

# Evaluation **Facile** - Florian POMPIDOU

---

A la demande du client, il a été demandé d'employer diverses méthodes d'analyse sur :

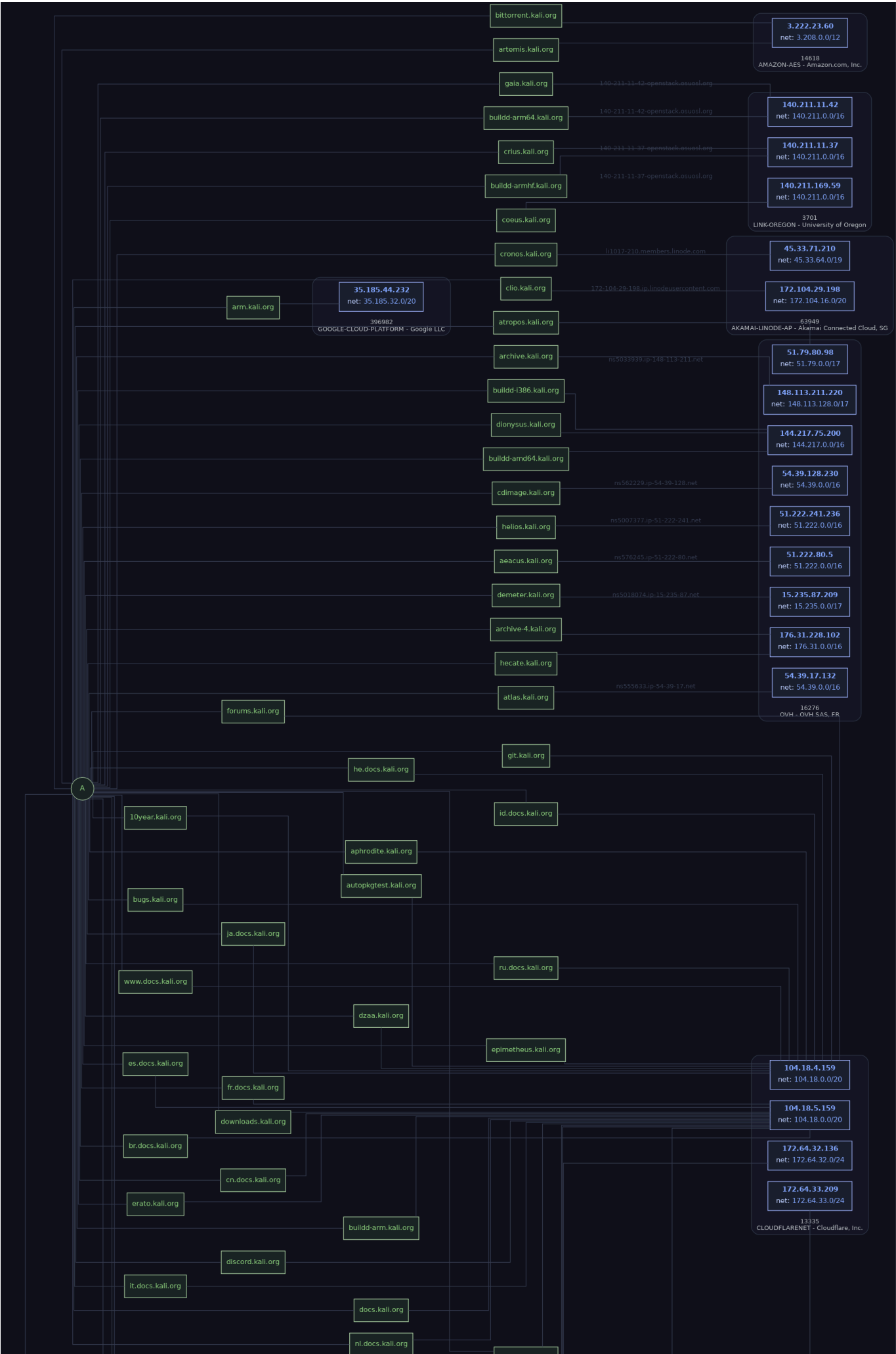
- Un domaine public **kali.org** par des moyens *passifs* (aucun scan actif n'est autorisé, faute d'accord avec la structure ciblée)
- Des actifs numériques sous forme de binaires

## 1. OSINT

### A. Reconnaissance

#### **a. Analyse des sous-domaines**

Pour l'analyse des sous-domaines, en l'absence d'accès à l'API de CRT.SH, j'ai utilisé l'outil **DNSDumpster**. Celui-ci affiche le contenu complet d'un WHOIS classique, mais mis en cache par plusieurs retours perpétuels sur le temps long, et donne un rendu sous forme de graphique pour montrer les interconnexions, en plus de proposer un export en .xlsx pour étude hors-ligne. Une conversion en .csv est jointe au rapport présent.





