RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE
LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOULOUD MAMMERI DE TIZI-OUZOU
FACULTE DE GENIE ELECTRIQUE ET INFORMATIQUE
DEPARTEMENT D'INFORMATIQUE

Mémoire

En vue de l'obtention du diplôme de
Master en informatique

Option : Réseaux, Mobilité et Systèmes Embarqués

Thème

Portail Captif et pare-feu netfilter sur un Raspberry PI

Réalisé par :
Mr OUKKAL Sofiane
Mr KHETTAB Houssem Eddine

encadré par :
Mr DAOUI Mehammed

Examinateurs :

Mme OUKFIF Karima	Présidente
Mme KHIALI Lynda	Examinatrice

Année universitaire 2018/2019

Table des matières

I. CHAPITRE 1 6

Introduction 7

1. Définition 7
2. Historique : 8
3. Caractéristiques d'un système embarqué 9
4. Contraintes des systèmes embarqués : 10
5. Types de systèmes embarqués : 11
6. Architecture d'un système embarqué : 13
   - 6.1. Architecture typique d'un système embarqué : 13
   - 6.2. Architecture matérielle : 14
   - 6.3. Architecture logiciel : 19
7. Système d'exploitation pour l'embarqué 21
8. Programmation 22
9. Domaine d'application 23

Conclusion 24

II. CHAPITRE 2 25

Introduction ..... **Erreur ! Signet non défini.**

1. Définition 26
2. Fonctionnement général d'un pare-feu : 26
3. Catégories de pare-feu 28
   - 3.1. Les pares-feux sans état (stateless firewall) 28
   - 3.2. Pare-feu à état (state full firewall) 28
   - 3.3. Pare-feu applicatif 28
   - 3.4. Pare-feu identifiant 29
   - 3.5. Pare-feu personnel 29
4. Portail captif : 29
   - 4.1. Définition : 29

4.2. Principe de fonctionnement : 29
4.3. Exemple d'application de portail captif : 31
4.4. Exemple de portail captif : 31
5. Pare-feu Netfliter : 33
5.1. Définition : 33
5.2. Structure de netfilter : 33
5.3. Point d'accrochage (Hook) : 33
5.4. Les tables, chaînes et cibles : 34
Conclusion 37
III. CHAPITRE 3 38
Introduction 39
1. Présentation de notre application 39
1.1. Un portail captif 39
1.2. Un pare-feu 39
2. Conception du portail captif 39
3. Conception de l'interface web de la configuration du pare-feu : 42
4. Fonctionnement de l'interface graphique 43
5. Architecture logicielle globale : 44
6. Architecture matérielle 44
Conclusion 45
IV. CHAPITRE 4 RÉALISATION 46
Introduction 47
1. Description du matériel 47
1.1. Raspberry 47
1.2. Les modèles de Raspberry Pi 47
2. Coté software : 51
2.1. Systèmes d'exploitation pour la Raspberry Pi 51
2.1.2. Kali linux 52
2.2. Installation de Raspbian 54
2.3. Accès à distance au Raspberry Pi 63

2.4. Les outils de développement : 65
2.5. Création de portail Captif : 66
3. Présentation de l'application (notre système) 72
Conclusion 75

Table des figures

Figure I-1: fonctionnement d'un système embarqué. 7
Figure I-2 architecture de base d'un système embarqué 13
Figure I-3 vue simplifiée d'une architecture d'un système embarqué 14
Figure I-4 architecture de Von Neumann 15
Figure I-5 Architecture de Harvard 16
Figure I-6 vue d'une architecture typique d'un système embarqué 17
Figure I-7 architecture SoC 18
Figure I-8 architecture logicielle d'un système embarqué 19
Figure II-1 rôle d'un pare-feu 26
Figure II-2 exemple de règles de pare-feu 27
Figure II-3 exemple des agissements d'un portail captif 30
Figure II-4 structure de netfilter 33
Figure II-5 point d'accrochage 34
Figure III-1 Schéma logiciel de notre portail captif 42
Figure III-2 Schéma logicielle de l'interface graphique de configuration 43
Figure III-3 schéma logiciel global de notre système 44
Figure III-4 architecture matérielle de notre système 45
Figure IV-1 Raspberry pi 1 model A 47
Figure IV-2 Raspberry pi 1 model B 48
Figure IV-3 comparaison entre Raspberry pi 1 model A et B 48
Figure IV-4 Raspberry pi 1 model A+ 49

Figure IV-5 Raspberry pi 1 model B+ 49
Figure IV-6 Raspberry pi 2 50
Figure IV-7 Raspberry pi 3 model B+ 51
Figure IV-8 interface d'installation de Raspbian 55
Figure IV-9 interface de configuration de Raspberry 56
Figure IV-10 les opération disponible dans l'outil Raspi-conf 56
Figure IV-11 logiciel PuTTY 63
Figure IV-12 logiciel VNC viewer 64
Figure IV-13 Interface graphique sur logiciel VNC VIEWER 65
Figure IV-14 script de configuration de Nginx 67
Figure IV-15 installation de nodejs et npm 68
Figure IV-16 programme de serveur du portail captif 69
Figure IV-17 page d'authentification de l'administrateur 72
Figure IV-18 page de configuration de netfilter 73
Figure IV-19 page de portail captif 74

Introduction Général

Le progrès de la technologie a permis une avancée sans précédente dans les domaines des réseaux et système embarqués, ces progrès sont bien-sûr suivis d'un nombre indéfini de défis. Parmi ces défis, on trouve le risque lié à la sécurité qui a induit à intégrer de nombreux mécanismes de protection.

