Open Source Intelligence

Building Your Own Vulnerability Radar

How to harvest, correlate, and visualize world-class threat intel for $0

Jerry Gamblin — Principal Engineer, Cisco & Founder, RogoLabs

BSides Galway

The Six-Figure Illusion

The Myth

"You need expensive subscriptions to get good threat intel."

The Reality

Most commercial feeds just repackage open data you can get for free. They aggregate public sources, add branding, and charge thousands monthly.

The Friction

Black-box feeds lack context. They don't know your stack, so they spam you with noise. Generic alerts about vulnerabilities you don't even run.

The "Builder" Mindset

Stop consuming feeds passively. Start generating insights that matter to your environment. This is about shifting from vendor dependency to engineering autonomy.

1

Harvest

Pull raw vulnerability data from open sources

2

Correlate

Match intel against your actual tech stack

3

Visualize

Surface actionable risks in real-time

The Stack: Python, GitHub Actions, and JSON.

Total Cost: $0. No subscriptions, no vendor lock-in, full control.

Why Open Source Intelligence?

The Commercial Trap

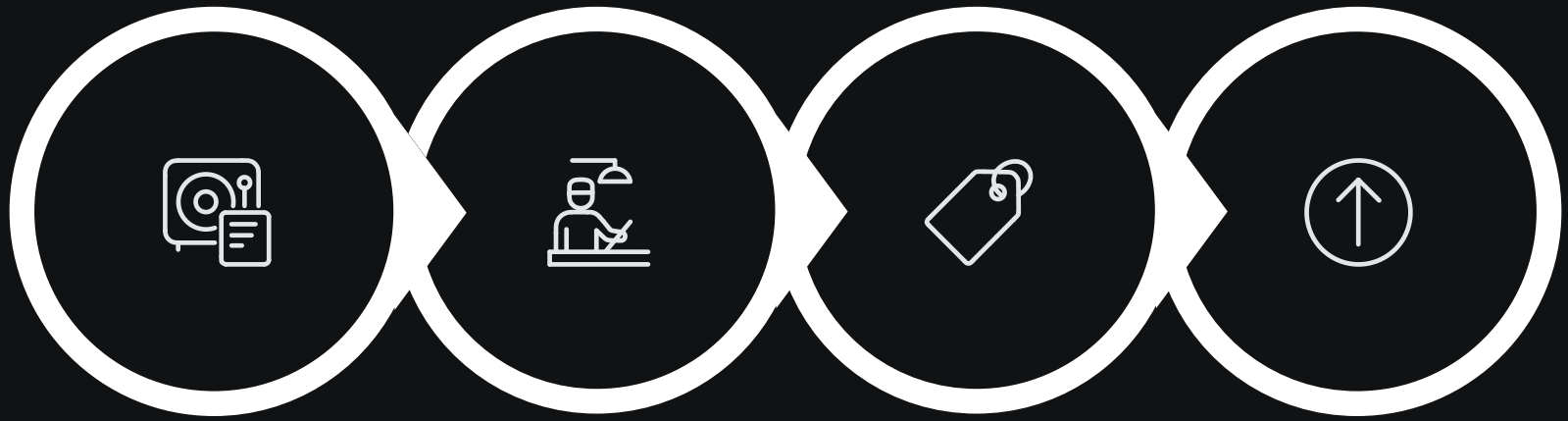
- Vendor lock-in and recurring costs
- Black-box algorithms you can't audit
- Data you could access yourself, repackaged
- Update delays when you need speed

The OSINT Advantage

- Zero licensing costs, infinite scale
- Full transparency and customization
- Real-time access to source data
- Community-driven improvements

You're not building a product. You're building a capability.

Architecture Deep Dive



Data Collection

Correlation Engine

Enrichment

Output

The entire system runs as a serverless workflow in GitHub Actions. Each phase is modular, testable, and extensible. Want to add a new data source? Drop in a new collector function. Need custom scoring? Modify the enrichment logic.

The Raw Materials: Open Data Sources

The foundation of your radar is high-quality, freely available threat intelligence. This approach leverages **direct data files and GitHub repositories**, ensuring **NO APIs, NO authentication, and NO rate limits**. These sources are maintained by government agencies, research institutions, and the security community.

