



Analyse de malware avec r2ai et MCP

Linux/Symbiote et BPFDoor en 2025

Axelle Apvrille

Cyb'Air, Novembre 2025

Présentation



- Chercheur Anti-Virus chez **Fortinet** (17 ans +)
- Rétro-ingénierie de malware sur téléphone mobile, objets connectés et **Linux**
- Organisatrice de **Ph0wn CTF** à Sophia Antipolis, le CTF pour le “smart” : objets connectés, téléphone et IA.
- @cryptax (Blue Sky, Mastodon, Discord)

Pico le Croco →



Contexte Anti-Viral

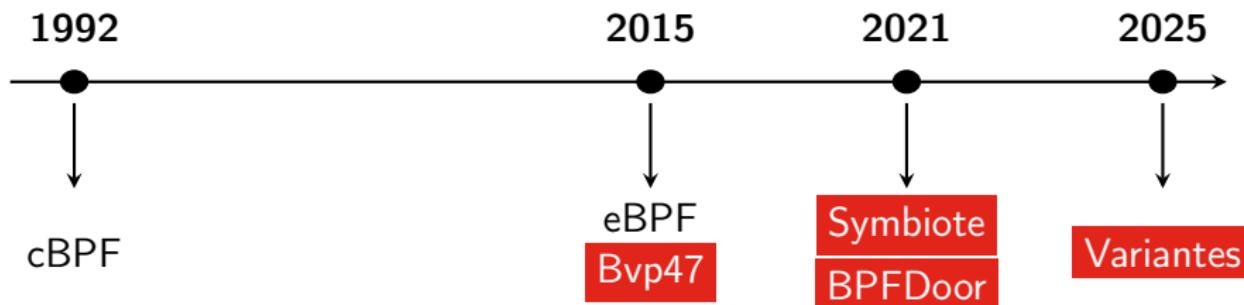


- 2 heures pour détecter (ou non) un nouveau malware.
- 10,000 nouveaux malware Android par jour.
- 20,000 sur Linux par mois (600-700 par jour).

Pourquoi détailler une analyse de malware ?

- Améliorer la détection de variantes.
- Trouver les nouveaux sites où les malwares seront dropés.
- Aider à arrêter les cybercriminels : surveillance du C2, *sinkhole* (dérivation)...
- Contribuer à améliorer la sécurité du développement, API, OS.
- L'analyse détaillée dure généralement entre 2 et 40 heures.

Berkeley Packet Filter (BPF)



- Chargement par l'utilisateur de petits programmes, tournant dans le noyau Linux en bac à sable.
- Sert à modifier les paquets réseau, mais plus : appels système, trace, perf...

Chargement + instructions BPF

```
const char *filePath = "bpf_prog.o";
struct bpf_object* bpfObject;
struct bpf_program* prog;

// bpf_prog.o contient des instructions eBPF
bpfObject = bpf_object__open_file(filePath, NULL);
if (!bpfObject) { ... }
// chargement
err = bpf_object__load(bpfObject);
if (err) { ... }
prog = bpf_object__find_program_by_name(bpfObject, "detect_execve");
if (!prog) { ... }
bpf_program_attach(prog);
```

Instructions BPF 🔪

- BPF_ADD, BPF_SUB, BPF_OR, BPF_JEQ, BPF_LD...
- Taille fixe : 64 bits
- Supporté nativement par Radare2 et Binary Ninja.