D'autre part, la notion d'Appliance vient compléter les deux mondes, celui de l'application et l'équipement. En effet, les systèmes embarqués deviennent de plus en plus puissants et sont ainsi capable aujourd'hui d'embarquer des solutions répondant aux défis de la sécurité cité plus haut, des solution tels que les pare-feu (CISCO , Juniper), système de détection d'intrusion et les portails captifs.

Dans ce contexte, nous avons pour objectif d'embarquer un pare-feu doublé d'un portail captif sur une Raspberry pi. Nous avons choisi cette solution car elle comporte beaucoup d'avantages, à commencer par la basse consommation d'énergie, le moindre cout, rend la tâche facile pour l'administrateur. Il sera d'une grande utilité dans la phase d'accès à un réseau public en assurant de nombreuse fonctionnalité comme par exemple, la sécurité et le contrôle d'accès au réseau, le payement d'un abonnement Internet ou encore l'affichage de spots de publicité aux clients se connectant au réseau.

Afin de mener à bien notre projet nous avons devisé notre travail en quatre chapitres, dans le premier chapitre nous avons présenté des généralités sur les systèmes embarqués telles que l'architecture matérielle, les systèmes. Le deuxième chapitre est consacré aux différents types de pare-feu et leurs fonctionnements. Le troisième chapitre est consacré à la conception de notre système. La réalisation de notre système sera présentée dans le quatrième chapitre, dans lequel nous présenterons l'environnement de développement et les divers composants implémentés dans l'architecture de notre système. Nous finirons ce rapport par une conclusion générale.

I. CHAPITRE 1

Généralités sur les Systèmes Embarqués

Introduction

Les technologies des systèmes embarqués, logiciel embarqué et microélectronique, ont la capacité de transformer tous les objets du monde physique du plus petit au plus grand, du plus simple au plus complexe en objets numériques, intelligents, autonomes et communicants. L'émergence des systèmes embarqués, amplifie de façon considérable cette révolution. De ce fait, le déploiement généralisé des systèmes embarqués modifie profondément notre environnement, et porte de très nombreuses innovations de produits et d'usages et impacte l'ensemble des activités industrielles et de services.

1. Définition

Un système embarqué est une combinaison de matériel et de logiciel, conçu pour une fonction ou des fonctions spécifiques. Contrairement à un ordinateur traditionnel, un système embarqué n'a pas de raison d'être seul, il s'agit toujours d'un composant faisant partie d'un système plus large, on ne le voit pas forcément même si ce dernier est de plus en plus présent dans notre vie quotidienne. [1]

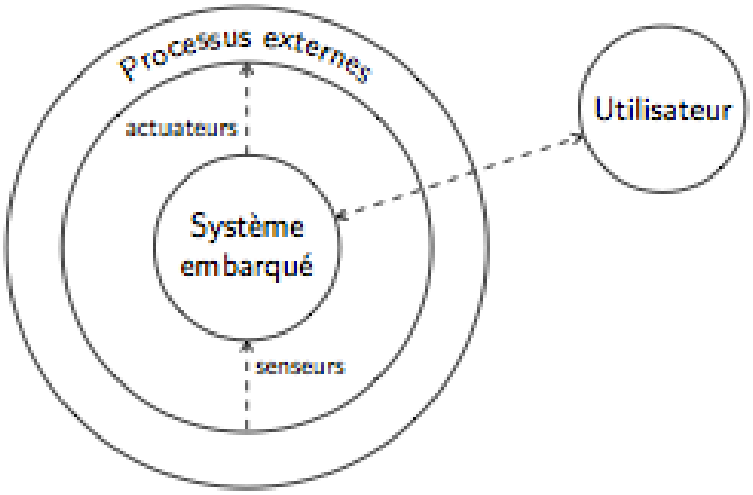


Figure I-1: fonctionnement d'un système embarqué.

2. Historique :

[1967] : <<Apollo Guidance computer>> l'un des premiers systèmes embarqués environ un millier de circuits intégrés identiques (portes NAND).

[1970] : Missile Minuteman, Il était le système de guidage informatisé Autonetics D-17, qui utilise des circuits logiques à base de transistor et disque dur comme la mémoire principale.

