

RANSOMWARE STOP/DJVU : ANALYSE ET SIMULATION DE CHIFFREMENT

PRESENTED BY

Ben Hamou Mehdi



Sommaire

Introduction	2
Découverte de l'infection	2
Recherche et identification du ransomware	2
Décryptage des fichiers	3
Analyse Technique Approfondie de STOP/DJVU	5
Simulation Pédagogique du Ransomware STOP/DJVU.....	6
Conclusion.....	11

1. Introduction

Dans le paysage actuel de la cybersécurité, les ransomwares représentent une menace critique pour les particuliers, les entreprises et les institutions. Parmi eux, la famille STOP/DJVU se distingue par sa persistance et sa capacité à causer des dommages irréversibles en l'absence de contre-mesures appropriées.

Ce mémoire s'appuie sur une étude de cas concret : l'infection du disque dur externe de mon père par une variante de STOP/DJVU. Cet incident a servi de base pour approfondir mes connaissances en analyse malveillante et en techniques de récupération de données. À travers une approche méthodique, ce document détaille les étapes d'identification, de déchiffrement et d'analyse technique du malware, tout en proposant une simulation pédagogique pour illustrer ses mécanismes sous-jacents.

2. Découverte de l'infection

L'incident a débuté lorsque mon père a remarqué que tous ses fichiers sur un disque dur externe avaient subi une modification suspecte : leurs extensions avaient été remplacées par des combinaisons aléatoires telles que .zzla, .opqz, etc. Par ailleurs, un fichier texte intitulé _readme.txt était apparu, contenant un message exigeant le paiement d'une rançon en Bitcoin pour récupérer l'accès aux données.

En tant qu'étudiant en cybersécurité, j'ai immédiatement identifié les signes caractéristiques d'un ransomware :

- **Modification des extensions de fichiers (signe de chiffrement).**
- **Présence d'une note de rançon (indiquant une demande de paiement).**
- **Absence de fichiers accessibles (confirmation d'un chiffrement massif).**

Cette situation a constitué une opportunité pratique pour appliquer mes connaissances théoriques et explorer les mécanismes de ce type de malware.

3. Recherche et identification du ransomware

Pour confirmer la nature de l'attaque, j'ai soumis un fichier chiffré et la note de rançon à ID Ransomware, un service en ligne permettant d'identifier les souches de ransomware. Les résultats ont confirmé qu'il s'agissait bien d'une variante de STOP/DJVU, une famille active depuis 2018 et responsable de milliers d'infections mondiales.

STOP/DJVU opère selon deux modes distincts :

Version Online (Connexion réussie au C2)

- Le malware contacte un serveur de commande et contrôle (C2) pour générer une clé de chiffrement unique.

- Le déchiffrement sans la clé privée est quasi impossible, rendant la récupération des données très difficile sans payer la rançon.

Version Offline (Connexion échouée)

- Si le ransomware ne parvient pas à joindre le C2, il utilise une clé statique prédéfinie.
- Certaines de ces clés ont été reverse-engineerées par des chercheurs, permettant le développement d'outils de déchiffrement gratuits.

Critère	Mode Online	Mode Offline
Connexion C2	Réussie	Échouée
Clé AES	Unique, chiffrée avec RSA	Statique, stockée localement
Récupération	Très difficile (nécessite la clé privée)	Possible (si clé offline connue)

Dans ce cas précis, heureusement l'analyse a révélé que l'infection était de type Offline, offrant une possibilité de récupération des données sans payer la rançon.

1 Result

STOP (Djvu)

⚠ This ransomware may be decryptable under certain circumstances.

Please refer to the appropriate guide for more information.

