TD Ransomware

Aujourd'hui, le temps d'un TD, vous allez vous glisser dans la peau d'un *black hat*. Vous allez faire le mal, mais du mieux que vous pouvez. A la clef, de grosses rançons et une retraite anticipée assurée. Sur une île paradisiaque ou dans la fraîcheur d'une maison d'arrêt, cela dépend avant tout de vous !

Avant de pouvoir réaliser des prises d'otages numériques, vous devez créer l'arme qui vous servira à réaliser vos braquages : un ransomware. En Français : Rançongiciel, mais ça c'est uniquement pour le fan club de l'Académie Française.

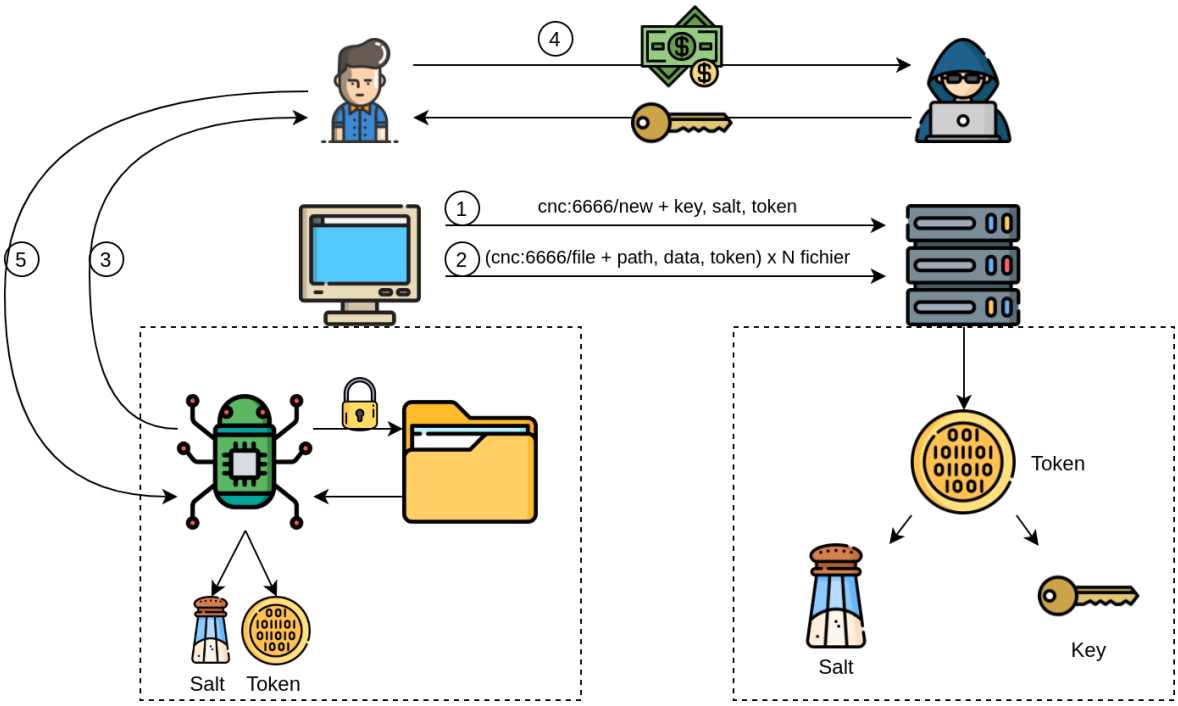
Un ransomware est un malware - contraction de malicious software - donc le but est de bloquer un système d'information. Pour en retrouver l'usage, le propriétaire doit payer un rançon. En supplément, depuis quelques années, les données sont exfiltrées; ainsi en cas de sauvegarde et de restauration, pour éviter la rançon, les attaquants disposent d'un levier supplémentaire : le chantage. "Si vous ne payer pas, on divulgue vos secrets". Dans les deux configurations, votre victime est assurément dans une situation inconfortable !

Certaines variantes plus simple vont venir bloquer le MBR ou encore verrouiller l'accès à l'OS.

L'objectif de ce TD est de coder un ransomware et le CNC (voir plus loin).