Linux/Symbiote

- Novembre 2021 - maintenu et actif en 2025
- Cible le secteur financier latino-américain
- Utilise un filtre BPF **MODIFIE** en 2025

Backdoor
Hook de
pam_authenticate

Keylogger
Hook de read

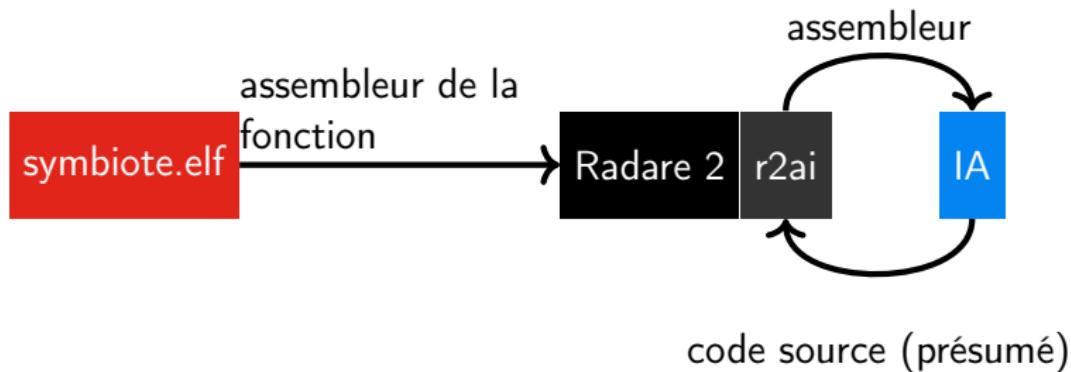
RC4

BPF Socket Filter



Demo : désassemblage par IA

- ① Utilisation de **Radare2** avec plugin IA : **r2ai**
- ② Récupérer des clés d'API
- ③ Décompiler la fonction courante : **r2ai -d**



Linux/Symbiote : la fonction apply_filter

```
void apply_filter(int fd, int opt_level, int third_param, void* data,  
→   int fifth_param, int filter_size) {
```

Un filtre de $352 = 0x160$ octets

```
memcpy(filter_buf, "filter", 0x160);
```

```
syscall(0x36, fd, opt_level, third_param, &size, fifth_param);
```

- Appel de $0x36 = 54$ setsockopt(). Vérifiez ici [🔗](#)
- Les autres arguments de l'appel système sont fournis par les arguments de apply_filter.



apply_filter() est appelée par setsockopt()

```
void setsockopt(int sockfd, int level, int optname, void *optval,  
→ socklen_t optlen) {  
    int result;  
    result = syscall(0x36, sockfd, level, optname, optval, optlen);  
    → setsockopt  
    if (result < 0) {  
        int *errno_location = __errno_location();  
        *errno_location = ~result;  
    } else if (optname == 0x1a) { SO_ATTACH_FILTER  
        syscall(0x36, sockfd, level, optname, optval, optlen);  
        → setsockopt  
        int *errno_location = __errno_location();  
        *errno_location = ~result;  
    }  
}
```



setsockopt() appelle setsockopt() ??? pourquoi pas apply_filter() ?



Retour à l'assembleur



| | | |
|------------|----------|-----------------------|
| 0x00002cca | 89de | mov esi, ebx |
| 0x00002ccc | 89c7 | mov edi, eax |
| 0x00002cce | e887ffff | call sym.apply_filter |

- ① Appel du *vrai* appel système `setsockopt()`
- ② Si `optname = SO_ATTACH_FILTER`, le malware appelle aussi `apply_filter()`, qui ajoute un filtre BPF !

Vérifiez tout ce qui a de l'importance



Désassembler les instructions BPF avec Radare2



```
[0x00004540]> s obj.filter
[0x00004540]> p8 0x160
280000000c0000001500000fdd8600003000000...
[0x00004540]> e asm.arch=bpf
[0x00004540]> pd 30
 0x00004540      280000000c..    ldh [0xc]
 0x00004548      1500000fdd..   jeq r0, 0x86dd,
  ↳ +0xf00
 0x00004550      3000000014..   ldb [0x14]
 0x00004558      1500020084..   jeq r0, 0x84, +0x2
 0x00004560      1500010006..   jeq r0, 0x6, +0x1
 0x00004568      1500002611..   jeq r0, 0x11,
  ↳ +0x2600
 0x00004570      2800000036..   ldh [0x36]
```

Ca veut dire quoi ?

