

**UNIVERSITÉ ABDELMALEK ESSAADI**

**ÉCOLE NATIONAL DES SCIENCES APPLIQUÉE DE TANGER**

**DÉPARTEMENT DE MATHÉMATIQUE ET INFORMATIQUE**

**MASTER CYBER SÉCURITÉ ET CYBERCRIMINALITÉ (MCSC)**



---

**La mise en place d'une architecture de surveillance intelligente des soins de santé  
IoT Sécurisé.**

---

**Issam Najah**

**Abdelaziz Laaychi**

---

**ANNEE UNIVERSITAIRE : 2020/2021**

**Mme S .Lazaar**

# **Abstract**

La pensée croissante de l'Internet des objets (IoT) découvre rapidement son chemin tout au long de notre vie moderne, visant à améliorer la qualité de vie en associant de nombreux appareils intelligents, technologies et applications. Dans l'ensemble, l'IoT permettrait l'automatisation de tout ce qui nous entoure. Actuellement, des chercheurs ont constaté qu'il existe une application potentielle de l'IoT dans l'industrie liée aux soins de santé. Un nombre des principes de sécurité doivent être appliqués à chaque niveau pour atteindre un Réalisation IoT sécurisée. L'avenir du framework IoT ne peut qu'être garanti si les problèmes de sécurité qui y sont associés sont résolus. De nombreux chercheurs ont tenté de résoudre les problèmes de sécurité spécifiques aux couches et aux appareils IoT par la mise en œuvre des contre-mesures correspondantes.

Dans cet article, nous avons présenté un modèle sécurisé qui automatise le problème de la collecte des données vitales des patients, livraison et traitement puis l'envoyer au Cloud. Aider ainsi le médecin à diagnostiquer et observer les problèmes médicaux où se trouve le patient.

# **Introduction**

La santé mobile (m-santé) fait référence à l'utilisation des dispositifs mobiles pour collecter les données de santé en temps réel auprès des patients, le stocker sur des serveurs réseau connectés à Internet. Les données peuvent être accessibles par un groupe hétérogène de clients (p. ex., hôpitaux, compagnies d'assurance maladie, etc.). Les données m-health sont utilisées par les médecins pour surveiller, diagnostiquer et traiter les patients. Disponibilité des dispositifs médicaux portables et des réseaux de capteurs corporels passés à la santé mobile. Intégration d'appareils mobiles de santé dans l'environnement du patient permet de prévoir des anomalies de santé en temps réel.

La technologie de la santé mobile sera la clé des services de santé personnels à l'avenir. L'Internet des objets (IoT) est un environnement composé de choses telles que les étiquettes RFID, les appareils médicaux, les téléphones portables, etc. Ces appareils se connectent via des identifiants uniques et interagissent entre eux. Les appareils connectés à l'IoT transfèrent des données entre eux, conduisant à leur tour à de nouvelles données dérivées. Les soins de santé sont l'un des domaines d'application les plus importants de l'IoT. Il offre des opportunités à plusieurs applications médicales comme la surveillance de la santé mobile à distance. Les dispositifs médicaux génèrent des données et les envoient à des serveurs informatiques désignés.

Détection médicale omniprésente et la technologie mobile déployée sur les systèmes IoT conduit à l'accumulation de mégadonnées . Les données fournissent des informations sur les patients en temps réel qui peuvent conduire au diagnostic, au traitement ou hospitalisation en cas d'urgence médicale. Par conséquent, l'infrastructure IoT, les dispositifs intelligents de surveillance de la santé, et les communications sans fil offrent d'énormes possibilités d'amélioration de la qualité des services de santé. Le concept de la m-santé dans le contexte de l'Internet des objets (m-IoT) apporte les promesses d'un meilleur système de gestion de la santé par planification des ressources limitées en assurant leur meilleure utilisation et service de plus de patients. Les services de m-santé basés sur l'IoT profitent aux gens en économisant leur temps et leur argent de visite les hôpitaux à moins qu'il n'y ait un réel besoin. Les éléments essentiels de la m-santé et de l'IoT sont les dispositifs médicaux équipés de capteurs et d'appareils de communication. Celles-ci les dispositifs médicaux sont en fait des dispositifs intelligents qui peuvent être utilisés pour surveiller divers paramètres tels que le taux de sucre, la pression artérielle, la fréquence cardiaque, etc. m-health data en temps réel pour identifier certains modèles et augmenter différents niveaux d'alerte tels que normal, prudent, urgence, etc., en fonction de l'état des patients observés. Le volume de données sur les serveurs IoT est très important car il est collecté des millions d'utilisateurs individuels ou de patients.

Un nombre des principes de sécurité doivent être appliqués à chaque niveau pour atteindre un Réalisation IoT sécurisée. L'avenir du framework IoT ne peut qu'être garanti si les problèmes de sécurité qui y sont associés sont résolus. De nombreux chercheurs ont tenté de résoudre les problèmes de sécurité spécifiques aux couches et aux appareils IoT par la mise en œuvre des contre-mesures correspondantes. Dans cet article, nous avons présenté un modèle sécurisé qui automatise le problème de la collecte des données vitales des patients, livraison et traitement puis l'envoie au Cloud. Aider ainsi le médecin à diagnostiquer et observer les problèmes médicaux où se trouve le patient. La section suivante décrit les architectures IoT. Dans la section II, on décrit les Défis de sécurité dans chaque couche de l'IoT . Ensuite, nous décrirons notre architecture avec les matériaux et les logiciels utilisés (section III). Ensuite, une implémentation sécurisée est faite, nous commençons par la collection des données d'après les capteurs jusqu'à l'envoi des données au Cloud. Enfin, cet article se termine par des conclusions.

## I. LES ARCHITECTURES IoT

Dans l'IoT, chaque couche est définie par ses fonctions et les appareils qui sont utilisés dans cette couche. Il y a des opinions différentes concernant le nombre de couches dans l'IoT. Cependant, selon de nombreux chercheurs, l'IoT fonctionne principalement sur trois niveaux appelés couches Perception, Réseau et Application. Chaque couche IoT est associée à des problèmes de sécurité inhérents. Figure 1 montre le cadre architectural de base à trois couches de l'IoT en ce qui concerne les appareils et les technologies qui englobent Chaque couche.

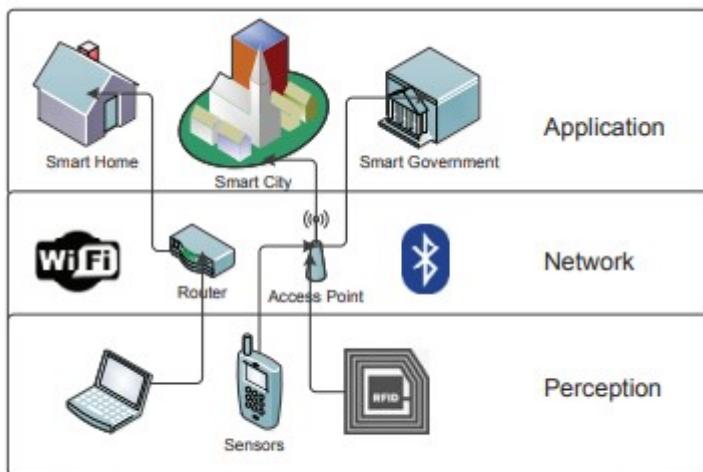


Figure 1. Three-layer IoT architecture.

### 1. Couche de perception

La couche de perception est également appelée couche «Capteurs» dans IoT. Le but de cette couche est d'acquérir les données de l'environnement à l'aide de capteurs et d'actionneurs. Cette couche détecte, recueille et traite les informations puis les transmet à la couche réseau. Cette couche exécute également le nœud IoT collaboration dans les réseaux locaux et à courte portée.

### 2. Couche réseau

La couche réseau de l'IoT remplit la fonction de routage des données et la transmission à différents hubs et appareils IoT sur le L'Internet. À cette couche, plates-formes de cloud computing, Internet, les passerelles, les dispositifs de commutation et de routage, etc. fonctionnent en utilisant certaines des technologies très récentes telles que le WiFi, LTE, Bluetooth, 4G, Zigbee etc. Les passerelles réseau servent de médiateur entre différents nœuds IoT par agrégation, filtrage, et la transmission de données vers et depuis différents capteurs.

### 3. Couche d'application

La couche application garantit l'authenticité, l'intégrité et la confidentialité des données. À ce niveau, l'objectif de l'IoT ou la création d'un environnement intelligent est réalisée.

## **II. Défis de sécurité dans chaque couche de l'IoT**

Chaque couche IoT est sensible aux menaces de sécurité et aux attaques. Ceux-ci peuvent être actifs ou passifs et peuvent provenir de sources externes ou réseau interne suite à une attaque du Insider. Une attaque active arrête directement le service tandis que le type passif surveille les informations du réseau IoT sans entraver son service. À chaque couche, les appareils et services IoT sont sensibles aux attaques par déni de service (DoS), qui font l'appareil, la ressource ou le réseau indisponible aux personnes autorisées utilisateurs. Les sections suivantes présentent une analyse détaillée des Problèmes de sécurité par rapport à chaque couche.