Identified by

- sample_extension: .opqz

[Click here for more information about STOP \(Djvu\)](#)

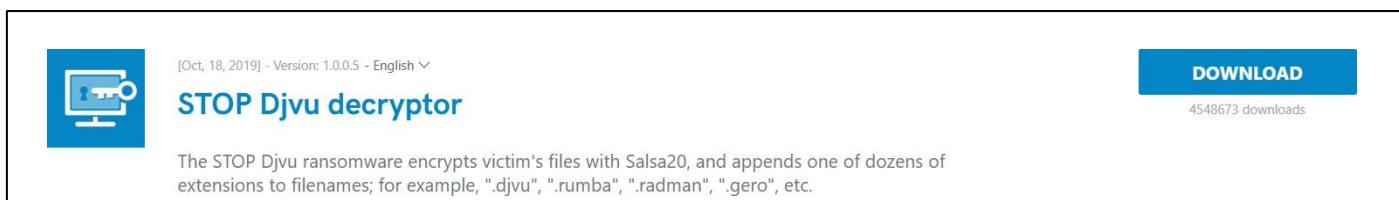
4. Décryptage des fichiers

DJVU Decryptor est un outil spécialisé permettant de déchiffrer les fichiers affectés par les ransomwares STOP/DJVU en mode Offline. Développé par Emsisoft, il exploite les failles des versions où le malware n'a pas pu contacter son serveur de commande, utilisant alors une clé AES statique stockée localement.

L'outil analyse la structure des fichiers infectés, identifie la variante spécifique, puis applique un processus de déchiffrement en trois étapes : extraction de la clé à partir des traces système, suppression du footer malveillant, et application de l'algorithme AES-256 en mode CTR pour restaurer les données originales il présente cependant des limites : inefficace contre les versions Online où la clé est chiffrée avec RSA, et moins performant sur les fichiers très volumineux.

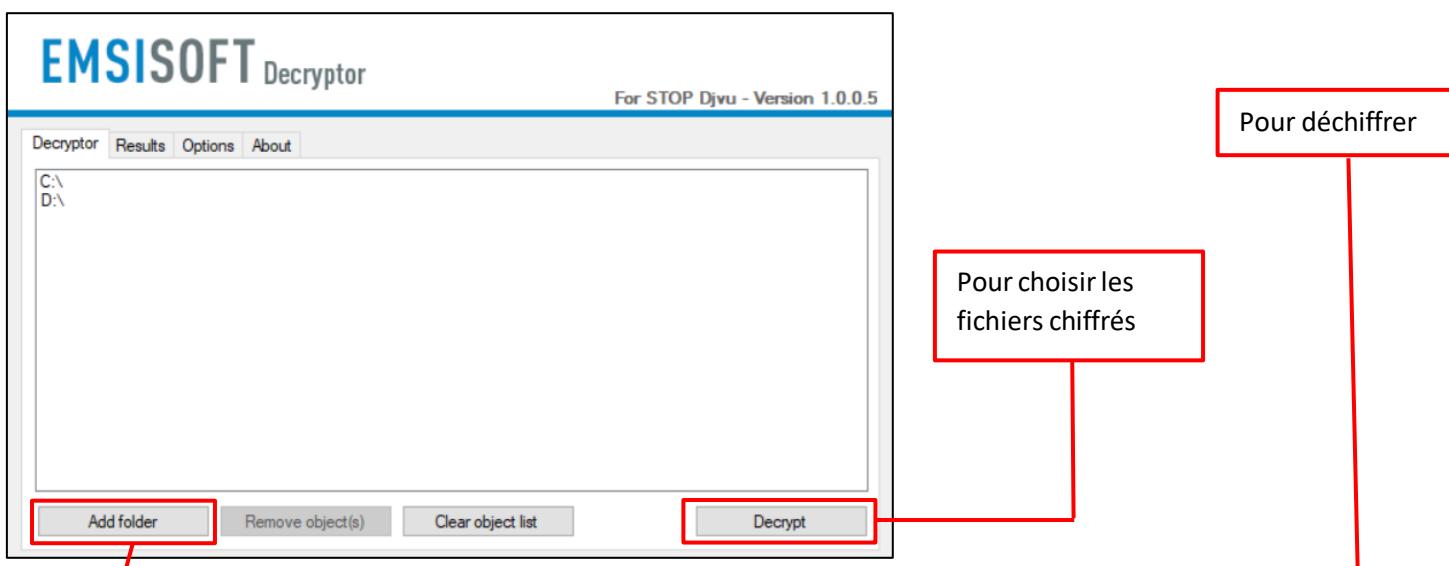
Voici les étapes suivies :

Téléchargement du DJVU Decryptor : Le logiciel est téléchargé depuis une source fiable.



