Rapport 3 : Ingénierie Crypto

Membres du groupe :

- *Papa Abdou CALLOGA*
- *Yacine DIAGNE*
- *Yaye Khadidiatou DIOP*
- *Aby NDIAYE*
- *Alioune SALL*

Professeur : *M. MENDY*

Compréhension des 3 scénarios d'attaque et rajout de 2 autres scénarios :

1. Scénarios d'attaque

Nous avons à disposition les 5 scénarios qui sont les suivants que nous allons implémenter dans la suite en tant que cas pratiques .

Scénario 1 :

Une application chiffre des numéros de cartes de crédit dans une base de données utilisant un chiffrement en base automatique. Cependant, ces données sont automatiquement déchiffrées lorsqu'elles sont récupérées, permettant, à une injection SQL de récupérer des numéros de carte de crédit en clair.

- Outils utilisés :
  - Base de données SQL : la base de données sera utilisée pour stocker les mots de passe
  - Langages de programmation :
    - HTML/CSS : Pour le formulaire de connexion d'un utilisateur (Mise en page et Mise en forme)
    - SQL : Pour gérer la base de données et les requêtes SQL
    - Php : Pour l'exécution de scripts qui permettent de récupérer les données utilisateurs (numéros de carte crédit)
  - Fonctions de hachage :
    - MD5 : Pour le cryptage des données
  - Docker : un conteneur Docker sera utilisé comme environnement de test
- Implementation: voir démo

Scénario 2 :

Un site n'utilise pas ou ne force pas l'utilisation de TLS sur toutes les pages ou supporte des protocoles de chiffrement faibles. Un attaquant surveille le trafic réseau (par exemple sur un réseau sans fil non sécurisé), dégrade les connexions de HTTPS à HTTP, intercepte les requêtes et vole le cookie de session d'un utilisateur. L'attaquant utilise alors ce cookie et détourne la session de l'utilisateur (authentifié), pouvant ainsi accéder aux données privées de l'utilisateur ou les modifier. Un attaquant pourrait également modifier toutes les données en transit,

par exemple le destinataire d'un virement d'argent.

- Outils utilisés:
  - Wireshark: pour surveiller le trafic réseau et capturer les cookies
  - EditThisCookie: Extension pour injecter des cookies
- Implementation: voir démo

Scénario 3 :

La base de données contenant les mots de passe n'utilise pas de sel, ou alors de simples hachés pour stocker les mots de passe de chacun. Une faille d'upload de fichier permet à un attaquant de récupérer la base de données de mots de passe. Tous les hachés non salés peuvent alors être révélés avec une rainbow table de hachés pré-calculés. Des hachés générés par des fonctions de hachage simples ou rapides peuvent être déchiffrés par des GPUs, même salés.

- Outils utilisés :
  - Bases de données SQL : la base de données sera utilisée pour stocker les mots de passe
  - Fonctions de Hachages : MD5
  - Langage de programmation :
- Implementation: voir démo

Cependant nous devons créer deux autres scénarios provenant de notre imagination et qui soit réalisable . Ci dessous les deux scénarios que nous avons créé

Scénario 4:

L'attaquant injecte un script JavaScript malveillant dans la base de données d'un site Web. Lorsque la victime demande une page du site Web, le site Web transmet la page à son navigateur avec le script malveillant intégré au corps HTML. Le navigateur de la victime exécute ce script, qui envoie par exemple le cookie de la victime au serveur de l'attaquant, qui l'extrait et l'utilise pour détourner la session. Ces vulnérabilités peuvent non seulement permettre à un attaquant de voler des cookies, mais aussi d'enregistrer les frappes de touches et des captures d'écran, de découvrir et de collecter des informations réseau et d'accéder et de contrôler à distance l'ordinateur de la victime.

Scénario 5:

L'application testée et ses clients effectuent des échanges via un protocole utilisant un algorithme de chiffrement par blocs faibles pour chiffrer les communications.

Une attaque Man in the Middle est alors mise en place par le hacker, pour capturer le trafic crypté entre l'utilisateur et l'application, puis craquer le cryptage facilement en raison de la faiblesse du chiffrement.

- Outils et concepts concernés :
  - Langages de programmation :
    - HTML/CSS;
    - SQL;
    - Php;
  - Fonctions de cryptage utilisée : DES;
  - Outil pour capturer le trafic :: Wireshark
  - Outil de reconnaissance de l'attaqueur : testssl.sh;
  - Exploit utilisés : Sweet32 (*CVE-2016-2183, CVE-2016-6329)*
  - Docker : un conteneur Docker sera utilisé comme environnement de test
- Implementation: voir démo