Le domaine **kali.org** fait ainsi état de 84 enregistrements DNS, répartis sommairement comme suit :

Top 5 Banners

Name	Count
cloudflare	60
nginx	26
SSH-2.0-OpenSSH_9.2p1 Debian-2+deb12u7	16

Top 5 ASN

ASN	ASN Name	Network Range	Count
13335	CLOUDFLARENET - Cloudflare, Inc.	172.64.32.0/24	30
16276	OVH - OVH SAS, FR	51.222.0.0/16	12
3701	LINK-OREGON - University of Oregon	140.211.0.0/16	5
15169	GOOGLE - Google LLC	192.178.155.0/24	5
14618	AMAZON-AES - Amazon.com, Inc.	3.208.0.0/12	2

Top 5 Countries

Name	Count
United States	15
Canada	10
France	2

b. Analyse de relations

A l'aide de l'outil **Maltego**, nous pouvons établir une suite de relations en partant du domaine racine.

On peut voir à l'origine de [kali.org](https://kali.org) une suite de sous-domaines actifs et contactables, ainsi que quelques adresses courriel menant à des individus depuis un domaine [@mailbox.org](mailto:@mailbox.org).

(Les crédits ont manqué à ce moment-là, si j'ai le temps de refaire un compte je poursuivrai, sinon j'ai la méthode.)

### c. Analyse d'infrastructure technique

Pour analyser la surface d'attaque probable de façon passive, nous pouvons utiliser [Censys](#) et filtrer :

1. [kali.org](#) => Avec le nom de domaine (résultats peu fiables, filtre aussi les serveurs qui utilisent la tech/le nom ou sont fans mais ne sont pas en liens)
2. [web.cert.names = kali.org](#) => Avec le dispensaire de certificats de noms de domaine (plus fiable, émane vraiment de lui et trouve les liens filiaux, mais pour peu que le serveur d'émission soit autre, il faut d'abord le trouver)
3. [\(kali.org\) and \(web.cert.names = kali.org\) and \(host.services.vulns.id: \\* or web.vulns.id: \\*\)](#) => Requête avancée, permettant de ne trier QUE les services avec des CVEs récoltées. Efficace pour qui veut prévenir les services de ces vulnérabilités, mais on peut se douter que les webmaîtres de chez Kali seront des cordonniers bien chaussés. De plus, cette requête demande un abonnement payant.

Prenant un résultat sur la deuxième requête :

Search Results | Report Builder

Filter Results

ASSET TYPES

- Hosts: 0
- Certificates: 0
- Web Properties: 44

SOFTWARE VENDORS

- cloudflare: 84
- vmware: 9
- apache: 3
- f5: 2
- php: 1

SOFTWARE PRODUCTS

- cloudflare\_load\_balancer: 42
- waf: 42
- horizon: 9
- http\_server: 3
- nginx: 2
- More

RESULTS: 44 • DURATION: 0.24s

Personalize Results

bugs.kali.org: 443 • WEB PROPERTY

AS OF 12 JUN 2026 | 08:59 UTC

WAF

HTML Title: My View - Kali Linux Bug Tracker

Browser Trust: Trusted

Ever Valid?: Yes

Endpoints (2): 443 / HTTP, 443 / HTTP /my\_vie...

Software (3): Cloudflare Load Balancer, Cloudflare Waf, Php

MATCHED FIELDS

web.cert.names: kali.org

www.kali.org: 443 • WEB PROPERTY

AS OF 12 JUN 2026 | 07:41 UTC

WAF

HTML Title: Kali Linux | Penetration Testing and Ethical Hacking Linux Distribution

Browser Trust: Trusted

Ever Valid?: Yes

Le premier lien [bugs.kali.org](https://bugs.kali.org) nous donne diverses informations :

- L'état du certificat TLS (empreinte, émission, émetteur, région, domaines couverts, validité, etc)
- Ports ouverts (ici, 443, HTTPS)

- Service actif (Cloudflare WAF, Web Application Firewall, donc un pare-feu actif servant de point d'entrée [à cheval entre pare-feu et répartiteur de charge])
- [Toutes les infos de la page d'entrée](#)

Pas de CVEs exposées, l'analyse peut s'arrêter là sur ce cas.

