

RAPTOR DEFENSE V2.0 // HANDBOOK

RAPTOR DEFENSE V2.0 // MANUEL DE RÉFÉRENCE ADMINISTRATEUR

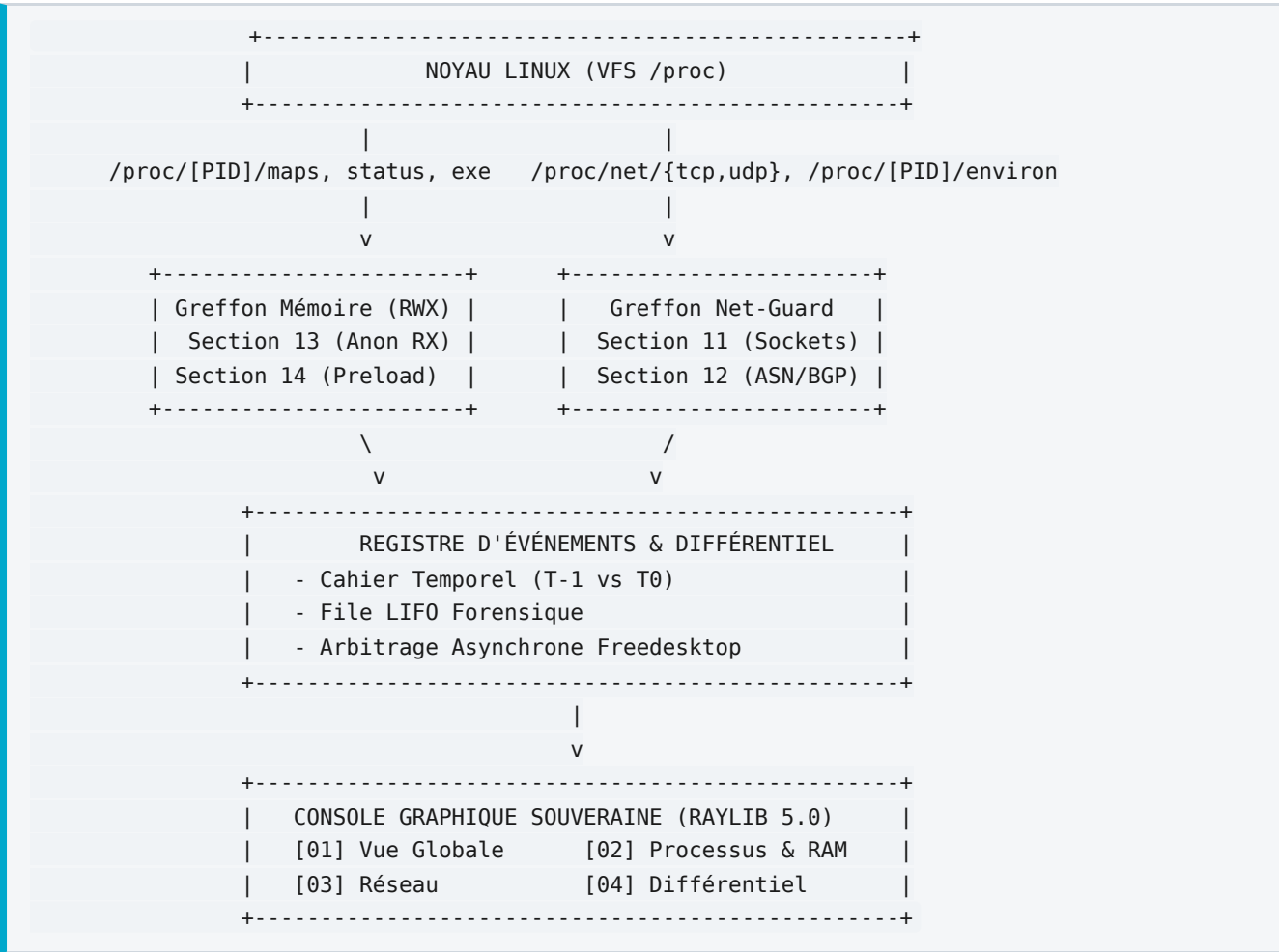
**Guide d'Exploitation Technique, Forensique Mémoire & Réseau
(Volumes 1, 2 & 3)**

VOLUME 1 : ARCHITECTURE FONDAMENTALE & INSPECTION MÉMOIRE (ONGLETS 1 & 2)

SECTION 1 : VUE D'ENSEMBLE ARCHITECTURALE & MODÈLE OPÉRATIONNEL

RAPTOR v2.0 est un agent défensif d'investigation en direct (*Host-based Intrusion Detection & Live Forensics*) conçu pour auditer les couches mémoires et réseau sous Linux sans dépendre d'agents distants, d'interpréteurs lourds ou de modules noyau propriétaires.

Le moteur repose sur une inspection passive directe des pseudo-systèmes de fichiers du noyau Linux (`/proc` et `/sys`). Chaque cycle d'audit extrait, normalise et corrèle l'état des descripteurs de processus, des tables d'allocation de pages virtuelles (`maps`) et des structures de communication réseau (`net/tcp` , `net/udp`).



SECTION 2 : ONGLET [01] VUE GLOBALE (DASHBOARD OPÉRATIONNEL)

L'onglet [01] **Vue Globale** constitue l'interface de contrôle tactique immédiate. Il condense en un panneau unique l'état de santé du poste, le volume de ressources matérielles mobilisées et les indicateurs de compromission active.

```
+-----+
+-----+
| RAPTOR DEFENSE // AUDIT MÉMOIRE & PROCESSUS   [ SURVEILLANCE ACTIVE ]           [ ? Guide
Analytique ] |
| [ 01 ] Vue Globale | [ 02 ] Processus & RAM | [ 03 ] Sockets & Réseau | [ 04 ] Diff | [ 05
] IA      |
+-----+
+-----+
| [ ALERTE DÉFENSIVE // ANOMALIES DÉTECTÉES ]
|
| Des processus suspects non officiels ou avec mémoire injectée sont actifs sur ce poste.
|
+-----+
+-----+
| CHARGE CPU          | MÉMOIRE VIVE (RAM)   | MENACES / RWX          | FLUX RÉSEAU ACTIFS
|
| 15.2 %              | 5.46 / 7.48 Go          | 1 ANOMALIE             | 3 SOCKETS
|
| [===                ] | [=====                ] | [!!!!!!!!!!!!!!!!!!!!!!] | [==
]|
+-----+
+-----+
| TOP 5 DES CONSOMMATEURS DE MÉMOIRE :
|
| PID   | Nom de l'exécutable | RAM (RSS) | Statut Dérive | Emplacement Binaire
|
| 6649  | chromium            | 2529 Mo  | Stable       | /usr/lib/chromium/chromium
|
| 18145 | chromium            | 399 Mo   | Stable       | /usr/lib/chromium/chromium
|
| 6328  | chromium            | 280 Mo   | Stable       | /usr/lib/chromium/chromium
|
| ...   | ...                  | ...      | ...          | ...
|
+-----+
+-----+
| [ LANCER UN NOUVEL AUDIT IMMÉDIAT ]   Horodatage du contrôle : 2026-10-23 18:04:12
|
+-----+
+-----+
```

2.1. Métriques Clés & Instrumentation

Charge Processeur (CPU)

- **Origine des données** : Parsing direct de `/proc/stat` sur la ligne d'agrégation `cpu`.
- **Algorithme de calcul** : Mesure différentielle sur deux échantillons temporels t_1 et t_2 espacés d'un intervalle de garde Δt :

Total = user + nice + system + idle + iowait + irq + softirq

$$\text{Charge \%} = \frac{\Delta \text{Total} - \Delta \text{idle}}{\Delta \text{Total}} \times 100$$

- **Comportement administrateur** : Un pic ponctuel (> 85%) est toléré lors de compilations ou d'audits complets. Une charge CPU soutenue sans processus visible dans le Top 5 signale un déni de service local ou une dissimulation par hooking d'appels système.

Mémoire Vive (RAM Physique)

- **Origine des données** : `/proc/meminfo` via les compteurs `MemTotal` et `MemAvailable`.
- **Formule** :

RAM Utilisée = MemTotal - MemAvailable

$$\text{Pourcentage} = \frac{\text{RAM Utilisée}}{\text{MemTotal}} \times 100$$

- **Interprétation** : RAPTOR n'utilise pas `MemFree`, car le noyau Linux alloue massivement la mémoire inactive au cache de pages (`buffers/cache`). `MemAvailable` reflète la mémoire réellement disponible pour les processus applicatifs sans forcer d'éviction sur la partition d'échange (*swap*).

Menaces / RWX

- **Indicateur d'anomalie** : Compteur consolidé `threats_count`.
- **Seuils d'alerte** :
 - `0 ANOMALIE` : Couleur d'accent vert (`COL_GREEN_OK`), système nominal.
 - `> 0 ANOMALIE` : Couleur d'accent rouge critique (`COL_CRIT_RED`), déclenchement de la bannière supérieure et notification système Freedesktop.

