What RSEs should know about supply chain security

Mark Woodbridge

STEP-UP RSLondon 2026

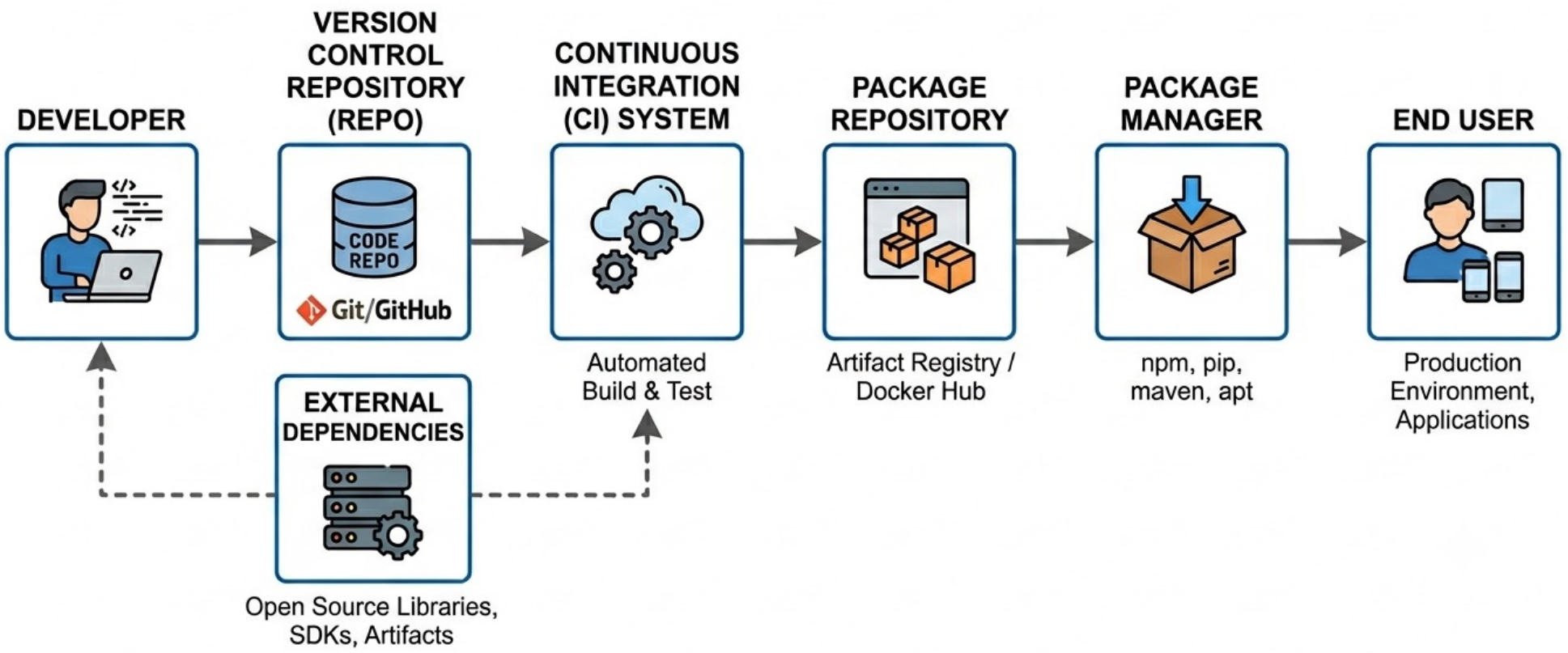
Monday 29th June 2026

PharosAI

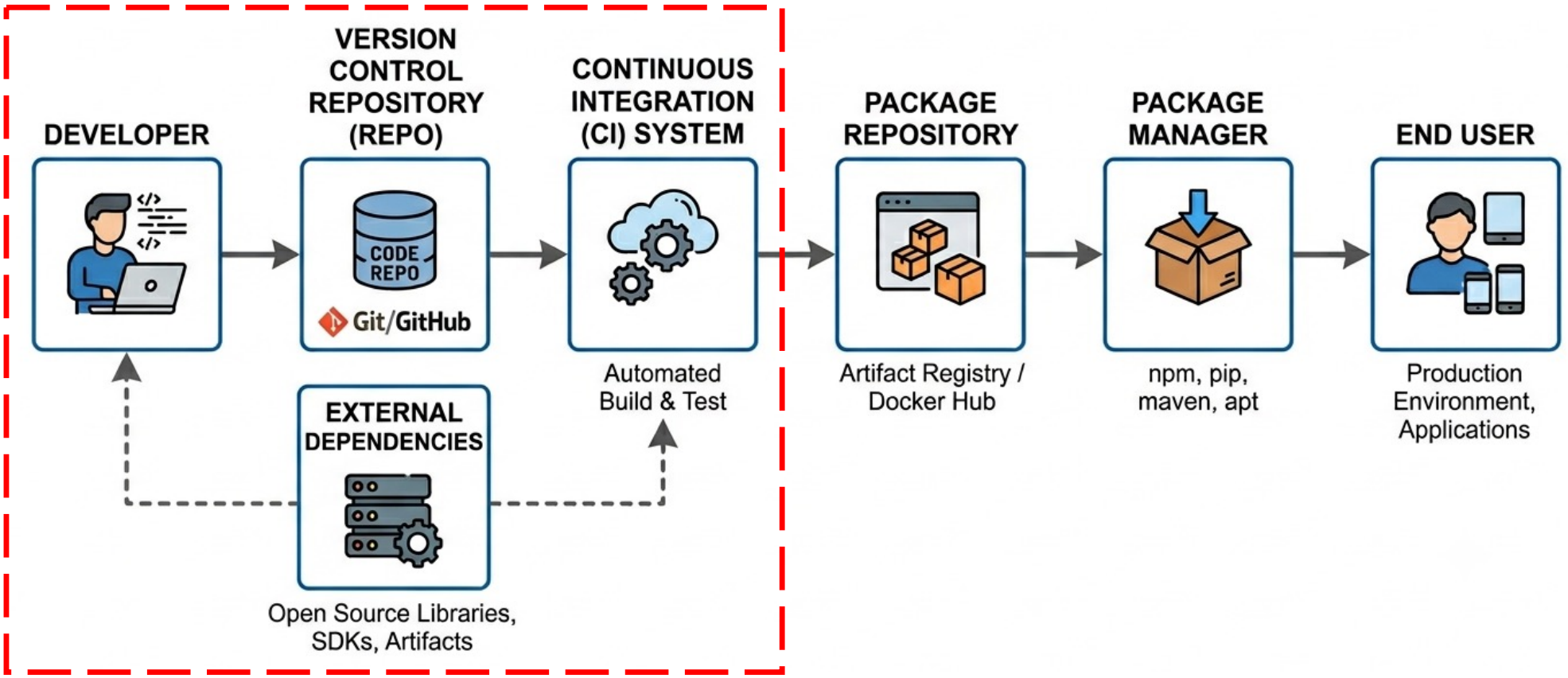
Part 1

Context

Supply Chain??



Supply Chain??



In practice

```
pip install …

code --install-extension …
pi install …

apt install …

git push
```

Cyber Threats

Axios NPM Package Compromised: Supply Chain Attack Hits JavaScript HTTP Client with 100M+ Weekly Downloads

A supply chain attack hit Axios when attackers used stolen npm credentials to publish malicious versions containing a phantom dependency. This triggered a cross-platform RAT during installation and replaced its files with clean decoys, making detection challenging.

By: Peter Girnus, Jacob Santos
Mar 31, 2026
Read time: 11 min (2913 words)

Red Hat Cloud Services npm Packages Hijacked

June 01, 2026 By Sonatype Security Research Team

5 minute read time

Miasma Supply Chain Worm Burrows Into 73 Microsoft Repositories

The attacks stemmed from a GitHub account that was also compromised in a previous Miasma attack on Microsoft last month.

Rob Wright, Senior News Director, Dark Reading

June 9, 2026

Bitwarden CLI Impersonation Attack Steals Cloud Credentials and Spreads Across npm Supply Chains

By Cameron Hyde

Apr 24, 2026 5 minutes

PyTorch dependency ‘torchtriton’ on PyPI Supply Chain Attack

Updated: May 14, 2024
by Mansi B.