CVE List V5

The authoritative source of truth for CVEs, available directly from its **GitHub repository**. JSON-formatted vulnerability records with structured metadata, descriptions, and references. **No API needed; simply clone the repository.**

NVD Data Dumps

The National Vulnerability Database provides comprehensive data. Access the bulk data feeds at **NVD Data Feeds**. **No REST API required; download the files directly.**

CISA KEV

Known Exploited Vulnerabilities catalog. Answers the critical question: "What is being hacked right now in the wild?" See the catalog at **CISA KEV**.

EPSS

Exploit Prediction Scoring System. Machine learning-based probability scores that forecast which vulnerabilities will be exploited next. Learn more at **FIRST EPSS**.

PatchThis

Crowd-sourced exploit availability tracker. Community-driven intelligence on which CVEs have working exploit code available.

Data Source Deep Dive

CVE/NVD

- 200K+ vulnerabilities cataloged
- CVSS scoring and metadata
- Vendor advisories aggregated
- Updated continuously

CISA KEV

- Known Exploited Vulnerabilities
- Government-verified active threats
- Mandatory patching timelines
- The "must-fix" list

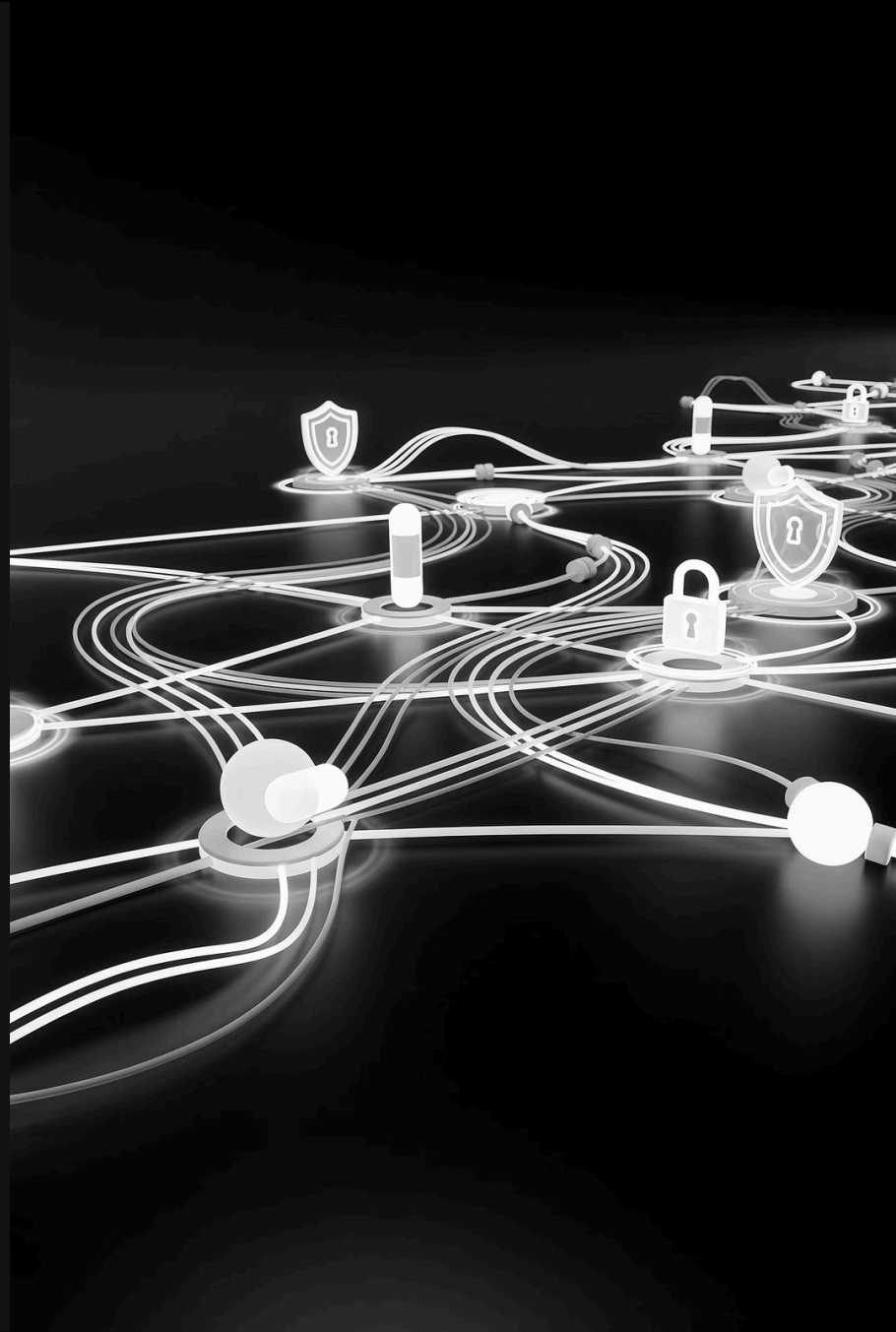
EPSS Scores

- Exploit Prediction Scoring
- Probability-based prioritization
- Machine learning driven
- Updates daily

PatchThis

- Vendor patch availability
- Release date tracking
- Remediation guidance
- Actionability signal

Four free feeds. Unlimited intelligence when combined correctly.



The "Harvest" Architecture

Sequential Data Collection

The harvester runs on a schedule, pulling fresh data every hour. No API keys means no rate limits, no authentication failures, no key rotation headaches. This is achieved by downloading public files directly or cloning public Git repositories.

Mechanism: Python with `requests` for direct file downloads and `subprocess` for Git operations. GitHub Actions provides the compute—free for public repos, generous limits for private.

Cron Schedule: `5 * * * *`

Scheduled to run 5 minutes after every hour to ensure reliable data ingestion.

Data is downloaded, validated, and stored as JSON artifacts for the correlation phase.

```
import requests
import subprocess
import os

# Git clone pattern
def fetch_cve_list():
    repo_url = "https://github.com/CVEProject/cvelistV5.git"
    target_dir = "cvelistV5_data"
    if not os.path.exists(target_dir):
        subprocess.run(["git", "clone", repo_url, target_dir])
    else:
        subprocess.run(["git", "-C", target_dir, "pull"])

