dpkg
cpe:2.3:a:vixie:cron:3.0pl1:*:*:*:*:*:*:*:*
5/19/2026, 11:00:00 AM
host
cron
running
service
cpe:2.3:a:vixie:cron:running:*:*:*:*:*:*:*:*
5/19/2026, 11:00:01 AM
host
curl
8.5.0-2ubuntu10.9
dpkg
cpe:2.3:a:haxx:curl:8.5.0:*:*:*:*:*:*:*:*
5/19/2026, 11:00:00 AM
host
dbus
running
service
cpe:2.3:a:*:dbus:running:*:*:*:*:*:*:*:*
5/19/2026, 11:00:01 AM
VMStat Monitor v1.4.0

Workflow 6 — Software Inventory (Agentless SSH) VMS

No agent binary — pulls package list via SSH using stored credential profiles
app/services/ssh_scanner_service.py

1 Trigger
Start inventory scan

`POST /software/scan` (single target)
or "scan all active targets" background task
: `app/api/v1/software.py:559`

2 Resolve credentials
Get SSH credentials securely

`TargetCredential` → `CredentialProfile`
(Fernet-encrypted SSH password
OR private key + passphrase)

3 asyncssh connect
Connect to target via SSH
(per-VM concurrent)

- timeout + retry
- audit row in `SoftwareScan` table

4 Host package discovery
Detect OS & list installed packages

`dpkg-query -W` (Debian/Ubuntu)
→ name + version
fallback: `rpm -qa --queryformat` (RHEL/CentOS)
tab-separated parse

5 Service + binary probes
Enumerate running services & binaries

`systemctl list-units --type=service`
Binary version probes:
nginx · mysql · postgres · docker · redis · haproxy · node · python3

6 Container scan (cascade)
Scan containers and packages

`docker ps` → list containers
per container try `dpkg` → fallback `rpm` → fallback `apk` (Alpine)
tag rows with `container_name`

7 CPE normalization
Normalize version to CPE

`cpe_normalizer.py` (name, version)
→ CPE 2.3 URI
vendor lookup table
fallback to wildcard: `cpe:2.3:a:*:<name>`

8 Soft-delete + insert
Maintain history & current state

Old rows: `is_current=false` (preserved for history)
New rows: `is_current=true`
Correlation engine reads `is_current=true`

Agentless via SSH = No binary to install/update across hundreds of VMs
Credentials rotated through one profile • Works on any *nix with sshd
Downside: Windows VMs not covered (would need WinRM / agent).

HOW IT WORKS

PROBES & TOOLS

Package Managers
- Debian/Ubuntu : `dpkg-query -W`
- RHEL/CentOS : `rpm -qa --queryformat`

Service & Binary Detection
- `systemctl list-units --type=service`
- Probes: nginx, mysql, postgres, docker, redis, haproxy, node, python3

Container Support
- `docker ps`
- `dpkg` / `rpm` / `apk` (Alpine)
- Tag by `container_name`

OUTPUT EXAMPLE (per target)

Package / Service	Version	CPE 2.3	Source	Status
nginx	1.24.0	cpe:2.3:a:nginx:nginx:1.24.0:*:*:*:*:*:*:*	package	●
postgresql	14.10	cpe:2.3:a:postgresql:postgresql:*:*:*:*:*:*:*:*	package	●
docker	24.0.7	cpe:2.3:a:docker:docker:24.0.7:*:*:*:*:*:*:*	service	●
redis	7.0.11	cpe:2.3:a:redis:redis:7.0.11:*:*:*:*:*:*:*	binary	●
myapp (ctr)	2.1.3	cpe:2.3:a:myapp:myapp:2.1.3:*:*:*:*:*:*:*	container	●
...	...	... ...	...	...

DATA STORED

SoftwareScan table
- target_id
- package / service / binary
- version
- cpe 2.3
- source (package / service / container)
- container_name (if any)
- is_current (true/false)
- first_seen / last_seen

BENEFITS

- Complete inventory without installing agents
- Works with encrypted SSH credentials
- Supports major Linux distributions
- Container-aware inventory
- Normalized to CPE for vulnerability correlation
- Historical tracking (soft-delete)
- Scalable, concurrent & scheduled

DATA FLOW (SCHEDULED)

Analysis
Capacity Planning
Research
Reports
Backup Compliance
Grafana

Security
Software Dashboard
Software Inventory
Vulnerabilities
CVE Matches
Feed Health
Correlation Precision
Supply Chain

ISO Compliance
Incident Log
Change Log

Administration
Administrator
admin

CVE ↔ Software Matches

Drill-down ของทุกคู่ (CVE, installed_software, target) ที่ correlation engine จับได้ — กรองด้วย match_type, severity, KEV หรือ search.