#### d. BONUS - Rapport technique centralisé

En outil complémentaire à [Censys](#) et [Maltego](#), j'ai fait le choix d'utiliser [BuiltWith](#), reprenant sous forme d'onglets les différents éléments précédemment vus, même si avec une moindre portée technique, mais également avec quelques éléments supplémentaires :

1. Les technologies de base utilisées sur le site, widgets, langue d'affichage, frameworks, CDN, librairies JavaScript, prestataire de courriels, hébergeurs, certificats SSL, fichiers robots.txt, etc.
2. [La synthèse des traqueurs et analyseurs présents sur le serveur](#), comme CloudFlare insights, Google Analytics jusqu'en 2020, ou d'autres traqueurs passifs/actifs. Le choix de non-renouvellement de certaines technologies (Google Analytics) et le choix des analyseurs actuels (CloudFlare uniquement) permet de renseigner sur le profil politique de l'organisation, qui boude les outils de surveillance publicitaire, information qui peut se révéler pertinente selon le contexte étudié.
3. [Les hyperliens de sortie](#) montrant une relation directe sur les sites [exploit-db.com](#), [backtrack-linux.org](#), [spamtool.net](#), [getspamtool.com](#) et [lynigma.com](#) permet d'établir un profil collaboratif autour du système Linux, avec l'écosystème de cybersécurité cadre "EXPLOITS", et autres partenariats (commerciaux ?) avec des outils tiers de protection.
4. [Les redirections présentes](#) permettant d'étendre un peu la "galaxie" Kali-Linux, avec des domaines comme [kali.me](#), [kali.us.org](#), [kali.ninja](#), [kali-linux.org](#) ou [nethunter.com](#). Redirections permanentes motivées possiblement par l'envie de ne pas voir de domaine pirate homonyme rediriger des gens honnêtes vers des outils vérolés (possiblement certains domaines sont même des domaines saisis) ou des domaines historiques sur lesquels découvrir de nouvelles étendues (API, serveurs de fichiers, etc)

#### Finalités

Catégorie	Éléments trouvés	Outil utilisé	Observations
Domaine	kali.org	DNSDumpster	RAS
Sous-domaines	10year.kali.org - aeacus.kali.org - aphrodite.kali.org - archive.kali.org - archive- 4.kali.org - arm.kali.org - artemis.kali.org - artifacts.kali.org - atlas.kali.org - atropos.kali.org - autopkgtest.kali.org - bittorrent.kali.org - bugs.kali.org - buildd- amd64.kali.org - buildd- arm.kali.org - buildd-	DNSDumpster Maltego	RAS

Catégorie	Éléments trouvés	Outil utilisé	Observations
	arm64.kali.org - buildd- armhf.kali.org - buildd- i386.kali.org - cdimage.kali.org - clio.kali.org - coeus.kali.org - crius.kali.org - cronos.kali.org - demeter.kali.org - dionysus.kali.org - discord.kali.org - docs.kali.org - ar.docs.kali.org - br.docs.kali.org - cn.docs.kali.org - de.docs.kali.org - en.docs.kali.org - es.docs.kali.org - fr.docs.kali.org - he.docs.kali.org - id.docs.kali.org - it.docs.kali.org - ja.docs.kali.org - nl.docs.kali.org - ru.docs.kali.org - www.docs.kali.org - downloads.kali.org - dzaa.kali.org - epimetheus.kali.org - erato.kali.org - forums.kali.org - gaia.kali.org - git.kali.org - hecate.kali.org - helios.kali.org		
Adresses IP	51.222.80.5 - 148.113.211.220 - 176.31.228.102 - 35.185.44.232 - 3.222.23.60 - 104.18.5.159 - 54.39.17.132 - 51.79.80.98 - 144.217.75.200 - 140.211.11.42 - 140.211.11.37 - 54.39.128.230 - 172.104.29.198 - 140.211.169.59 - 45.33.71.210 - 15.235.87.209 - 104.18.4.159 - 51.222.241.236 -	DNSDumpster Meltago	RAS
Services exposés	HTTPS	Censys.io	Il s'agit de Kali géré par OffSec. Je ne pense pas être en mesure de trouver plus que ce qu'ils veulent que je vois.
Courriel de	magamabazarov@mailbox.org mahamabazarov@mailbox.org	Maltego	Gérant d'un paquet intégré à la suite : <a href="https://github.com/caster0x00/Nihilist/releases">https://github.com/caster0x00/Nihilist/releases</a>

Catégorie	Éléments trouvés	Outil utilisé	Observations
contact			<a href="https://gitlab.com/kalilinux/packages/sara">https://gitlab.com/kalilinux/packages/sara</a>
Fichiers exposés	X	X	(Crédits insuffisants)

## B. Trouvailles

(On parle de *kali.org*, géré par OffSec. C'est la suite de hacking la plus connue du monde. Je ne pense PAS avoir de quoi trouver la moindre chose à mon niveau, et si une faille s'ouvrait, leur BugBounty s'illuminerait comme un sapin de Noël un soir d'assaut lors d'une guerre de tranchées. J'ai donc trouvé des failles historiques mais patchées depuis pour répondre à la question.)

### MantisBT sur bugs.kali.org

MantisBT est un service de gestion des tickets.

- **Description** : Une faille dans l'authentification liée à du *Type Juggling* en PHP (usage de comparaisons `==` au lieu de `===`) permet à un attaquant de se connecter au compte admin à partir de son hash MD5
- **Source/Preuve** : [CVE-2025-47776](#)
- **Criticité** : 91/100 (Critique)
- **Impact Potentiel** : Connexion au compte admin MantisBT sans mot de passe.