### **1. Couche de perception**

Il y a trois problèmes de sécurité dans la couche de perception IoT. La première est la force des signaux sans fil. La plupart des signaux sont transmis entre les nœuds de capteurs de l'IoT à l'aide du sans-fil technologies dont l'efficacité peut être compromise par vagues dérangeantes. Deuxièmement, le nœud de capteur dans les appareils IoT peut être intercepté non seulement par le propriétaire mais aussi par les attaquants car les nœuds IoT fonctionnent généralement en externe et en environnements extérieur, conduisant à des attaques physiques sur les capteurs IoT et appareils dans lesquels un attaquant peut altérer le matériel composant de l'appareil. Troisièmement, la nature inhérente de topologie de réseau qui est dynamique car les nœuds IoT sont souvent déplacés à différents endroits. La couche de perception IoT se compose principalement de capteurs et de RFID, grâce auxquels leur capacité de stockage, consommation d'énergie et calcul les capacités sont très limitées, ce qui les rend sensibles à de nombreux types de menaces et d'attaques. La confidentialité de cette couche peut facilement être exploitée par Replay Attack qui peut être effectué par usurpation, modification ou rejouer les informations d'identité de l'un des appareils dans l'IoT. Ou l'attaquant peut obtenir la clé de chiffrement en analysant le temps nécessaire pour effectuer le chiffrement, ce que l'on appelle Attaque chronométrée. Une autre attaque menaçant la confidentialité est lorsque l'attaquant prend le contrôle du nœud et capture toutes informations et données qui sont essentiellement une attaque de capture de nœud. L'attaquant peut ajouter un autre nœud au réseau qui menace le l'intégrité des données de cette couche en envoyant des données malveillantes. Cela peut également conduire à une attaque DoS, en consommant l'énergie de les nœuds du système et le priver du mode veille que les nœuds utilisent pour économiser l'énergie. Les problèmes de sécurité énumérés ci-dessus au niveau de la couche de perception peuvent être adressés à l'aide du cryptage (qui peut être point à point ou de bout en bout), authentification (pour vérifier la véritable identité de l'expéditeur) et contrôle d'accès.

### **2. couche réseau**

Comme mentionné précédemment, la couche réseau de l'IoT est également sensible aux attaques DoS. Outre les attaques DoS, l'adversaire peut également attaquer la confidentialité et la vie privée à couche réseau par analyse du trafic, écoute clandestine et surveillance passive. Ces attaques ont une forte probabilité d'occurrence en raison des mécanismes d'accès à distance et des données d'échange d'appareils. La couche réseau est très sensible aux attaques Man-in-the-Middle, qui peut être suivie d'écoutes clandestines. Si le matériel de saisie des appareils est écouté, le canal de communication sécurisé sera complètement compromis. Le mécanisme d'échange de clés L'IoT doit être suffisamment sécurisé pour empêcher tout intrus d'écoute clandestine, puis vol d'identité.

La communication dans l'IoT est différente de celle de l'Internet car il ne se limite pas à la machine à l'homme. Cependant, la fonction de communication de machine à machine que l'IoT introduit a un problème de sécurité de compatibilité. L'hétérogénéité des composants du réseau le rend difficile

d'utiliser les protocoles réseau actuels tels quels, et toujours produire des mécanismes de protection efficaces. Les attaquants peuvent également profiter du fait que tout est connecté dans l'ordre pour obtenir plus d'informations sur les utilisateurs et utiliser ces informations pour de futures activités criminelles. Protéger le réseau est important dans l'IoT, mais aussi la protection des objets dans le réseau est tout aussi importante. Les objets doivent avoir la capacité de connaître l'état du réseau et la capacité de se protéger de toute attaque contre le réseau. Ce peut être réalisé en ayant de bons protocoles ainsi que des logiciels qui permettent aux objets de répondre à toutes les situations et comportements qui peuvent être considérés comme anormaux ou peuvent affecter leur sécurité.

### **3. Couche d'application**

Étant donné que l'IoT n'a toujours pas de politiques mondiales et normes qui régissent l'interaction et le développement d'applications, il existe de nombreux problèmes liés à l'application Sécurité. Différentes applications ont une authentification différente mécanismes, ce qui rend l'intégration de tous très difficile d'assurer la confidentialité des données et l'authentification de l'identité. La grande quantité d'appareils connectés partageant des données surchargées ceux qui analysent les données, ce qui peut avoir un impact important sur la disponibilité des services. Un autre problème qui doit être pris en compte lors de la conception de l'application dans l'IoT est la façon dont différents utilisateurs interagissent avec eux, la quantité de données qui seront révélées, et qui sera responsable de la gestion de ces applications. Les utilisateurs doivent disposer d'outils pour les données qu'ils souhaitent divulguer et doivent être conscient de la manière dont les données seront utilisées, par qui et quand.

## **III. Notre Architecture IoT**

Au cours des dernières années, le système de santé mobile basé sur l'IoT a attiré l'attention de plusieurs chercheurs pour étudier le potentiel avantages et défis du futur du système de soins de santé. Le déploiement de la technologie IoT dans le domaine médical est très dépendant des technologies de l'information et de la communication (TIC). Des chercheurs concluent que la portée de la télémédecine continuera de s'étendre à l'avenir en raison des dernières évolutions des nanotechnologies, traitement de l'information et la communication sans fil. Une plateforme d'acquisition et de gestion des connaissances pour une santé personnalisée basée sur l'Internet des objets (IoT) c'est Le travail de présent. Le système utilise plusieurs capteurs pour lire le rythme cardiaque, température corporelle, etc., et transmet les données surveillées à une infrastructure cloud via smartphone . Analyse des problèmes de sécurité et de confidentialité de l'IoT du point de vue des soins de santé et propose un modèle de sécurité collaborative intelligent pour minimiser les risques de sécurité. Une architecture de sécurité pour un système de cybersanté mobile pour gérer le service de prescription de médicaments via les dossiers de santé personnels électroniques est déjà présentée par des chercheurs .L'architecture prend en charge l'application mobile d'e-santé (m-health) grâce à une technologie RFID.

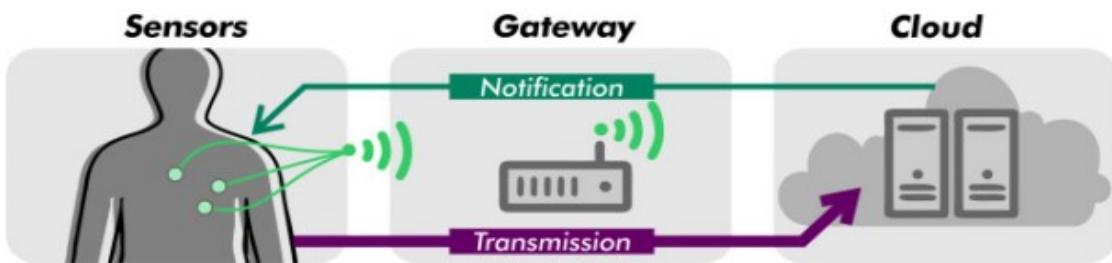


Fig. 1. IoT-based system for remote patient monitoring.

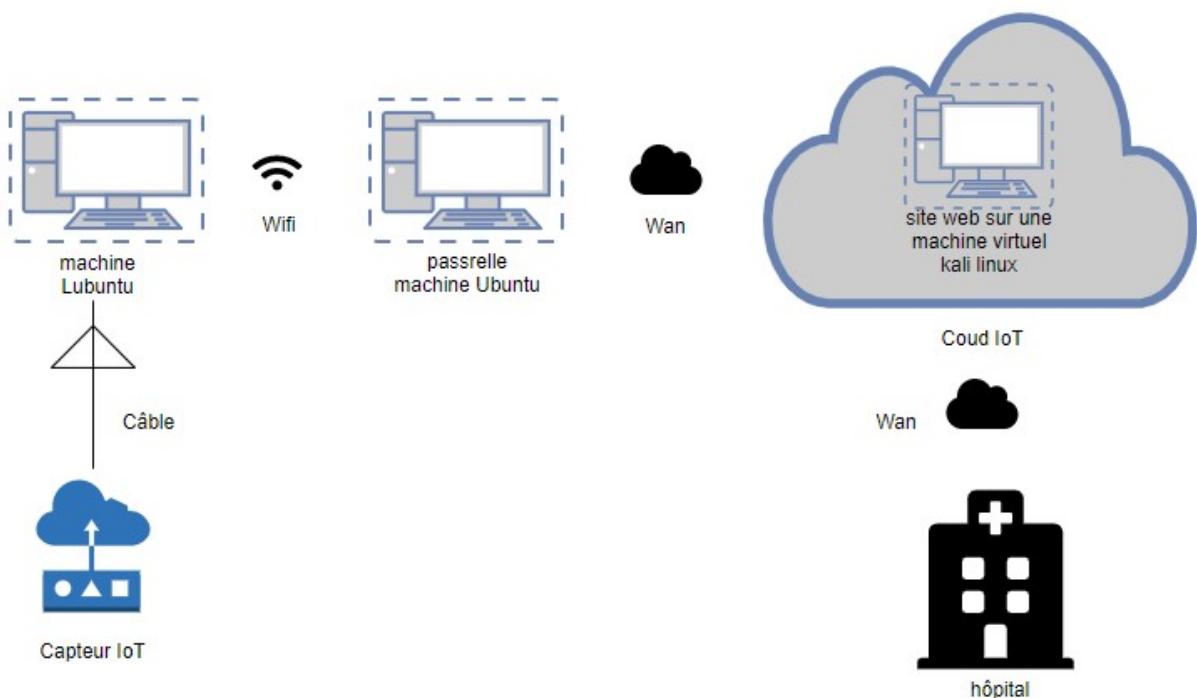


Fig 2 : Notre architecture IoT proposée.