ldh [0xc]

jeq r0, 0x86dd, +0xf00

ldb [0x14]

jeq r0, 0x84, +0x2



Demandons à Claude*

AA This is the packet filter bytecode. Probably it was decoded wrong. Can you decode it?
(eBPF)

```
[0x00002b5a]> pxQ 352 @ obj.filter  
0x00004540 0x0000000c00000028  
0x00004548 0x000086dd0f000015  
0x00004550 0x0000001400000030  
0x00004558 0x0000008400020015  
0x00004560 0x0000000600010015
```

...
Show more

*Anthropic



Demandons à Claude*

What This Filter Does:

This is a **port-based packet filter** that:

1. Accepts **IPv4 and IPv6** packets
2. Filters **TCP, UDP, and SCTP** protocols
3. Matches packets destined to specific ports:
 - **54778, 58870, 59666, 54879, 57987, 64322, 45677, 63227**
4. **Accepts matching packets, drops others**

What This Suggests:

These are **unusual high-numbered ports** (all above 45000). This could be:

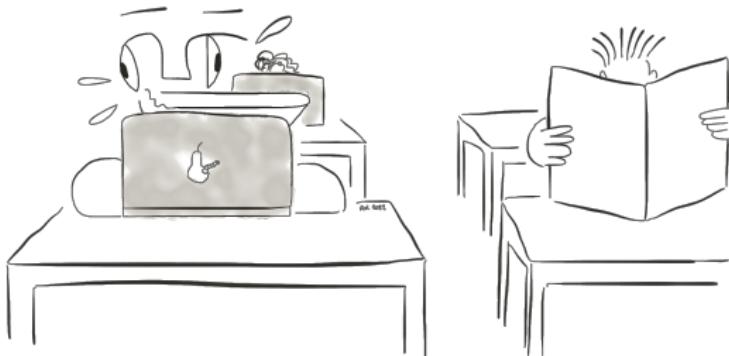
- **P2P application** (BitTorrent, custom protocol)
- **Game server** or streaming application
- **Covert communication channel** using non-standard ports
- **Malware C2** (command & control) traffic
- **Port-knocking sequence** or similar security mechanism

The specific ports don't match common services. Would you like me to check if these ports are associated with known applications?

*Anthropic



Communiquer - sans se faire pincer



- Communication du botnet sur des hauts ports.
- Plus de ports qu'en 2021 : **port hopping**.
- Ajout du protocole **UDP** : pas de connexion, moins de traces.

Quelques erreurs fréquentes de l'IA

- Confusion entre *paquet perdu* et *paquet sélectionné*.
Opposé!!!
- Prétendu filtrage sur le *port 2048*. Mauvaise compréhension des instructions BPF.
0x0800 = identifiant pour paquet IPv4. Pas de filtrage du port 2048.

Les hallucinations, erreurs ou omissions (importantes) sont présentes **chaque fois**.

Il faut toujours vérifier ce qui a de l'importance.



Linux/BPFDoor

- Version de Juin 2025
- 150 nouveaux échantillons en 2025
- Attribué à un groupe chinois 🔒

Références

- Analyse du malware en 2021 🔒
- Une version d'avril 2025 🔒



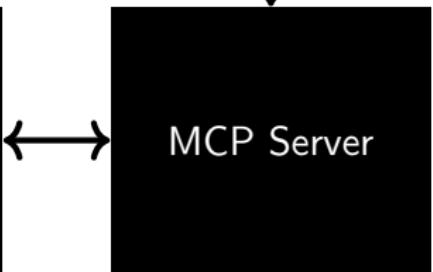
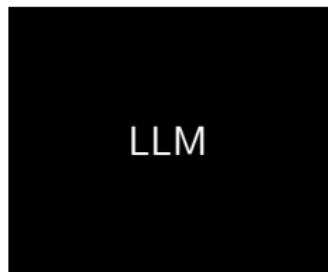
MCP : comment ça marche

qwen, claude sonnet...