- 
- **Description** : Faille de déni de service liée à une taille illimitée des notes attachées dans les tickets (+ de 4 millions de caractères) capables de corrompre l'affichage de l'histoire du ticket, et le rend inopérant
  - **Source/Preuve** : [CVE-2025-46556](#)
  - **Criticité** : 75/100 (Élevé)
  - **Impact Potentiel** : Rend des tickets inopérants, mais pas d'impact global.

### Discourse sur forums.kali.org

Discourse est un service de forum/plateforme de discussion basé sur Ruby.

- **Description** : Faille de divulgation des fichiers de sauvegarde liée à une mauvaise interaction entre la directive interne de NGinx et une fonction de Ruby. En devinant le nom du fichier de sauvegarde (qui suit une structure de base), on peut tromper le serveur et télécharger le dump complet de la base de données du forum.
- **Source/Preuve** : [CVE-2024-53991](#)
- **Criticité** : 75/100 (Élevé)
- **Impact Potentiel** : Intégralité de la base de données (incluant forums secrets/admins et messages privés [si disponibles], hash de mots de passe, adresses courriels, adresses IP, etc) fuitée.

### Paquets sur pkg.kali.org

Porte dérobée **xz-utils** découverte par l'ingénieur PHP sur base d'une requête ayant pris 500ms de trop pour lui.

- **Description** : La porte dérobée de **xz-utils** a frappé la branche **Kali Rolling**, gestionnaire des paquets de la distribution. Autrement dit, tout paquet téléchargé depuis cette branche était compromis,

et chaque système était vérolé.

- **Source/Preuve** : CVE-2024-3094
- **Criticité** : 100/100 (Maximal)
- **Impact Potentiel** : Tous les systèmes ayant téléchargé un paquet de la branche **Rolling** sont considérés comme pénétrables par une porte dérobée. Impact global et majeur.

Cette affaire a été à l'origine de plusieurs prises de conscience :

- les paquets librement maintenus peuvent être repris par des acteurs malveillants faute d'incitation à une maintenance soutenue de leur créateur
- des dépendances de dépendances de dépendances sont une faille de sécurité obfusquée à haute dangerosité
- tout le monde peut passer à côté, même les plus experts

Les équipes de Kali ont réagi en révoquant les signatures des paquets et en purgeant les dépôts.

C. Recommendations

##	Recommandation	Trouvaille associée	Priorité
01	Mise à jour de version	MantisBT	Haute
02	Sauvegarde externe avec nom de fichier aléatoire	Discourse	Moyenne
03	Purge des dépôts	xz-utils	Très haute
04	Signatures diversifiées pour chaque paquet	xz-utils	Moyenne

2. Ingénierie Inversée

Le client nous demande d'étudier le fichier **crackme\_3**, avec décompilation et ingénierie inversée.

Profil du fichier

Tout d'abord, les métadonnées classiques sont extraites :

```
$ file crackme_3
crackme_3: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked,
interpreter /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=45939497a0b02f28b8cc23629acfbcb2e39ca2072, for GNU/Linux 3.2.0, not
stripped

$ strings crackme_3 | head -16
r/lib64/ld-linux-x86-64.so.2
strcpy
puts
__libc_start_main
stderr
fprintf
strcmp
libc.so.6
GLIBC_2.2.5
GLIBC_2.34
```



```

__gmon_start__
PTE1
H=0@@@
Usage: %s <password>
[-] Acces refuse.
[+] Acces accorde ! Flag : CTF{ESDI_R3v3rs3_2026}

```

J'utilise également un duo de commande avec **objdump** pour obtenir quelques informations, comme les fonctions en cas de binaire non empaqueté :

```

$ objdump -f crackme_3

crackme_3:      file format elf64-x86-64
architecture: i386:x86-64, flags 0x00000112:
EXEC_P, HAS_SYMS, D_PAGED
start address 0x0000000000401080

$ objdump -d crackme_3 | grep ">:"
0000000000401000 <_init>:
0000000000401020 <strcpy@plt-0x10>:
0000000000401030 <strcpy@plt>:
0000000000401040 <puts@plt>:
0000000000401050 <strcmp@plt>:
0000000000401060 <fprintf@plt>:
0000000000401080 <_start>:
00000000004010b0 <_dl_relocate_static_pie>:
00000000004010c0 <deregister_tm_clones>:
00000000004010f0 <register_tm_clones>:
0000000000401130 <__do_global_ctors_aux>:
0000000000401160 <frame_dummy>:
0000000000401180 <main>:
000000000040123c <_fini>:

```

Et une dernière passe avec **checksec** :

```

$ checksec --file=crackme_3
RELRO           STACK CANARY      NX            PIE            RPATH          RUNPATH
Symbols         FORTIFY Fortified   FortifiableFILE
Partial RELRO   No canary found   NX enabled    No PIE         No RPATH       No
RUNPATH        39 Symbols       No            0              2              crackme_3

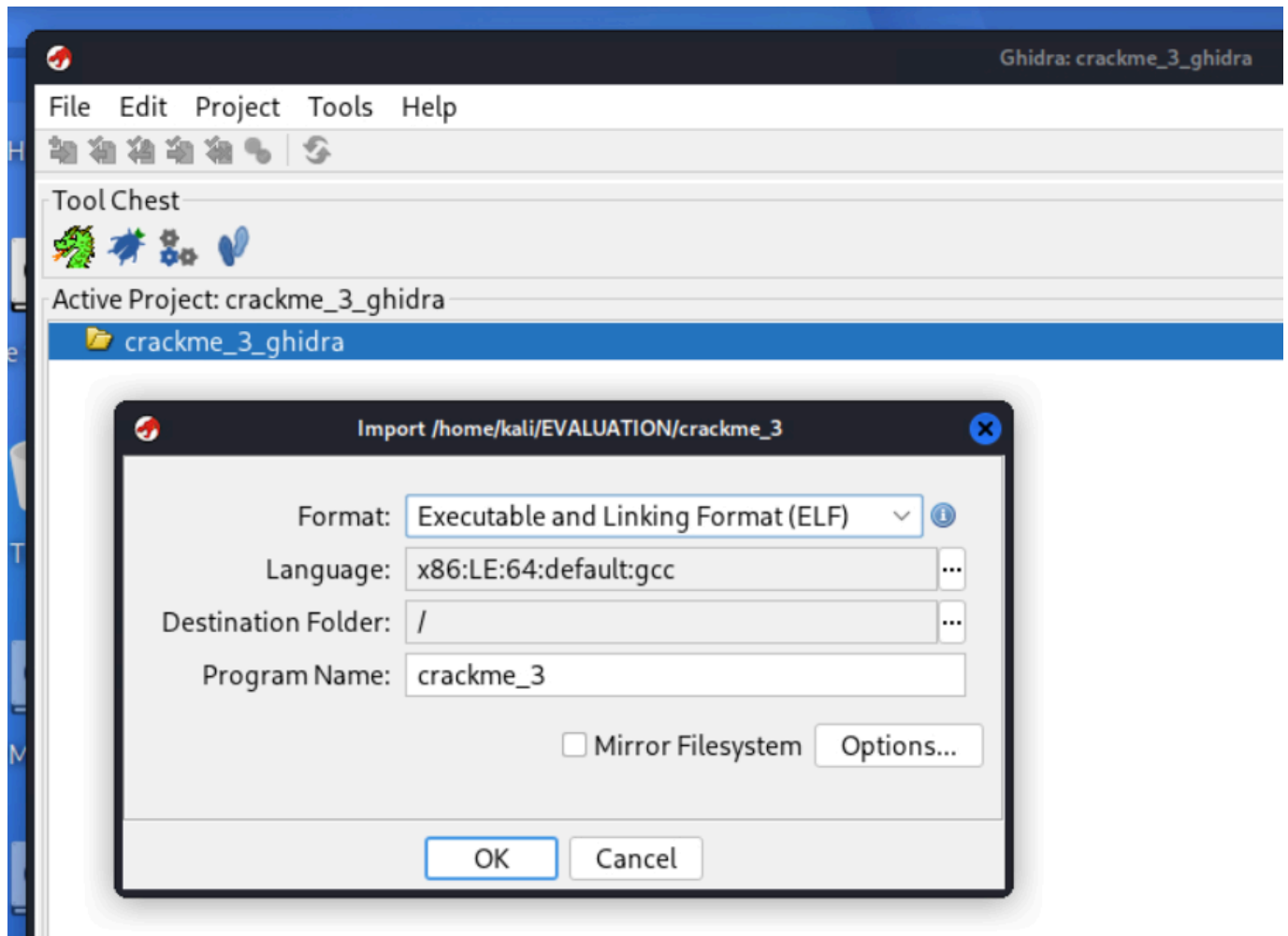
```

Ce que cela nous apprend :