Sélection des fichiers cryptés : J'ai sélectionné les fichiers cryptés avec les extensions suspectes (comme .zzla et .opqz).

Lancement du décryptage : J'ai lancé le processus de décryptage, et après un certain temps, les fichiers ont été restaurés.



Résultats du décryptage :

Le processus de décryptage a été couronné de succès, comme en témoignent les captures d'écran des fichiers récupérés. Cet exemple les fichiers avant et après le décryptage, prouvant ainsi l'efficacité du DJVU Decryptor.



5. Analyse Technique Approfondie de STOP/DJVU

Mécanismes de Chiffrement

STOP/DJVU utilise une architecture de chiffrement à double couche pour maximiser son impact. La première couche repose sur l'algorithme AES-256 en mode CBC (Cipher Block Chaining), un standard industriel robuste. Dans ce mode, chaque bloc de données est combiné via une opération XOR avec le bloc précédent avant chiffrement, ce qui complique les attaques par analyse de motifs. En mode Online, une clé AES unique est générée pour chaque victime, tandis qu'en mode Offline, une clé statique prédefinie est utilisée.

La seconde couche, réservée aux versions Online, implique du chiffrement RSA-2048. Le malware génère une paire de clés RSA (publique/privée) : la clé AES est chiffrée avec la clé publique et transmise au serveur C2 (Command & Control). Sans la clé privée, détenue exclusivement par les attaquants, le déchiffrement devient mathématiquement impossible, rendant la récupération des données extrêmement difficile sans paiement de la rançon.

Persistance et Propagation

Pour assurer sa survie dans le système infecté, STOP/DJVU déploie plusieurs techniques avancées :

- Installation furtive : Le malware se copie dans des dossiers système masqués (ex: `%AppData%\Local\[NomAléatoire]`), exploitant les permissions utilisateur pour éviter une détection manuelle.
- Persistance via tâches planifiées : Il crée des entrées dans le Windows Task Scheduler (ex: `schtasks /create /tn "Mise à jour système"`) pour se réexécuter après chaque redémarrage.
- Neutralisation des défenses : Des commandes comme `net stop "Windows Defender"` désactivent les antivirus, tandis que `vssadmin delete shadows /all` supprime les sauvegardes Volume Shadow Copy (VSS), bloquant toute restauration via les "versions précédentes" de Windows.

Ces mécanismes transforment STOP/DJVU en une menace résiliente, capable de résister aux tentatives de suppression classiques.

Techniques d'Évasion et Ciblage

STOP/DJVU excelle dans l'art de passer inaperçu et de maximiser ses dommages :

- **Ciblage étendu** : Il scanne activement les disques et réseaux pour chiffrer plus de 400 types de fichiers (documents, images, bases de données, multimédias), paralysant ainsi les activités professionnelles et personnelles.
- **Furtivité** : Contrairement à des ransomwares comme WannaCry, il opère silencieusement sans fenêtres pop-up. Il se camoufle en injectant son code dans des processus légitimes (ex: `svchost.exe`) ou en usurpant leurs noms.
- **Destruction proactive des sauvegardes** : Via des outils système détournés (ex: `wmic shadowcopy delete`), il élimine toute possibilité de récupération par snapshots Windows, forçant les victimes à négocier.

6. Simulation Pédagogique du Ransomware STOP/DJVU

Pourquoi cette Simulation ?