[1971] : le premier microcontrôleur, était le premier circuit intégré incorporant tous les éléments d'un ordinateur dans un seul boîtier : unité de calcul, mémoire, contrôle des entrées /sorties.

[1972] : l'Intel 8008 fut le premier microprocesseur 8 bits à voir le jour. Deux fois plus puissant que le 4004, le 8008 était composé de 3300 transistors et fabriqué lui aussi en 10 000 nm, et sa vitesse atteignait 0,5 MHz (500 kHz), voire 0,8 MHz un peu plus tard.

[1978] : Le Zilog Z80 est un microprocesseur 8 bits conçu et fabriqué par Zilog. Une des particularités de ce processeur est le couplage de certains registres (2 fois 8 bits) et le bus d'adresses 16 bits, permettant un traitement de l'information nettement plus rapide qu'avec un processeur 8 bits classique.

[1979] : le Motorola 68000 est un microprocesseur CISC 16/32 bits développé par Motorola. C'est le premier de la famille de microprocesseurs souvent appelée m68k ou 680x0, qui comprend notamment les microprocesseurs Motorola 68010, Motorola 68020, Motorola 68030, Motorola 68040 et Motorola 68060. Le nom du 68000 vient à la fois de la continuité avec la famille de microprocesseurs Motorola 6800 et du nombre de transistors qu'il contient, un peu plus de 68 000. [2]

De 80 à aujourd'hui : avènement d'une industrie nouvelle, utilisant la puissance, la miniaturisation et la robustesse des puces électroniques pour rendre intelligents, communicants et sûrs tous les objets de notre quotidien. [3]

3. Caractéristiques d'un système embarqué

La majorité des systèmes embarqués ont des caractéristiques communes, peu importe leur type. Tout d'abord, un système embarqué doit être fiable, généralement il doit pouvoir fonctionner en autonomie, en particulier, on peut mesurer :

- Sa **robustesse** : la probabilité qu'il fonctionne correctement à un certain temps t, sachant qu'il fonctionnait depuis t=0.
- Sa **maintenabilité** : la probabilité qu'il fonctionne correctement un certain temps après qu'une erreur se soit produite.
  Sa **disponibilité :** la probabilité qu'il fonctionne correctement à tout moment
- Sa **sûreté :** il ne cause pas de nuisances.
- Sa **sécurité** : ses communications sont confidentielles et authentifiées.

Lors de la conception d'un système embarqué, il faut penser à ces critères et les maximiser, car après, il n'est pas évident de les modifier pour une meilleure satisfaction, en effet, étant donné les contraintes du système embarqué, surtout en termes de puissance de calcul, il ne sera pas forcément possible de tous les maximiser, donc un choix devra être posé dépend de la tache visée.

Ensuite, puisque le SE est conçu pour un but spécifique et précis, il faut qu'il soit efficace plus qu'un ordinateur standard, prenant en compte les aspects suivants :

- La **consommation énergétique** ;
- La **taille du code**, en particulier pour des systèmes embarqués de bas niveau déployés sur une petite puce.
- L'**exécution** du programme tant au niveau du temps que de la mémoire consommée.
- Le **poids et la taille**.
- Le **cout** de fabrication, mais aussi de conception et design.

Nous illustrons dans la figure suivante une ou plusieurs mesures qui permettent d'évaluer ces aspects :

CRITERES	MESURE
CONSOMMATION ENERGETIQUE	Watt
TAILLE DU CODE	Octets Lignes de code
EXECUTION DU PROGRAMME	MIPS Nombre de read/write par seconde
POIDS ET TAILLE	Kilogramme, centimètre
COUT DE FABRICATION	$, €, DA …

Tableau 1 : Critère d'efficacité d'un système embarqué.

Enfin, toutes ces caractéristiques d'efficacité et de fiabilité, doivent évidemment être considérées en lien avec l'application pour laquelle le système embarqué est développé. En effet, certains critères ont sans doute moins de sens ou sont moins importants pour une application donnée. [4]

4. Contraintes des systèmes embarqués

Les systèmes embarqués sont soumis à plusieurs contraintes par rapport à l'environnement et aux critères d'exécution, Parmi ces contraintes :

- **L'espace mémoire :** à cause de leurs capacité mémoire très limitée, les systèmes embarqués doivent être conçus de manière à optimiser le plus possible l'espace dédié pour répondre aux demandes du système sans surcoût.
- **La puissance de calcul :** il est préférable qu'un système embarqué aie une puissance de calcul raisonnable pour répondre aux demandes des tâches à exécuter. L'optimisation de cela revient à éviter les surcoûts financiers et énergétiques.
- **L'énergie :** une consommation énergétique la plus basse possible est nécessaire pour une plus grande autonomie à cause des systèmes embarqués utilisant les batteries/panneaux solaires.
- **Le temps de réponse :** le temps d'exécution des tâches prédéfinie sont souvent connus voir limités en vue d'avoir un temps de réponse comparable au temps réel.