ⓘ *ค่าเริ่มต้นแสดงเฉพาะ `exact` + `version_range` (high-confidence); ติ๊ก `vendor_product` ด้านล่างเพื่อดู low-confidence pairs (มักเป็น false positive เช่น Shellshock บน bash 5.x).*

← PRECISION OVERVIEW

Match type: ✓ Exact | ✓ Version Range | Vendor Product

Severity: Critical | High | Medium | Low

KEV: ANY | KEV

Search

Patch status (Ubuntu version_range only): Patched | Unpatched | Unknown

Hide patched (Canonical-backported FPs)

417 pairs match current filter

CLEAR | APPLY

CVE	Severity	CVSS	Match	Software	Target	Patch status	Last matched
CVE-2025-48384 KEV EPSS 0.60%	HIGH	8.0 v3	version_range cpe:2.3:a:git-scm:git:*:*:*:*:*:*:*:* git < 2.43.7 (+7 more)	git 1:2.43.0-1ubuntu7.3	NC[redacted] 10.1.[redacted]	patched	19/05/2026, 14:00:00
CVE-2025-32463 KEV EPSS 57.34%	HIGH	7.8 v3	version_range cpe:2.3:a:sudo_project:sudo:*:*:*:*:*:*:*:* sudo ≥ 1.9.14, < 1.9.17	sudo 1.9.15p5-3ubuntu5.24.04.2	NO[redacted] 10.1.1[redacted]	patched	19/05/2026, 14:00:00
CVE-2025-32463 KEV EPSS 57.34%	HIGH	7.8 v3	version_range cpe:2.3:a:sudo_project:sudo:*:*:*:*:*:*:*:* sudo ≥ 1.9.14, < 1.9.17	sudo 1.9.15p5-3ubuntu5.24.04.2	ISA[redacted] 10.1.[redacted]	patched	19/05/2026, 14:00:00
CVE-2026-4176 EPSS 0.03%	CRITICAL	9.8 v3	version_range cpe:2.3:a:perl:perl:*:*:*:*:*:*:*:* perl ≥ 5.9.4, < 5.40.4 (+2 more)	perl 5.38.2-3.2ubuntu0.2	NC[redacted] 10.1.1[redacted]	unknown	19/05/2026, 14:00:00
CVE-2026-4176 EPSS 0.03%	CRITICAL	9.8 v3	version_range cpe:2.3:a:perl:perl:*:*:*:*:*:*:*:* perl ≥ 5.9.4, < 5.40.4 (+2 more)	perl 5.38.2-3.2ubuntu0.2	IS[redacted] 10.1.1[redacted]	unknown	19/05/2026, 14:00:00
CVE-2026-31789 EPSS 0.01%	CRITICAL	9.8 v3	version_range cpe:2.3:a:openssl:openssl:*:*:*:*:*:*:*:* openssl ≥ 3.0.0, < 3.0.20 (+4 more)	libssl3 3.5.5-r0 · vmstat-db-1	NC[redacted] 10.1.10[redacted]	unknown	19/05/2026, 14:00:00

Workflow 7 — CVE Correlation (3-Tier Match Hierarchy) VMS

app/services/correlation_service.py — daily 07:00 UTC
O(N_sw + N_cve) via in-memory indexes

1. **Build 3 indexes (in-memory)**
   exact_idx: CPE URI → CVE
   vp_idx: vendor:product → CVE
   ranges_idx: vendor:product → [(CVE, versionStart/End)]
2. **Iterate `is_current=true` software**
   Parse installed version: PEP 440 first
   Fall back to Debian dpkg rules
   (handles `1:1:2.3-1ubuntu5.24.04.2`)
3. **Tier 1 — exact CPE match**
   Installed CPE URI matches one of CVE's `affected_cpes`
   → claim · `match_type=exact`
4. **Tier 2 — version-range**
   Same vendor:product, installed version inside
   [versionStartIncluding/Excluding, versionEndIncluding/Excluding]
   → claim · `match_type=version_range`
5. **Tier 3 — vendor:product fallback**
   Same `cpe:2.3:<vendor>:<product>` prefix
   NO range data OR outside all ranges
   → capped at VP_FANOUT_CAP=100 (prevents lib/openssl explosion)
6. **Upsert vulnerability_matches**
   Columns: match_type, matched_cpe,
   first_matched_at, last_matched_at, idempotent across runs
7. **Ubuntu USN post-filter**
   For Debian/Ubuntu rows: if USN has `fixed_version`
   AND installed >= `fixed` → `patch_status=patched`
   (bridges backport blind spot)
8. **Overlay KEV + EPSS**
   KEV: if CVE in CISA `catalog` → `is_kev=true` + due_date
   EPSS: daily CSV → exploit probability % + percentile
   (update-only, never creates)