La figure 2 montre notre architecture. Notre travail consiste à collecter des données de Fréquence Cardiaque à partir d'un capteur connecté avec Arduino. Ces données sont envoyées au Smartphone ou à un ordinateur (machine Virtuel Lubuntu dans notre architecture) via un câble. Ce Smartphone ou ordinateur est connecté avec une passerelle (machine Virtuel Ubuntu dans notre architecture) via réseau sans fil pour transmettre les données surveillées à une infrastructure cloud (machine Virtual Kali Linux hébergeant un site web dans notre architecture pour le stockage des données médicales). La simulation est faite avec GNS3. Nous ajoutons des mécanismes pour garantir la confidentialité, l'intégrité, l'authentification et la détection d'intrusion, comme l'authentification et le cryptage des données par Poly1305 et AES-128, l'installation et la configuration du système de détection d'intrusion Snort, puis nous ajoutons une test (avec une machine Virtual Kali Linux qui essaie de faire une scanne des ports avec nmap), en plus la sécurisation de notre site web créé pour le stockage des données médicales par SSL/TLS.

## IV. Implémentation d'architecture IoT sécurisé

Nous commençons notre implémentation par la collection des données par un capteur cardiofréquencemètre .

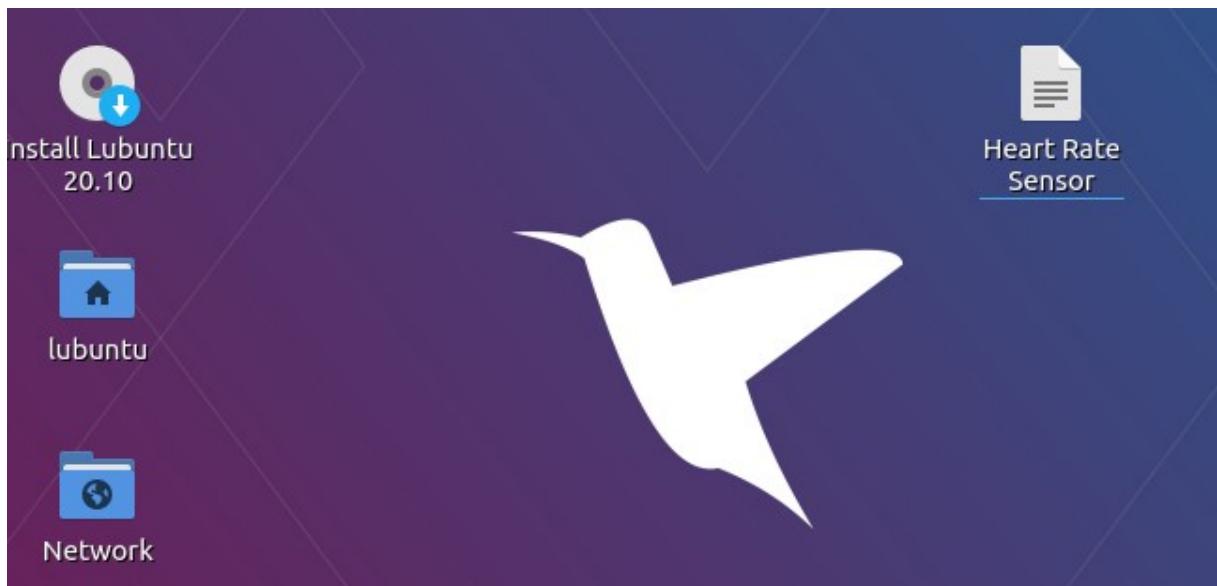
Le cardiofréquencemètre : Aussi appelé « cardio », le cardiofréquencemètre est un capteur de fréquence cardiaque, utilisé depuis longtemps chez les sportifs pour leur permettre de suivre régulièrement leur aptitude cardiaque à l'effort. En effet c'est le champion cycliste italien Moser qui utilisa pour la première fois le relevé de sa fréquence cardiaque pour battre le record de l'heure sur piste (1984) en utilisant un rythmostat.

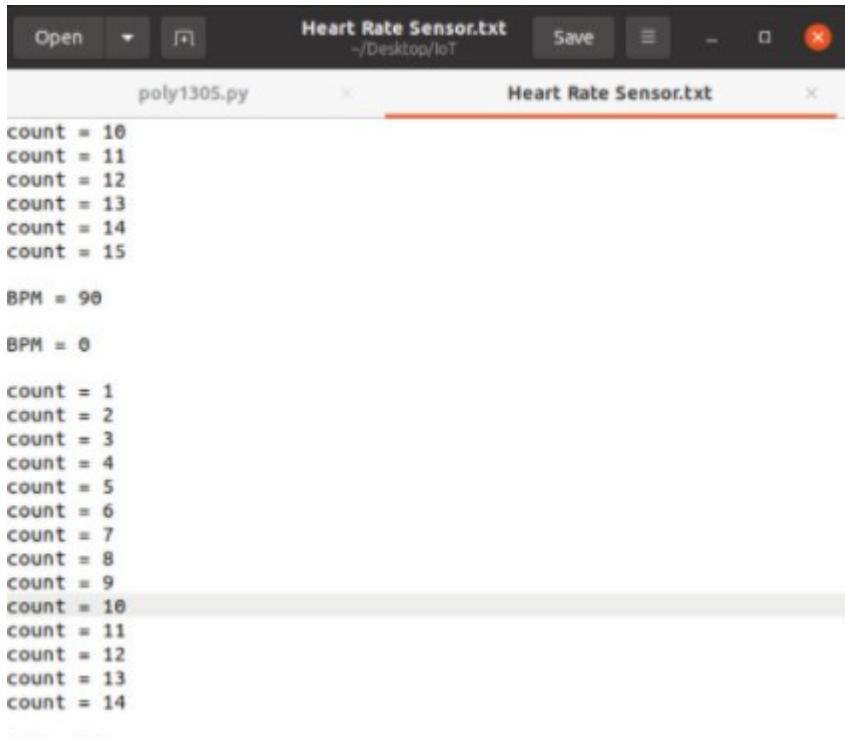
Le cardiofréquencemètre se compose le plus souvent d'un capteur/émetteur posé sur le thorax, maintenu par des ceintures élastiques et d'un récepteur qui a la forme d'une montre et que l'on porte au poignet. Le capteur enregistre la fréquence cardiaque du cœur et transmet au récepteur le signal en affichant cette fréquence cardiaque.

Maintenant nous programmons l'Arduino pour communiquer avec le Le cardiofréquencemètre , lui faire prendre une mesure, récupérer le résultat sous forme d'une trame de 40 bits et la convertir pour obtenir la mesure cardiaque relative.

Après la collecte des données nous connectons capteur+arduino avec une machine Lubuntu avec un câble.

Les données se trouvent actuellement à la machine virtual lubuntu.





A screenshot of a terminal window titled "Heart Rate Sensor.txt" located at "/Desktop/IoT". The window contains two tabs: "poly1305.py" and "Heart Rate Sensor.txt", with "Heart Rate Sensor.txt" being the active tab. The content of the active tab shows a sequence of data lines:

```
count = 10
count = 11
count = 12
count = 13
count = 14
count = 15

BPM = 98

BPM = 0

count = 1
count = 2
count = 3
count = 4
count = 5
count = 6
count = 7
count = 8
count = 9
count = 10
count = 11
count = 12
count = 13
count = 14
```

## 1. Authentification et l'envoie des données chiffrées

### A. Authentification

L'authentification pour un système informatique est un processus permettant au système de s'assurer de la légitimité de la demande d'accès faite par une entité (être humain ou un autre système...) afin d'autoriser l'accès de cette entité à des ressources du système (systèmes, réseaux, applications...) conformément au paramétrage du contrôle d'accès. L'authentification permet donc, pour le système, de valider la légitimité de l'accès de l'entité, ensuite le système attribue à cette entité les données d'identité pour cette session (ces attributs sont détenus par le système ou peuvent être fournis par l'entité lors du processus d'authentification). C'est à partir des éléments issus de ces deux processus que l'accès aux ressources du système pourra être paramétré (contrôle d'accès).

### B. Chiffrements des données

Le chiffrement (ou cryptage) est un procédé de cryptographie grâce auquel on souhaite rendre la compréhension d'un document impossible à toute personne qui n'a pas la clé de (dé)chiffrement. Ce principe est généralement lié au principe d'accès conditionnel.

