Rapport Projet Cryptographie – Algo vérification Certificat

Par : Palayer François

1°) Introduction :

Dans le contexte de la sécurité informatique et des communications numériques, les certificats numériques jouent un rôle crucial. Ils assurent l'authenticité, l'intégrité et la confidentialité des échanges en ligne. Cependant, la confiance accordée à ces certificats repose sur la validité de la chaîne de certificats, un processus essentiel qui implique plusieurs niveaux de vérification. Une chaîne de certificats est une suite de certificats reliant un certificat utilisateur à une autorité de certification (CA) racine de confiance. Vérifier cette chaîne est fondamental pour garantir la sécurité des communications numériques. Mais pour cela, des vérifications doivent être faites comme :

- La première et principale raison de vérifier une chaîne de certificats est d'assurer la confiance et l'authenticité des certificats numériques. Chaque certificat dans la chaîne est signé par l'entité suivante, jusqu'à arriver à une autorité de certification racine, largement reconnue et de confiance. En validant cette chaîne, on s'assure que le certificat présenté est légitime et qu'il n'a pas été falsifié. Cela protège contre les attaques de type "man-in-the-middle" et autres tentatives de fraude L'analyse de l'intégrité de la chaine de certificats, cela en vérifiant les signatures numériques de chaque certificat pour garantir que rien n'a été altéré ou falsifié depuis l'émission du certificat.

- La vérification de l'intégrité des certificats est une autre étape critique. Elle consiste à s'assurer que les certificats n'ont pas été modifiés ou corrompus depuis leur émission. Les signatures numériques apposées par les autorités de certification jouent un rôle clé dans cette vérification. Si un certificat a été altéré, la signature numérique ne correspondra plus, ce qui invalidera la chaîne de certificats. L'analyse de la chaine permet aussi de reconnaitre une autorité de confiance ou une ancre de confiance, pour établir une confiance utilisée pour la sécurité des communications en ligne.

- Les certificats numériques ont une durée de vie limitée. La vérification de la validité temporelle consiste à s'assurer que les certificats ne sont pas expirés et qu'ils sont valides à la date de vérification. Cela inclut également la vérification des listes de révocation de certificats (CRL) et des statuts OCSP.

II°) Le choix des outils

Le choix du langage :

D'après les conseils de camarades qui ont eu plus de facilité à gérer ça, le langage choisit pour élaborer le programme a été le Python, notamment grâce à sa portabilité sur plusieurs système mais aussi pour simplifier la lisibilité du code et profiter des librairies concernant la cryptographie et les gestion d'erreur étant plus efficace via les outils de débogages que propose python.

Le choix des outils :

Le choix de l'environnement de développement s'est porté sur Visual Studio Code avec des extensions d'export Github et d'aide sur le code appelé Co-pilot.

Le choix des librairies :

Comme bibliothèque utilisé il y a principalement « cryptography », elle regroupe sous python l'essentielle des modules et fonctions concernant la cryptographie.

III°) Etapes réalisées du projet :

A l'heure actuelle les étapes du projet qui ont été réalisés sont celles demandés dans le sujet, à l'exception de la vérification manuelle de signature RSA et ECDSA et de la révocation et étapes bonus

Pour chaque parties, la partie 1 :

- Ici, le code vérifie que le nombre d'arguments passés est exactement 3. Si ce n'est pas le cas, il affiche un message d'usage correct et termine le programme avec '`sys.exit(1)`':

```
if len(sys.argv) != 3:
    print("Usage: python validate-cert.py format[DER|PEM] chemin_du_certificat")
    sys.exit(1)
```

Ensuite, pour déterminer si l'argument de format est "DER" ou "PEM", le code utilise une condition à l'intérieur de la fonction '`charger_certificat`':

```
if format == 'DER':
    cert = x509.load_der_x509_certificate(contenu, default_backend())
elif format == 'PEM':
    cert = x509.load_pem_x509_certificate(contenu, default_backend())
else:
    raise ValueError("Format non supporté")
```

- Ensuite, La récupération de la clé publique du certificat se fait dans la fonction '`verifier_signature`'. Si tout a bien fonctionné, nous devrions avoir comme retour :

```
[][08:10:25] []Francois []/media/sf_vm_dossier/test []>python3 validate-cert.py PEM blizzard.pem
La signature du certificat est considérée valide (auto-signé).
Sujet : CN=blizzard.com
Émetteur : CN=Amazon RSA 2048 M02,O=Amazon,C=US
Valide du 2024-06-07 00:00:00 au 2025-07-06 23:59:59
Usages de la clé :
  Signature numérique : True
  Non-répudiation : False
  Chiffrement de clé : True
  Chiffrement de données : False
  Authentification de clé : False
  Signature de certificat : False
  Signature de CRL : False
[][08:12:05] []Francois []/media/sf_vm_dossier/test []>python3 validate-cert.py DER blizzard.cer
La signature du certificat est considérée valide (auto-signé).
Sujet : CN=Blizzard Battle.net Local Cert,OU=Battle.net,O=Blizzard Entertainment,L=Irvine,ST=California,C=US
Émetteur : CN=Blizzard Battle.net Local Cert,OU=Battle.net,O=Blizzard Entertainment,L=Irvine,ST=California,C=US
Valide du 2018-01-01 21:32:19 au 2027-12-30 21:32:19
L'extension Key Usage n'est pas présente dans ce certificat.
[][08:12:19] []Francois []/media/sf_vm_dossier/test []>
```

- Ensuite, La vérification de l'usage de la clé dans le certificat est effectuée dans la fonction '`verifier_key_usage`', si l'extension n'est pas présente, il y a un message sur la sortie standard disant « L'extension Key Usage n'est pas présente dans ce certificat ».