- **La sûreté :** certains systèmes embarqués sont au cœur de tâches critiques voire vitales. En conséquence, la sûreté du fonctionnement de celui-ci doit être assurée. [5]

5. Types de systèmes embarqués

Généralement les systèmes embarqués sont classés de trois façons différentes, par rapport aux tâches pour lesquelles ils sont conçus et utilisés, aux types de fonctions qu'ils calculent et par rapport à leurs tailles et complexité.

Pour le classement par type de fonctions à remplir, les systèmes embarqués sont catégorisés en quatre usages :

- Les systèmes embarqués qui exécutent des tâches prédéfinies comparable aux ordinateurs communs, sauf qu'ils sont plus compacts puisqu'ils sont embarqués dans des packages de petites tailles. Appelé **systèmes embarqués à usage général**, en prend comme exemple les ATM : guichets automatiques bancaires.
- Les systèmes embarqués qui effectuent une vérification voir, un contrôle en boucle rétroactive fermée sur les systèmes en temps réel. Appelé **système embarqué de contrôle**. Ils sont souvent retrouvés dans les moteurs de voitures, le contrôle aérien ou les centrales nucléaires.
- Les systèmes embarqués qui traitent un très grand nombre de données. **Les systèmes embarqués pour traiter les signaux**, utilisés généralement dans le traitement audio et vidéo et dans les sonars et les radars.
- Les systèmes embarqués utilisés dans le domaine réseaux et de la communication, pour la transmission de données et la réalisation de communications. Retrouvé dans la téléphonie et internet.

Ce classement, comme cité ci-dessus, regroupe les systèmes embarqués en fonction de la tâche à remplir, ceci dit certains prototypes peuvent être classés dans plusieurs catégories en même temps. Par exemple, une console de jeu vidéo peut être classée dans les usages généraux puisqu'elle fonctionne comme un ordinateur classique et peut être classée dans les systèmes embarqués pour traitement de signaux. Certains appareils peuvent aussi contenir plusieurs systèmes embarqués tel que l'exemple de la voiture cité précédemment.

En ce qui concerne le classement par rapport aux types de fonctions que les systèmes embarqués calculent, on dénombre cinq catégories :

- Il y a la possibilité qu'un système embarqué ai pour but de s'assurer qu'une **loi de contrôle** soit respectée. En prend l'exemple d'un contrôleur PID (Proportional-

Integral-Derivative) dans un drone, celui-ci calcule une valeur d'erreur et une valeur mesurée, à partir de ces valeurs il produit et exécute des commandes pour se rapprocher de l'objectif. Un contrôleur PID permet d'assurer une stabilité en vol stationnaire.

- Un système embarqué peut aussi avoir une **logique séquentielle** à la manière d'une machine à laver alternant entre différents modes de fonctionnement.
- On trouve également les systèmes embarqués qui ont pour fonction le **traitement de signaux**, voir le traitement de contenu multimédia. La compression, la décompression ou l'applications de filtre sonore.
- Les systèmes embarqués peuvent également être utilisés comme **interface** pour des applications précises. Cela a pour but la gestion du thermomètre, lampes, alarmes, music, etc. Par exemple, dans les maisons connectées ou les tableaux de bord des voitures.
- Finalement, il existe des systèmes embarqués qui **réagissent aux fautes**. Leur rôle est de diagnostiquer et réparer les erreurs si le système à un comportement anormale.

Ces classements précédents montrent qu'il existe une très grande variété de systèmes embarqués. Cependant, ils jouissent évidemment tous d'un grand nombre de caractéristiques communes, les faisant appartenir au monde des systèmes embarqués.

Si on est plutôt concerné par le côté technique, on a la possibilité de classer les systèmes embarqués selon leurs complexités :

- Les systèmes embarqués de **petite échelle** sont caractérisés par un simple microcontrôleur sur 8 ou 16 bits, avec une partie physique et logique très simple. Le design de ces systèmes se retrouve au niveau d'une carte et les ressources, notamment en termes de mémoire, sont très limitées pour limiter la dissipation de puissance.
- Les systèmes embarqués de **moyenne échelle** sont caractérisés par un simple microcontrôleur sur 16 ou 32 bits, un *Digital Signal Processor* (processeur de signal numérique) ou un processeur avec un jeu d'instructions réduit (RISC). La partie physique et logique sont plus complexes. Côté hardware, on retrouve des *Application Specific Standard Product* (produit standard spécifique à une application, par exemple pour smartphones) (ASSP) ou *Instruction Processor* (IP) et ils peuvent être contrôlés par des *Real-Time Operating Systems* (systèmes d'exploitation temps réel) (RTOS).