Flux Réseau Actifs

- **Origine des données** : Décompte instantané des sockets à l'état `ESTABLISHED` corrélés à un processus utilisateur valide via `/proc/net/tcp` et `/proc/net/udp`.

2.2. La Bannière de Statut Défensif

Le bandeau horizontal situé sous la barre de navigation reflète en temps réel l'évaluation heuristique du poste :

```
Statut Nominal (Sain) :  
[ SYSTÈME SÉCURISÉ // AUCUNE INTRUSION DÉTECTÉE ]  
Tous les processus actifs sont conformes et adossés à des binaires signés ou répertoriés.  
  
Statut Compromis (Alerte Active) :  
[ ALERTE DÉFENSIVE // ANOMALIES DÉTECTÉES ]  
Des processus suspects non officiels ou avec mémoire injectée sont actifs sur ce poste.
```

Toute modification de cet état ($0 \rightarrow N$ menaces) réveille immédiatement le gestionnaire d'événements asynchrones (`raptor_notify`) pour projeter une alerte vers l'environnement de bureau.

2.3. Top 5 des Consommateurs de Mémoire

Cette table synthétique permet à l'administrateur de surveiller les processus susceptibles d'épuiser les ressources de la machine (*OOM Killer*).

- **Critère de classement** : Tri décroissant strict basé sur la mémoire résidente réelle en RAM (`rss` , Resident Set Size).
- **Filtrage automatique** : Les threads du noyau Linux (PID ayant un `rss == 0` et pointant vers `[système/restreint]`) sont systématiquement exclus du tableau afin de ne pas masquer les applications consommatrices réelles.
- **Statut de dérive** : Indique si l'empreinte mémoire est stable ou en phase d'expansion critique par rapport au cycle d'échantillonnage précédent (T_{-1}).

2.4. Procédure Administrateur : Audit Forcé & Réinitialisation

En bas à gauche de la vue, le bouton [**LANCER UN NOUVEL AUDIT IMMÉDIAT**] réinitialise les tampons temporaires et force la relecture complète des structures du noyau : 1. Parcours itératif de l'arborescence `/proc` . 2. Re-qualification des chemins de confiance et des autorisations d'exécution. 3. Actualisation de l'horodatage système (Horodatage du contrôle : AAAA-MM-JJ HH:MM:SS).

SECTION 3 : ONGLET [02] PROCESSUS & RAM (INSPECTION FORENSIQUE)

L'onglet [02] **Processus & RAM** est la console d'investigation mémoire de RAPTOR. Il fournit une cartographie rigoureuse de chaque tâche résidente, vérifie la légitimité de ses segments de mémoire virtuelle et neutralise les contournements d'injection sans fichier.

+-----+					
-----+					
1. TOP 10 DES PLUS GROS CONSOMMATEURS DE MÉMOIRE					
PID	Nom du Processus	RAM (RSS)	Statut	Emplacement Binaire	
6649	chromium	2018 Mo	Stable	/usr/lib/chromium/chromium	
18145	chromium	391 Mo	Stable	/usr/lib/chromium/chromium	
6328	chromium	282 Mo	Stable	/usr/lib/chromium/chromium	
7506	chromium	245 Mo	Stable	/usr/lib/chromium/chromium	
1966	cinnamon	234 Mo	Stable	/usr/bin/cinnamon	
+-----+					
-----+					
2. PROCESSUS SUSPECTS ET ANOMALIES DÉTECTÉES					
PID	Nom du Processus	RAM (RSS)	Mémoire Virtuelle	Statut Dérive	Emplacement Binaire
18859	test_preload	2 Mo	0 Mo	HIJACK LD_PRELOAD	
/home/dan/GAMES/...					
+-----+					
-----+					
3. INSPECTION GLOBALE DES PROCESSUS (SURVEILLANCE EXHAUSTIVE DE L'HÔTE)					
PID	Nom du Processus	RAM (RSS)	Statut Dérive	Emplacement Binaire	
1	systemd	14 Mo	Stable	/usr/lib/systemd/systemd	
452	systemd-journald	22 Mo	Stable	/usr/lib/systemd/systemd-journald	
...	...	...	...	...	
+-----+					
-----+					

3.1. Mécanique d'Extraction des Métriques sous /proc

Pour chaque identifiant de processus rencontré lors de l'ouverture de `/proc`, RAPTOR applique la séquence d'extraction normalisée suivante :

LECTURE CHIRURGICALE DES STRUCTURES PAR PROCESSUS

<code>/proc/[PID]/comm</code>	---> Nom court de l'exécutable (ex: chromium)
<code>/proc/[PID]/exe</code>	---> <code>readlink()</code> vers le binaire réel sur disque
<code>/proc/[PID]/status</code>	---> Extraction de la ligne "VmRSS: [valeur] kB"
<code>/proc/[PID]/maps</code>	---> Parcours des descripteurs de segments VMA

1. Extraction Typographique (`comm`)

- Le fichier `/proc/[PID]/comm` est lu en premier. Si le fichier est inaccessible ou corrompu, RAPTOR se replie sur le dernier composant du chemin résolu par `/proc/[PID]/exe`. Si celui-ci est également indisponible, l'identifiant est normalisé sous la forme `proc_[PID]`.

2. Résolution du Binaire Réel (`exe`)

- L'appel système `readlink("/proc/[PID]/exe", out_exe, sizeof(out_exe))` résout le lien symbolique vers le fichier physique exécuté.
- Si le binaire appartient à un thread noyau ou dispose de permissions restreintes interdisant la lecture du lien, RAPTOR qualifie la cible en `[système/restreint]`.

3. Calcul de l'Empreinte Mémoire (`VmRSS`)

- RAPTOR analyse `/proc/[PID]/status` et isole le champ `VmRSS`.
- La valeur extraite en kilo-octets est convertie en octets ($kB \times 1024$) pour permettre des comparaisons arithmétiques 64 bits rigoureuses.

3.2. Taxonomie des Menaces & Qualification des Statuts

Le moteur forensique qualifie chaque processus selon un ordre de priorité déterministe. Une seule anomalie critique suffit à basculer la ligne dans le tableau des alertes.

Statut Qualifié	Niveau de Gravité	Vecteur d'Attaque Traqué	Signature Détectée
HIJACK LD_PRELOAD	CRITIQUE	Détournement de chargeur dynamique (Section 14)	Présence de la variable LD_PRELOAD= dans /proc/[PID]/environ sur un binaire non exempté.
MUTATION R-X	CRITIQUE	Injection mémoire sans fichier / Inode 0 (Section 13)	Page mémoire exécutable r-xp orpheline de fichier disque après bascule dynamique mprotect() .
INJECTION RWX	CRITIQUE	Violation de la politique $W \oplus X$	Présence d'un segment cumulant rwx sur la pile ([stack]), sur un binaire supprimé ou sans support JIT.
BINAIRE SUPPRIMÉ	ÉLEVÉ	Effacement de traces après exécution (Fileless dropper)	Le binaire exécuté est marqué (deleted) et n'existe plus physiquement sur le disque.
NON OFFICIEL	AVERTISSEMENT	Exécution en dehors des chemins système certifiés	Processus s'exécutant depuis /tmp , /dev/shm , ou un répertoire personnel non référencé.
CONFORME	NOMINAL	Processus intègre	Binaire situé dans les chemins officiels ou explicitement validé par mem_whitelist.conf .

3.3. Analyse Approfondie des Vecteurs d'Attaque

1. Détournement LD_PRELOAD (Section 14)

Le mécanisme de préchargement de bibliothèques est le vecteur privilégié des rootkits en espace utilisateur (*userland rootkits* tels que Jynx, Azazel ou Reptile en mode partagé).

- **Au niveau global :** Le fichier `/etc/ld.so.preload` force le chargeur dynamique (`ld.so`) à insérer une bibliothèque partagée (`.so`) dans l'espace mémoire de **tous** les processus démarrés sur l'hôte.
- **Au niveau ciblé :** La définition de la variable d'environnement `LD_PRELOAD=/chemin/lib.so` dans un processus permet de surcharger des symboles libc critiques (`read` , `write` , `connect` , `kill`) pour dissimuler des sockets ou des fichiers aux commandes d'administration (`ls` , `ps` , `netstat`).

RAPTOR intercepte ces attaques en lisant directement la mémoire d'environnement du noyau via `/proc/[PID]/environ`, où les variables sont stockées de façon contiguë sous la forme `CLÉ=VALEUR\0`.