La validité temporelle du certificat est vérifiée dans la fonction ‘`verifier_validite_temporelle`’. Il faut tout d’abord récupérer la date actuelle avec ‘`datetime.utcnow()`’.

```
maintenant = datetime.utcnow()
```

Ensuite, il faut la comparer avec les champs renvoyant la dates pas avant «`not_valid_before` » et pas après « `not_valid_after` ».

```
if certificat.not_valid_before <= maintenant <= certificat.not_valid_after:
    print("Le certificat est temporellement valide.")
    return True
else:
    print("Le certificat n'est pas temporellement valide.")
    return False
```

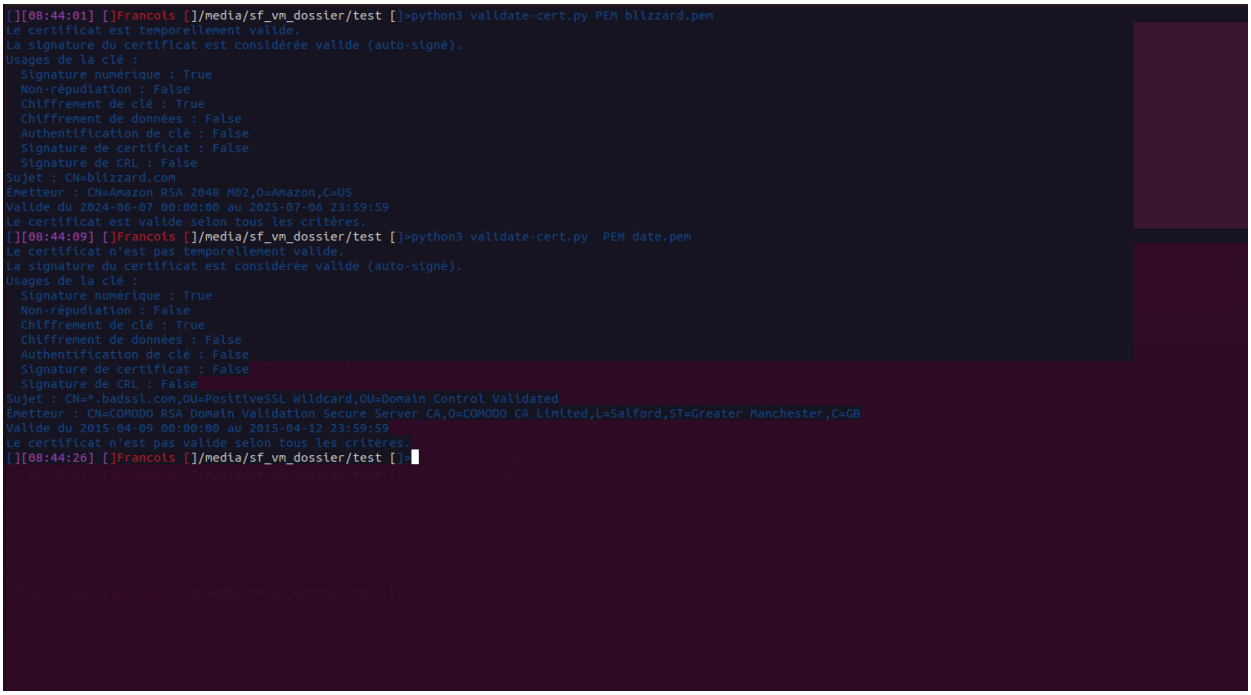
Enfin, pour vérifier l’état du certificat on appelle la fonction ‘`verifier_certificat`’, qui vérifie tous les critères de validité du certificat faisant un opérateur logique AND de tous les retours booléens des fonctions correspondant aux critères de validité du certificat observé :

```
def verifier_certificat(certificat):
    """Vérifie tous les aspects du certificat et affiche un message final."""
    valide_temporellement = verifier_validite_temporelle(certificat)
    signature_valide = verifier_signature(certificat)
    key_usage_valide = verifier_key_usage(certificat)

    afficher_details(certificat)

    if valide_temporellement and signature_valide and key_usage_valide:
        print("Le certificat est valide selon tous les critères.")
    else:
        print("Le certificat n'est pas valide selon tous les critères.")
```

Voici deux cas de figures, l’un où l’on inspecte un certificat valide venant d’un site commun blizzard.com et l’autre issue d’un site ayant une période de validité expiré :



On peut voir que le certificat n°2 n’est pas valide temporellement et donc ne coche pas toutes les case, il n’est pas valide.