# Direct download pattern
def fetch_nvd_data_dump(year):
    url = f"https://nvd.nist.gov/feeds/json/cve/1.1/nvdcve-1.1-{year}.json.zip"
    file_path = f"nvd_data_{year}.zip"
    response = requests.get(url, stream=True)
    with open(file_path, 'wb') as f:
        for chunk in response.iter_content(chunk_size=8192):
            f.write(chunk)

# Example execution
fetch_cve_list()
fetch_nvd_data_dump(2023)
```

The Watchlist: Your Filter

Generic vulnerability feeds generate thousands of alerts daily. Your watchlist transforms noise into signal by filtering for what actually matters to YOUR infrastructure.

```
# watchlist.yml
vendors:
  - Microsoft
  - Apache
  - Cisco
  - VMware

products:
  - Windows Server
  - Exchange
  - IIS
  - Log4j

keywords:
  - remote code execution
  - privilege escalation
  - authentication bypass
```

Why This Works

- Reduces alert volume by 95%+
- Focuses on your actual attack surface
- Eliminates irrelevant CVEs instantly
- Scales with your environment

Start with 10-20 critical vendors. Expand based on your asset inventory.

The Secret Sauce: Context

The vulnerability landscape is overwhelming. Over 200,000 CVEs exist, with thousands added monthly. Without filtering, you drown in noise.

The Problem

Too many vulnerabilities to track manually. Generic feeds alert on everything, creating alert fatigue.

The Fix

watchlist.yaml defines your actual attack surface. List the technologies you run—specific vendors, products, versions.

The Result

Intelligence becomes actionable. You only see vulnerabilities that affect systems you actually operate.

Prioritization Logic: The Critical Formula

Not all vulnerabilities deserve the same attention. Our enrichment engine applies multi-factor scoring to surface what truly matters.

Core Logic

```
# enrichment.py
def is_critical(cve):
    # Critical if exploited + on watchlist
    is_watchlist_exploit = cve.get('in_patchthis', False) and \
                           cve.get('in_watchlist', False)

    # Critical if CISA KEV + very high CVSS
    is_kev_high_cvss = cve.get('in_cisa_kev', False) and \
                       cve.get('cvss_score', 0.0) >= 9.0

    # Critical if high EPSS and high CVSS
    is_high_epss_cvss = cve.get('epss_score', 0.0) >= 0.5 and \
                        cve.get('cvss_score', 0.0) >= 7.0 # Adjusted for conciseness

    return is_watchlist_exploit or is_kev_high_cvss or is_high_epss_cvss
```

Advanced Filtering

Threshold Tuning: Adjust CVSS and EPSS thresholds based on your risk appetite and team capacity.

