I - Introduction

Question 1 : D’après vous, Debian est-il mieux sécurisé qu’un système Windows ?

Selon moi, Debian est mieux sécurisé que Windows dû à la possibilité de gestion des droits d’accès sur les fichiers. En effet, à l’aide de chmod, on peut facilement décider qui à accès à quels fichiers. De plus, sur Linux, les utilisateurs ne sont pas administrateurs contrairement à Windows et permet donc d’éviter de faire des choses irréparables sur l’ordinateur. Sur Windows, le lancement des fichiers .exe se fait avec un double clic donc très facilement alors que sur Linux c’est plus difficile. On sait que les fichiers .exe sont très souvent source de virus. Enfin, Linux est un système d’exploitations Open Source, ainsi il y a des mises à jour très régulièrement et tous les meilleurs informlaticiens du monde peuvent détecter des bugs et les résoudre.

Question 2 : D'après votre compréhension du business de SRIOEYE, quelle sont les deux principes de sécurité que vous jugez plus délicat (commentez votre réponse)?

SRIOEYE est une entreprise spécialisée dans la maintenance de caméras de surveillance. Pour moi, le premier principe de sécurité que je juge délicat est la disponibilité. La disponibilité désigne le fait qu’un système doit être disponible lorsque l’on en a besoin. Etant donné que la société utilise des caméras de surveillance, il est dans l’obligation d’obtenir les vidéos en direct au moment où on en a besoin. L’autre principe est la confidentialité. Cela consiste à ce qu’un système ne soit accessible seulement par les personnes autorisées (notion de LLP). Il est logique que seuls les personnes pouvant regarder les vidéos des caméras soient les personnes qui sont autorisées à le faire et non pas des individus malintentionnés.

Question 3 : Si vous devez résumer en un mot la raison pour laquelle les caméras de SRIOEYE sont vulnérables, quel mot utiliseriez-vous ?

En un mot : l’interconnexion. L’interconnexion est en fait un processus de connexion de différents équipements, nous ici il s’agit de caméras, qui peuvent communiquer entre eux. Cependant, l’interconnexion a des faiblesses notamment au niveau de l’envoi des données obtenues par le capteur, au niveau de l’envoi des données à travers les actionneurs, lors de l’échange de données entre le Cloud et les autres caméras et enfin lors du traitement des données sensibles.

Question 4 : Pour un équipement IoT comme une caméra connectée à faible puissance, quelles sont les conséquences sur les protocoles de sécurité ?

Les conséquences d’une caméra connectée à faible puissance sont que la capacité de stockage et sa faible puissance vont engendrer des protocoles de sécurité peu fiable et à bas coût. On va donc négliger les bons protocoles de sécurités qui sont trop groumands en puissance.

Question 5 : Quel peut être l’avantage d’un protocole comme LORA pour SRIOEYE

LoRa est un protocole de communication sans fil qui permet aux appareils de communiquer sur de longues distances et a une faible consommation d’énergie. Ainsi, SRIOEYE doit utiliser ce protocole pour pouvoir avoir accès aux caméras depuis de longue listance. De plus, LoRa utilise un chiffrement solide pour protéger les données transmises ce qui permet de garantir la confidentialité et l’intégrité des données.

II - Obtention de secret d’architecture

Question 1 : Comment appelle-t-on cette stratégie façon d’extirper les informations sur une cible ?

Ce phénomène s’apelle le social engineering. Il consiste a sous-tirer des informations confidentielles à une personne en s’appuyant sur ses faiblesses.

III - Gestion de la base de données

Question 1 : Sachant qu’on sait que l’algorithme RC4 est utilisé, quelle est la longueur de la clé utilisée par les caméras ?

La longueur de la clé par les caméras est de 512 bits car il y a le vecteur IV qui fait 24 bits et la clé de chiffrement de 488 bits.

Question 2 : Si les caméras doivent changer des clés, chaque une minute et garder les anciennes clés de la journée pour un souci de reporting, quelle est la taille en MBytes que consommerons le stockage des clés ?

Une clé par minutes => 1440 clés par jour (car 1440 minutes en 24h) => clé sotckée sur 64 octets donc 64 x 1440 = 92 160 octets => Le stockage des clés consommera 92 Méga octets

Question 3 : Quel est le nombre combinaisons pour une attaque de type force brute si on veut deviner la clé utilisée par les caméras ?

Le nombre de combinaisons pour une attaque brute force si on veut deviner la clé utilisée est de 128^512 car 128 caractères possibles avec 512 bits.

Question 4 : Avec un CPU cadencé à 2.8 GHz par seconde, pourrais-je réussir à obtenir la clé utilisée par une caméra en moins d'une minute ? Sinon, quelle puissance de calcul me faudrait-il ?

Il est impossible d’obtenir la clé en moins d’une minute au vu du nombre de combinaisons différentes. Il faudrait des années lumières pour réussir à obtenir la clé.

Question 5 : En hexadécimal, réaliser la conversion des chiffres ci-dessous (en expliquant votre cheminement) : (1044), (481), (844), (24)

1044 :

1044 = 16 * 65 + 4
65 = 16 * 4 + 1
4 = 16 * 0 + 4

Ainsi, 1044 en base 10 équivaut à 414 en hexa.

481:

481 = 16 * 30 + 1
30 = 16 * 1 + 14
1 = 16 * 0 + 1

Ainsi, 481 en base 10 équivaut à 1E1 en hexa.

<u>844 :</u>

844 = 16 * 52 + 12
52 = 16 * 3 + 4
3 = 16 * 0 + 3

Ainsi, 844 en base 10 équivaut à 34C en hexa.

<u>24 :</u>

24 = 16 * 1 + 8
1 = 16 * 0 + 1

Ainsi, 24 en base 10 équivaut à 18 en hexa.

Question 6 : Vous avez mis la main sur les bouts de code suivant de monitoring ci-dessous. Analyser et déterminer à quelle faille il s'expose avec un exemple concret. Corrigez-les.

Cf rapport de l'équipe.

Question 7 : Quelles sont les techniques dans la littérature pour contrer le type de problème que présentaient les deux codes précédents.

Le problème dans les codes précédents est le buffer overflow. Les techniques pour contrer ce problème sont les suivants :

- *Les canaries*: ces derniers consistent en une page mémoire placées en début et/ou en fin de zone mémoire afin de détecter les débordements. Si la valeur du canarie est modifiée, alors il y a eu buffer overflow.

- *KASLR* : le KASLR ou autrement dit le Kernel Adress Space Layout Randomization consiste à changer régulièrement et aléatoirement l'emplacement des adresses mémoires du tas d'éxécution pour bloquer les attaques liées aux anciens emplacements.