Face à la complexité croissante des ransomwares comme STOP/DJVU, j'ai décidé de créer une simulation pédagogique pour :

- Comprendre en profondeur les mécanismes de chiffrement/déchiffrement
- Expérimenter sans risque les techniques utilisées par les cybercriminels
- Développer des compétences pratiques en analyse de malware
- Préparer des défenses efficaces contre ce type de menace

Création du Simulateur (`fake_rans.py`)

Ce script simule le comportement d'un ransomware de la famille **STOP/Djvu**, largement répandu. Il chiffre les fichiers d'un dossier en utilisant l'algorithme **AES en mode CTR** (Counter), puis ajoute un **footer simulé** contenant des métadonnées comme une signature factice (`b'djvus'`), une clé chiffrée et un identifiant machine. Enfin, les fichiers sont renommés avec une extension fictive. `zzla`, et une note de rançon éducative est générée.

Fonctionnalités principales :

- **Génération d'une clé AES aléatoire (256 bits)** pour le chiffrement.
- **Chiffrement de fichiers** en mode AES-CTR avec un nonce de 8 octets (comme utilisé dans le vrai ransomware).

```

# Configuration STOP Djvu simulée
FAKE_SIGNATURE = b'djvus'
FAKE_ID = get_random_bytes(32) # ID machine simulé (32 octets)
FAKE_KEY_ENCRYPTED = get_random_bytes(32) # Clé AES simulée chiffrée

# Génère une clé AES pour le chiffrement
def generate_key():
    return get_random_bytes(32)

```

- **Ajout d'un footer** contenant :

- Une signature fictive (b'djvus')
- Une "clé chiffrée" simulée
- Un identifiant machine simulé (32 octets)

```

# Construction du footer simulé STOP Djvu
fake_footer = FAKE_SIGNATURE + FAKE_KEY_ENCRYPTED + FAKE_ID

with open(file_path, 'wb') as f:
    f.write(nonce + encrypted_data + fake_footer)

```

- **Renommage des fichiers** avec l'extension .zzla pour simuler l'infection.
- **Création d'un fichier README.txt** pour simuler une fausse note de rançon.
- **Sauvegarde de la clé AES** dans un fichier texte (utile pour la phase de déchiffrement).

```

# Renomme les fichiers avec ".zzla"
def rename_files(directory, ext=".zzla"):
    for filename in os.listdir(directory):
        path = os.path.join(directory, filename)
        if os.path.isfile(path) and filename != "README.txt":
            new_name = filename + ext
            os.rename(path, os.path.join(directory, new_name))
            print(f"[+] Renommé : {filename} -> {new_name}")

# Crée une fausse note de rançon comme dans STOP Djvu
def create_readme(directory):
    ransom_note = f"""
    ATTENTION !
    Vos fichiers ont été chiffrés avec .zzla
    Ce simulateur est à but éducatif uniquement.
    Aucune donnée n'a été compromise.
    Votre ID personnelle : {base64.b64encode(FAKE_ID).decode()[:32]}
    Pour tout renseignement, contactez : fake-decrypt-support@education.org
    """
    with open(os.path.join(directory, "README.txt"), 'w') as f:
        f.write(ransom_note.strip())

```

Interface graphique de déchiffrement

Le second script est une **interface graphique en Python (Tkinter)** permettant à l'utilisateur de déchiffrer les fichiers .zzla précédemment chiffrés. L'utilisateur doit fournir la **clé AES originale au format hexadécimal**.

Fonctionnalités principales :