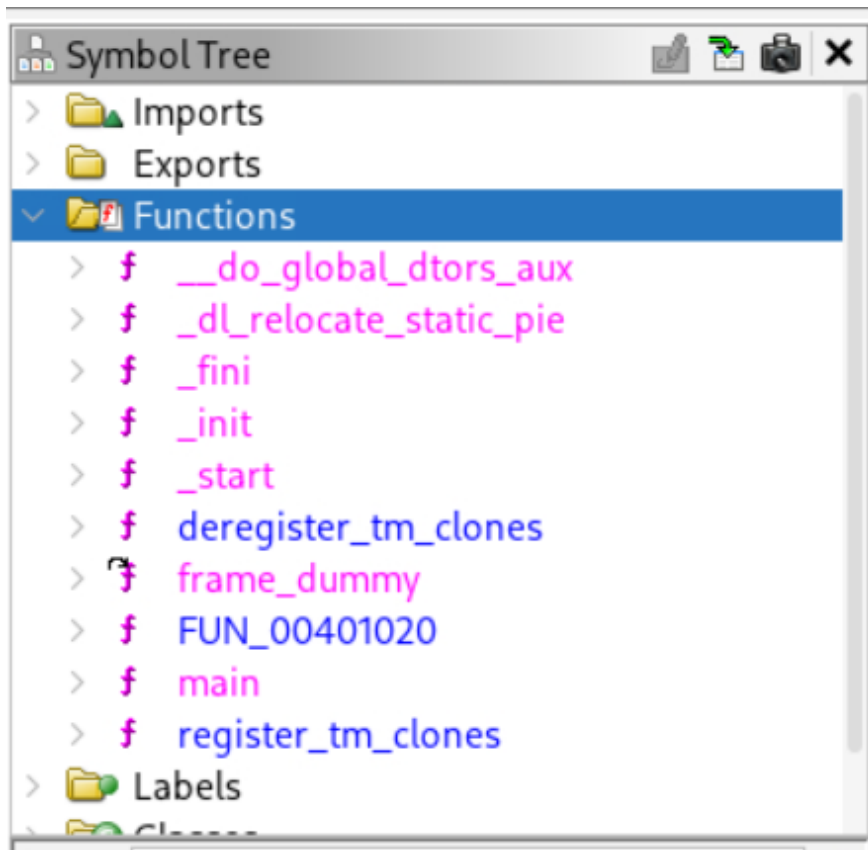
- Exécutable ELF 64bit, architecture moderne
- Non épuré, donc avec le noms des fonctions, variables et symboles d'origine (plus facile à décompiler)
- 

## A. GHIDRA

Je crée un nouveau projet dans Ghidra et j'importe le fichier selon les infos recueillies plus tôt (détectées nativement) :



J'analyse le programme, j'utilise l'outil "Code Analyzer" et j'en extraies les données essentielles, notamment les fonctions :



Une rapide cartographie nous permet d'établir ceci :

Nom de la fonction	Adresse	Rôle
deregister_tm_clones	0x4010c0	(fonction d'usine)
register_tm_clones	0x4010f0	(fonction d'usine)
frame_dummy	0x401160	(fonction d'usine)
main	0x401180	Fonction principale, coeur du code
FUN_00401020 (non réel inconnu)	0x401202 (d'où le nom)	Fonction secondaire, nécessaire à main(). Une étude plus approfondie apprend qu'il s'agit d'une fonction PLT (Procedure Linkage Table), dites "chef de gare"

### main()

Etendue de 0x401180 à 0x401236.

- Ligne 0x401188 : JZ (JUMP ZERO) => Saute si le résultat d'un précédent CMP (COMPARE) ou TEST est égal à 0. Ici donc, il évalue un argument == 2 pour poursuivre le programme, sinon il saute sur un stderr, met une valeur EBX (code d'erreur) à 1 et RET (quitte le programme)
- Ligne 0x4011b2 : Si l'argument est présent, on récupère l'argument et on le stocke dans la RSI. On met le RAX à 0 pour servir de compteur à une boucle (i = 0) avant de charger une variable `enc.0` chiffrée.
- Ligne 0x4011e0 : Prend le RAX (i = n) comme index de la chaîne chiffrée `enc.0`, applique un XOR avec la clef 0x42 pour déchiffrer, stocke le caractère en clair sur la pile RSP, incrémente le RAX (i++) et recommence si le RAX n'a pas atteint 0xf (i = 15) => La boucle prend une chaîne de 15 octets cachée dans le binaire, applique un XOR sur chaque octet avec 0x42, et reconstruit le vrai mot de passe attendu dans la variable `local_38`

- Ligne 0x401201 : CALL (APPEL) la `strcpy`, autrement dit notre argument du programme et la stocke de la RSI vers la RSP
- Ligne 0x40120c : CALL la `strcmp`, donc une comparaison de *string* entre `local_38` et le mot de passe interne
- Ligne 0x401213 : Les deux chaînes sont identiques ? Renvoi de 0, le JZ est valide et affiche la victoire. Les deux chaînes diffèrent ? Pas de JUMP ZERO, on affiche l'accès refusé avec PUTS et on saute à la fin.