MCP : comment ça marche

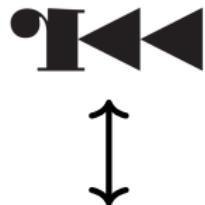
qwen, claude sonnet...



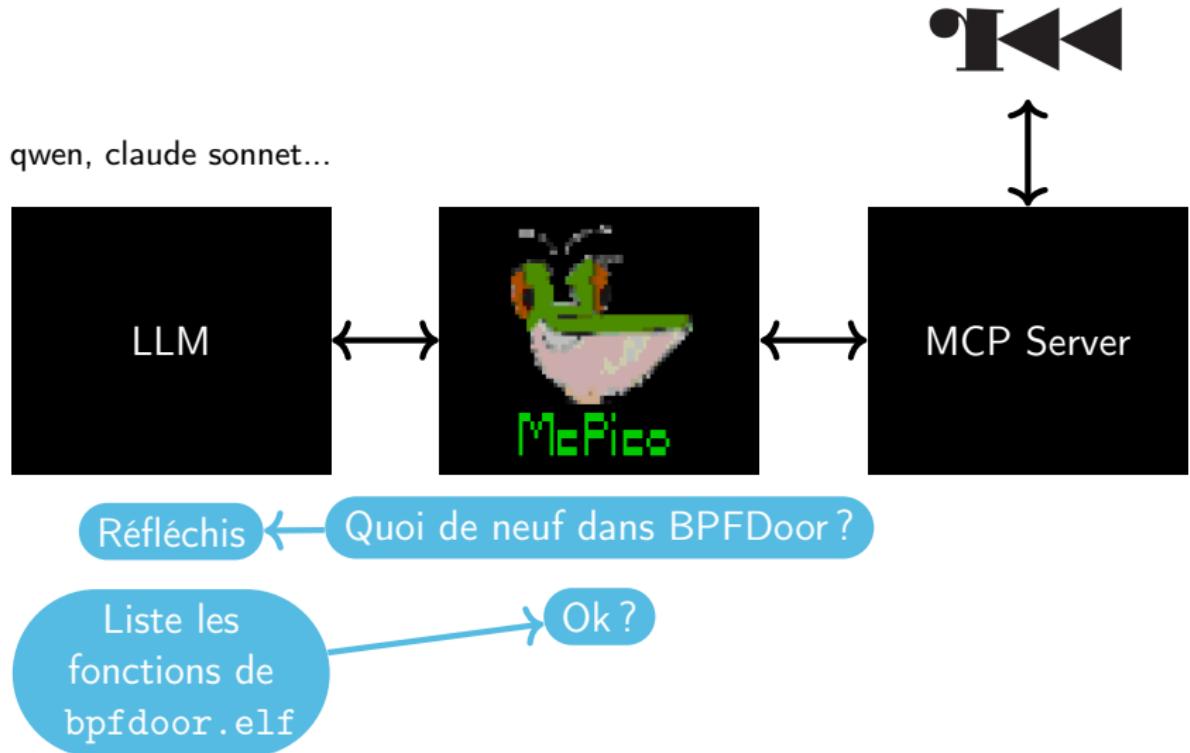
Quoi de neuf dans BPFDoor ?