- **Interface simple et intuitive** avec :
 - Saisie de la clé AES

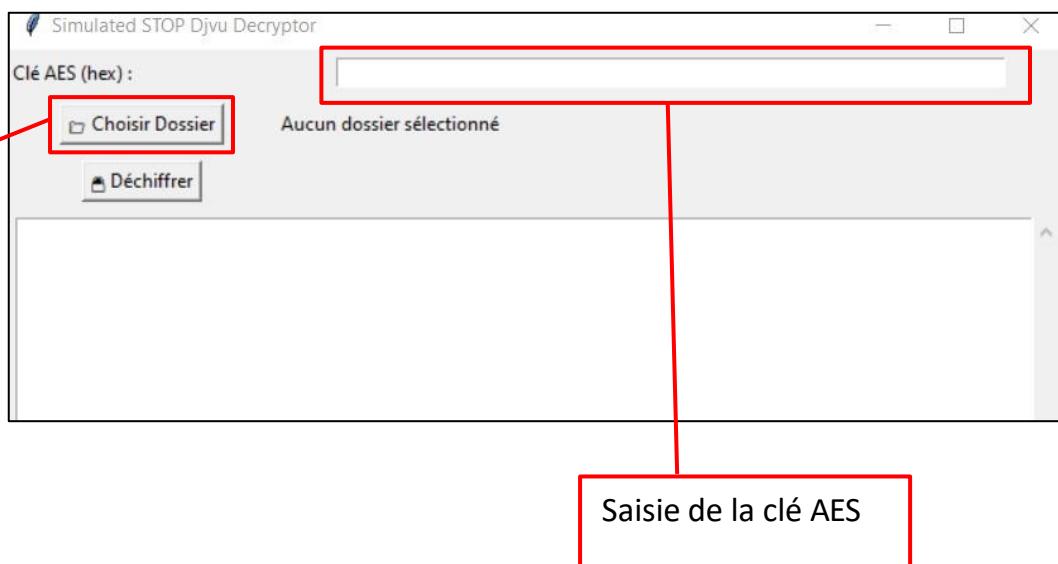
- Sélection du dossier contenant les fichiers .zzla
- Bouton de déchiffrement
- Affichage des résultats dans une console intégrée
- Extraction du nonce et des données chiffrées à partir de chaque fichier .zzla
- Déchiffrement avec AES-CTR pour reconstituer le fichier original

```
def decrypt_file(self, path):
    try:
        with open(path, 'rb') as f:
            data = f.read()
        if len(data) < NONCE_SIZE + FOOTER_SIZE:
            self.log(f"[+] Trop court ou non valide : {os.path.basename(path)}")
            return
        nonce = data[:NONCE_SIZE]
        ciphertext = data[NONCE_SIZE:-FOOTER_SIZE]
        cipher = AES.new(self.key, AES.MODE_CTR, nonce=nonce)
        plaintext = cipher.decrypt(ciphertext)
```

- Suppression du footer simulé lors de la récupération du contenu initial
- Log des opérations réussies ou échouées avec messages d'erreur clairs

```
        self.log(f"[+] Déchiffré : {os.path.basename(path)}")
    except Exception as e:
        self.log(f"[!] Échec : {os.path.basename(path)} ({e})")
```

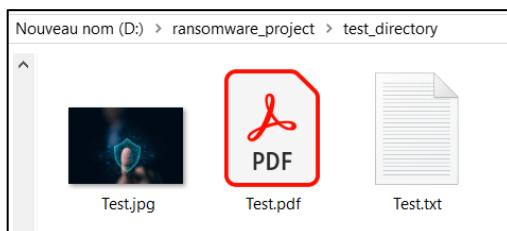
Vue de l'application graphique de déchiffrement :



Afin de tester concrètement le fonctionnement de la simulation, j'ai créé un répertoire nommé `test_directory` contenant trois fichiers exemples. Ce dossier a servi de terrain d'expérimentation pour observer le processus complet de chiffrement puis de déchiffrement à l'aide des scripts précédemment développés.



Les fichiers de test (.jpg, .pdf, et .txt)



Chiffrement des fichiers

Le script de chiffrement a parcouru tous les fichiers du répertoire `test_directory` et a appliqué un chiffrement AES-CTR sur chaque fichier avec une clé AES générée aléatoirement. Après le chiffrement, un footer simulé a été ajouté à chaque fichier, contenant une fausse signature (`djvus`), une clé simulée chiffrée, et un identifiant aléatoire. Tous les fichiers ont ensuite été renommés avec l'extension `.zzla`, simulant ainsi un comportement typique des ransomwares comme STOP Djvu.