Bien que le chiffrement puisse rendre secret le sens d'un document, d'autres techniques cryptographiques sont nécessaires pour communiquer de façon sûre. Pour vérifier l'intégrité ou l'authenticité d'un document, on utilise respectivement un Message Authentication Code (MAC) ou une signature numérique. On peut aussi prendre en considération l'analyse de trafic dont la communication peut faire l'objet, puisque les motifs provenant de la présence de communications peuvent faire l'objet d'une reconnaissance de motifs. Pour rendre secrète la présence de communications, on utilise la stéganographie. La sécurité d'un système de chiffrement doit reposer

sur le secret de la clé de chiffrement et non sur celui de l'algorithme. Le principe de Kerckhoffs suppose en effet que l'ennemi (ou la personne qui veut déchiffrer le message codé) connaisse l'algorithme utilisé.

### C. Poly1305

Poly1305 est un code d'authentification de message cryptographique (MAC) créé par Daniel J. Bernstein . Il peut être utilisé pour vérifier l'intégrité des données et l'authenticité d'un message. Une variante du Poly1305 de Bernstein qui ne nécessite pas d'AES a été normalisée par l'Internet Engineering Task Force dans la RFC 8439.

La proposition originale, Poly1305-AES, qui utilise la fonction de chiffrement AES (avant) comme source de pseudo - aléatoire, calcule un authentificateur de 128 bits (16 octets) d'un message de longueur variable. En plus du message, il nécessite une clé AES de 128 bits, une clé supplémentaire de 128 bits  $r$  (avec 106 bits de clé effectifs) et un nonce de 128 bits qui doit être unique parmi tous les messages authentifiés avec la même clé. Le message est découpé en blocs de 16 octets qui deviennent des coefficients d'un polynôme évalué en  $r$ , modulo le nombre premier  $2^{130}-5$ . Le code d'authentification est la somme de cette évaluation polynomiale, plus une valeur pseudo-aléatoire calculée en passant le nonce à travers le chiffrement par bloc AES. Le nom Poly1305-AES est dérivé de son utilisation de l'évaluation polynomiale, du module  $2^{130}-5$  et de l'AES. Dans NaCl , Poly1305 est utilisé avec Salsa20 à la place d'AES, et dans TLS et SSH, il est utilisé avec la variante ChaCha20 de la même chose. Google a sélectionné Poly1305 avec le chiffrement symétrique ChaCha20 de Bernstein en remplacement de RC4 en TLS / SSL , qui est utilisé pour la sécurité Internet. La mise en œuvre initiale de Google est utilisée pour sécuriser le trafic https ( TLS / SSL ) entre le navigateur Chrome sur les téléphones Android et les sites Web de Google. L'utilisation de ChaCha20 / Poly1305 a été normalisée dans la RFC 7905.

La sécurité de Poly1305-AES est très proche de l'algorithme de chiffrement par bloc AES sous-jacent. Par conséquent, le seul moyen pour un attaquant de casser Poly1305-AES est de casser AES. Poly1305 .

### D. AES-128

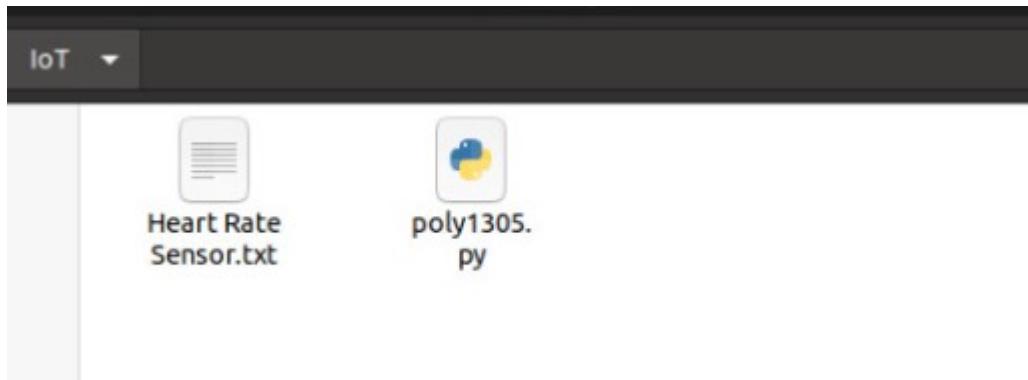
Maintenant que nous savons ce qu'est la norme AES, parlons un peu de la version 128 bits. Avec la création de la norme AES-128, la clé a atteint une longueur de 128 bits, d'où le suffixe. Sans clé cryptographique, les données chiffrées sont complètement incompréhensibles.

En appliquant 10 cycles de chiffrement lors de son processus de compression de données, la norme AES-128 découpe les données en morceaux et les mélange selon une méthode déterminée par le sous-type de chiffrement. À ce stade, une clé est générée, elle permet à la personne recevant les données de démêler le chiffrement.

Les algorithmes de clé symétrique (comme la norme AES-128) utilisent la même clé pour chiffrer et déchiffrer le message. Ils sont donc plus rapides que les chiffrements asymétriques et donc parfaits pour le chiffrement de données VPN.

## E. Implémentation de l'authentification et le chiffrement par poly1305-AES-128

Après la collecte des données à partir du capteur et arduino nous avons transmis les données à l'ordinateur (machine Virtual Lubuntu) par un câble, maintenant nous envoyons ces données au passerelle (machine Virtuel Ubuntu) . Mais l'envoie se fait de manière sécurisée avec des mécanismes d'authentification et cryptage des données pour assurer l'intégrité, la confidentialité et l'authentification.



Voici le script qui permet de générer une balise MAC (notons comment le nonce est généré automatiquement au hasard):

```
from Crypto.Hash import Poly1305
from Crypto.Cipher import AES
from binascii import unhexlify

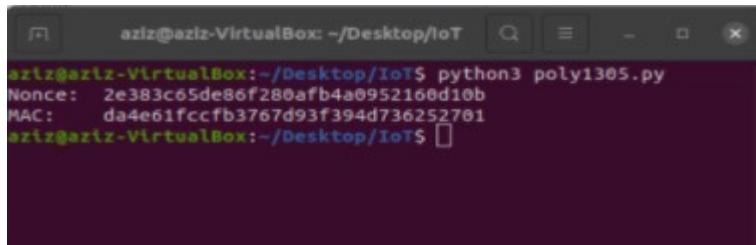
f= open("Heart Rate Sensor.txt","r")
contents = f.read()
contents=bytes(contents, "ascii")

secret = b'Thirtytwo very very secret bytes'
mac = Poly1305.new(key=secret, cipher=AES)
mac.update(contents)
print("Nonce: ", mac.nonce.hex())
print("MAC:   ", mac.hexdigest())
f.close()
```

- 
- **key** ( bytes / bytearray / memoryview ) - La clé de 32 octets pour l'objet Poly1305.
  - **cipher** (module from Crypto.Cipher) - L'algorithme de chiffrement à utiliser pour dériver la paire de clés Poly1305 (r,s). Cela ne peut être que Crypto.Cipher.AES ou Crypto.Cipher.ChaCha20.

- **nonce** ( bytes / bytearray / memoryview ) - Facultatif. Valeur non répétable à utiliser pour le MAC de ce message. Il doit faire 16 octets pour AESet 8 ou 12 octets pour ChaCha20. S'il n'est pas passé, un nonce aléatoire est créé; vous le trouverez dans l' `nonce` attribut du nouvel objet.
- **data** ( bytes / bytearray / memoryview ) - Facultatif. Le tout premier morceau du message à authentifier. Cela équivaut à un appel précoce à `update()`.

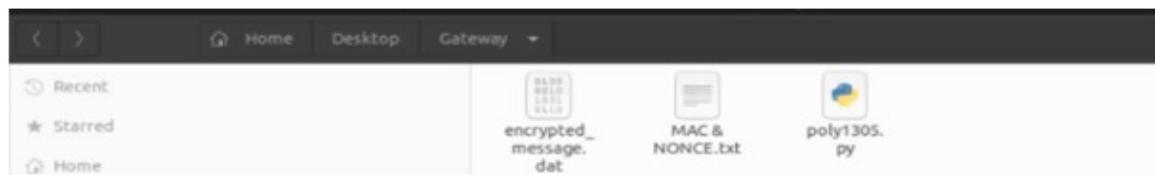
La génération d'une balise MAC avec nonce par l'exécution de script sur la machine Lubuntu:



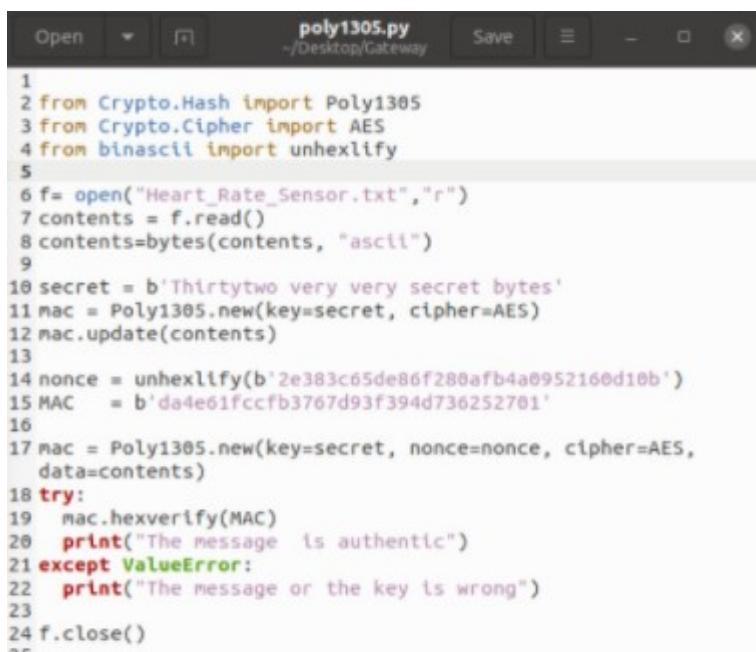
```
aziz@aziz-VirtualBox: ~/Desktop/IoT$ python3 poly1305.py
Nonce: 2e383c65de86f280afb4a0952160d10b
MAC: da4e61fccfb3767d93f394d736252701
aziz@aziz-VirtualBox:~/Desktop/IoT$
```

Maintenant on va passer à la passerelle pour la validation de Mac.