2. Mutations R-X & Inode 0 (Section 13)

Pour contourner les mécanismes de détection statique des segments RWX, les implants modernes appliquent la séquence d'évasion en deux temps :

```
ÉTAPE A (Injection) : Allocation d'une page anonyme standard en lecture/écriture :
                        mmap(NULL, size, PROT_READ | PROT_WRITE, MAP_ANONYMOUS | MAP_PRIVATE,
-1, 0)
                        -> Permissions affichées : rw-p (Totalement légitime, non détecté par
RWX)

ÉTAPE B (Exécution) : Copie du shellcode en mémoire puis mutation des droits :
                        mprotect(addr, size, PROT_READ | PROT_EXEC)
                        -> Permissions affichées : r-xp (Inode 0, aucun fichier adossé sur
disque)
```

RAPTOR inspecte `/proc/[PID]/maps` ligne par ligne :

```
7f9d8a200000-7f9d8a204000 r-xp 00000000 00:00 0 <--- ALERTE RAPTOR : INODE 0 + R-X SANS
NOM
```

Dès qu'un segment présente les permissions `r-xp`, un décalage nul, un numéro d'inœud égal à `0` et une absence de nom de fichier disque (hors pages système certifiées `[vdso]` et `[vsyscall]`), RAPTOR lève l'alerte **MUTATION R-X** et transmet l'adresse mémoire exacte au registre forensique.

3. Intelligence Contextuelle : Mises à Jour Système vs Malwares

Lorsqu'un gestionnaire de paquets (APT, Dpkg, Flatpak) met à jour une application ouverte en mémoire (par exemple un navigateur web ou un environnement de bureau), Linux remplace le fichier sur le stockage. L'ancien binaire déjà ouvert par le processus en cours d'exécution est dissocié du système de fichiers (*unlinked*), et le noyau lui attribue automatiquement le suffixe (*deleted*).

Pour éliminer les faux positifs sans affaiblir la détection, RAPTOR applique une contre-vérification sur disque :

```
/* Extrait de la logique d'arbitrage de RAPTOR */
if (strstr(exe, "(deleted)") != NULL) {
    char clean_path[512];
    /* Extraction du chemin d'origine sans la mention "(deleted)" */
    raptor_strip_deleted_suffix(exe, clean_path);

    /* Si le fichier existe toujours sur disque et provient d'un chemin de confiance */
    if (access(clean_path, F_OK) == 0 && proc_mem_is_trusted_path(clean_path)) {
        is_deleted = false; /* Mise à jour de paquet officielle : alerte neutralisée */
    } else {
        is_deleted = true; /* Fichier introuvable sur disque : malware furtif confirmé */
    }
}
```

3.4. Registre Souverain d'Exclusion (`mem_whitelist.conf`)

Pour autoriser les moteurs légitimes exécutant du code généré à la volée (navigateurs Chromium/Firefox avec compilateurs JIT V8/SpiderMonkey, conteneurs Flatpak isolés, éditeurs basés sur `libffi`), RAPTOR utilise le fichier de configuration externe `assets/rules/mem_whitelist.conf` .

Syntaxe Officielle des Règles

Chaque directive est déclarée sur une ligne indépendante avec séparateurs verticaux (`|`) :

```
# =====
# RAPTOR DEFENSE // REGISTRE SOUVERAIN D'EXCLUSION MÉMOIRE (mem_whitelist.conf)
# =====
# MOTIF | TYPE | DESCRIPTION
# -----+-----+-----
cinnamon | JIT | Environnement de bureau Mint
github-desktop | JIT | Client Flatpak Electron / V8
chromium | JIT | Navigateur Web (Moteur V8)
firefox | JIT | Navigateur Web (SpiderMonkey)
xed | FFI | Éditeur de texte GNOME/Mint (libffi)
python3 | JIT | Interpréteur et modules dynamiques
node | JIT | Moteur serveur JavaScript
```

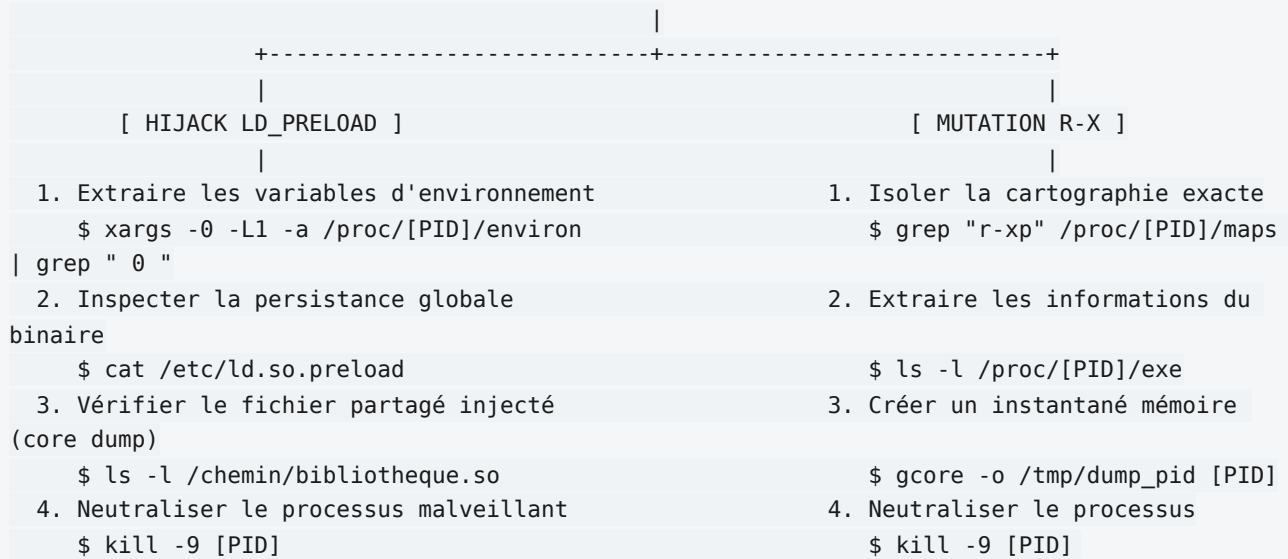
- **Fonctionnement du filtre** : RAPTOR effectue une comparaison insensible à la casse sur le nom d'exécutable (`comm`) et sur le chemin absolu (`exe`).
- **Souveraineté d'exploitation** : Aucune chaîne n'est codée en dur dans le binaire C. L'administrateur peut adapter ou restreindre le registre à chaud sans recompiler l'application.

3.5. Guide d'Intervention Administrateur Face aux Alertes

Lorsqu'une anomalie est signalée dans le tableau **2. PROCESSUS SUSPECTS ET ANOMALIES DÉTECTÉES** , appliquez la méthodologie d'investigation suivante depuis un terminal d'administration :

GUIDE DE RÉPONSE SUR INCIDENT

[Anomalie affichée dans RAPTOR]



Commandes Pratiques de Vérification Directe

1. Confirmer une injection `LD_PRELOAD` sur un processus suspect (ex: PID 18859) :

```
grep -z "^LD_PRELOAD=" /proc/18859/environ | tr '\0' '\n'
```

Si la commande retourne un chemin vers un fichier non officiel (`/tmp/...` , `/dev/shm/...` , ou bibliothèque orpheline), l'infection est confirmée.

2. Confirmer une mutation mémoire clandestine Inode 0 :

```
awk '$2 ~ /\.x./ && $5 == 0 && $6 == "" { print $0 }' /proc/[PID]/maps
```

Cette commande filtre instantanément les plages mémoire qui possèdent le droit d'exécution (`x`), sont rattachées à l'inode `0` et n'ont aucun fichier de référence sur le stockage.

3. Éliminer la menace :

```
kill -SIGKILL [PID]
```

Une fois le processus arrêté, cliquez sur [**LANCER UN NOUVEL AUDIT IMMÉDIAT**] dans RAPTOR : l'anomalie bascule au statut résolu et le compteur d'alertes revient au niveau nominal.

VOLUME 2 : SÉCURITÉ RÉSEAU, NET-GUARD & CAHIER FORENSIQUE (ONGLETS 3 & 4)

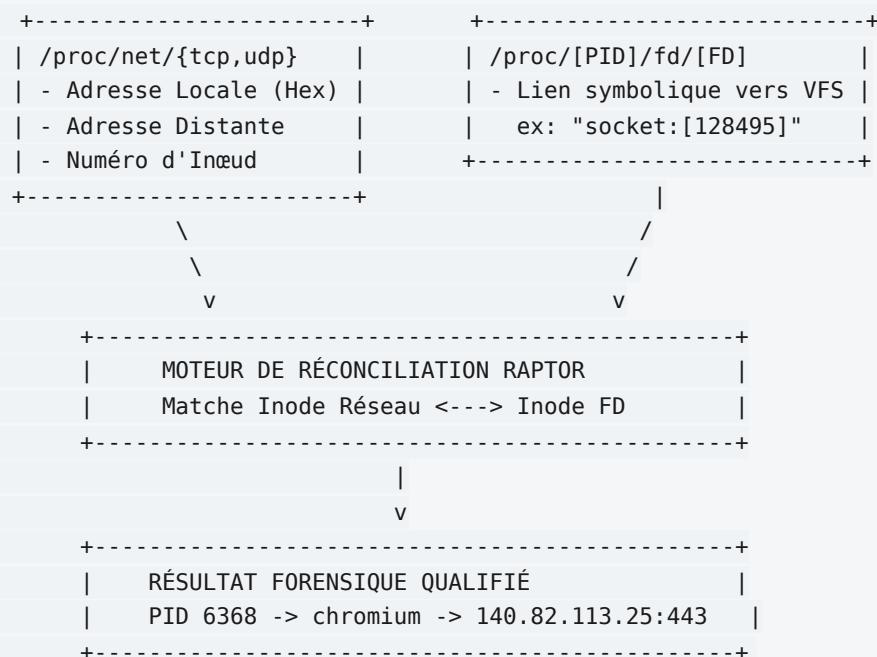
SECTION 4 : ONGLET [03] SOCKETS & RÉSEAU (NET-GUARD & SOUVERAINETÉ)