### Vulnérabilité principale, MEMORY CORRUPTION

- La pile offrant une comparaison entre la chaîne passée en argument et la chaîne déchiffrée du mot de passe se joue avec un `strcpy`, avec une étape intermédiaire de comparaison renvoyant une valeur (0 ou non) pour "valider" que la chaîne correspond ou non, avant de sauter vers le résultat final.
- Cependant, avec un `break` au bon endroit, on peut définir un résultat et sauter à l'endroit souhaité en modifiant ce qu'on fait écrire à la fonction quand le résultat est reçu par les pointeurs.
- En bref : nous entrant une donnée externe non validée pour sauter les "sécurités" en place en modifiant le RIP (Register Instruction Pointer)

### Vulnérabilité secondaire, BUFFER OVERFLOW

Le `checksec` a mis en évidence un manque de STACK CANARY. De fait, le code est vulnérable aux coups de grisou :

- La méthode `strcpy` est un code historique qui copie une chaîne de caractères jusqu'à y trouver un octet nul, sans vérifier la possibilité que l'espace soit un temps soit peu suffisamment large pour accueillir la chaîne en argument.
- Prenant la fonction `main()`, `SUB RSP, 0x30` signale un espace alloué de 48 octets sur la pile pour les variables locales
- Toujours dans `main()`, `local_28` est le buffer qui commence à l'offset déclaré `[RSP + 0x10]`, donc 16 octets plus bas que la pile RSP.
- Avec une taille allouée de 48 octets et un début de buffer 16 octets avant la fin de la pile, on a 32 octets pour stocker le mot de passe dans `local_28` avant de déborder de l'espace alloué et renvoyer un SIGSEV.

## B. GDB

### Trouver le mot de passe caché

```
$ chmod +x crackme_3

$ gdb -q ./crackme_3
Reading symbols from ./crackme_3...
(no debugging symbols found in ./crackme_3)
(gdb) info functions
All defined functions:

Non-debugging symbols:
0x000000000401000  _init
0x000000000401030  strcpy@plt
0x000000000401040  puts@plt
0x000000000401050  strcmp@plt
```

```

0x0000000000401060  fprintf@plt
0x0000000000401080  _start
0x00000000004010b0  _dl_relocate_static_pie
0x00000000004010c0  deregister_tm_clones
0x00000000004010f0  register_tm_clones
0x0000000000401130  __do_global_ctors_aux
0x0000000000401160  frame_dummy
0x0000000000401180  main
0x000000000040123c  _fini
(gdb) run Albathar
Starting program: /home/kali/EVALUATION/crackme_3 Albathar
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib/x86_64-linux-gnu/libthread_db.so.1".
[-] Acces refuse.
[Inferior 1 (process 38985) exited with code 01]
(gdb) break *0x00401215
Breakpoint 1 at 0x401215
(gdb) run Albathar
Starting program: /home/kali/EVALUATION/crackme_3 Albathar
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, 0x0000000000401215 in main ()
(gdb) set $rip = 0x0040122a
(gdb) continue
Continuing.
[+] Acces accorde ! Flag : CTF{ESDI_R3v3rs3_2026}
[Inferior 1 (process 39225) exited with code 0374]

```

Pour expliquer mon raisonnement :

```

0x0000000000401215 <+149>:  je      0x40122a <main+170>

```

1. Nous mettons un break sur l'adresse du dessus où un JUMP EQUAL avec s'effectuer en cas de mauvais mot de passe
2. Nous venons d'exécuter le `strcmp` qui compare mon argument et la chaîne attendue. Le résultat est négatif, évidemment. Le JUMP ZERO va marcher et sauter à "[-] Accès refusé".
3. Le BREAK arrête la lecture du programme, et on déplace la tête de lecture du processeur (l'endroit où il pointe, comme une tête de lecteur sur un sillon de vinyle)
4. En changeant la valeur du RIP, on a "attrapé" la tête de lecture du processeur, et on le fait pointer non plus vers le JUMP ZERO mais sur l'adresse de fin. On déplace de force la tête de lecture du vinyle pour passer au morceau final.
5. Le message de victoire s'affiche.

En bref, on saute le JUMP ZERO, on contourne toute la suite logique du programme pour atterrir à la fin, située ici :

```
0x000000000040122a <+170>: lea    0xdff(%rip),%rdi    # 0x402030
```

Ainsi le mot de passe est récupéré sans avoir besoin de relever de flag ni de faire de force brute :