Comme vous voyez on a un fichier nommé `MAC&nONCE.txt` qui contient le Mac et le nonce pour la validation de l'authentification :



Voici le script python montrant comment valider le MAC ci-dessus:



```
1 from Crypto.Hash import Poly1305
2 from Crypto.Cipher import AES
3 from binascii import unhexlify
4
5 f= open("Heart_Rate_Sensor.txt","r")
6 contents = f.read()
7 contents=bytes(contents, "ascii")
8
9 secret = b'Thirtytwo very very secret bytes'
10 mac = Poly1305.new(key=secret, cipher=AES)
11 mac.update(contents)
12
13 nonce = unhexlify(b'2e383c65de86f280afb4a0952160d10b')
14 MAC = b'da4e61fccfb3767d93f394d736252701'
15
16 mac = Poly1305.new(key=secret, nonce=nonce, cipher=AES,
17 data=contents)
18 try:
19     mac.hexverify(MAC)
20     print("The message is authentic")
21 except ValueError:
22     print("The message or the key is wrong")
23
24 f.close()
~
```

La validation de Mac en utilisant le script Poly1305 :

Comme le voit la Mac est authentique donc l'authentification se fait de manière normale.

```
aziz@aziz-VirtualBox:~/Desktop/Gateway$ python3 poly1305.py
The message is authentic
aziz@aziz-VirtualBox:~/Desktop/Gateway$
```

Après l'authentification. On va passer au cryptage de fichier Heart\_Rate\_Sensor.txt avec AES-128, nous utilisons la commande openssl, comme vous voyez le fichier de sortie nommé encrypt de message .dat :

```
aziz@aziz-VirtualBox:~/Desktop/IoT$ openssl enc -aes-128-cbc -in ~/Desktop/IoT/Heart_Rate_Sensor.txt -out encrypted_message.dat
enter aes-128-cbc encryption password:
verifying - enter aes-128-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
aziz@aziz-VirtualBox:~/Desktop/IoT$
```

Maintenant on va déchiffrer le message avec AES-128 comme voyez on a un troisième fichier qui est ajouté au liste des fichiers, c'est le fichier qui contient les données en clair.

```
aziz@aziz-VirtualBox:~/Desktop/Gateway$ openssl enc -aes-128-cbc -d -in encrypted_message.dat > Heart_Rate_Sensor.txt
enter aes-128-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
aziz@aziz-VirtualBox:~/Desktop/Gateway$
```

Voilà le contenu en clair :

```
aziz@aziz-VirtualBox:~/Desktop/Gateway$ openssl enc -aes-128-cbc -d -in encrypted_message.dat > Heart_Rate_Sensor.txt
enter aes-128-cbc decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
aziz@aziz-VirtualBox:~/Desktop/Gateway$
```

1 count = 10  
2 count = 11  
3 count = 12  
4 count = 13  
5 count = 14  
6 count = 15  
7  
8 BPM = 90  
9  
10 BPM = 0  
11  
12 count = 1  
13 count = 2  
14 count = 3  
15 count = 4  
16 count = 5  
17 count = 6  
18 count = 7  
19 count = 8  
20 count = 9  
21 count = 10  
22 count = 11  
23 count = 12  
24 count = 13  
25 count = 14  
26  
27 BPM = 84  
28  
29 BPM = 0

## 2. L'envoi des au Cloud IoT .

Dans cette partie nous essayons d'envoyer les données cardiaques qui se trouvent actuellement au niveau passerelle à un site web [www.life-care.com](http://www.life-care.com) qui se trouve au niveau d'une machine Virtuel kali linux.

L'adresse IP de machine qui héberge le site web [www.life-care.com](http://www.life-care.com) :

```
kali㉿kali:~$ sudo ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.1.4 netmask 255.255.255.0 broadcast 192.168.1.255
        ether 08:00:27:5c:65:26 txqueuelen 1000 (Ethernet)
          RX packets 40439 bytes 2431758 (2.3 MiB)
```

### a) Installation de serveur web apache

Apache existe depuis plus de 20 ans et même s'il perd des parts de marché depuis quelques années, il reste le serveur web le plus utilisé au monde.

Commençons par installer Apache et les autres packages comme PHP par la commande :

```
kali㉿kali:/var/www/html$ sudo apt install apache2 php libapache2-mod-php mysql-server php-mysql
Reading package lists... Done
Building dependency tree
Reading state information... Done
Package mysql-server is not available, but is referred to by another package
or
```

### b) Configurons notre site web life-care sous Apache

Apache est capable de gérer plusieurs sites web sur la même machine, c'est ce qu'on appelle des hôtes virtuels ou virtual hosts. Dans la requête que fera le client sur la machine, un entête HTTP précisera si le client veut plutôt consulter toto.com ou tata.org qui sont tous les deux hébergés sur la machine. La configuration de chaque virtual host se fait dans un fichier \*.conf dans /etc/apache2/sites-available/. Créons donc le fichier [/etc/apache2/sites-available/01-www.life-care.com.conf](#) contenant les paramètres suivants :

```
GNU nano 4.9.3 /etc/apache2/sites-available/life-care.com.conf Modified
<VirtualHost *:80>
    ServerName life-care.com
    ServerAlias www.life-care.com
    ServerAdmin webmaster@life-care.com
    DocumentRoot /srv/web/life-care.com/www
    <Directory /srv/web/life-care.com/www>
        Options -Indexes +FollowSymLinks +MultiViews
        AllowOverride none
        Require all granted
    </Directory>
    ErrorLog /var/log/apache2/error.life-care.com.log
    CustomLog /var/log/apache2/access.life-care.com.log combined
</VirtualHost>
```

Pour pouvoir tester notre site web, il nous reste à :

La racine de notre arborescence et y déposer un premier fichier `.html`

```
kali@kali:/var/www/html$ ls  
contact.php  css  fonts  images  index.html  js  style.css  
kali@kali:/var/www/html$
```

Activons la configuration de notre VirtualHost :

Cette commande crée automatiquement le lien symbolique du répertoire `sites-enabled` vers le répertoire `sites-available` .

```
kali@kali:~$ sudo a2enmod  
Considering dependency set  
Module setenvif already en  
Considering dependency mim
```

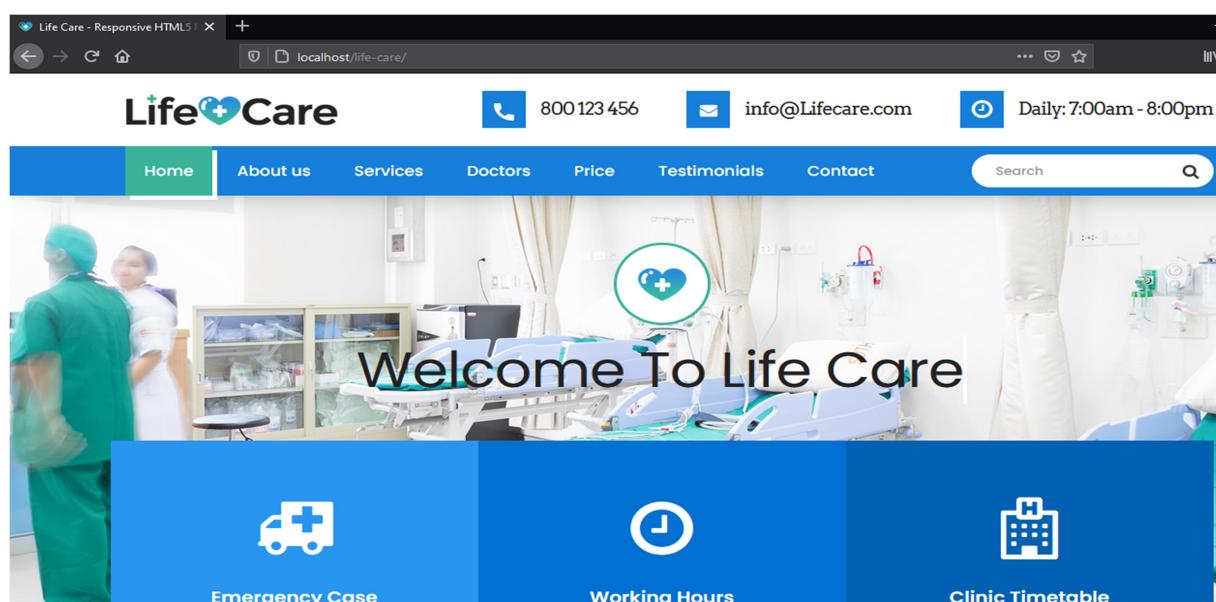
Rechargeons maintenant la configuration d'Apache pour prendre en compte nos modifications :

```
kali@kali:/var/www/html$ sudo systemctl start apache2  
kali@kali:/var/www/html$
```