Réfléchis

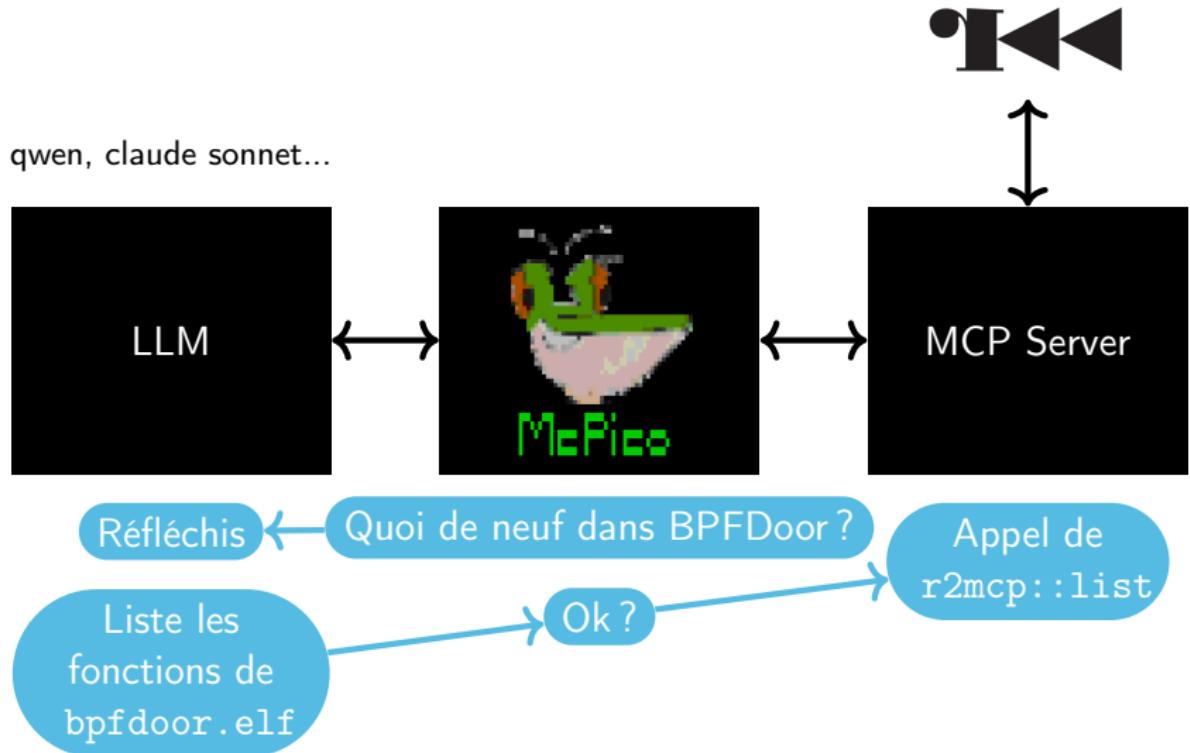
Liste les fonctions de bpfdoor.elf



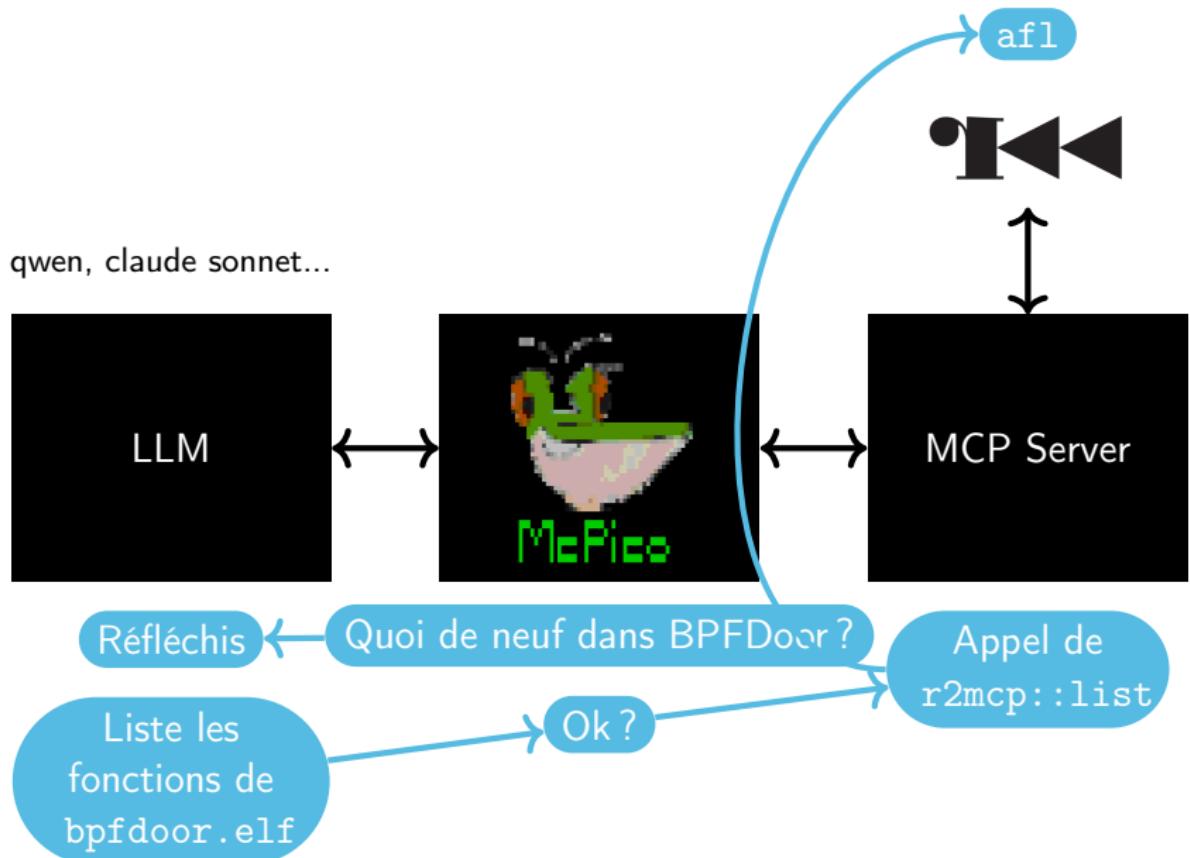