Comment ça fonctionne ?

Ce malware est la phase finale d'un attaque, à savoir l'exécution de la charge active. Le ransomware va échanger avec un CNC (C&C, command and control) qui sera ici un serveur http exposant une API Rest. Le CNC est stratégique pour l'attaquant. Voici le schéma des échanges entre l'utilisateur, le système, le malware et le CNC :



Lors de l'exécution du malware, il va dans un premier temps (1) créer trois données importantes d'un point de vue cryptographique :

- une clef privée, servant au chiffrement/déchiffrement

- un sel, permettant la création d'un token à partir d'une KDF
- un token, servant d'identifiant unique, à partir de la clef et du sel.

Ces trois informations suivantes sont envoyés au CNC; En local, seul le token et sel sont stocké : il ne faudrait pas que la victime ai la clef privé lui permettant de déchiffrer ses données !

Une seconde étape (2), optionnelle, est de copier vers le CNC toutes les informations qui seront chiffrées. Cela vous assure un moyen de pression suplémentaire pour arriver à vos fins.

Chiffrer et informer (3) : c'est le début de la catastrophe pour votre victime mais aussi le début de votre fortune. Toutes les données sont maintenant chiffrées et un petit message laconique informe le client de ce qu'il doit faire. Envoyer un mail à une adresse particulière avec le token afin de réaliser le payement. Au fond, c'est pas différent des impôts avec votre numéro fiscal ?

Après des échanges entre vous et la victime, qui ne pouvait pas payer ce que vous lui demandiez, vous avez trouvez un compromis. Il vaut mieux en effet être payé moins que pas du tout. De toute façon, il sera toujours temps de revenir l'année prochaine. Toujours comme les impôts, d'ailleurs. Après payement, vous lui faite parvenir la clef de déchiffrement (4) en lui expliquant avec beaucoup de bienveillance comme retrouver ses données. Après tout, vous n'êtes pas un monstre mais un businessman.

La clé saisie, le ransomware libère les fichiers et supprime ses traces sur le systèmes (5).

Les règles du jeu

Oui c'est un TD, mais c'est aussi un jeu, avant tout. Vous ne serez pas payé en 'coins mais avec des points. C'est bien aussi ?

Pour maximiser vos gains, vous devez respecter un certain nombre de règles :

- Soyez claire dans votre code et vos commentaires. Et oui, votre code doit être commenté. Dit autrement, si je ne peux pas comprendre votre code, je ne peux pas vous mettre de point
- Respectez les coding rules
- Je vous fourni un ensemble de template, avec des fonctions à trou. Normalement, vous n'avez pas besoin d'en écrire d'autre (sauf bonus). Vous être libre de le faire mais utiliser les notions pour le typage (ex : `def func(a:int, b:str)->list:`)
- Expliquez ce que vous faite et pourquoi : le readme est là pour ça
- Vous êtes un hacker dans l'âme, vous avez fait tout le TD en 1h chrono, et vous voulez le faire savoir ? Prenez des initiatives, ajouter des fonctions qui vous feront gagner plus d'argent (ou de point) ... et bien sûre expliquez-les (CF bonus).

En prérequis vous devez :

- Avoir une machine physique ou une VM base Debian (ubuntu, mint, kali, etc)
- Avoir docker d'installer
- Connaître les bases de python
- Avoir un compte github

/!\ IMPORTANT /!\ :

Une fois le TD fini, vous pouvez m'envoyer le lien de votre repository par mail avec comme titre « [XA-YYYY] TD Ransomware prenom nom », ou X est votre promo (ex: 5A) et YYYY est l'année courante. Votre repository devra porter le nom td-ransomware-ABB, où A est la première lettre de votre prénom et BB les deux premières lettres de votre nom. Un date limite de retour vous sera donnée : passé ce délai, plus aucun mail et plus aucun commit ne sera pris en compte. Soyez à l'heure.

Le non respect de ces règles entraîne un zero !