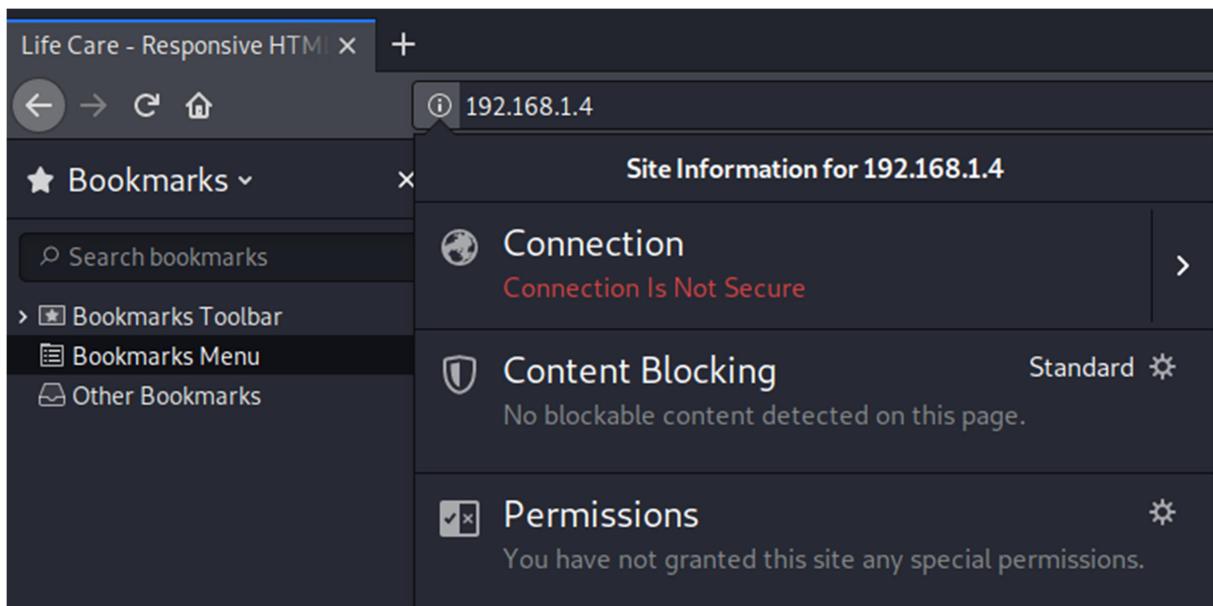
Notre serveur est prêt. Maintenant si nous ajoutons `www.life-care.com` en alias à notre serveur dans le fichier `/etc/hosts` de notre client (passerelle kali linux), nous pouvons admirer la page d'accueil de notre site dans notre navigateur.

Maintenant nous avons accédé au site web à travers la machine passerelle.

```
kali@kali:~$ sudo ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500  
      inet 192.168.1.2  netmask 255.255.255.0  broadcast 192.168.1.255  
        ether 08:00:27:5c:65:26  txqueuelen 1000  (Ethernet)
```



Le problème c'est que notre site web n'est pas sécurisé. Une attaque peut récupérer facilement les données des patients.



### c) Sécurisons notre serveur web

Nous venons de mettre en ligne un site web qui nous permettra de partager des informations avec le monde entier. Malheureusement, le monde n'est pas constitué que de personnes bien intentionnées. Dans ce chapitre, je vais sécuriser notre site web avec trois façons de renforcer la sécurité :

- En évitant de donner des informations sur votre architecture
- En restreignant l'accès à certaines ressources
- En utilisant le protocole HTTPS

#### ❖ Sécurisez la configuration de notre serveur web

En matière de sécurité, il est toujours préférable de donner le moins d'informations possibles sur son système à de potentiels attaquants. Or, par défaut, Apache renvoie des entêtes HTTP contenant le nom et la version du serveur web ainsi que le nom du système d'exploitation : autant d'informations qui peuvent intéresser une personne mal intentionnée.

Pour éviter ça, kali linux Nous facilite la vie en regroupant les directives sensibles en terme de sécurité dans le fichier `/etc/apache2/conf-available/security.conf`. Dans ce fichier, configurons les directives comme ceci, c'est la configuration qui donne le moins d'informations :

```
GNU nano 4.9.3      /etc/apache2/conf-available/security.conf      Modified
#ServerTokens Minimal
ServerTokens prod
#ServerTokens Full
#
# Optionally add a line containing the server version and virtual host
# name to server-generated pages (internal error documents, FTP directory
# listings, mod_status and mod_info output etc., but not CGI generated
# documents or custom error documents).
# Set to "EMail" to also include a mailto: link to the ServerAdmin.
# Set to one of: On | Off | EMail
#ServerSignature Off
ServerSignature off
```

### ❖ Restreignez l'accès aux ressources sensibles

Apache permet de restreindre l'accès à des ressources en fonction d'un grand nombre de critères. Les modes d'authentification les plus courants sont certainement par IP et par login/mot de passe.

Commençons par créer les données à protéger :

```
kali㉿kali:~$ sudo mkdir /var/www/html/www.life-care.com/top_secret
kali㉿kali:~$ echo "voici mon secret" > /var/www/html/www.life-care.com/top_
secret/index.html
```

Pour protéger notre répertoire `top_secret` , ajoutons la section suivante à la configuration de notre `<VirtualHost />` dans `/etc/apache2/sites-available/www.life-care.com.conf` et rechargeons la configuration Apache :

```
/etc/apache2/sites-available/www.life-care.com.conf      Modified
<Directory /var/www/html/www.life-care.com/top_secret>
AuthType Basic
    AuthName "Accès restreint aux utilisateurs authentifiés"
    AuthBasicProvider file
    AuthUserFile "/etc/apache2/passwords"
    Require ip 192.168.1.4
    Require valid-user
</Directory>
```

Ici, toutes les requêtes venant de l'IP 192.168.1.4 sont automatiquement acceptées. Une fenêtre d'authentification est proposée aux requêtes venant d'autres IP. Un couple login/mot de passe valide doit être entré pour que l'accès soit autorisé.

Les directives `Auth*` précisent le fonctionnement de cette autorisation par mot de passe :

`AuthType Basic` : indique une authentification par login/mot de passe simple

`AuthName` : permet de définir le titre de la fenêtre d'authentification qui sera affiché aux clients

`AuthBasicProvider file` : indique que les comptes utilisateurs sont définis dans un fichier

`AuthUserFile` : précise le fichier où sont enregistrés les comptes utilisateurs

Il Nous faut donc maintenant créer ce fameux fichier de mots de passes grâce à la commande htpasswd :

```
kali㉿kali:~$ sudo htpasswd -c /etc/apache2/passwords issam
New password:
Re-type new password:
Adding password for user issam
kali㉿kali:~$
```

Nous pouvons tester en nous connectant à [http://www.life-care.com/top\\_secret/](http://www.life-care.com/top_secret/) depuis notre machine cliente :



### ❖ Sécurisons nos connexions grâce à HTTPS

Il y a plusieurs intérêts à configurer notre site en HTTPS :

- nos connexions sont chiffrées ce qui empêche de pouvoir intercepter nos mots de passe mais aussi nos informations de session, nos cookies, etc.
- le serveur est identifié par un certificat ce qui permet au client de pouvoir s'assurer que le serveur auquel il se connecte est bien celui qu'il prétend être
- HTTPS est maintenant la norme et les moteurs de recherche, Google en tête, référencent mieux les sites HTTPS que les sites HTTP.

Pour pouvoir configurer notre site en HTTPS, nous devrons générer un certificat pour notre serveur. Tout le monde peut générer un certificat mais pour que les clients soient sûrs que le certificat qui leur est présenté à la connexion correspond au serveur qu'ils veulent joindre, ils font confiance à une autorité de certification pour valider le certificat du serveur.

**Le protocole TLS** permet à deux machines de communiquer de manière sécurisée. Les informations échangées entre les deux machines sont de ce fait pratiquement inviolables. Il doit assurer l'authentification du serveur grâce à un certificat. La confidentialité des données grâce au chiffrement et l'intégrité des données.

**Un certificat** permet de fournir diverses informations concernant l'identité de son détenteur (la personne qui publie les données). Ce certificat s'accompagne d'une **clé publique** qui est indispensable pour que la communication entre les machines soit chiffrée.

Afin de garantir l'authenticité du certificat, ce dernier est signé numériquement provenant soit par une autorité de certification (Société spécialisée dans la certification) soit par le détenteur du certificat lui-même. Dans ce dernier cas, on parlera de certificat auto-signé.

Dans la plupart des cas, l'obtention d'un certificat certifié par une AC (autorité de certification) ayant un prix assez élevé, les webmasters auront tendance à vouloir signer eux-même leur certificat. Ce faisant, il est à noter que dans ce cas, le certificat ne sera pas reconnu par les navigateurs web comme étant certifié.