Watchlist Matching: String matching against vendor/product names from your watchlist.yaml configuration.

Exploit Signal: Prioritize CVEs with confirmed exploit code or active exploitation over theoretical risks.

This beats standard feeds because it adapts to your specific environment and risk tolerance. No more generic alerts.

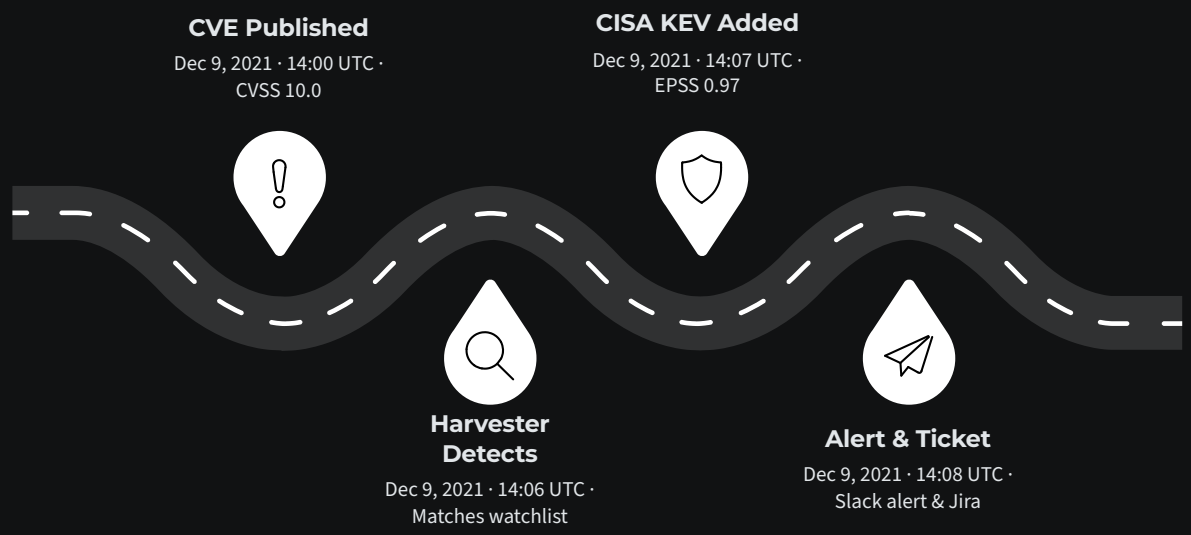
Enrichment in Action

The correlation engine transforms raw CVE data into contextualized intelligence. Each vulnerability gets scored, tagged, and linked to your infrastructure.


```
{
  "cve_id": "CVE-2024-1234",
  "cvss_score": 9.8,
  "epss_score": 0.89,
  "watchlist_hit": true,
  "in_cisa_kev": true,
  "priority": "CRITICAL",
  "recommendation": "Patch immediately - active exploitation
detected"
}
```


The result is a JSON object that tells a complete story, not just a score. It connects the dots between what's exploited in the wild and what you actually run.

Real-World Scenario: Log4Shell



From CVE publication to team notification in **8 minutes**. No human intervention required.

*8-minute notification is a best-case alignment with our hourly heartbeat. Real-world latency is guaranteed under 2 hours.

Dashboards for Humans

The "No-Server" Approach

Jinja2 templates generate `radar_report.md` with metrics, severity breakdowns, and actionable alerts.

Markdown reports render natively in GitHub, offering zero hosting costs, version control, and global accessibility, with visual status badges.

23

Critical Alerts

Active threats

47

High Priority

Exploits available

156

Watchlist CVEs

Monitored

Operationalizing the Data

Intelligence isn't useful until it triggers a workflow. Transform your radar from a static dashboard into an automated response system.

01

GitHub Issues

Automatically create issues for critical findings. Include CVE details, CVSS scores, and remediation steps. Assign to relevant teams.