Coding rules

Le code devra respecter un certain nombre de règles (arbitraires) pour ne pas perdre de point :

- Le code doit être commenter (j'insiste)
- Les noms des fonctions et de variables s'écrivent en en snake case (nom_de_fonction)
- Les noms de constantes s'écrivent en snake case majuscule (NOM_DE_CONSTANTES)
- Les noms de classe s'écrivent en camel case (MaClass)
- Pas de valeurs littérales dans le code, utilisez toujours des constantes pour les expliciter
- Une classe, un fichier
- Une fonction ne doit pas dépasser 20 lignes
- Pas plus de deux imbrications de bloc
- Utilisez des variable intermédiaires lors d'enchainement de fonction. Exemple à ne pas faire : x = y(z(w(g,h,j,k)))
- Les noms (variables, fonctions, etc.) doivent avoir du sens (voir règle précédente)
- Utilisez les log (self._log) plutôt que les print() si c'est possible
- Factoriser votre code : si un bout de code est utilisé plusieurs fois, faites en une fonction.
- Utilisez Github

Action !

Github

Tout professionnel, même black hat, utilise git pour son code. Et ce TD ne va pas y déroger. Vous allez utiliser github pour publier votre code (repo public) et le `README` vous servira à répondre aux questions.

Il est recommandé de publier sa clef public ssh pour pouvoir faire les opérations sur le repo. Si vous n'avez pas de clef, vous pouvez taper la commande `ssh-keygen -b 511 -t ed25519`, en laissant le mot de passe vide. Pour le reste, RTFM :)

Les commandes qui vous serviront :

- git clone
- git add
- git commit
- git push

Si vous avez visual code, vous pouvez ajouter le module gitlens pour tout faire sans une ligne de commande.

Dans le `README`, n'oubliez pas de préciser la question à la quelle vous répondez ! Le `README` est en markdown, profitez-en pour soigner la présentation.

Votre premier commit doit consister à ajouter les fichiers joins à ce PDF. Faite régulièrement des petits commits et décrivez ce que vous faite : en cas de problème vous pourrez revenir à une itération précédente (git revert).