L'onglet [03] **Sockets & Réseau** assure la surveillance passive et la qualification souveraine de l'ensemble des communications réseau de l'hôte. Contrairement aux pare-feux conventionnels basés sur Netfilter/iptables qui inspectent les paquets au niveau de la couche réseau du noyau, RAPTOR corrèle en temps réel chaque flux réseau avec le processus local propriétaire, son binaire physique sur disque et son intégrité mémoire.

```
+-----+
+-----+
| RAPTOR DEFENSE // AUDIT MÉMOIRE & PROCESSUS   [ SURVEILLANCE ACTIVE ]       [ ? Guide
Analytique ] |
| [ 01 ] Vue Globale | [ 02 ] Processus & RAM | [ 03 ] Sockets & Réseau | [ 04 ] Diff | [ 05
] IA      |
+-----+
+-----+
| SÉCURITÉ RÉSEAU & CONTRÔLE SOUVERAIN (NET-GUARD / RWX)      [ SYNCHRONISER REGISTRE ] [ +
NOUVELLE ] |
| Analyse en direct des sockets ouverts, détection anti-phishing et traque d'injections
mémoire.      |
+-----+
+-----+
| Connexions actives et évaluation Net-Guard (3 flux surveillés) :
|
| PID / Processus          | Proto | Adresse Locale      | Adresse Distante    | État          |
Statut |
| PID 6368 (chromium)      | TCP   | 192.168.40.2:34902  | 140.82.113.25:443    | ESTABLISHED  |
SURVEILLÉ
| PID 17903 (github-desk...) | TCP   | 192.168.40.2:45430  | 140.82.114.25:443    | ESTABLISHED  |
SURVEILLÉ
| PID 17903 (github-desk...) | TCP   | 192.168.40.2:49272  | 140.82.114.21:443    | ESTABLISHED  |
SURVEILLÉ
+-----+
+-----+
| Processus avec mémoire exécutable non officielle (RWX) :
|
| PID   | Nom du Processus | Statut d'Injection | Chemin de l'exécutable
|
| 17841 | test_injection   | INJECTION RWX      | /tmp/payload_dropper
|
+-----+
+-----+
```

4.1. Architecture de Corrélation Triangulaire Réseau

Pour relier une adresse IP distante à un processus local sans jamais altérer le flux de données, RAPTOR exécute une corrélation triangulaire en trois étapes :



1. Analyse des tables `/proc/net/tcp` et `/proc/net/udp`

Le noyau Linux maintient l'état des sockets ouverts sous format hexadécimal condensé. Exemple d'enregistrement brut sous `/proc/net/tcp` :

```

sl  local_address rem_address  st tx_queue rx_queue tr tm->when retrnsmt  uid  timeout
inode
1: 0228A8C0:8856 1971528C:01BB 01 00000000:00000000 00:00000000 00000000 1000 0
128495

```

• Conversion mathématique :

- `local_address` (0228A8C0:8856) est converti en petit-boutiste (*little-endian*) : C0.A8.28.02 → 192.168.40.2 sur le port 0x8856 → 34902 .
- `rem_address` (1971528C:01BB) → 8C.52.71.19 → 140.82.113.25 sur le port HTTPS 0x01BB → 443 .
- `st` (01) identifie rigoureusement l'état `TCP_ESTABLISHED` .
- `inode` (128495) constitue la clé d'ancrage universelle.

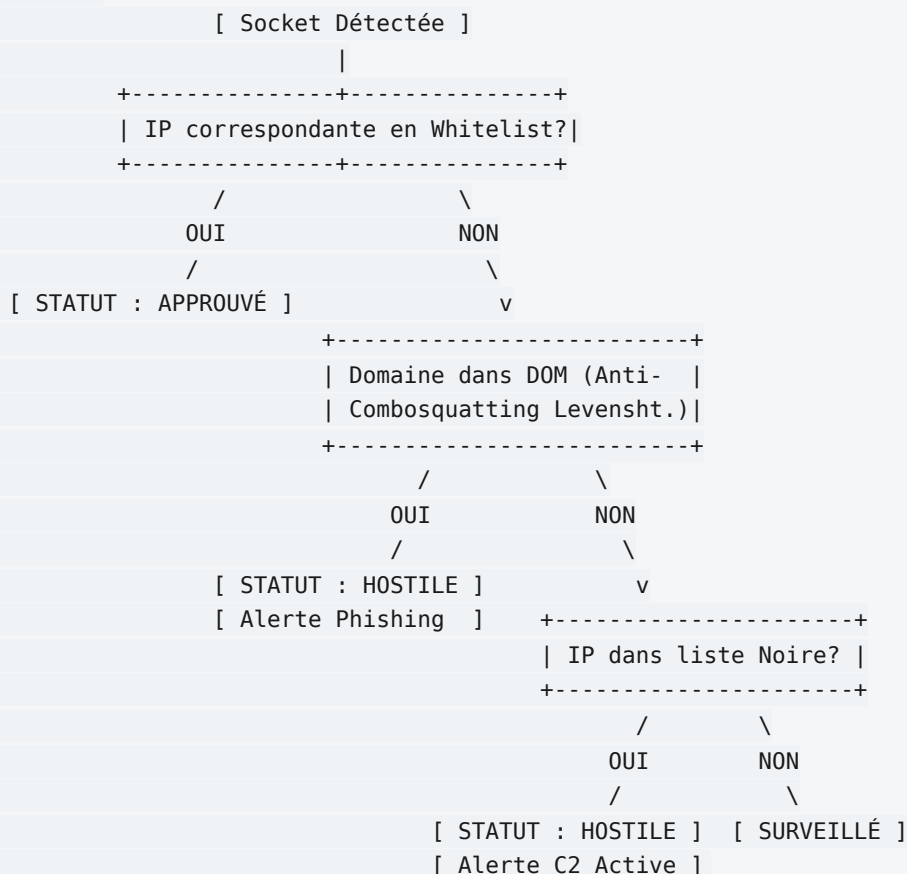
2. Résolution Inœud → PID (`/proc/[PID]/fd/`)

RAPTOR parcourt les répertoires `/proc/[PID]/fd/` de l'ensemble des tâches actives et lit les liens symboliques via l'appel système `readlink()` . Lorsqu'un descripteur de fichier pointe vers la chaîne `socket:[128495]` , le propriétaire légal du flux est formellement identifié sans ambiguïté.

4.2. Moteur Net-Guard (Section 11) : Filtrage Binaire & Anti-Phishing

Dès qu'une socket est résolue, elle traverse le filtre Net-Guard afin de déterminer sa légitimité.

FLUX D'ÉVALUATION D'UNE SOCKET NET-GUARD



Grille d'Évaluation des Statuts Réseau

Statut Net-Guard	Niveau de Menace	Description & Conséquence Opérationnelle
APPROUVÉ	NOMINAL	Flux dirigé vers une infrastructure institutionnelle, un réseau d'entreprise autorisé ou un serveur de mise à jour officiel certifié.
SURVEILLÉ	ATTENTION	Flux légitime sortant vers des services Web, des hébergeurs tiers ou des réseaux CDN ne figurant pas sur liste noire.
HOSTILE	CRITIQUE	Connexion établie vers un relais de commande et contrôle (C2), une passerelle d'exfiltration ou un domaine d'hameçonnage identifié.

Format des Règles Souveraines (fw_officiel.conf et fw_custom.conf)

Le filtrage réseau de RAPTOR repose sur deux fichiers au format texte modifiables à chaud :

1. `assets/rules/fw_officiel.conf` : Registre de sécurité maintenu et mis à jour de façon souveraine.
2. `assets/rules/fw_custom.conf` : Règles personnalisées de l'organisation locale.

```
# =====
# RÈGLES DE SOUVERAINETÉ RÉSEAU (SYNTAXE OFFICIELLE RAPTOR)
# =====

# Règle DOM (Anti-Combosquatting / Phishing) :
# DOM | Marque Cible | Domaines Légitimes Autorisés | Mots-clés Suspects | Description
DOM | hydroquebec | hydroquebec.com,hydro.qc.ca | paiement,facture,remboursement | Portail
officiel d'énergie
DOM | desjardins | desjardins.com,desjardins.ca | securite,validation,acces | Espace
transactionnel bancaire

# Règle IP (Filtrage Binaire / Masques CIDR) :
# IP | Adresse IP ou Sous-réseau CIDR | Statut | Description
IP | 192.168.40.0/24 | APPROUVÉ | Réseau local d'administration sécurisé
IP | 45.33.32.156 | HOSTILE | Serveur de contrôle Cobalt Strike
identifié
IP | 185.220.101.0/24 | HOSTILE | Nœuds de sortie Tor d'exfiltration
```

Ajout de Règles à Chaud via l'Interface Graphique

Le bouton **[+ NOUVELLE RÈGLE]** situé en haut à droite de l'onglet 3 ouvre une boîte modale interactive permettant d'intégrer immédiatement une règle de blocage ou d'autorisation sans redémarrer le programme : * **Type de cible** : Sélection par commutateur entre **DOM** (anti-phishing) et **IP** (blocage réseau). * **Validation dynamique** : RAPTOR vérifie la cohérence du masque CIDR et enregistre la nouvelle directive directement dans `assets/rules/fw_custom.conf`.