On peut aussi construire un prototype plus sophistiqué qui a une complexité physique et logique très grande. Ils sont construits à partir de processeurs évolutifs et configurables ou de *Programmable Logic Array* (PLA). Pour ces systèmes, un Co-design et une intégration hardware/software sont importants. [4]

6. Architecture d'un système embarqué

6.1. Architecture typique d'un système embarqué

Un système embarqué typique constitué de deux parties principales : le matériel embarqué et le logiciel embarqué. La figure suivante montre une architecture de base d'un système embarqué

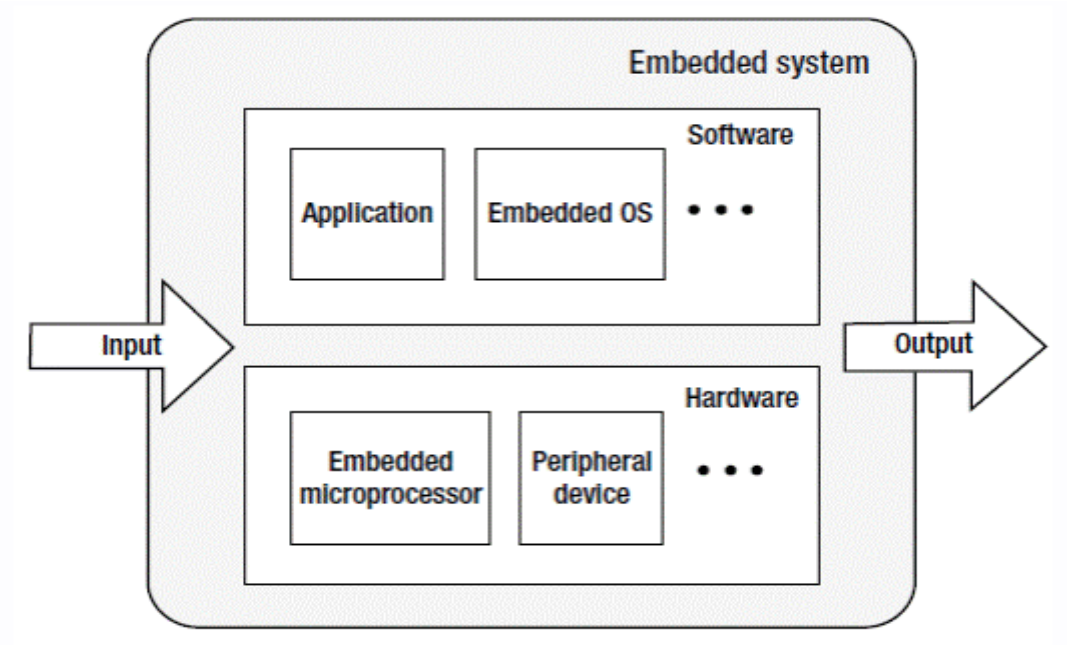


Figure I-2 architecture de base d'un système embarqué

Le matériel et le logiciel collaborent toujours pour traiter divers signaux d'entrée de l'extérieur et produire le résultat de traitement sous une forme ou une autre, le signal d'entré peut être un dispositif ergonomique comme clavier, souris, ou une sortie dans un circuit dans un autre système intégré. La sortie peut être sous forme de son, lumière, électricité, ou un autre signal pour un autre circuit, ou un fichier pour une base de données.[6]

6.2.Architecture matérielle :

6.2.1. Architecture matériel typique :

Les composants de base du système informatique tels que processeur, une mémoire, et E/S qui sont interconnecté avec un bus, afin que toutes les parties puissent communiquer et exécuter un programme, comme illustré dans la figure suivante :

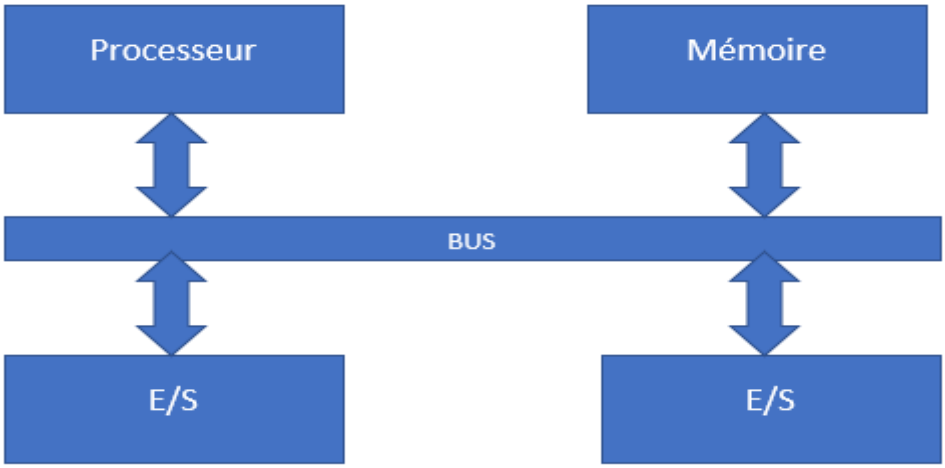


Figure I-3vue simplifiée d'une architecture d'un système embarqué

Le rôle et la fonction de micro-processeur dans un système embarqué sont généralement les mêmes que ceux de la CPU d'un ordinateur ordinaire : contrôle du fonctionnement de l'ordinateur, exécution des instructions et traitement des données. Dans de nombreux cas, le microprocesseur dans un système intégré est également appelé CPU.