MCP : comment ça marche



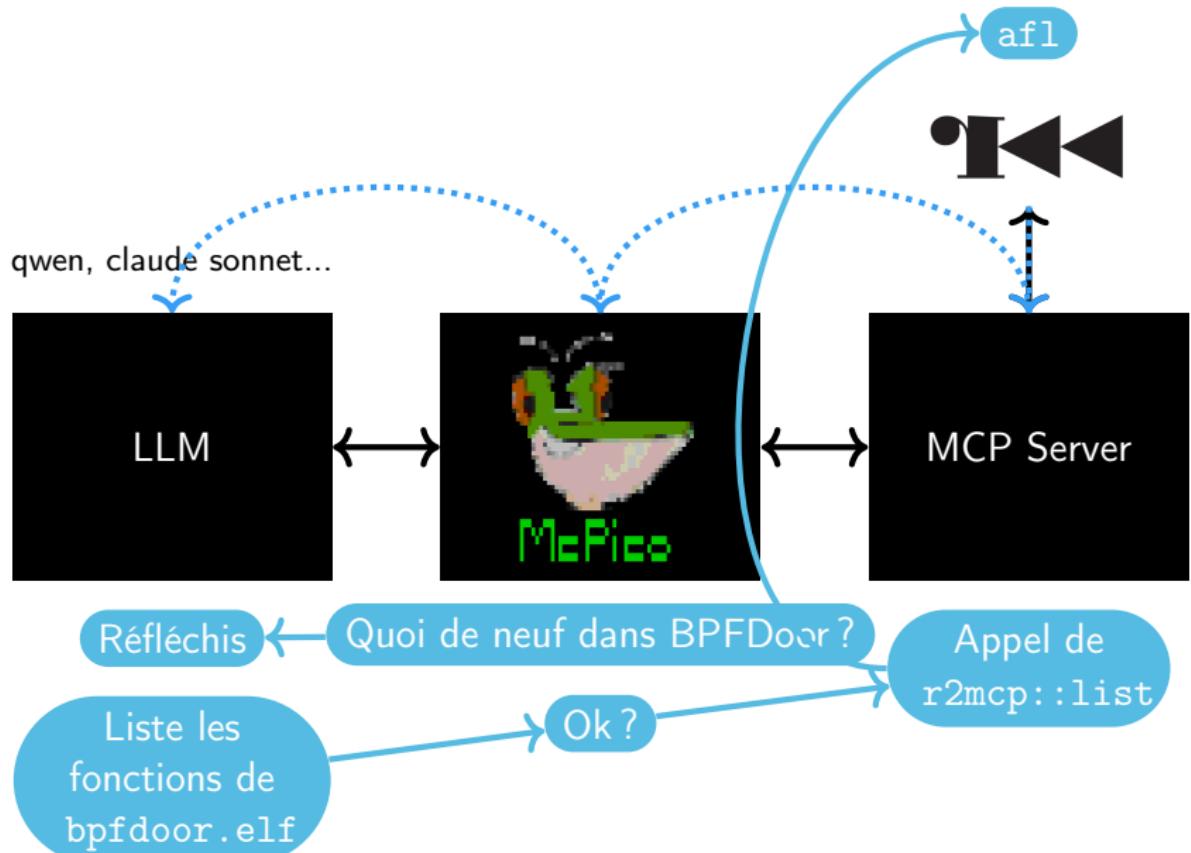
MCP : comment ça marche