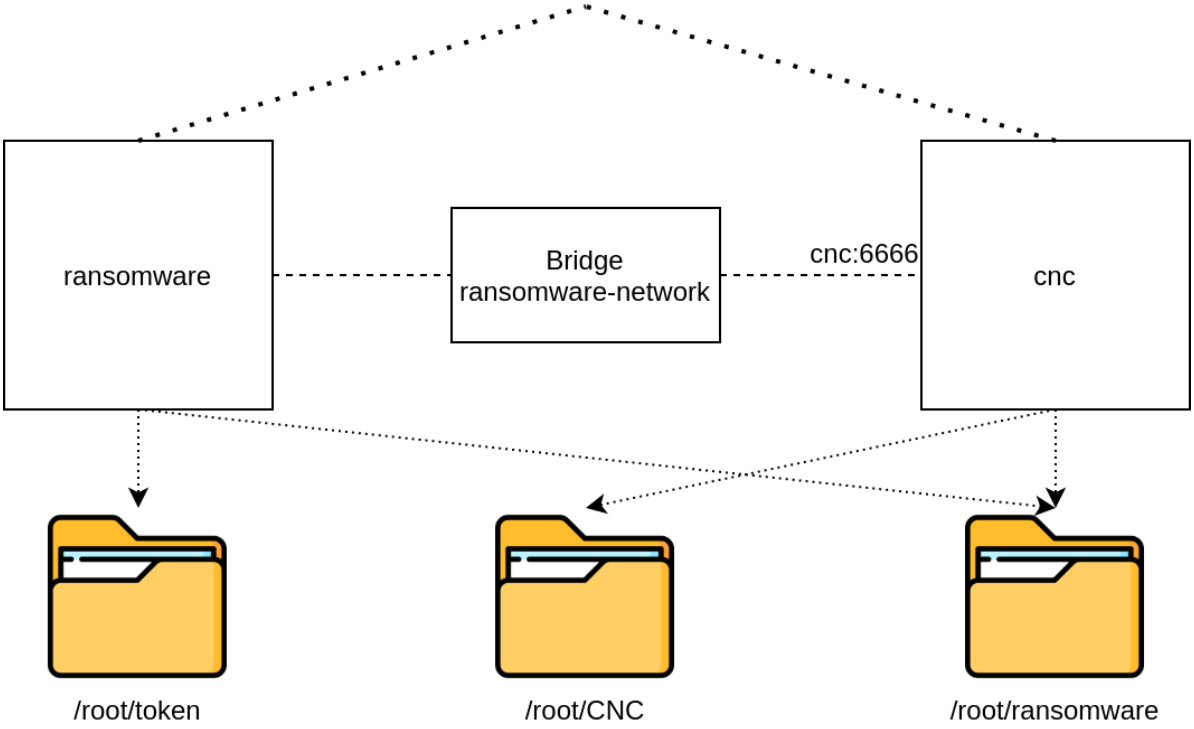
Environnement

Un ransomware, comme tous les malwares, sont des logiciels **DANGEREUX**. En effet, ils sont par définition envahissants et destructeurs. Pour éviter tout risque à vos machines, vous allez travailler dans un environnement virtualisé léger : Docker

image : ransomware

```
FROM python:3

RUN apt update && apt install -y python3-pip
RUN pip3 install cryptography requests
```



A partir d'un `Dockerfile`, on va créer une image `ransomware` qui va nous servir à instancier le CNC et la machine de la victime. Les deux conteneurs seront interconnectés sur un réseau bridge `ransomware-network`. Grâce au serveur DNS inclus dans docker, il sera possible de résoudre le nom "cnc" (i.e trouver l'IP du conteneur à partir de son nom de domaine). Enfin, pour permettre la persistance des informations qui nous intéresse - et facilité le développement - on montera deux volumes dans chaques conteneurs :

- sources sera monté dans chaque conteneurs et permettra d'exécuter les sources depuis un environnement *sandboxé*
- token contiendra les informations cryptographiques chez la victime
- CNC contiendra les informations cryptographique de l'attaquant (dont la clé privée)

La construction de l'image se fait via le script `build.sh` et le fichier `Dockerfile`, de même que le réseau `ransomware-network`. L'exécution du CNC se fait via le script `run_cnc.sh`; De même, l'exécution du ransomware en one-shot se fait avec `run_ransomware.sh`. Toutefois il peut être intéressant de rester dans l'environnement de la cible pour pouvoir observer les effets ou débugger : on utilisera alors `exec_target.sh` (il ne faut pas exécuter `run_ransomware.sh` et `exec_target.sh` en même temps !). De part l'environnement virtualisé, vous ne devez rien installé sur votre machine, tout est inclus dans Docker.

Chiffrement

Dans le repertoire `sources` se trouve un fichier `xorcrypt.py` :

```python
from itertools import cycle


def xorcrypt(data:bytes, key:bytes)->bytes:
    # encrypt and decrypt bytes

    # Loop the key (abc become abcabcabcab....)
    infinite_key = cycle(key)
    # create couple from data and key.
    match = zip(data, infinite_key)
    # XOR key and data
    tmp = [a ^ b for a,b in match]
    # return encrypted or decrypted data
    return bytes(tmp)

def xorfile(filename:str, key:bytes)->bytes:
    # encrypt and decrypt file

    # load the file
    with open(filename, "rb") as f:
        plain = f.read()

    # Do the job
    encrypted = xorcrypt(plain, key)

    # write the result on the same file
    with open(filename, "wb") as f:
        f.write(encrypted)
```

Q1 : Quelle est le nom de l'algorithme de chiffrement ? Est-il robuste et pourquoi ?

Trouver les fichiers

Écrire la methode `get_files(self, filter:str)->list` dans la classe `Ransomware` afin d'être en mesure de récupérer tout les fichiers donc l'extension match le filtre. Dans le cadre de notre TD, on souhaites trouver tous les fichiers `*.txt` avec leur chemin absolue. La sortie attendu est une liste de string.

Tips : La bibliothèque standard `Path` contient un foncttion `rglob`.

Génération des secrets

L'efficacité d'un ransomware repose en partie sur la sécurité de ses secrets. Bien sûre, différentes approches existent mais nous allons ici rester simple. Un sel et une clef privée doivent être générée à partir d'une source aléatoire cryptographique. A l'issue, ces deux données doivent être hachées avec la fonction `PBKDF2HMAC`. Implémentez la méthode `do_derivation(self, salt:bytes, key:bytes)->bytes` et la méthode `create(self)->Tuple[bytes, bytes, bytes]` dans la classe `secret_manger`.