Apres le lancement de script

```
[Running] python -u "d:\ransomware_project\fake_rans.py"
[+] Renommé : Test.jpg -> Test.jpg.zzla
[+] Renommé : Test.pdf -> Test.pdf.zzla
[+] Renommé : Test.txt -> Test.txt.zzla
[+] Fichier chiffré façon STOP Djvu : Test.jpg.zzla
[+] Fichier chiffré façon STOP Djvu : Test.pdf.zzla
[+] Fichier chiffré façon STOP Djvu : Test.txt.zzla
```

Création automatique de la note de rançon

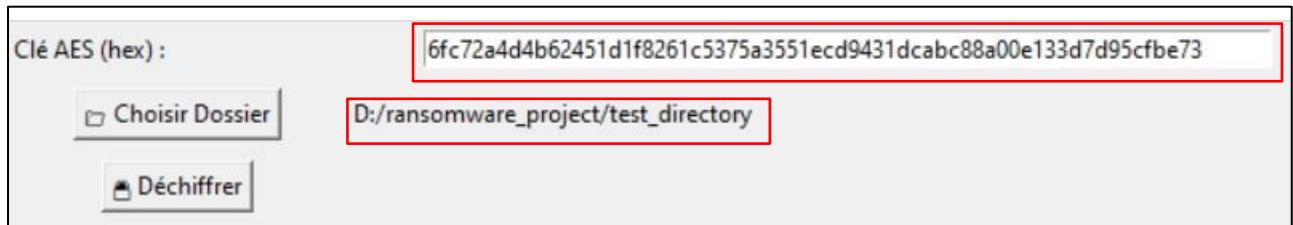
Une fois le chiffrement effectué, une **note de rançon** nommée `README.txt` a été générée automatiquement. Cette note contient un message pédagogique expliquant que le chiffrement est simulé à des fins éducatives. Elle inclut également un faux identifiant unique encodé en base64 et un contact fictif pour renforcer l'illusion d'un ransomware actif.



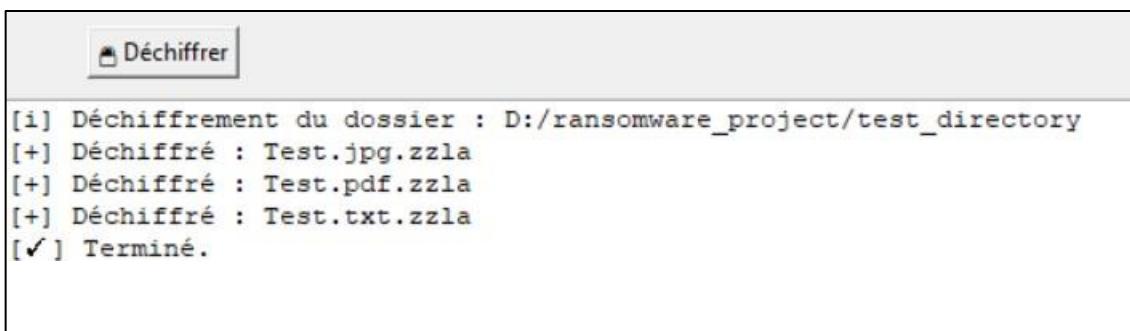
Test de Déchiffrement via l'Interface Graphique

Une fois les fichiers chiffrés et la note de rançon créée, j'ai créé un fichier python pour décrypter (derypt.py) pour tester le processus de déchiffrement. Cette interface permet à l'utilisateur de fournir la clé AES (au format hexadécimal) utilisée lors du chiffrement, et de déchiffrer les fichiers .zzla en suivant ces étapes :

- **Saisie de la clé AES** : L'utilisateur entre la clé AES en hexadécimal dans un champ de texte prévu à cet effet.
- **Sélection du dossier de fichiers à déchiffrer** : L'utilisateur choisit le répertoire `test_directory` contenant les fichiers .zzla à déchiffrer.