3-TIER MATCH HIERARCHY

TIER 1 — Exact CPE Match
Highest Confidence
CPE URI EXACTLY matches CVE affected_cpes
match_type = exact

→

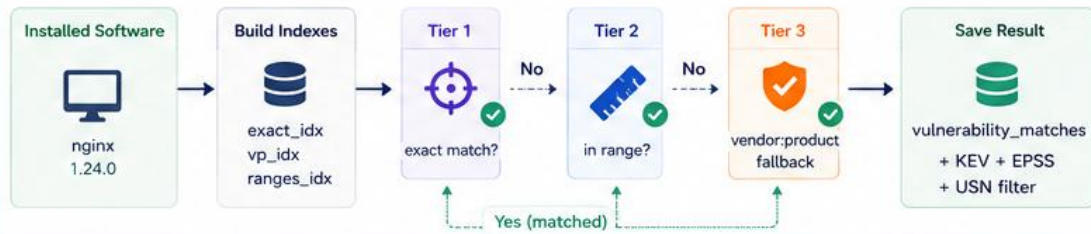
TIER 2 — Version-Range Match
High Confidence
Version is within CVE version range
match_type = version_range

→

TIER 3 — Vendor:Product Fallback
Lower Confidence
Same vendor:product prefix (no range data or outside all ranges)
match_type = vendor_product

Stop at first successful tier (1 →2 → 3). Cap fallback at 100 CVEs per product.

EXAMPLE MATCH FLOW



DATA SOURCES

- Installed Software (is_current=true)
- CVE Database (NVD)
- CPE Dictionary
- CISA KEV Catalog
- EPSS Daily CSV
- Ubuntu USN Feeds

OUTPUT FIELDS (vulnerability_matches)

- cve_id
- installed_cpe
- match_type (exact / version_range / vendor_product)
- matched_cpe
- first_matched_at
- last_matched_at
- is_kev (true/false)
- epss_score, epss_percentile
- patch_status (patched / unpatched / NULL)
- due_date (from KEV)

KEY BENEFITS

- High accuracy with 3-tier matching strategy
- Handles version ranges and backports
- Reduces false positives from vendor fallback
- Risk prioritization with KEV + EPSS
- Fast & scalable using in-memory indexes

KNOW LIMITATIONS (BLIND SPOTS)

- **AND-node configurations** (~5–10% of CVEs with composite "OS AND app" conditions) are skipped at range extraction
- **Unparseable installed versions** fall to vendor:product tier
- **non-Debian distros** (Alpine, RHEL custom patches) not USN-filtered → patch_status=NULL
- **Windows VMs not covered** (would need WinRM / agent)

SCHEDULE & SCALE

Daily 07:00 UTC
Complexity: O(N_sw + N_cve)
In-memory indexes for speed

Analytics
Analysis
Capacity Planning
Research
Reports
Backup Compliance
Grafana

Security
Software Dashboard
Software Inventory
Vulnerabilities
CVE Matches
Feed Health
Correlation Precision
Supply Chain

ISO Compliance
Incident Log
Change Log

Administration
Administrator

Feed Health

Sync status of every threat-intelligence feed powering CVE correlation — NVD corpus, EPSS scores, CISA KEV, and MISP instances.

REFRESH

9	0	1	9
Healthy feeds	Failed feeds	Stale (>24h)	Total feeds

Feed detail

Feed	Status	Last sync	Age	Items	Duration	Error	Actions
alpine	success	19/05/2026, 12:45:00	3h ago	42,757	12.4s	—	
epss	success	19/05/2026, 13:00:00	3h ago	333,848	27.2s	—	SYNC
kev	success	19/05/2026, 13:30:00	2h ago	1,592	2.1s	—	SYNC
misp:1 (MISP_SWU)	success	19/05/2026, 15:45:00	39m ago	0	0.1s	—	SYNC
mitre	success	19/05/2026, 11:30:00	4h ago	351,483	419.9s	—	
nvd	success	19/05/2026, 13:15:00	3h ago	74	2.7s	—	SYNC
usn	success	19/05/2026, 12:30:00	3h ago	13	10.3s	—	
vulnrichment	success	10/05/2026, 10:30:00	9d ago	146,877	166.6s	—	
correlation	success	19/05/2026, 14:00:00	2h ago	5,929	64.4s	—	