Tips : Le rédacteur du TD à négligemment oublié de retirer les imports. Vous avez donc directement les bonnes lib/fonctions de disponibles.

D'un côté, il doit être facile de trouvé un exemple avec `PBKDF2HMAC` et `hashes.SHA256` pour faire une dérivation de clef; D'un autre côté, la bibliothèque standard `secrets` fait des miracles avec `token_bytes`. N'oubliez pas d'utiliser les constantes déjà définies !

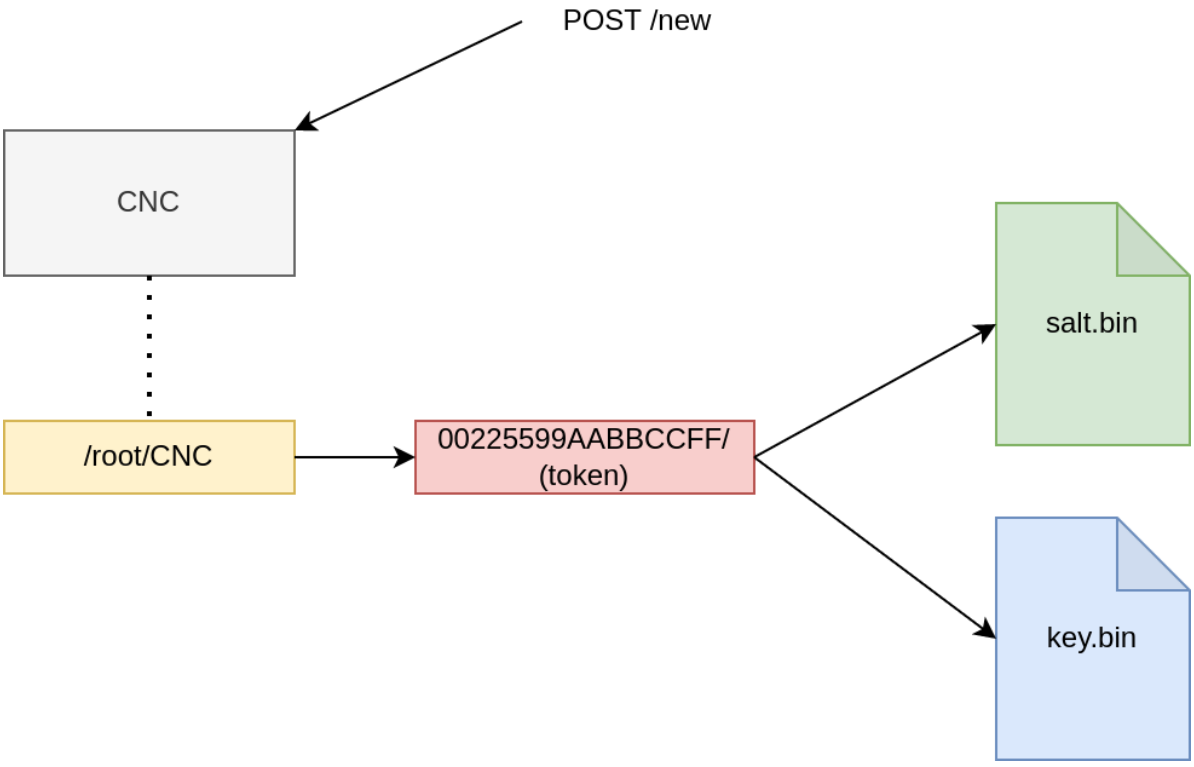
Q2 : Pourquoi ne pas hacher le sel et la clef directement ? Et avec un hmac ?