02

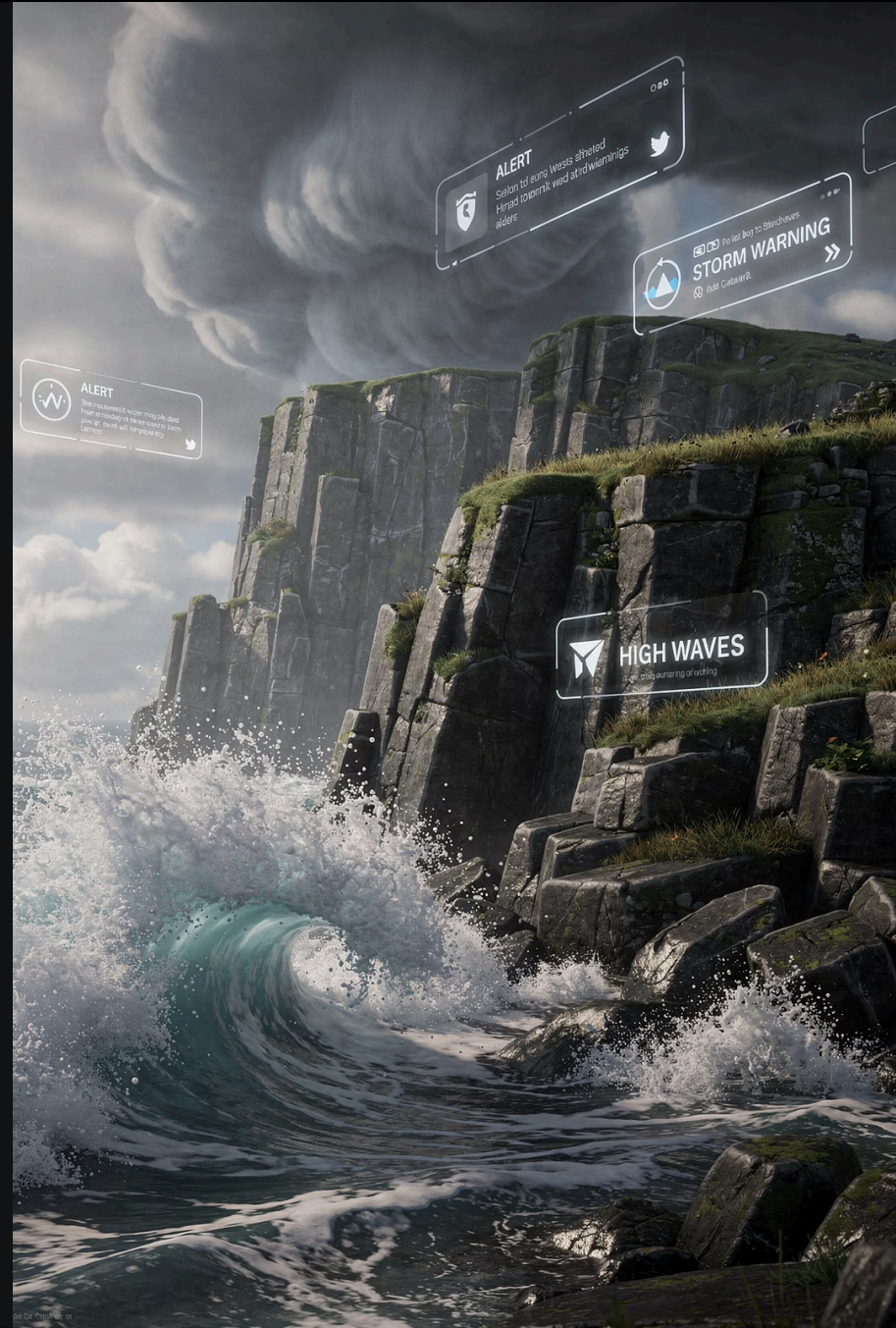
ChatOps Integration

Webhook alerts to Slack, Discord, or Microsoft Teams. Real-time notifications when high-priority vulnerabilities hit your watchlist.

03

SIEM Integration

Export enriched JSON to your security information and event management system for correlation with logs and alerts.



Integration Points

VulnRadar doesn't live in isolation. It feeds your existing security workflow.