Neutralisation Chirurgicale d'un Flux Hostile

Lorsqu'un socket est qualifié **HOSTILE**, la colonne *Action Défensive* propose l'abattage immédiat du processus : * En un clic, RAPTOR transmet un signal d'arrêt non différable :

kill(PID, SIGKILL)

- La connexion TCP est immédiatement interrompue par l'envoi d'un paquet **RST** par la pile réseau du noyau Linux, empêchant toute exfiltration de données.

4.3. Renseignement BGP & Rapprochement Lexical (Section 12)

Moteur Anti-Combosquatting

Pour neutraliser les attaques d'ingénierie sociale et les domaines de phishing imitant des organisations légitimes, RAPTOR calcule la distance lexicale de **Levenshtein** entre le nom d'hôte résolu et les marques enregistrées dans la configuration : * Si un domaine distant contient la marque cible (ex: `desjardins-verification-compte.com`), mais ne correspond à aucun des domaines certifiés de la liste blanche (`desjardins.com`), le moteur lève instantanément une alerte rouge d'usurpation d'identité institutionnelle.

Cascade Anti-Opacité & Résolution BGP (Net-Intel)

Pour identifier instantanément les serveurs distants qui ne possèdent aucun enregistrement DNS inverse (PTR), RAPTOR déploie une cascade d'identification en trois couches :

CASCADE DE RENSEIGNEMENT RÉSEAU SANS PTR

[IP Distant non résolue par PTR]

|
v

```
+-----+
| ÉTAGE 1 : Cache Dynamique Mémoire |
| Recherche locale immédiate (0 ms) |
+-----+
```

| (Non trouvé)
v

```
+-----+
| ÉTAGE 2 : Table Statique CIDR (0 ms) |
| assets/rules/table_static.conf       |
| Match contre CDN mondiaux / Opérateurs |
+-----+
```

| (Non trouvé)
v

```
+-----+
| ÉTAGE 3 : Sonde DNS BGP (Team Cymru) |
| Résolution asynchrone non bloquante  |
| Query : [IP-inversée].origin.asn.cymru.com|
+-----+
```

|
v

[ASN Identifié : ex AS13335 (Cloudflare)]

1. **Étage 1 (Cache L1)** : Table de hachage interne thread-safe conservant les résolutions précédentes afin d'éliminer les requêtes réseau répétitives.
2. **Étage 2 (Table Statique 0 ms)** : Fichier `assets/rules/table_static.conf` contenant les allocations IP des infrastructures mondiales majeures (AWS, Cloudflare, Akamai, OVH, Hetzner, Fastly, GitHub).
3. **Étage 3 (Sonde BGP Asynchrone)** : Si l'adresse est inconnue, un thread détaché interroge le service d'infrastructure de **Team Cymru** via une requête DNS `TXT` au format :

```
host -t txt 25.113.82.140.origin.asn.cymru.com
```

La réponse extrait directement le numéro de système autonome (ASN), la plage CIDR allouée et le code pays officiel d'enregistrement.

4.4. Tableau Inférieur : Corrélation Sockets et Injections RWX

La partie inférieure de l'onglet 3 isole spécifiquement les processus qui détiennent une connexion réseau active **tout en présentant des segments de mémoire altérés (RWX)** :

- **Gravité tactique maximale** : Un processus réseau normal (comme `curl`, `firefox` ou `ssh`) possède des segments mémoire stricts ($W \oplus X$). La présence simultanée d'une connexion réseau externe et d'une allocation `rwX` (Lecture, Écriture, Exécution) est le marqueur caractéristique d'un **implant distant**, d'un **shell interactif inversé** (*reverse shell*) ou d'un **relais d'exfiltration C2**.

- **Visibilité immédiate** : Tout binaire présent dans ce tableau doit faire l'objet d'un abattage prioritaire (SIGKILL) et d'un examen forensique approfondi.
-

SECTION 5 : ONGLET [04] CAHIER DIFFÉRENTIEL (FORENSIQUE TEMPOREL & MUTATIONS)

L'onglet [04] **Cahier Différentiel** est l'instrument de diagnostic temporel de RAPTOR. Il offre une visibilité sur les anomalies dynamiques qui ne peuvent pas être détectées par un audit instantané ponctuel : les gonflements progressifs de mémoire (fuites et injections par étapes) et l'historique chronologique des mutations d'exécution clandestines.

```
+-----+
+-----+
| CAHIER DIFFÉRENTIEL TEMPOREL (T-1 vs T0) // SUIVI DE DÉRIVE RAM
|
| Mesure différentielle active entre les cycles d'échantillonnage de /proc (analyse
d'inflation RAM) |
| PID   | NOM PROCESSUS   | RAM T-1   | RAM T0     | VARIATION | STATUT
|
| Système stable. Aucune fuite mémoire critique d'expansion détectée sur le dernier cycle.
|
+-----+
+-----+
| REGISTRE FORENSIQUE : MUTATIONS EN VOL & INJECTIONS R-X SANS FICHIER (1 ÉVÉNEMENT) [ Purger
|
| PID   | EXÉCUTABLE   | PLAGE MÉMOIRE (RAM)   | TAILLE | DIAGNOSTIC FORENSIQUE
|
| 18859 | test_preload | N/A                   | N/A    | Détournement LD_PRELOAD :
/lib/...
|
+-----+
+-----+
```

5.1. Volet Supérieur : Suivi Différentiel de Dérive RAM (T_{-1} vs T_0)

Le premier tableau capture l'empreinte mémoire résidente (RSS) de chaque processus au cycle d'échantillonnage précédent (T_{-1}) et la confronte au cycle courant (T_0).

1. Formule d'Analyse d'Inflation

Pour chaque processus actif présent sur les deux cycles :

$$\Delta \text{RSS} = \text{RSS}(T_0) - \text{RSS}(T_{-1})$$

$$\text{Variation \%} = \frac{\Delta \text{RSS}}{\text{RSS}(T_{-1})} \times 100$$

2. Typologie des Dérives Surveillées

- **Expansion Normale** : Augmentation modérée lors du chargement de documents ou d'onglets de navigation.
- **Fuite Mémoire Continue (*Memory Leak*)** : Croissance ininterrompue de la mémoire vive sans stabilisation lors des phases d'inactivité, conduisant à l'instabilité du système et au

déclenchement de l'OOM-Killer (*Out Of Memory*).

- **Allocation Furtive par Paliers** : Comportement typique d'implants de collecte qui mettent en mémoire tampon des frappes clavier (*keylogging*), des captures d'écran ou des fichiers avant exfiltration groupée.

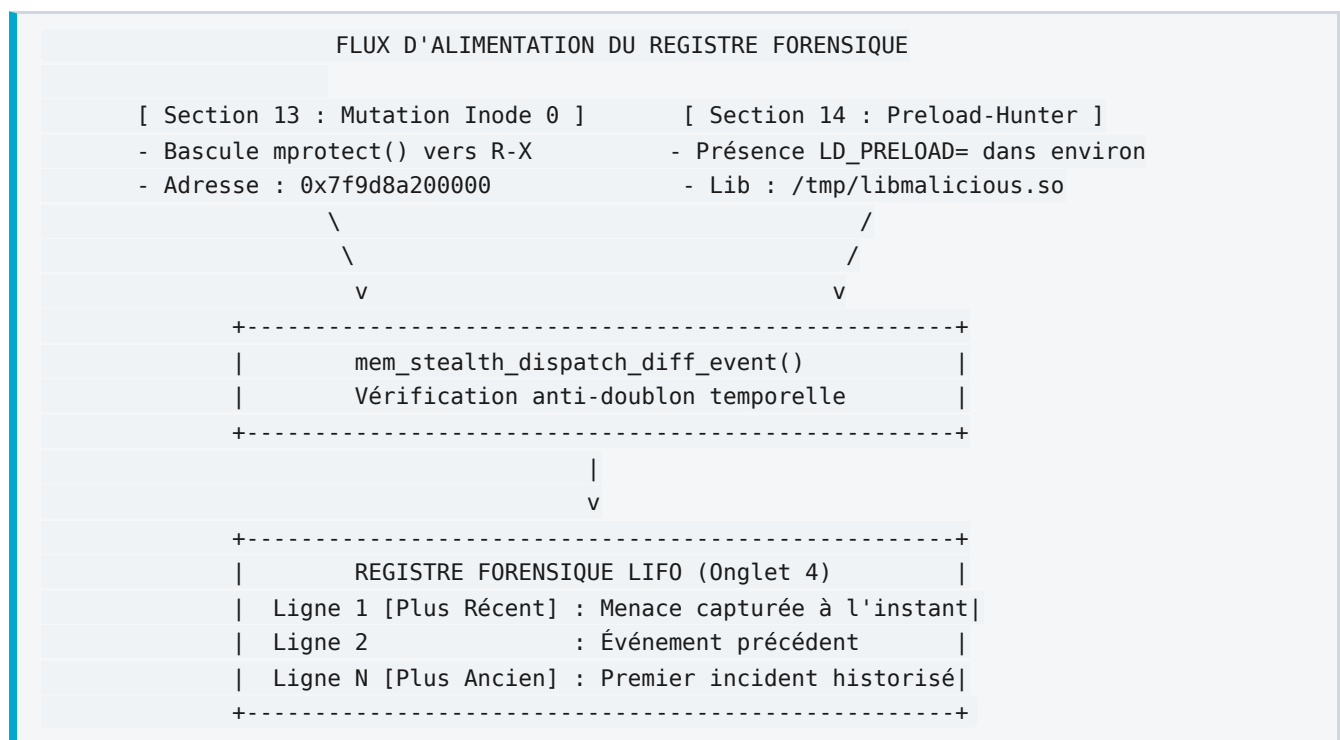
Lorsque toutes les variations demeurent dans les seuils nominaux, RAPTOR affiche un message de conformité vert : "Système stable. Aucune fuite mémoire critique d'expansion détectée sur le dernier cycle."