Maintenant, voyons pour la partie 2 :

Tout d'abord, on vérifie bien qu'au lancement du programme, on a bien le bon nombre d'arguments.

```
if __name__ == '__main__':
    if len(sys.argv) != 4:
        afficher_usage()
        sys.exit(1)
```

Si, l'utilisateur n'a pas rentré les arguments nécessaires, il s'affiche alors le format demandé et un exemple pour le guider.

```
[][08:52:58] []Francois []/media/sf_vm_dossier/test []>python3 validate-cert-chain.py
Usage: python validate-cert-chain.py certificat_final certificat_intermediaire certificat_racine
Exemple: python validate-cert-chain.py blizzard.pem AmazonRSA.crt AmazonRootCA.crt
[][08:53:06] []Francois []/media/sf_vm_dossier/test []>
```

Ensuite, Les certificats sont chargés dans la section principale du script, où les chemins des certificats sont extraits des arguments de la ligne de commande et passés à '`charger_certificat`'.

- Dès lors, nous commençons par vérifier la validité de la chaine en vérifiant la clef publique de chaque certificat et en s'assurant que le sujet du certificat est l'emetteur du certificat précédent. Si le moindre test ne passe pas, on retourne alors

`La chaîne de certificats invalide.`

Si par contre, aucun échec n'est rapporté, alors on retourne

`La chaîne de certificats valide.`

- L'extension '`BasicConstraints`' est récupérée à l'aide de la méthode '`get_extension_for_class`' de la bibliothèque '`cryptography`'

```
[][09:11:02] []Francois []/media/sf_vm_dossier/test []>python3 validate-cert-chain.py blizzard.pem AmazonRSA.crt AmazonRootCA.crt
Certificat émetteur 1 a l'extension BasicConstraints et est autorisé à signer d'autres certificats.
Signature du certificat 0 vérifiée avec succès.
Certificat émetteur 2 a l'extension BasicConstraints et est autorisé à signer d'autres certificats.
Signature du certificat 1 vérifiée avec succès.
Chaîne de certificats valide.
[][09:11:04] []Francois []/media/sf_vm_dossier/test []>python3 validate-cert-chain.py  invalid_leaf.pem  AmazonRSA.crt  AmazonRootCA.crt
Certificat émetteur 1 a l'extension BasicConstraints et est autorisé à signer d'autres certificats.
Erreur de validation de la signature pour le certificat 0:
Certificat émetteur 2 a l'extension BasicConstraints et est autorisé à signer d'autres certificats.
Signature du certificat 1 vérifiée avec succès.
Chaîne de certificats invalide.
[][09:11:16] []Francois []/media/sf_vm_dossier/test []>
```

IV°) Autres choix techniques et structure de programme

Il a été décidé que chaque fonctionnalité se suive l'une après l'autre dans son développement. En suivant la temporalité du sujet. Beaucoup de retour en arrière ont été fait pour tenter de développer le souci de la création des fonctionnalités concernant les vérifications des signatures RSA et ECDSA. Ainsi un github a été utilisé en local pour ce projet.

V°) Les difficultés rencontrées

Les difficultés rencontrées se sont concentrés sur l'élaboration du code notamment la vérification de la chaine de certificat qui a donné du fil à retordre de par les différents scénarios possibles à prendre compte pour discréditer la validité d'un certificat. Il y a eu la compréhension des vérifications des chaines de certificat x509 qui a occupé une période non-négligeables durant l'élaboration de ce projet .

VI°) Les axes d'amélioration

Il serait d'avis de finir le projet avec la vérification RSA et ECDSA et l'ajout d'un statut vérifiant la révocation d'un certificat en plus du bonus restant. Ensuite, Il faudrait couvrir un plus grand axe de scénario pouvant causer une discréditation d'une chaine de certificat et proposer une application avec une interface, permettant à un utilisateur lambda de vérifier la validité d'une chaîne, Cela donnerait un gain de temps conséquent et donc une application possible dans le monde professionnelle.

VII°) Ressources utilisées

Les ressources utilisés ont été multiples, il y a eu la recherche sur de nombreux sites internet en voici quelques exemples :

https://blog.eleven-labs.com/fr/comprendre-le-ssltls-partie-3-certificats/

https://www.globalsign.com/fr/centre-information-ssl/definition-certificat-ssl

https://chatgpt.com/

https://badssl.com/

VIII°) Conclusion

Bilan technique : Sur le plan technique le projet n'est qu'en partie réussie, il manque des éléments cruciaux tels que la vérification manuel des signatures RSA et ECDSA et la révocation, caractéristiques très importantes dans la vérification d'une chaine de certificat X509. On peut néanmoins noter l'existence d'un moyen subsidiaire pour vérifier l'état d'un certificat racine et une chaîne de certificats.

Bilan pédagogique : Sur le plan pédagogique le projet m'a permis d'incorporé le fonctionnement du certificat x509, j'ai pu ainsi comprendre comment la présence de ces documents numériques sont vitaux pour tout échanges numériques sécurisés sur le web. J'ai pu me familiarisé avec de nombreux nouveaux outils de développement et continuer d'approfondir ce que je connaissais déjà.