La mémoire est utilisée pour stocker des instructions et des données. Les modules d'E/S sont responsables de l'échange de données entre le processeur, la mémoire et les périphériques externes.
Les périphériques externes comprennent les périphériques de stockage secondaires (tels que les disques flash et les disques durs), les équipements de communication et les équipements terminaux. Le bus système fournit des données et contrôle la communication et la transmission du signal pour le processeur, la mémoire et les modules d'E / S.

Il existe essentiellement deux types d'architecture qui s'appliquent aux systèmes embarqués : l'architecture Von Neumann et l'architecture Harvard.[6]

6.2.2. Architecture de von Neumann :

Cette architecture a été proposée pour la première fois par John von Neumann. Son principe c'est d'utiliser une structure de stockage unique, pour conserver à la fois les instructions et les données demandées ou résultat de calcul, et ces derniers partagent le même bus de transmission, en effet la transmission d'information devient le goulot d'étranglement des performances et cela affecte la vitesse de traitement de données. Nous illustrons cette architecture dans la figure suivante :

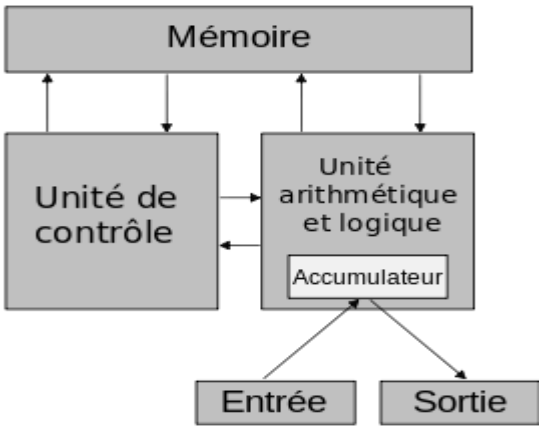


Figure I-4 architecture de Von Neumann

6.2.3. Architecture de Harvard :

L'architecture de Harvard est une conception des processeurs qui sépare physiquement la mémoire de données et la mémoire programme. L'accès à chacune des deux mémoires s'effectue via deux bus distincts, comme illustré dans la figure suivante :

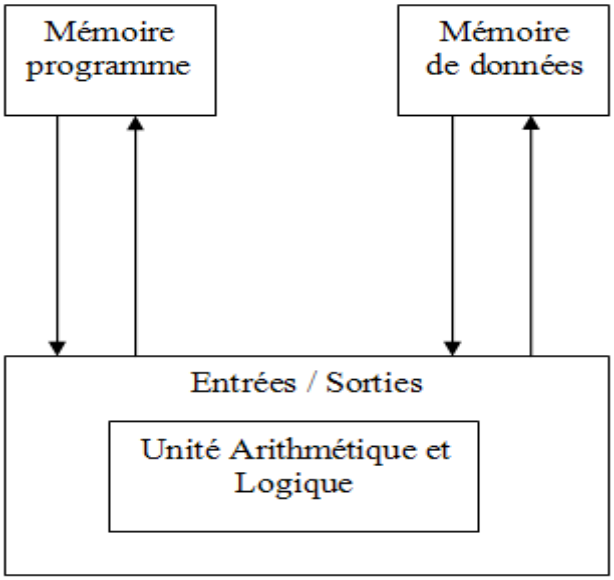


Figure I-5 Architecture de Harvard

Cette architecture peut fournir une bande de mémoire de donnée plus large. Elle est utilisée généralement dans les DSP.

6.2.4. Structure typique d'un système embarqué :

La structure d'un système embarqué varie selon les besoins de l'application, par exemple, il est possible d'éliminer quelques composants dans certains cas et de les utiliser dans d'autre. Dans la figure suivante nous illustrons une structure typique d'un système embarqué :