Enrollement

Il vous faut maintenant faire parvenir à votre CNC tous les éléments cryptographiques. Vous devez donc coder la fonction `post_new(self, salt:bytes, key:bytes, token:bytes)->None` dans la classe `SecretManager`. Pour cela vous pouvez utiliser la fonction `post` de la bibliothèque `requests` pour envoyé le json suivant :

```
{
    "token" : self.bin_to_b64(token),
    "salt" : self.bin_to_b64(salt),
    "key" : self.bin_to_b64(key)
}
```

La réciproque de cette methode doit être réalisée du côté du CNC avec une fonction `post_new(self, path:str, params:dict, body:dict)->dict`. Elle doit créer un répertoire à partir du token (ex: sha256) et y stocker le sel et la clef.



Si vous êtes observateurs, vous avez sûrement remarqué que l'on faisait un **post** vers une url **new** et que la fonction à créer dans la classe CNC s'appel **post_new**. Cela pourrait peut être vous servir plus tard. Concernant les arguments, voici leur explications :

- path : chemin complet de l'url. Il serait possible (mais inutile) d'avoir une url comme /new/foo/bar
- params : dictionnaire contenant des paramètres et leur valeurs passées en paramètre. Pas forcement utile ici.
- body : contient le json - la payload - envoyé par le client. C'est ici que l'information se trouve.

La méthode peut retourner un dictionnaire, même vide, ou None (convertie en dictionnaire vide).

Il faudra être vigilant avec le json qui ne supporte pas les données binaires. Pour éviter les problèmes, vous devez sûrement encoder/décoder en base64.

Setup

Le setup est constitué de la création des éléments cryptographiques, de leur sauvegarde en local et de l'envoi au CNC. Complétez la fonction `setup(self)->None` de la classe `SecretManager`.

Les fichiers token et salt seront respectivement token.bin et salt.bin dans le repertoire self._path

Tips : le chemin */root/token* n'est peut être pas présent et nécessite peut être d'être créé si ce n'est pas le cas.

Q3 : Pourquoi il est préférable de vérifier qu'un fichier token.bin n'est pas déjà présent ?

Chiffrement des fichiers

Implémentez la fonction `xorfiles` dans la classe `SecretManager`. A partir d'une liste de string (path vers le fichier), chiffrer les fichiers avec la clef `self._key`.

Rendre le token affichable

Le token est une donnée binaire de base. Comme proposé dans la phase d'enrollement, vous pouvez hacher le token avec du sha256.

Implémentez la fonction `get_hex_token(self)->str`.

Encrypt !

C'est le grand moment que vous attendez : implémentez la fonction `encrypt` de la classe `Ransomware`. Cette fonction va :

- Lister les fichiers txt
- Créer le SecretManager
- Appeler setup()
- Chiffrer les fichiers
- Afficher un message permettant à la victime de contacter l'attaquant avec le token au format hex.

L'étape d'après, c'est de partir avec la caisse … ou pas. Ce n'est que la moitié du travail : si vous ne pouvez pas déchiffrer votre œuvre. Vous devez en effet échanger la réparation de votre œuvre contre monnaie sonnante et trébuchante; Sinon, rapidement vous aurez une mauvaise réputation et personne ne vous payera plus.

Pour valider votre travail, lancez le CNC dans un terminal, et dans un autre le ransomware. Vous devez voir les éléments crypto sur le CNC et avoir le message chez la victime. Pour être sûre, lancez l'environnement de la cible (exec_target.sh), exécutez le ransomware et vérifiez sur un fichier texte qu'il est bien illisible.

Charger les éléments cryptographiques

Une fois le ransomware exécuté, il faut permettre à la victime d'insérer la clef que vous avez échanger contre votre cryptomonnaie préférée.

En premier lieu, il faut que la classe `SecretManager` puisse charger self.*salt et self.*token depuis les fichiers locaux. Implémenter la méthode `load(self)->None`.

Vérifier et utiliser la clef

Une fois les éléments cryptographiques chargés, il est possible de charger une clef qui serait fournie par la victime. Mais avant de définir la variable self._key de SecretManager, il est impératif de la valider, sans quoi les fichiers seraient déchiffré avec la mauvaise clef …

La clef sera fournie en base64 (c'est une clef binaire originalement). Si la clef n'est pas bonne, vous devez levez une exception.

Q4 : Comment vérifier que la clef la bonne ?