MCP : comment ça marche



MCP : comment ça marche

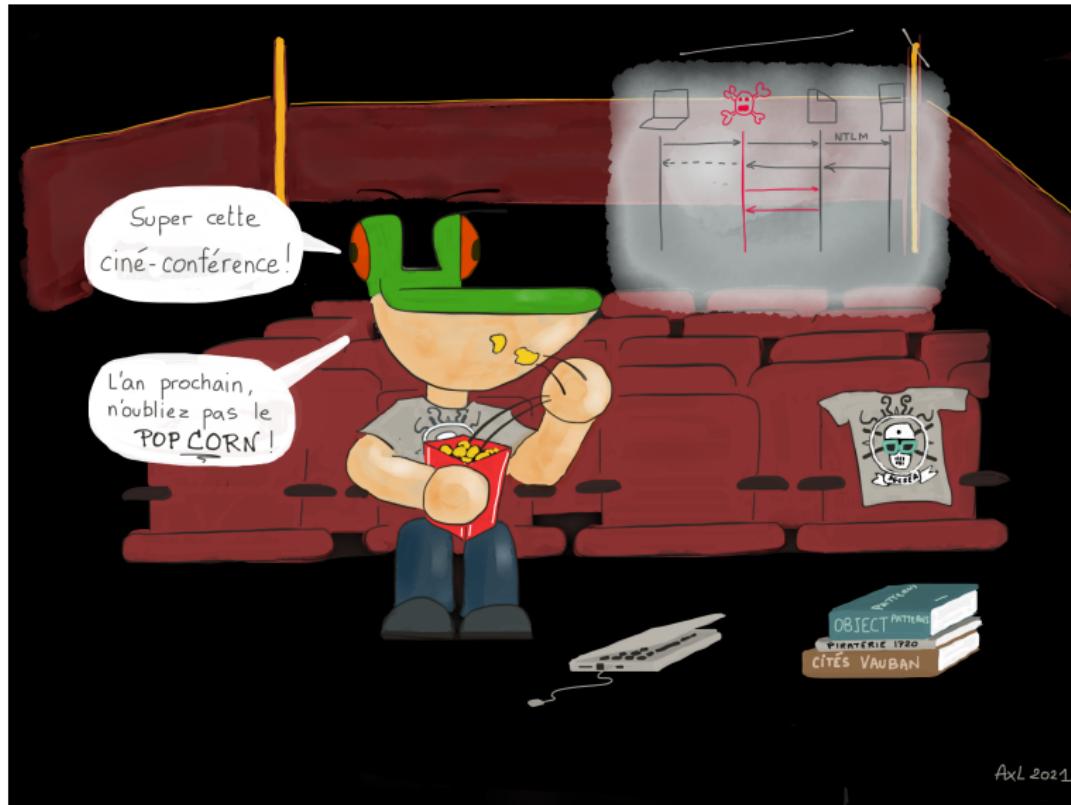


Petite précision d'importance...