CA Cert permet d'obtenir des certificats gratuits. Il vous faudra néanmoins installer le certificat racine dans votre navigateur.

Let's encrypt permet également d'obtenir des certificats gratuits. En outre Let's Encrypt fournit l'application cerbot qui simplifie grandement la création et la gestion des certificats.

## Activation du module SSL/TLS

Pour que le protocole TLS puisse fonctionner avec le Serveur HTTP Apache2, il faut activer le module SSL avec la commande :

```
kali㉿kali:~$ sudo a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
```

Puis recharger la configuration d'Apache2 :

```
kali㉿kali:~$ systemctl restart apache2
kali㉿kali:~$
```

## Création du certificat avec Let's encrypt

En voici, un récapitulatif. Il faut d'abord installer le dépôt ppa officiel (sans risques) puis installer le paquet certbot :

```
kali㉿kali:~$ sudo apt install software-properties-common
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required
:
```

```
kali㉿kali:~$ sudo add-apt-repository ppa:certbot/certbot
The PPA has been DEPRECATED.

To get up to date instructions on how to get certbot for your systems, please see https://certbot.eff.org/docs/install.html.
```

```
kali㉿kali:~$ sudo apt install certbot
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required
:
```

## La Génération de certificat avec Certbot

Le script est très bien fait, ce qui implique qu'il est possible de simplement lancer, pour un serveur avec apache:

```
kali㉿kali:~$ sudo certbot certonly --webroot -w /var/www/life-care -d life-care -d www.life-care.com
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator webroot, Installer None
Enter email address (used for urgent renewal and security notices)
(Enter 'c' to cancel): issam.njh@gmail.com

-----
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf. You must
agree in order to register with the ACME server. Do you agree?
-----
(Y)es/(N)o: █
```

Cette commande va générer un certificat valable pour les domaines (option -d) life-care.com et www. life-care, qui correspondent à un site web existant et fonctionnel placé dans le dossier racine (option -w) var/www/ life-care.

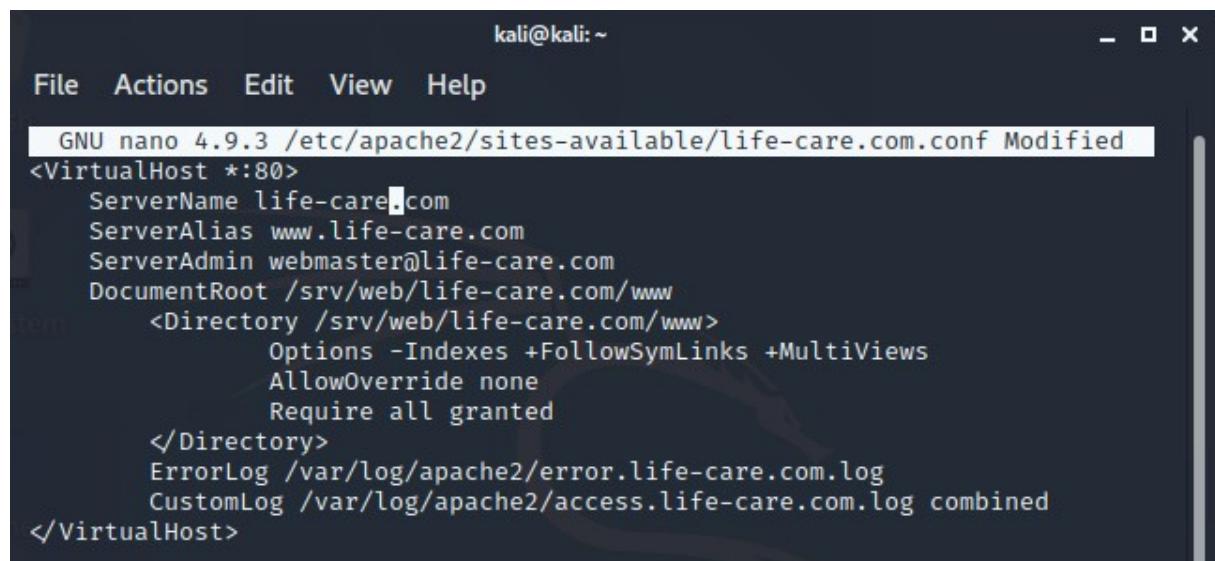
À l'issue de la commande précédente sera créé une arborescence sous /etc/letsencrypt qui contient nos certificats et des fichiers de configuration nécessaires aux procédures de renouvellement et de révocation. Sera créé également une tâche cron (/etc/cron.d/certbot) permettant de renouveler automatiquement les certificats avant qu'ils n'arrivent à échéance (les certificats Let'sEncrypt ne sont valables que 3 mois).

L'emplacement du certificat et de la clé privée est indiqué à la fin de la procédure, typiquement :

/etc/letsencrypt/live/ life-care.com/privkey.pem pour la clé privée  
/etc/letsencrypt/live/ life-care.com/fullchain.pem pour le certificat complet.

#### ❖ Configuration de l'hôte virtuel pour HTTPS

Ouvrons le fichier de configuration de votre hôte virtuel, [/etc/apache2/sites-available/life-care.com.conf](#) :



```
kali㉿kali: ~
File Actions Edit View Help
GNU nano 4.9.3 /etc/apache2/sites-available/life-care.com.conf Modified
<VirtualHost *:80>
    ServerName life-care.com
    ServerAlias www.life-care.com
    ServerAdmin webmaster@life-care.com
    DocumentRoot /srv/web/life-care.com/www
        <Directory /srv/web/life-care.com/www>
            Options -Indexes +FollowSymLinks +MultiViews
            AllowOverride none
            Require all granted
        </Directory>
        ErrorLog /var/log/apache2/error.life-care.com.log
        CustomLog /var/log/apache2/access.life-care.com.log combined
</VirtualHost>
```

Et ajoutons à la suite l'hôte virtuel pour le port 443 (port standard pour le HTTPS) :

```
File Actions Edit View Help
GNU nano 4.9.3 /etc/apache2/sites-available/life-care.com.conf Modified
<VirtualHost *:80>
    ServerName life-care.com
    ServerAlias www.life-care.com
    ServerAdmin webmaster@life-care.com
    DocumentRoot /srv/web/life-care.com/www
        <Directory /srv/web/life-care.com/www>
            Options -Indexes +FollowSymLinks +MultiViews
            AllowOverride none
            Require all granted
        </Directory>
        ErrorLog /var/log/apache2/error.life-care.com.log
        CustomLog /var/log/apache2/access.life-care.com.log combined
</VirtualHost>
<VirtualHost *:443>
    ServerName life-care.com
    ServerAlias www.life-care.com
    ServerAdmin webmaster@life-care.com
    DocumentRoot /srv/web/life-care.com/www
        <Directory /srv/web/life-care.com/www>
            Options -Indexes +FollowSymLinks +MultiViews
            AllowOverride none
            Require all granted
        </Directory>
```

Enregistrons le fichier et rechargeons la configuration d'Apache :

```
kali㉿kali:~$ systemctl restart apache2
kali㉿kali:~$
```

Notre site devrait maintenant être accessible en HTTP et en HTTPS :



Forcer la connexion en HTTPS :

Si nous voulons que notre site ne soit accessible qu'en HTTPS et que les internautes soient redirigés automatiquement, modifions ainsi notre fichier d'hôte virtuel :

```
GNU nano 4.9.3 /etc/apache2/sites-available/life-care.com.conf Modified
<VirtualHost *:80>
    ServerName life-care.com
    ServerAlias www.life-care.com
    ServerAdmin webmaster@life-care.com
    # Redirection 301 vers le site en HTTPS
    Redirect permanent / https://life-care.com/
    DocumentRoot /srv/web/life-care.com/www
```

Enregistrons le fichier et rechargeons la configuration d'Apache :

```
kali㉿kali:~$ systemctl restart apache2
kali㉿kali:~$ █
```

Maintenant on accède au site web pour envoyons les données médicaux :

The screenshot shows a web application for managing medical patients. At the top, there's a navigation bar with links for Services, Doctors, Price, Testimonials, Contact, and a search bar. Below the navigation, there are four service icons: a monitor with a heart rate graph labeled 'LARGE LABORATORY', a clipboard labeled 'DETAILED SPECIALIST', a capsule labeled 'FINE INFRASTRUCTURE', and a syringe labeled 'ANYTIME BLOOD BANK'. To the right, a sidebar titled '+ remote patients' contains fields for 'name' (issam najah), 'email' (issam.najah2802@gmail.com), 'days' (Sunday), 'time' (PM), 'doctor' (Mr.abdelaziz Iaaychi), and a file input field ('Choisir un fichier') containing 'Heart Rate Sensor.txt'. A message box below says 'les résultats d'aujourd'hui 01/01/2021'. At the bottom right is a 'Submit' button.