5.2. Volet Inférieur : Registre Forensique LIFO (Sections 13 & 14)

Le second volet constitue le registre immuable des événements de sécurité critiques détectés sur le sous-système mémoire.

Architecture de la File LIFO (*Last In, First Out*)

Les alertes forensiques sont conservées au sein d'un tampon circulaire protégé en mémoire (`MAX_STEALTH_EVENTS`). Le rendu à l'écran applique un ordonnancement LIFO strict : **la dernière attaque interceptée apparaît systématiquement sur la première ligne rouge du tableau**, garantissant à l'administrateur une lecture chronologique immédiate sans nécessiter de défilement.



Signification des Colonnes Forensiques

1. **PID** : Identifiant numérique du processus audité au moment de l'infraction.
2. **EXÉCUTABLE** : Nom court de la commande ou du binaire (`comm`).
3. **PLAGE MÉMOIRE (RAM)** :
 - Pour une Mutation R-X (Section 13) : Adresse hexadécimale exacte de début et de fin de la page anonyme devenue exécutable (ex: `0x7f10b000 - 0x7f10c000`).
 - Pour un Détournement LD_PRELOAD (Section 14) : Affichage normalisé **N/A** . Cette mention signifie formellement **Non Applicable**, car l'injection s'est opérée au niveau de

l'environnement d'exécution du chargeur dynamique et ne possède pas d'adresse virtuelle isolée.

4. TAILLE :

- Pour une Mutation R-X : Taille physique de la plage injectée en kilo-octets (calculée par $\frac{\text{Adresse Fin} - \text{Adresse Début}}{1024}$).
- Pour un Détournement LD_PRELOAD : Affichage normalisé **N/A**.

5. DIAGNOSTIC FORENSIQUE :

- Indication claire de la signature d'intrusion et, dans le cas d'un LD_PRELOAD, le chemin complet vers la bibliothèque partagée injectée.

5.3. Procédures Administrateur : Purge et Traçabilité

1. Purge Manuelle de l'Histoire

Lorsqu'un incident a été investigué, corrigé et que le binaire malveillant a été neutralisé, l'administrateur peut réinitialiser la vue à l'aide du bouton **[Purger l'historique]** situé en haut à droite du panneau forensique. * Cette action appelle la fonction `mem_stealth_clear_diff_events()`. * Le tampon mémoire est remis à zéro. * L'encadré repasse immédiatement en vert avec la mention : "• AUCUNE MUTATION CLANDESTINE DÉTECTÉE : Aucun processus n'a converti de mémoire anonyme vers R-X (W^X respecté)."

2. Traçabilité Post-Incident en Ligne de Commande

Avant d'effectuer la purge de l'historique dans RAPTOR, l'administrateur peut exporter l'état des descripteurs pour archivage légal :

```
# Exporter la cartographie mémoire complète du processus suspect (ex: PID 18859)
cat /proc/18859/maps > /var/log/raptor_incident_18859_maps.log

# Exporter l'environnement d'exécution complet
xargs -0 -L1 -a /proc/18859/envron > /var/log/raptor_incident_18859_env.log

# Extraire les bibliothèques partagées ouvertes
lsof -p 18859 > /var/log/raptor_incident_18859_lsof.log
```

VOLUME 3 : HEURISTIQUES D'ANOMALIE, NOTIFICATIONS & BANCS D'ESSAIS (ONGLET 5, SECTION 15 & OUTILS)

SECTION 6 : ONGLET [05] RAPPORT IA & HEURISTIQUES D'ANOMALIE

L'onglet [05] **Rapport IA** concentre l'intelligence décisionnelle et la synthèse prédictive de RAPTOR v2.0. Alors que les onglets précédents fournissent une vision forensique granulaire (pages mémoire, sockets bruts, descripteurs de fichiers), le moteur heuristique corrèle ces signaux faibles pour produire une évaluation d'intégrité globale et orienter les actions de remédiation de l'administrateur.

```
+-----+
| RAPTOR DEFENSE // AUDIT MÉMOIRE & PROCESSUS   [ SURVEILLANCE ACTIVE ]      [ ? Guide
Analytique ] |
| [ 01 ] Vue Globale | [ 02 ] Processus & RAM | [ 03 ] Sockets & Réseau | [ 04 ] Diff | [ 05
] IA      |
+-----+
| SYNTHÈSE D'ÉVALUATION SOUVERAINE // MOTEUR D'ANALYSE PRÉDICTIVE ET HEURISTIQUE
|
| Évaluation algorithmique multi-vectorielle en temps réel (0 télémétrie distante, 100 %
local).      |
+-----+
| INDICE DE CONFIANCE HÔTE : 72 / 100 [ STATUT : VIGILANCE RENFORCÉE // COMPROMISSION
PROBABLE ]    |
| [=====]
|
+-----+
| VECTEURS DE RISQUE ANALYSÉS :
|
| 1. INTÉGRITÉ MÉMOIRE VIVE (RWX / R-X) : ALERTE CRITIQUE (1 processus hostile détecté)
|
| 2. CHARGEUR DYNAMIQUE (LD_PRELOAD) : ALERTE CRITIQUE (Détournement d'environnement
actif)      |
| 3. POSTURE RÉSEAU (SOCKETS SORTANTS) : CONFORME (Aucun flux C2 ou combosquatting
répertorié) |
| 4. PERSISTANCE DISQUE ET RACINES HOST : INTÈGRE (Fichier /etc/ld.so.preload vierge)
|
+-----+
| RECOMMANDATIONS OPÉRATIONNELLES IMMÉDIATES :
|
| [!] Isoler et neutraliser le PID 18859 (test_preload) : injection LD_PRELOAD suspecte.
|
| [i] Contrôler l'intégrité de la bibliothèque préchargée : /lib/x86_64-linux-gnu/libm.so.6
|
| [i] Procéder à un redémarrage des applications Flatpak si des mises à jour système sont en
cours.      |
+-----+
+-----+
```

6.1. Algorithme de Calcul de l'Indice de Confiance

L'indice de santé système (noté de 0 à 100) est évalué à chaque cycle d'audit par pondération différentielle des anomalies constatées sur les 4 vecteurs critiques d'intrusion :

$$\text{Indice Global} = 100 - \sum_{i=1}^n (\text{Pondération}_i \times \text{Gravité}_i)$$

Barème des Pénalités Heuristiques

Vecteur d'Analyse	Condition Déclenchante	Pénalité Déduite	Statut Résultant
Persistance Globale	Infection active de /etc/ld.so.preload	-50 pts	COMPROMISSION CRITIQUE
Détournement Ciblé	Variable LD_PRELOAD sur binaire hors whitelist	-30 pts / proc	VIGILANCE RENFORCÉE
Mutation Clandestine	Plage r-xp orpheline Inode 0 (mprotect)	-35 pts / proc	COMPROMISSION PROBABLE
Violation W^X (RWX)	Plage mémoire cumulant rwx suspecte	-25 pts / proc	VIGILANCE RENFORCÉE
Communication C2	Socket vers IP ou domaine qualifié HOSTILE	-30 pts / flux	VIGILANCE RENFORCÉE
Combosquatting	Domaine distant imitant une marque protégée	-20 pts / flux	ATTENTION
Dérive Anormale	Inflation RAM continue inexpliquée (> 200%)	-10 pts / proc	ATTENTION