- On n'envoie **PAS** le malware.
- On envoie des portions désassemblées par Radare2.
- Si on utilise une IA locale, on *n'envoie rien du tout*.
- On **n'exécute PAS** le malware (heureusement).
- Il faut quand même vérifier les commandes r2 que l'IA **veut lancer**. Approbation / modification de l'utilisateur. *On ne sait jamais, ex : mode debug.*



Demo



Résultats

| Fonction trouvée par l'IA | Mon commentaire |
|---------------------------------|--|
| Filtre BPF | Oublie de signaler que le filtre est différent |
| Reverse Shell | ✓ |
| Magic pattern X (nouveau) | ✓ |
| Port par défaut 44448 | A vérifier |
| Messages d'erreurs explicites | Pas important |
| Informations du compilateur GCC | Pas important |

Bon début, mais je voulais les **différences fonctionnelles**



2ème itération

| Feature | 2022 Version (Article) | 2025 Sample (bpfdoor.elf) | Status | Notes |
|----------------------|--|--|---|--|
| BPF Packet Filter | IPv4 only (0x800) | IPv4 + IPv6 (0x800, 0x86dd) | ✓ Enhanced | Major upgrade - now supports IPv6 traffic monitoring |
| Magic Packet Values | UDP: 0x7255 ICMP: 0x7255 TCP: 0x5293 | 'X' (0x58) character + colon delimiter | ⚠ Changed | Completely different packet identification mechanism |
| Packet Protocols | ICMP, UDP, TCP | ICMP, UDP, TCP (via AF_PACKET) | ✓ Same | Still monitors all three protocols |
| RC4 Encryption | Present | Present (sym.rc4 @ 0x0000169c) | ✓ Present | Encryption layer still used |
| Reverse Shell | Yes (connect-back) | Yes (sym.reverse_shell @ 0x00401562) | ✓ Present | Connects back using /bin/sh |
| Process Masquerading | 10 fake process names | None found | ✗ Removed | No masquerading strings in binary |
| File Location | /dev/shm/kdmtmpflush | Not present | ✗ Removed | No /dev/shm hardcoded paths |
| PID Dropper | /var/run/haldrund.pid | Not visible | ✗ Removed | A vérifier No dropper file references |
| Timestomping | Yes (Oct 30 2008 date) | Not visible | ✗ | No timestamp code found |

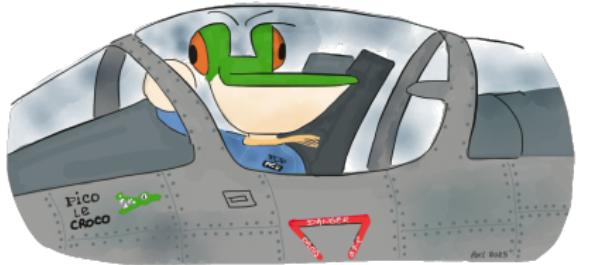


(Ma) Conclusion

- L'IA fait gagner beaucoup de temps (même avec les erreurs).
- Les coûts sont faibles (pour le reverse), les problèmes de confidentialité sont gérables.
- Difficilement utilisable par un novice. *Problème récurrent de l'enseignement et de l'apprentissage avec IA...*
- L'IA sait convaincre. Résistez et vérifiez tous les propos qui ont de l'importance.



← Non ! Ne lui faites pas confiance !



Thank You

- <https://github.com/radareorg/radare2>
- <https://github.com/radareorg/r2ai>
- <https://github.com/radareorg/radare2-mcp>
- <https://github.com/cryptax/mcpico>
- Télécharger la présentation :
<https://www.fortiguard.com/events>
- Ph0wn CTF - Sophia Antipolis : 13-14 mars 2026