Les données sont stockées à la base de données :

<b>id</b>	<b>email</b>	<b>name</b>	<b>days</b>	<b>time</b>	<b>doctor</b>	<b>file</b>	<b>message</b>
1	issam.njh@gmail.com	issam najah	sunday	pm	Mr.abdelaziz Iaaychi	Heart_Rate_Sensor.txt	les résultats d'aujourd'hui 01/01/2021

### **3. SNORT –la détection d'intrusion Lightweight pour les réseaux**

#### **a. Qu'est-ce que la détection d'intrusion «légère»?**

Un système de détection d'intrusion léger peut facilement être déployé sur la plupart des nœuds d'un réseau, avec perturbation minimale des opérations. «IDS léger» doit être multiplateforme, avoir une faible empreinte système et être facilement configurée par les administrateurs système qui ont besoin de mettre en œuvre une solution de sécurité spécifique dans un court laps de temps. Ils peuvent être n'importe quel ensemble d'outils logiciels qui peuvent être assemblés et intégrés action en réponse à l'évolution des situations de sécurité. Les IDS légers sont petits, puissants et flexibles suffisamment pour être utilisé comme éléments permanents de l'infrastructure de sécurité du réseau.

Snort est bien adapté pour remplir ces rôles, pesant dans à environ 100 kilo-octets dans sa source compressée Distribution. Sur la plupart des architectures modernes, Snort prend seulement quelques minutes pour compiler et mettre en place, et peut-être encore dix minutes pour configurer et activer. Comparez cela avec de nombreux NIDS commerciaux, qui nécessitent des plates-formes dédiées et une formation des utilisateurs pour déployer d'une manière significative. Snort peut être configuré et laissé fonctionnant pendant de longues périodes sans nécessiter surveillance ou maintenance administrative, et peut donc également être utilisé comme partie intégrante de la plupart infrastructures de sécurité réseau.

#### **b. Qu'est-ce que Snort?**

Snort est un renifleur de paquets basé sur libpcap et enregistreur pouvant être utilisé comme réseau léger système de détection d'intrusion (NIDS). Il comporte des règles journalisation basée pour effectuer une correspondance de modèle de contenu et détecter une variété d'attaques et de sondes, telles que le tampon débordements, scans de ports furtifs, attaques CGI, Sondes SMB et bien plus encore. Snort a le temps réel capacité d'alerte, avec des alertes envoyées à syslog, Messages Server Message Block (SMB) «WinPopup», ou un fichier «d'alerte» distinct. Snort est configuré en utilisant commutateurs de ligne de commande et paquet Berkeley en option Filtrer les commandes. Le moteur de détection est programmé en utilisant un langage simple qui décrit tests et actions par paquet. La facilité d'utilisation simplifie et accélère le développement de nouvelles règles de détection des exploits. Par exemple, lorsque le Showcode IIS des exploits Web ont été révélés sur le Bugtraq liste de diffusion, règles Snort pour détecter les sondes étaient disponibles en quelques heures.

#### **c. Installation et configuration de snort**

Premièrement Installation de package Snort .

```
kali㉿kali:~$ sudo apt-get install snort
[sudo] password for kali:
Reading package lists... Done
Building dependency tree
Reading state information... Done
snort is already the newest version (2.9.15.1-4).
```

Vérification de l'installation comme indiqué ci-dessous.

```
kali㉿kali:~$ sudo snort --version
      _--> Snort! <--_
  o" )~ Version 2.9.15.1 GRE (Build 15125)
  ... By Martin Roesch & The Snort Team: http://www.snort.org/contact#
team
  Copyright (C) 2014-2019 Cisco and/or its affiliates. All rights
reserved.
  Copyright (C) 1998-2013 Sourcefire, Inc., et al.
  Using libpcap version 1.9.1 (with TPACKET_V3)
  Using PCRE version: 8.39 2016-06-14
  Using ZLIB version: 1.2.11
```

La création des fichiers snort.conf et icmp.rules suivants:

```
kali㉿kali:~$ sudo nano /etc/snort/snort.conf
```

Nous avons vérifié le fichier de configuration snort.conf et que les règles icmp sont incluses ou non.

```
#####
# Step #3: Configure the base detection engine. For more information, see>
#####
#include /etc/snort/rules/icmp.rules
```

Nous avons ajouté une règle sur le fichier de règles icmp, par exemple la règle de base ci-dessus alerte quand il y a un paquet ICMP (ping).

```
kali㉿kali:~$ sudo nano /etc/snort/rules/icmp.rules
```

```
alert icmp any any → any any (msg:"ICMP Destination Unreachable Communicate");
alert icmp $EXTERNAL_NET any → $HOME_NET any (msg:"ICMP digital island base");
alert icmp $EXTERNAL_NET any → $HOME_NET any (msg:"ICMP Large ICMP Packet");
alert icmp any any → any any (msg:"ICMP Packet"; sid:477; rev:3;)
```

La vérification que la configuration de SNORT se fait avec succès.

```
kali㉿kali:~$ sudo snort -T -c /etc/snort/snort.conf =i eth0
```

```
Preprocessor Object: SF_BNS Version 1.1 <Build 4>
Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>
```

```
Snort successfully validated the configuration!
Snort exiting
kali㉿kali:~$
```

Maintenant nous ajoutons une deuxième machine Virtual KALI LINUX pour testons notre SNORT déjà configuré.

L'adresse IP de machine utilisée pour commencer le scanne : 192.168.1.2

```
kali㉿kali:~$ sudo ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.1.2 netmask 255.255.255.0 broadcast 192.168.1.255
        ether 08:00:27:5c:65:26 txqueuelen 1000 (Ethernet)
          RX packets 7125 bytes 437872 (427.6 KiB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 247 bytes 38343 (37.4 KiB)
            TX errors 0 dropped 0 overruns 0 frame 0
```

L'adresse IP de machine sur laquelle nous avons déjà configuré SNORT : 192.168.1.3

```
kali㉿kali:~$ sudo ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.1.3 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fd24:df6a:be89:8800:db:914c:9e8f:706c prefixlen 64 scopeid
          0x0<global>
        inet6 fd24:df6a:be89:8800:8051:8af:45ef:2c prefixlen 64 scopeid 0
          x0<global>
```

Nous avons fait un scanne de ports du machine avec nmap :

**Nmap** est un scanner de ports libre créé par Fyodor et distribué par Insecure.org. Il est conçu pour détecter les ports ouverts, identifier les services hébergés et obtenir des informations sur le système d'exploitation d'un ordinateur distant. Ce logiciel est devenu une référence pour les administrateurs réseaux car l'audit des résultats de Nmap fournit des indications sur la sécurité d'un réseau. Il est disponible sous Windows, Mac OS X, Linux, BSD et Solaris.

```
kali㉿kali:~$ sudo nmap 192.168.1.3
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-29 05:37 EST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.1.3
Host is up (0.0019s latency).
All 1000 scanned ports on 192.168.1.3 are filtered
MAC Address: 08:00:27:5C:65:26 (Oracle VirtualBox virtual NIC)
```

Nous avons déjà Exécuté le SNORT à partir de la ligne de commande, comme indiqué ci-dessous.

```
kali㉿kali:~$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i eth0
12/29-05:36:42.330980  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP} 0.0.0.0:68 → 255.255.255.67
12/29-05:37:46.764487  [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP} 0.0.0.0:68 → 255.255.255.67
12/29-05:38:33.833175  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.1.2:4566 → 192.168.1.3:705
```

Comme vous voyez la machine détecte toutes les intrusions arrivent de réseau extérieur.

# **Conclusion**

L'IoT est une combinaison de diverses technologies qui une gamme diversifiée d'appareils, et d'objets pour interagir et communiquer les uns avec les autres en utilisant différents réseaux les technologies. Jusqu'à présent, une grande partie des informations trouvées sur l'Internet est fourni par des êtres humains. En cas d'IoT smart les objets fournissent les informations. Il existe une grande variété d'applications basées sur l'IoT, y compris la santé, qui est l'objectif principal de ce travail. Les systèmes de santé utilisent appareils intelligents interconnectés pour établir un réseau IoT pour analyse des soins de santé, suivi des patients et identification automatique des situations où l'intervention d'un médecin est nécessaire.

Cet article a analysé les aspects de sécurité pour chaque couche de l'architecture IoT, et sur cette base, nous avons proposé une architecture IoT sécurisée qui permet de minimiser les risques de chaque couche d'architecture IoT. Et de garantir la confidentialité, l'intégrité et l'authentification par POLY1305 , le chiffrement des messages par AES-128 , la sécurisation de notre site web avec différentes méthodes enfin la détection d'intrusion par un système Lightweight.

# **Référence**

- 1) Ravi Kishore Kodali, Govinda Swamy and Boppana Lakshmi. An Implementation of IoT for Healthcare. 2015 IEEE Recent Advances in Intelligent Computational Systems (RAICS) | 10-12 December 2015 | Trivandrum.
- 2) Mohamed Tahar Hammi, Erwan Livolant, Patrick Bellot, Ahmed Serhrouchni, Pascale Minet. A Lightweight IoT Security Protocol. \*LTCI, Télécom ParisTech, Université Paris-Saclay, 75013, Paris, France.
- 3) Ivan Cvitić, Miroslav Vujić, Siniša Husnjak. CLASSIFICATION OF SECURITY RISKS IN THE IOT ENVIRONMENT . University of Zagreb, Faculty of Transport and Traffic Sciences, Vukelićeva 4, 10000 Zagreb, Croatia, EU