Workflow 8 — Vulnerability Feed Sync (6 Sources) VMS

Layered ingestion • NVD authoritative • MISP/Vulnrichment fill CVSS gaps
EPSS/KEV overlay signals • USN/Alpine post-filter

WHAT IT DOES

Aggregates vulnerability and exploit intelligence from 6 trusted sources and normalizes them into VMS. Each feed has a specific role. Together they provide **complete, prioritized**, and **actionable** vulnerability data.

LAYERED INGESTION ARCHITECTURE

1. **Ingest** — Pull from 6 sources by cadence
2. **Normalize** — Standardize fields, dedupe, map CVE
3. **Enrich** — Overlay EPSS/KEV, fill CVSS gaps
4. **Filter** — Post-filter by USN/Alpine
5. **Store & Use** — Save to VMS DB & dashboards

KEY OUTCOMES

- **Complete Coverage** — Combines authoritative (NVD) + community + threat intel
- **Better Prioritization** — EPSS, KEV and exploit signals raise true risk
- **Fewer False Positives** — USN/Alpine post-filter marks patched/fixed versions
- **Actionable & Timely** — Appropriate cadence for each source, always up to date

DATA FLOW OVERVIEW

THE 6 FEEDS

#	Feed	Endpoint / Source	Cadence	Role – What it does
1	NVD 2.0 (NIST NVD)	services.nvd.nist.gov/rest/json/cves/2.0	Daily incremental	**Authoritative** • lastModStartDate windowed • 120-day max window • rate-limit 5 req/30s • without key
2	MISP (MISP Threat Sharing)	User-configured instance	Hourly (per config)	**Curated Threat Events** • Fills CVSS gap when • NVD hasn't scored yet
3	EPSS	epss.empiricalsecurity.com/epss_scores-current.csv.gz	Daily	**Update-only** • Adds exploit-probability % + percentile • Never creates rows
4	KEV (CISA)	cisa.gov/…/known_exploited_vulnerabilities.json	Daily	**Actively Exploited CVEs** • Sets is_kev=true + due_date • Can stub rows NVD fills later
5	Ubuntu USN (ubuntu)	ubuntu.com/security/notices.json	Daily	**Backport Tracking** • Post-filter marks Debian/Ubuntu matches as patched if installed ≥ fixed_version
6	Vulnrichment (Vulnrichment)	CISA tarball (manual)	On-demand	**Fills NVD-deprioritized gaps** • Provides CVSS v3/v4 for lower priority CVEs

FIELDS ENRICHED IN VMS

- CVSS v3 / v4
- CVSS Source (NVD / MISP / Vulnrichment)
- EPSS Score & Percentile
- KEV Flag (is_kev)
- Due Date (KEV)
- USN Patch Status
- References & Links

HOW IT WORKS TOGETHER

NVD provides the baseline. MISP + Vulnrichment fill scoring gaps. EPSS and KEV add risk and context. USN/Alpine post-filter reduces noise by marking patched versions.

WHY POST-FILTER MATTERS

Many distributions backport fixes.
USN (Ubuntu) identifies patched packages so we don't flag already-protected systems.

NOTES

- All feeds are idempotent and deduplicated by CVE + data_source.
- Resilient to failures with backoff and retry.
- In-memory indexes for fast correlation and low latency.

GOAL

Complete, accurate, prioritized vulnerability data—ready for detection, reporting, and remediation.

Q&A

Thank You

AI in NOC Life

จากข้อมูล → วิเคราะห์ → ตัดสินใจ →
ลงมือแก้ไขอย่างปลอดภัย

Key Takeaways

1. AI ใช้งานได้จริงเมื่อมี Model + Context + Harness + Control
2. เปลี่ยนงาน NOC จาก Reactive เป็น Proactive & Automated
3. Automation ต้องมี Guardrail, Audit Log และ Human-in-the-loop

Next Step: เริ่มจาก Use Case ที่วัดผลได้ เช่น Vulnerability Scan, Threat Intel Block, Credential Monitor และ VM Right-Sizing

Mahutthawat Raksakiettisak | Computer Center, Srinakharinwirot University