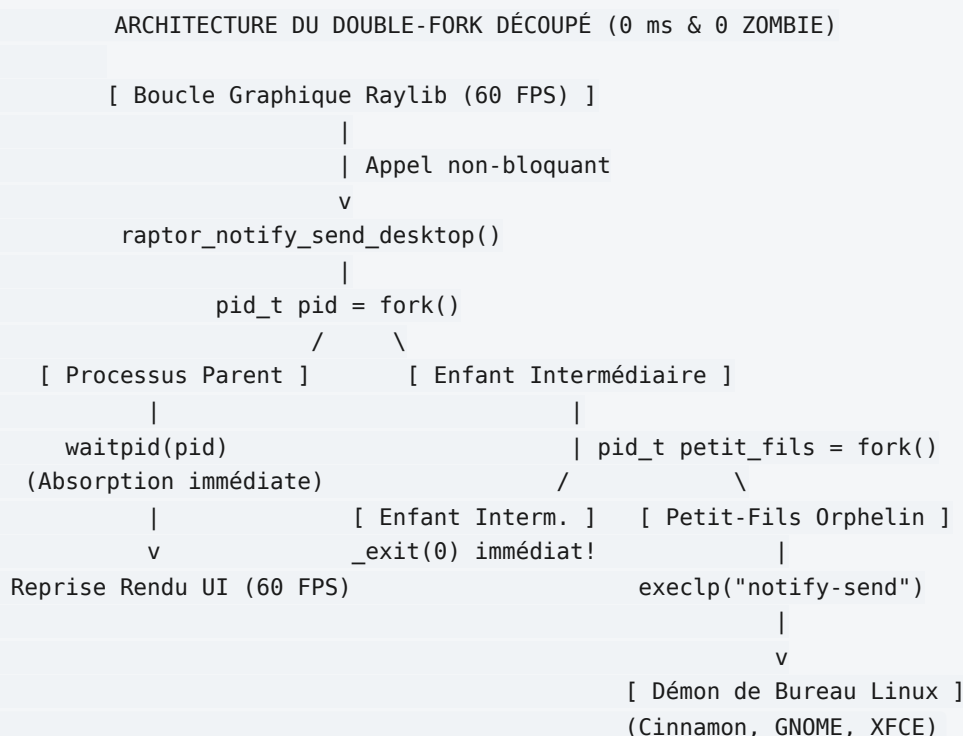
6.2. Moteur d'Export et Rapports Forensiques

L'administrateur peut exporter l'intégralité du rapport d'audit vers des formats structurés pour archivage, conformité réglementaire (ISO 27001, ANSSI, NIST) ou partage au sein du SOC :

1. **Rapport Console Formaté (CLI)** : Sortie directe sur `stdout` avec codes d'échappement ANSI en couleur.
2. **Journal Forensique Daté (JSON / Texte brut)** : Archivé par défaut sous `/var/log/raptor/audit_YYYYMMDD_HHMMSS.log`.
3. **Rapport Exécutif Autonome (HTML / CSS Cyber Dark)** : Fichier autonome intégrant graphiques vectoriels et tables dynamiques sans dépendance JavaScript externe.

SECTION 7 : NOTIFICATIONS SYSTÈME ASYNCHRONES FREEDESKTOP (SECTION 15)

Pour garantir une protection continue du poste même lorsque RAPTOR fonctionne en tâche de fond ou avec sa fenêtre Raylib minimisée, la Section 15 introduit le moteur de notifications asynchrones `raptor_notify`.



7.1. Mécanisme du Double-Fork & Préservation du Moteur Graphique

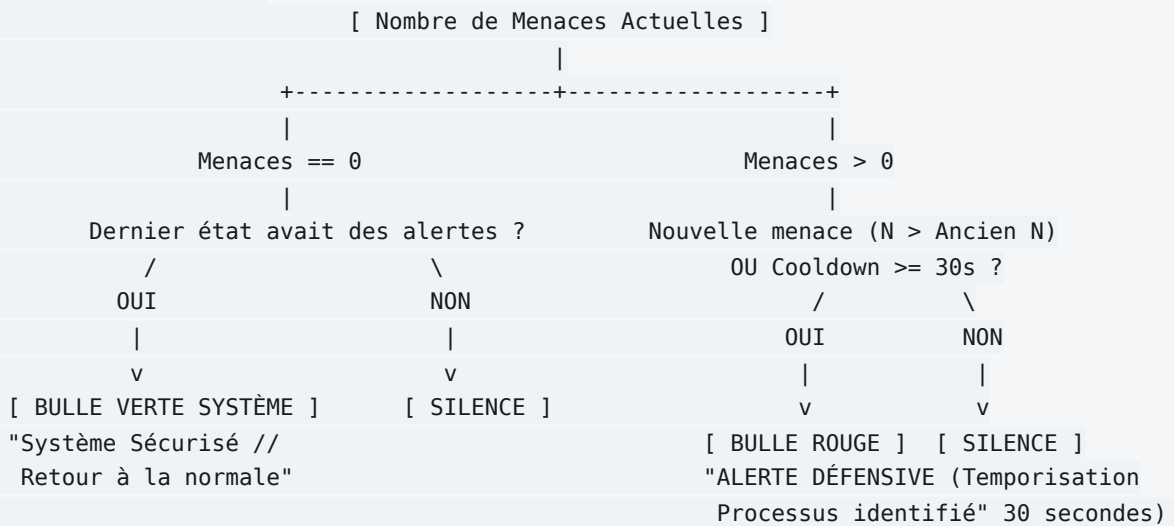
Un appel standard à `system("notify-send ...")` figerait la boucle de rendu Raylib pendant 100 à 300 ms, provoquant un décrochage visible du rafraîchissement d'écran et des saccades d'animation.

Pour contourner ce verrouillage, la fonction 86 (`raptor_notify_send_desktop`) implémente la technique UNIX du double découplage (`double-fork`) : 1. **Premier fork()** : Le processus parent crée un enfant intermédiaire puis l'attend immédiatement via `waitpid(pid, NULL, 0)`. 2. **Second fork()** : L'enfant intermédiaire crée immédiatement un petit-fils, puis s'éteint sans délai (`_exit(0)`). 3. **Adoption par init / systemd (PID 1)** : Le petit-fils devient orphelin et se retrouve automatiquement rattaché au gestionnaire racine du système. 4. **Exécution isolée** : Le petit-fils appelle `execlp("notify-send")`. À sa terminaison, le système d'exploitation nettoie automatiquement ses structures sans générer aucun processus zombie (`<defunct>`) dans la table des processus.

7.2. Filtre Anti-Rebond Temporel (Debounce Logic)

Pour éviter de saturer le serveur d'affichage et l'attention de l'opérateur par des centaines de bulles consécutives, la fonction 87 (`raptor_notify_evaluate_threats`) arbitre les émissions d'alerte :

LOGIQUE DU FILTRE ANTI-REBOND TEMPOREL



- **Déclenchement Instantané** : L'apparition d'une nouvelle menace (passage de 0 → N ou élévation du nombre d'anomalies) déclenche la notification sans aucun délai d'attente.
- **Temporisation de Rappel (Cooldown)** : Si l'anomalie persiste sans être neutralisée, un rappel n'est émis qu'une fois toutes les 30 secondes au maximum.
- **Notification de Résolution** : Dès que l'administrateur neutralise le processus pirate et que le compteur retombe à zéro, une notification informative verte est transmise au bureau pour confirmer le retour au statut nominal.

SECTION 8 : BANCS D'ESSAIS FORENSIQUES & VALIDATION OPÉRATIONNELLE

RAPTOR est livré avec deux outils d'injection inoffensifs autonomes situés dans `tools/` permettant de certifier le bon fonctionnement des sondes en conditions réelles sans compromettre la machine.

```
RAPTOR/tools/  
├─ test_stealth.c    # Banc d'essai Section 13 : Mutation de permissions mprotect()  
└─ test_preload.c    # Banc d'essai Section 14 : Détournement de chargeur LD_PRELOAD
```

8.1. Épreuve 1 : Validation de la Traque de Mutations R-X (`test_stealth.c`)

Le banc d'essai `test_stealth.c` reproduit fidèlement la technique de camouflage utilisée par les injecteurs de shellcodes sans fichier :

```
/* Extrait fonctionnel de test_stealth.c */  
void simuler_mutation_clandestine(void) {  
    size_t taille = 4096; // 1 page standard  
  
    // 1. Allocation d'une page anonyme sans fichier sur disque (Inode 0)  
    void* ptr = mmap(NULL, taille, PROT_READ | PROT_WRITE,  
                     MAP_ANONYMOUS | MAP_PRIVATE, -1, 0);  
  
    // 2. Écriture d'un octet de test (simulation d'injection de payload)  
    ((char*)ptr)[0] = 0x90; // NOP (No Operation)  
  
    // 3. Bascule clandestine vers Exécutable (Violation W^X indirecte)  
    mprotect(ptr, taille, PROT_READ | PROT_EXEC);  
  
    // 4. Maintien en mémoire vive pendant 60 secondes pour audit  
    sleep(60);  
  
    munmap(ptr, taille);  
}
```

Procédure de Test Pas-à-Pas

1. Compiler le banc d'essai :

```
gcc -Wall -O2 tools/test_stealth.c -o tools/test_stealth
```

2. Lancer RAPTOR dans un terminal.
3. Exécuter le test dans un second terminal :

```
./tools/test_stealth
```

4. Comportement attendu dans RAPTOR :

- L'onglet 1 passe immédiatement à **1 ANOMALIE** avec bandeau rouge d'alerte défensive.
- L'onglet 2 référence `test_stealth` avec le statut **MUTATION R-X**.
- L'onglet 4 (Cahier Différentiel) affiche l'événement en première ligne avec la plage d'adresses hexadécimales exactes (ex: `0x7fa2c000 - 0x7fa2d000`) et la taille de `4 Ko`.
- Une bulle de notification rouge s'affiche sur le bureau de l'opérateur.