Slack/Discord

Real-time critical alerts

Threaded discussions

Jira/ServiceNow

Auto-ticket creation

Priority assignment

SIEM Integration

Splunk/Elastic ingestion

Correlation with logs

GitHub Ops

Automated issue creation

Vulnerability tracking

Every integration is a webhook or API call away. No vendor partnerships required.

Let's See It Run

Pray to the demo gods 🙏

Live Demo: VulnRadar in Action

1

Step 1: Edit Watchlist

Add your technology stack to `watchlist.yaml` - vendors, products, and specific versions you want to track.

2

Step 2: Trigger Action

Manually trigger the GitHub Action or let the cron schedule run. Watch the harvest, correlation, and enrichment phases execute.

3

Step 3: Review Results

Check the generated `radar_report.md` for prioritized vulnerabilities. Critical alerts appear at the top with full context.

4

Step 4: Respond

Review automatically created GitHub issues. Click through to CVE details, vendor advisories, and patch information.

Demo uses `notify.py --demo` *to simulate a critical watchlist hit when live data is quiet.*

Performance & Scale

Built for production environments. Handles enterprise-scale vulnerability tracking without breaking a sweat.

200K+

CVEs Processed

Full NVD catalog indexed and searchable in under 3 minutes

24x...

Update Cycle

Automated harvesting runs 24x daily via GitHub Actions

<100...

Resource Footprint

Runs on free GitHub infrastructure. Zero hosting costs.

Tech Stack

- Python 3.11+
- Pandas for data manipulation
- Jinja2 for templating
- GitHub Actions for orchestration
- Markdown for output

Scaling Considerations

- Add more watchlist entries without performance hit
- Parallel data fetching for speed
- Incremental updates reduce bandwidth
- Static output = infinite read scalability

You Can Build This Today

We've covered the complete architecture of a zero-cost vulnerability intelligence platform. Let's recap the engineering wins.

1. Harvest

Free, open-source threat data from authoritative sources. No subscriptions, no rate limits.

2. Correlate

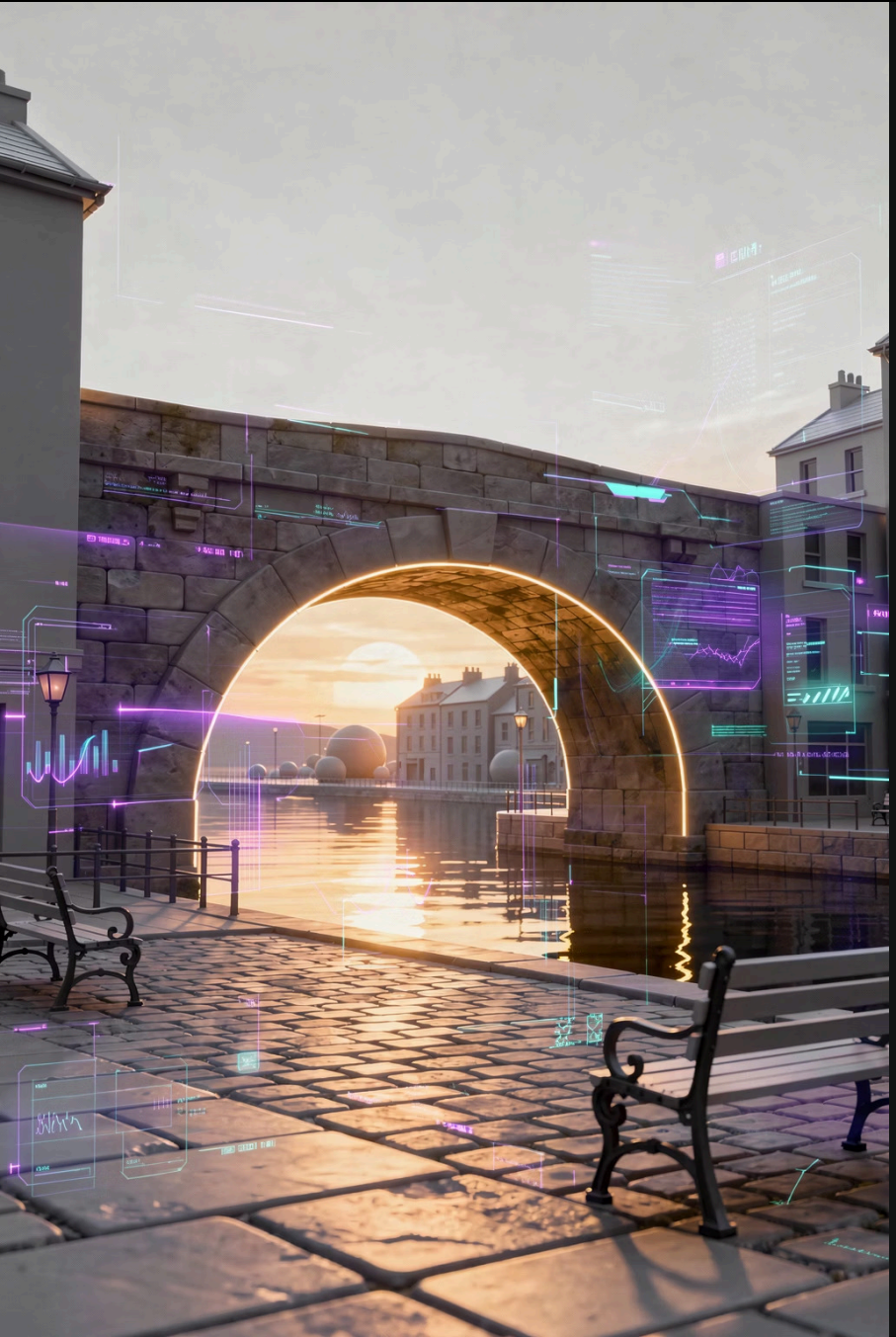
Context-aware filtering using your actual tech stack. Signal over noise.