Shai-Hulud "Hades" Wave Hits Six PyPI Bioinformatics Packages via Stolen Tokens

Written by

Kiran Raj

Published on
June 8, 2026

Updated on
June 8, 2026

Topics
Malware Security

Compromised litellm PyPI Package Delivers Multi-Stage Credential Stealer

March 24, 2026 By Sonatype Security Research Team

6 minute read time

Part 2

Case Study

LiteLLM

“AI gateway”

Downloads last month: 574M(!)
Downloads during exposure window: 47k

2,337	88%	1,671	283	46 min
Depend on litellm	Exposed	Unpinned	Protected	Window

Used by langchain, browser-use, graphrag, adk-google etc

Vector 1: Function injection

Credential stealing payload inserted into `proxy_server.py`

Trigger:

```
pip install litellm
litellm --model gpt-4o
```

Vector 2: Interpreter injection

Engineered `.whl` such that `pip install` placed `litellm_init.pth` file directly under `site_packages`:

```
import os, subprocess, sys;
subprocess.Popen([sys.executable, "-c", "import
base64; exec(base64.b64decode('...'))"])
```

Trigger:

```
pip install litellm
```

Vector 2: Interpreter injection

https://docs.python.org/3/library/site.html

*An executable line in a **path configuration file** is run at every Python startup, regardless of whether a particular module is actually going to be used. Its impact should thus be kept to a minimum.*

Lines starting with import (followed by space or tab) are executed. Limiting a code chunk to a single line is a deliberate measure to discourage putting anything more complex here.

Step 1: Harvesting

System info	Environment variables	SSH keys
Git credentials	AWS credentials	Kubernetes secrets
GCP credentials	Azure credentials	Docker configs
Package manager configs	Shell history	Crypto wallets
SSL/TLS private keys	Database credentials	CI/CD secrets