8.2. Épreuve 2 : Validation de Preload-Hunter (`test_preload.c`)

Le banc d'essai `test_preload.c` est un binaire inoffensif qui boucle pendant 60 secondes, utilisé pour vérifier la détection de l'injection d'une bibliothèque partagée dans son environnement d'exécution.

```
/* Extrait de test_preload.c */
int main(void) {
    printf("[BANC D'ESSAI PRELOAD] PID %d actif pour 60 secondes...\n", getpid());
    sleep(60);
    return 0;
}
```

Procédure de Test Pas-à-Pas

1. Compiler le binaire :

```
gcc -Wall -O2 tools/test_preload.c -o tools/test_preload
```

2. Lancer le binaire en lui injectant dynamiquement la bibliothèque mathématique standard du système via `LD_PRELOAD` :

```
LD_PRELOAD=/lib/x86_64-linux-gnu/libm.so.6 ./tools/test_preload
```

3. Comportement attendu dans RAPTOR :

- L'onglet 1 passe à **1 ANOMALIE**.
- L'onglet 2 liste `test_preload` avec le statut prioritaire **HIJACK LD_PRELOAD**.
- L'onglet 4 référence l'événement avec les mentions **N/A** dans les colonnes *Plage Mémoire* et *Taille*, et affiche dans *Diagnostic* : "Détournement `LD_PRELOAD : /lib/x86_64-linux-gnu/libm.so.6`".
- Le bureau Linux affiche une alerte identifiant nommé `test_preload`.
- Dès que les 60 secondes s'écoulent et que le programme se termine, le compteur revient à 0 et une notification verte confirme la normalisation du poste.

SECTION 9 : GOUVERNANCE TECHNIQUE & DICTIONNAIRE GLOBAL DES FONCTIONS (1 À 87)

L'architecture logicielle de RAPTOR est soumise à une gouvernance stricte : **aucun code non tracé, chaque fonction possède un identifiant unique immuable au dictionnaire et toute modification incrémente la révision du fichier d'en-tête.**

MATRICE DES 15 SECTIONS DE RAPTOR

[Sec 01-10 : Socle Universel, Forensique RWX & Rapports]	----> Fonctions 01 à 57
[Sec 11 : Net-Guard & Filtrage Souverain Sockets]	----> Fonctions 58 à 70
[Sec 12 : Net-Intel & Renseignement BGP/ASN Team Cymru]	----> Fonctions 71 à 79
[Sec 13 : Mem-Stealth & Traque Mutations R-X Inode 0]	----> Fonctions 80 à 83
[Sec 14 : Preload-Hunter & Audit Chargeur Dynamique]	----> Fonctions 84 à 85
[Sec 15 : Notifications Asynchrones Freedesktop]	----> Fonctions 86 à 87

9.1. Index Exhaustif des Fonctions Métier

Section 11 — Moteur Net-Guard & Filtrage Sockets (`raptor_net_guard.c`)

- **58.** `net_guard_init` : Initialisation des descripteurs et structures de hachage réseau.
- **59.** `net_guard_cleanup` : Libération des registres et tampons de filtrage mémoire.
- **60.** `net_guard_rules_load` : Chargement et parsing des fichiers `fw_officiel.conf` et `fw_custom.conf`.
- **61.** `net_guard_rule_add_memory` : Insertion atomique d'une règle en mémoire sans interruption de service.
- **62.** `net_guard_rule_add_custom` : Persistance sur disque d'une nouvelle règle dans `fw_custom.conf`.
- **63.** `net_guard_eval_socket` : Évaluation d'un flux selon la cascade Whitelist → DOM → IP.
- **64.** `net_guard_kill_process` : Émission du signal `SIGKILL` vers un PID hostile identifié.
- **65.** `net_guard_sync_official` : Téléchargement sécurisé (`curl`) et mise à jour atomique du registre.
- **66.** `net_guard levenshtein_dist` : Calcul mathématique de distance pour la détection de typosquatting.
- **67.** `net_guard_normalize_string` : Nettoyage typographique et passage en minuscules pour comparaison.
- **68.** `net_guard_parse_ip_port` : Décomposition d'une chaîne IP:Port vers les types binaires POSIX.
- **69.** `net_guard_ip_in_cidr` : Vérification binaire d'appartenance d'une adresse à un masque de sous-réseau.
- **70.** `net_guard_dns_worker_thread` : Résolution inverse asynchrone non-bloquante (`pthread`).

Section 12 — Moteur Net-Intel & Cascade BGP (`raptor_asn_intel.c`)

- **71.** `asn_intel_init` : Initialisation du cache de renseignement et chargement de la table statique.
- **72.** `asn_intel_cleanup` : Libération des tables de hachage de l'infrastructure de renseignement.

- **73. `asn_intel_table_load`** : Parsing et indexation en mémoire du fichier `assets/rules/table_static.conf`.
- **74. `asn_intel_cache_lookup`** : Recherche ultra-rapide en mémoire (0 ms) dans le cache L1.
- **75. `asn_intel_cache_insert`** : Insertion thread-safe d'une résolution validée dans le cache.
- **76. `asn_intel_static_lookup`** : Confrontation binaire 0 ms contre les plages CIDR des opérateurs mondiaux.
- **77. `asn_intel_query_cymru_sync`** : Émission d'une requête DNS TXT de secours vers Team Cymru.
- **78. `asn_intel_resolve_ip_async`** : Déclenchement de la cascade anti-opacité sur thread détaché.
- **79. `asn_intel_format_descriptor`** : Mise en forme unifiée du libellé ASN et code pays pour l'interface.

Section 13 — Traque Mémoire Anonyme R-X (`mem_stealth.c`)

- **80. `mem_stealth_whitelist_load`** : Lecture et indexation du registre souverain `assets/rules/mem_whitelist.conf`.
- **81. `mem_stealth_is_whitelisted`** : Validation d'innocuité d'un processus vis-à-vis des moteurs JIT et FFI certifiés.
- **82. `mem_stealth_scan_maps_anon_rx`** : Traque des segments exécutables `r-xp` Inode 0 dans `/proc/[PID]/maps`.
- **83. `mem_stealth_dispatch_diff_event`** : Enregistrement et déduplication des alertes dans la file LIFO de l'onglet 4.

Section 14 — Traque du Chargeur Dynamique (`preload_hunter.c`)

- **84. `preload_hunter_check_global`** : Contrôle passif direct de `/etc/ld.so.preload` pour neutraliser les rootkits hôtes.
- **85. `preload_hunter_scan_process_env`** : Inspection unitaire de `/proc/[PID]/environ` découpée sur les octets nuls.

Section 15 — Notifications Bureau Asynchrones (`raptor_notify.c`)

- **86. `raptor_notify_send_desktop`** : Émission de la bulle Freedesktop via double `fork()` sans processus zombie.
- **87. `raptor_notify_evaluate_threats`** : Filtre anti-rebond (30s) et arbitrage lors des transitions d'alertes.

9.2. Règles de Gouvernance pour le Développement Futur

Toute extension ou nouveau greffon intégré à RAPTOR doit respecter scrupuleusement la charte d'ingénierie du projet :

1. **Modularité Stricte par Greffons (`RaptorPlugin`)** : Aucun outil d'analyse ou d'export ne doit être codé directement dans la boucle graphique principale `ui_engine.c`. Tout module s'enregistre via la macro officielle `RAPTOR_PLUGIN_INIT`.
2. **Indépendance des Fichiers de Configuration** : Zéro chaîne d'adresses IP, de domaines ou de noms d'exécutables codée en dur dans le langage C. Tout élément configurable réside dans `assets/rules/`.
3. **Absence Totale de Télémétrie** : Le code ne doit émettre aucune requête vers des serveurs d'analyse tiers, des régies publicitaires ou des métriques cloud. La souveraineté des données de l'hôte audité est absolue.

4. **Zéro Régression de Performance** : L'interface doit maintenir en permanence 60 FPS constants sans saccade visuelle. Tout traitement bloquant (réseau, DNS, lecture disque lourde) doit être déporté sur un thread secondaire ou asynchrone.
5. **Documentation Systématique en En-Tête** : Chaque script C ou en-tête doit comporter son bloc descriptif standardisé précisant le nom du fichier, le numéro de révision, la responsabilité système et la liste numérotée exhaustive de ses fonctions.