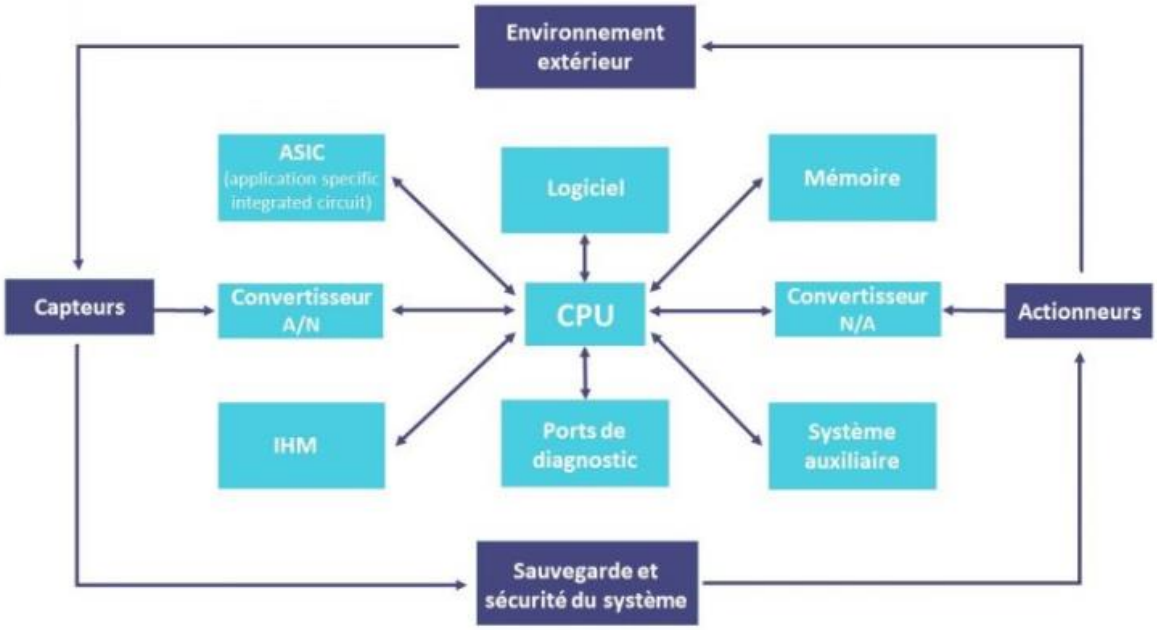


Figure I-6 vue d'une architecture typique d'un système embarqué

Avec le développement de la technologie de conception et de fabrication de circuits intégrés, la conception de circuits intégrés est passée de l'intégration à transistors à l'intégration à portes logiques, en passant par l'intégration IP (intellectuel property) ou le système sur puce (SoC).[6]

6.2.5. Architecture SoC (System on Chip):

Soc (système on chip) ou système sur puce, est un circuit intégré qui rassemble les principaux composants d'un ordinateur ordinaire sur une seule puce électronique, Les composants sont bien plus interconnectés, Actuellement ils sont les plus répandu, on les retrouve dans la plupart des objets utilisés tels que les smartphones, les tablettes, ainsi que les systèmes embarqués tels que Raspberry, banana Pi. Nous illustrons dans la figure suivante l'architecture d'un Système sur puce :