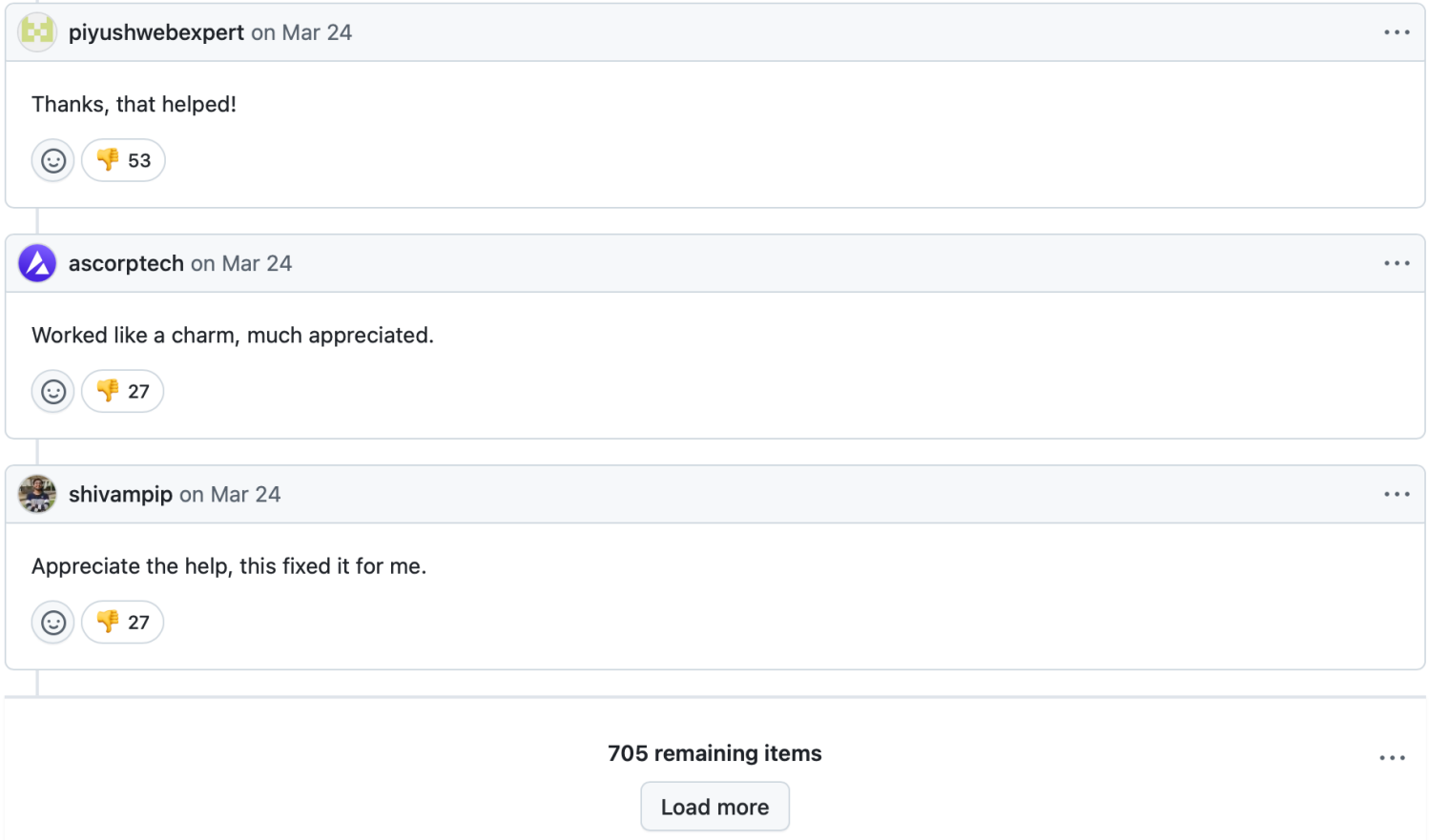
Step 2: Exfiltration

Encrypted secrets POSTed to https://models.litellm.cloud

Step 3: Persistence

- Installs scheduled service that executes code retrieved from https://checkmarx.zone/raw, bypassing DNS blocklists
  - systemd on Linux
  - Variants also install equivalent on macOS via launchd
- This backdoor is installed locally and in any accessible Kubernetes clusters (one pod per node)

Step 4: Suppression



Step 5: Magnification

Use stolen tokens to publish compromised packages

Recovery (in this case)

- Remove virtual environments, caches
- Remove scheduled service
- Rotate all secrets
- Remove Kubernetes pods
- Invoke incident response process, if appropriate

Upstream compromise

Actually *another* supply chain attack, this time on Trivy(!)

Exfiltrated PyPI publishing token (`pull_request_target`)

- litellm was not using trusted publishing (cf. bitwarden-cli)

But as we've seen there are many other vectors:

- Direct token theft (PyPI, GitHub)
- Phishing
- Account takeover
- Infiltration

Part 3

Advice

It could be you

```
pip install litellm

pip install reqeusts

pip install pandas-sdk

pip install torchtriton
```

Consequences

Threats to research integrity/reputation:

- Compute theft
- Code theft
- Ransomware
- Downtime
- Altered outputs
- Model tainting (data poisoning)

Recommendations (1)

Configure dependency cooldowns

`pyproject.toml` (project) or `uv.toml` (global: uvx, agents etc):

```
[tool.uv]
exclude-newer = "1 week”
```

Dependency Cooldowns (pip, npm, cargo, VS Code, dependabot…)

Recommendations (2)

Pin dependencies (packages, containers, actions…)

```
pypackage.toml:   [[package]]
                  name = "django"
                  wheels = [{hash = "sha256:251…", }]

    Dockerfile:   FROM python:3.14.6-slim@sha256:b87…

      gha.yaml:   - uses: actions/checkout@de0… # v6.0.2
```

Update via uv, Dependabot, pinact etc

Recommendations (3)

- Audit packages prior to installation and in CI
  - Vulnerability and malware checks (Astral)
  - Securing your end-to-end supply chain (GitHub)
  - First steps with Trivy
- Limit/vendor/inline deps (batteries included)
- Use reputable package managers/repositories

Recommendations (4)

- Environment isolation
  - Locally: dev containers, sandboxing
  - CI: permissions (zizmor etc)
- SSH passphrases or hardware security devices
- Encrypted secrets, narrowly scoped/expiring tokens

Recommendations (publishers)

Everything applying to consumers plus:

- Maximise the security of your GitHub account (including PATs)
- Don’t use `pull_request_target` unless you know what you’re doing
- Use Trusted Publishing (and therefore digital attestations)
- Perform contributor/successor due diligence (infiltration)
- Publish SBOMs
- Create immutable releases
- Protect branches
- Use MFA for everything

Takeaways

- Security only as strong as weakest link in supply chain
- Reusable code is an asset and a liability
- We've become accustomed to running untrusted code
- Explosion in exploitation of vulnerable processes
- Vendor response (cooldown etc) has been very reactive
- Many tools are (still) not secure by default
- But the risks can be managed/mitigated

Acknowledgements

Thanks to Glyn Chudleigh and everyone in e-Research at KCL and PharosAI

Your AI Gateway Was a Backdoor: Inside the LiteLLM Supply Chain Compromise (Trend Micro)

We May Be Living Through the Most Consequential Hundred Days in Cyber History, and Almost Nobody Has Noticed (Patrick Quirk)

mark.woodbridge@kcl.ac.uk

Guy's Cancer Charity

PharosAI

KING'S College LONDON

Department for Science, Innovation & Technology