ESDI\_R3v3rs3\_2026

```
(kali@GAWIN)-[~/EVALUATION]
$ chmod +x crackme_3

(kali@GAWIN)-[~/EVALUATION]
$ gdb -q ./crackme_3
Reading symbols from ./crackme_3...
(No debugging symbols found in ./crackme_3)
(gdb) info functions
All defined functions:

Non-debugging symbols:
0x0000000000401000 _init
0x0000000000401030 strcpy@plt
0x0000000000401040 puts@plt
0x0000000000401050 strcmp@plt
0x0000000000401060 fprintf@plt
0x0000000000401080 _start
0x00000000004010b0 _dl_relocate_static_pie
0x00000000004010c0 deregister_tm_clones
0x00000000004010f0 register_tm_clones
0x0000000000401130 __do_global_dtors_aux
0x0000000000401160 frame_dummy
0x0000000000401180 main
0x000000000040123c _fini
(gdb) run Albathar
Starting program: /home/kali/EVALUATION/crackme_3 Albathar
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib/x86_64-linux-gnu/libthread_db.so.1".
[-] Acces refuse.
[Inferior 1 (process 38985) exited with code 01]
(gdb) break *0x00401215
Breakpoint 1 at 0x401215
(gdb) run Albathar
Starting program: /home/kali/EVALUATION/crackme_3 Albathar
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, 0x0000000000401215 in main ()
(gdb) set $rip = 0x0040122a
(gdb) continue
Continuing.
[+] Acces accorde ! Flag : CTF{ESDI_R3v3rs3_2026}
[Inferior 1 (process 39225) exited with code 0374]
(gdb) q
```

### 3. Synthèse

#### A. Rapport de synthèse

A la demande du client, nous avons mené deux missions conjointes sur :

1. Le domaine public kali.org => Les rapports préliminaires permettent d'établir que l'environnement web de Kali est de Kalité. Aucun service n'est exposé sauf justification, les CVEs sont patchées rapidement, les

équipes sont fiables et répondent vite aux problèmes exposés dans leur programme de BugBounty.

2. L'étude d'un binaire suspect => Le binaire est un classique des programmes CTF, avec deux vulnérabilités détectées dans l'usage de méthodes historiques non-sécurisées et d'une possibilité de corruption de mémoire pour obtenir une validation finale.

## B. Complémentarité OSINT / RE

Dans une vraie mission de pentest, comment les informations OSINT collectées en Partie 1 pourraient-elles guider l'analyse RE de la Partie 2 ? Citez un exemple concret où l'OSINT aurait permis de cibler directement une vulnérabilité RE.

Nous abordons deux méthodologies et finalités qui diffèrent, et sont donc potentiellement complémentaires :

- L'OSINT fournit un **contexte**
- L'ingénierie inversée fournit un **mécanisme**
- Un outil comme **Censys** détecte des services, des systèmes d'exploitation, des compilateurs éventuellement, des versions. **Un ingénieur décompilateur peut définir un environnement de décompilation taillé sur mesure.**
- **Censys** toujours renseigne sur les produits utilisés (VPN, WAF, firmware IoT, etc), aussi peut-on **rechercher les binaires d'origine des firmwares pour les étudier à froid et en trouver des failles.**
- Utiliser **Maltego** ou du **dorking** permet de profiler un codeur précis sur une infrastructure précise, et avec, en tirer **des conclusions sur les langages utilisés, les empaqueteurs ou les librairies, ou même carrément les erreurs du passé.**

Imaginons un cas concret :

1. Un service sur le port 28576 est exposé sur un serveur trouvé par **Censys**, je vais sur le port et je trouve une page de connexion d'un vieux routeur **TP-Link** ou **Cisco** ou **Netgear**
2. Je fouille sur des forums ou sur la page "téléchargements" du site constructeur le binaire qui correspond à la version affichée sur le portail, mettons 3.6.1 (alors que la dernière version est 4.5.7)
3. Je décompile le binaire avec GHIDRA, je cherche et trouve une méthode **strcpy**

Je sais maintenant comment faire planter le service. Pire encore : je peux déplacer un RIP et aller sur le panneau admin.

## C. Respect du cadre légal

1. Quelle loi encadre l'accès frauduleux à un système informatique en France ? Citez l'article.
2. Dans votre reconnaissance OSINT de la Partie 1, avez-vous respecté les limites légales ? Justifiez.
3. Un binaire trouvé sur un serveur compromis peut-il être analysé légalement ? Dans quel contexte ?

1. **Loi Godfrain** (*le lien LEGIFRANCE ne pointe sur rien*)
2. Oui. Je n'ai effectué aucun test moi-même, depuis ma machine, sur la "cible". Je n'ai fait que relever des rapports établis par des outils de cartographie, sans lire de rapports actifs/agressifs non plus (pas de NMAP rapporté ni équivalent).
3. Ca dépend.
  - Ai-je le droit d'être sur ce serveur ?

- Non : conséquemment, non.
- Oui :
  - Le binaire a t-il une signature numérique d'un éditeur (re)connu ?
    - Non : Il s'agit certainement d'un outil d'attaque et on peut l'analyser librement. Les circonstances joueront en notre faveur en cas de désaccord légal.
    - Oui : On peut l'analyser de façon encadrée par le Code de la Propriété Intellectuelle. Par défaut, décompiler un logiciel propriétaire est interdit, **SAUF** si l'éditeur ne met plus à jour son binaire (exception de **sécurité**). > [Resource utile](#) <