- **Lancement du déchiffrement** : En cliquant sur le bouton "Déchiffrer", l'application extrait le **nonce** (premiers 8 octets) et les **données chiffrées** des fichiers .zzla. Le processus de déchiffrement est effectué en utilisant la clé AES fournie, avec le même mode **CTR** que lors du chiffrement.
- **Récupération des données originales** : Une fois déchiffrées, les données sont restaurées dans leur format d'origine et écrites dans un fichier sans l'extension .zzla, le footer (avec la fausse signature et les autres informations) est ignoré et non inclus dans le fichier déchiffré.



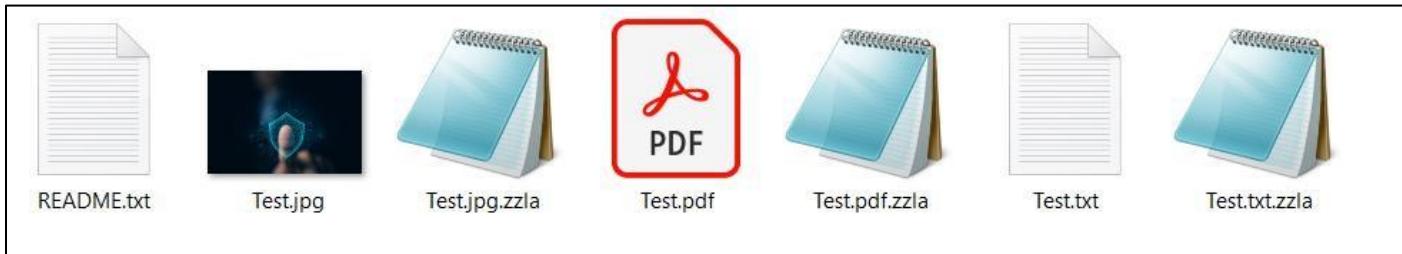
- **Log des opérations** : Tous les fichiers traités sont enregistrés dans une **console intégrée**, indiquant si le déchiffrement a réussi ou échoué, avec des messages d'erreur clairs en cas de problème (par exemple, si la clé est incorrecte ou si le fichier n'est pas valide).

Résultats et Validation

Le test a permis de valider que l'interface fonctionne correctement :

- Les fichiers .zzla ont été déchiffrés avec succès lorsque la clé correcte était fournie.
- Les fichiers originaux ont été restaurés sans erreur, et le footer simulé a bien été ignoré.

- L'interface graphique a montré des messages détaillés sur chaque fichier traité, confirmant que le processus de déchiffrement s'était déroulé sans accrocs.



7. Conclusion

Ce rapport a permis de mettre en lumière les mécanismes complexes des ransomwares, notamment la famille STOP/DJVU, et d'illustrer, à travers un cas concret, l'importance d'une réponse méthodique face à ce type de menace. Grâce à l'analyse détaillée de l'infection et à l'utilisation d'outils comme DJVU Decryptor, j'ai pu récupérer les données chiffrées et mieux comprendre le fonctionnement interne du malware.

La simulation pédagogique que j'ai développée constitue un outil précieux pour appréhender les étapes du chiffrement et du déchiffrement, ainsi que pour expérimenter, en toute sécurité, les techniques employées par les cybercriminels. Ce projet a non seulement enrichi ma compréhension des ransomwares, mais m'a aussi permis de mettre en pratique des compétences essentielles en cybersécurité, telles que l'analyse de malware, la gestion de clés cryptographiques, et la conception d'interfaces graphiques.

En somme, cette étude de cas concrète démontre l'importance de l'éducation et de la prévention face aux ransomwares. Si de telles attaques peuvent causer des pertes irréversibles, elles offrent également une occasion d'approfondir nos connaissances techniques et de renforcer nos capacités à contrer ce type de cybermenace. La cybersécurité est un domaine en constante évolution, et il est essentiel de continuer à développer des outils et des approches innovantes pour mieux protéger les systèmes et les données.