3. Visualize

Markdown dashboards that render in GitHub. Zero infrastructure overhead.

Total Cost: $0 in cloud spend. Just your engineering time to customize and deploy.

The Code: Available at `github.com/RogoLabs/VulnRadar`



Getting Started: 30-Minute Setup

From zero to operational threat intelligence in half an hour. No infrastructure required.

Step 1: Fork the Repo

Clone github.com/RogoLabs/VulnRadar
No API keys or credentials needed

Step 2: Customize Watchlist

Edit data/watchlist.yml
Add your vendors and products

Step 3: Enable GitHub Actions

Turn on workflow in repo settings
Runs automatically every hour

Step 4: Configure Outputs

Set Slack webhook (optional)
Add Jira integration (optional)

Step 5: First Run

Trigger manual workflow run
Review generated dashboard

Pro tip: Start with a minimal watchlist. Add entries as you identify gaps in coverage.

Common Pitfalls & Solutions

Mistakes to Avoid

Problem: Watchlist Too Broad
Solution: Start narrow, expand based on actual assets

Problem: Alert Fatigue
Solution: Tune EPSS thresholds, focus on KEV first

Problem: No Integration
Solution: Intelligence without action is just data

Problem: Set and Forget
Solution: Review and refine watchlist quarterly

Optimization Tips

- Use CVSS + EPSS + KEV together for scoring
- Create separate watchlists for different teams
- Archive old vulnerabilities after patching
- Monitor false positive rates
- Document your tuning decisions
- Share findings with the community

The system improves with feedback. Treat it like a living tool, not a static dashboard.

Key Takeaways for Builders

Stop Buying Black Boxes

Commercial threat intel platforms are expensive repackaging of free data. Build your own pipeline with full transparency and control.

Context Is Everything

Generic feeds create noise. Filtering through your watchlist transforms data into actionable intelligence specific to your environment.

Automate Everything

Intelligence without workflow integration is just interesting reading. Connect your radar to ticketing, ChatOps, and SIEM for real impact.

Start Small, Iterate Fast

Begin with a minimal watchlist and basic scoring. Tune thresholds based on your team's capacity and organizational risk appetite.

Questions & Contact

Get the Code Repository:

github/RogoLabs/VulnRadar

Fork it, customize it, break it, improve it. Pull requests welcome.

Connect

Twitter/X: @JGamblin
Email: jerry.gamblin@gmail.com

Resources

- See data/sources/ for collector implementations
- CISA KEV catalog
- EPSS scoring methodology
- .github/workflows/update.yml for the cron logic
- templates/report.md.j2 for dashboard code

Final Thought: Stop renting intelligence. Start building radars.