Implémentez `check_key(self, candidate_key:bytes)->bool` et `set_key(self, b64_key:str)->None`

Mr Propre

« On fait le mal mais on le fait bien » © : implémenter la fonction clean pour supprimer les éléments cryptographiques locaux.

Decrypt

Cette fonction va venir charger les éléments cryptographiques locaux et la liste des fichiers qui ont été chiffré. Dans un second temps, dans un boucle avec un try/except, on va réaliser les actions suivantes :

- Demander la clef,
- appeler set_key,
- appeler xorfiles sur la liste de fichier,
- appeler clean,
- afficher un message pour informer que tout c'est bien passé,
- sortir du ransomware

En cas d'echec :

- Afficher un message indiquant que la clef est mauvaise,
- recommancer la boucle.

Votre victime est soulagé d'avoir retrouvé ces précieux README et vous, vous avez maintenant les poches pleines de points. Mais vous en voulez plus ? Alors améliorez votre ransomware !

Bonus

Pour gagner des points, vous pouvez vous attaquez aux bonus. Ce sont des propositions : si vous avez d'autres [meilleurs] idées, allez y et maximisez vos profits !

Voler les fichiers

Une bonne politique de sécurité implique de faire régulièrement des sauvegardes, à chaud et à froid. Ce dernier point implique, par exemple, un disque dure USB donc hors d'atteinte. Cela casse donc votre modèle économique. Un bon moyen est de revendre à votre victime ses propres données : personne n'a envie de voir ses listing clients, sa compta ou les feuilles de payes être mis en place publique. Ou pire encore.

Une solution est d'ajouter une fonction `leak_files(self, files:List[str])->None` dans la classe `SecretManager`, devant envoyer les fichiers au CNC (ex : `post_file(self, path:str, params:dict, body:dict)->dict`).

B1 : Expliquez ce que vous faite et pourquoi

Chiffrement

Le chiffrement proposé est peut être ... perfectible.

B2 : Expliquez comment le casser et écrivez un script pour récupérer la clef à partir d'un fichier chiffré et d'un fichier clair.

Maintenant, il faut l'améliorer le chiffrement.

B3 : quelle(s) option(s) vous est(sont) offerte(s) fiable(s) par la bibliothèque `cryptographie` ? Justifiez

Implémentez votre solution.

Packer

Jusqu'à présent, vous avez utiliser le code directement. Mais ce n'est pas discret, pas pro, et cela implique d'avoir python et les bibliothèques d'installer. Pour éviter cela, on utiliser un packer, qui va produire un binaire *standalone*.

B4 : Quelle ligne de commande vous faut-il avec pyinstaller pour créer le binaire ?

B5 : Où se trouve le binaire créer ?

Dropper

On arrive rarement sur une machine avec directement le malware final; par de multiple rebond et mouvements latéraux, on arrive à lancer un Dropper qui va lui même aller chercher le malware en remote. Ce dernier est envoyé obfusqué, de façon à éviter les anti-virus (si si, un anti-virus ça se contourne très bien).

Ecrivez un dropper qui viendra demande au CNC le malware obfusquer (base64, xor, ...) qui l'installe et qui le lance. Compilez votre dopper comme le packet. Le binaire du ransomware se trouvera dans /root/CNC.

L'installation du binaire du ransomware peut être fait dans /usr/local/bin pour rendre le binaire accessible dans le path courant. Et n'oublier pas de le rendre exécutable !

Moyens de pression

Message au logging

C'est quand même sympa de rappeler à root qu'il s'est fait pwned et qu'il doit payer. Pour l'aider à payer, vous pouvez lancer `ransomware –decrypt`.

Tips : bashrc

Bloquer le terminal

Afficher la demande de payement, c'est pas mal, mais le mieux c'est qu'il ne puissent rien tenter de plus. « Catcher » les signaux (sigint, sigterm, etc) afin de lui interdire de quitter votre merveilleux logiciel.

Compte à rebours

Que serait une prise d'otage sans un compte à rebours ? Plusieurs choix d'offre à vous : doubler la rançon, lui dire que ses données seront détruites … à vous de voir.

Contact

Laurent MOULIN

lmoulin@spartan-conseil.fr