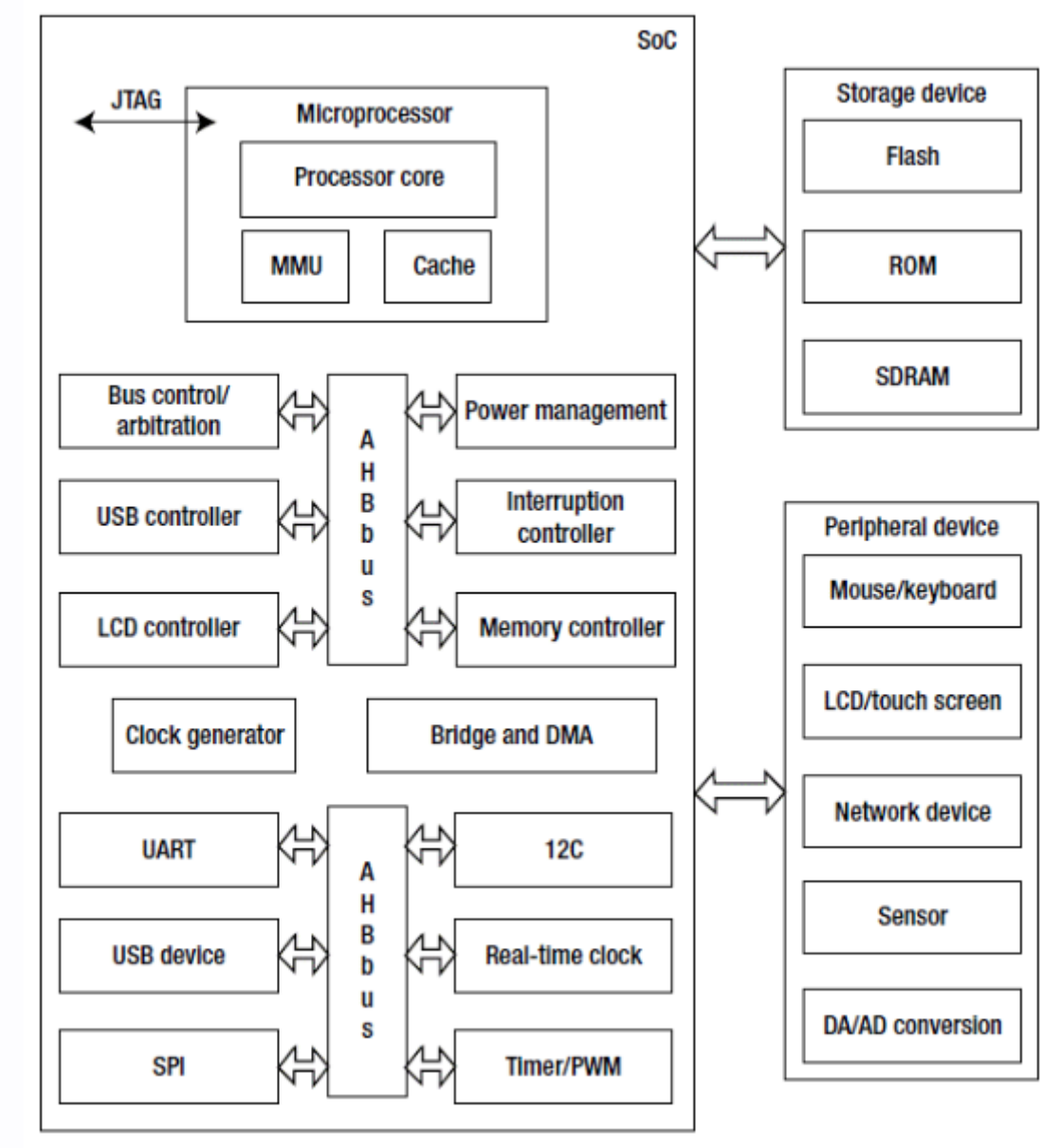


Figure I-7 architecture Soc

Un système on chip est plus spécialisé qu'un microcontrôleur classique, il peut être conçu de façon personnalisée en utilisant les technologies d'intégration actuelles, tels que ASIC, FPGA etc … [6]

6.3.Architecture logiciel :

L'architecture logicielle embarquée est extrêmement flexible. Les logiciels intégrés simples (tels que les jouets électroniques, les calculatrices, etc.) peuvent ne contenir que quelques milliers de lignes de code et effectuer des fonctions d'entrée et de sortie simples. D'autre part, les systèmes embarqués complexes (tels que les smartphones, les robots, etc.) nécessitent une architecture logicielle plus complexe, similaire aux ordinateurs de bureau et aux serveurs. Le logiciel intégré simple convient au matériel de puce à faible performance, aux fonctionnalités très limitées, et nécessite un développement secondaire fastidieux. Les systèmes embarqués complexes offrent des fonctions plus puissantes, nécessitent des interfaces plus pratiques pour les utilisateurs et nécessitent la prise en charge de matériel plus puissant. Avec l'amélioration des capacités d'intégration et de traitement du matériel, le goulot d'étranglement du matériel s'est progressivement relâché et même rompu, de sorte que les logiciels de systèmes intégrés ont désormais tendance à être entièrement fonctionnels et diversifiés.

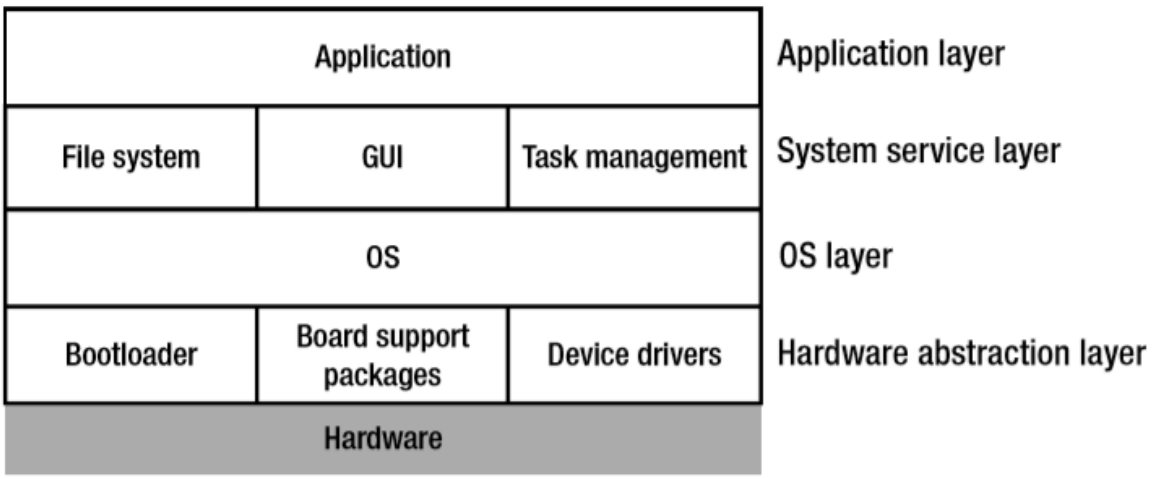


Figure I-8 architecture logicielle d'un système embarqué

Un système logiciel embarqué est composé de quatre couches, de bas en haut :

6.3.1. Couche d'abstraction matérielle (HAL)

La couche d'abstraction matérielle en anglais hardware abstraction layer (HAL), C'est un logiciel intermédiaire entre le matériel du système intégré et le système d'exploitation. Il offre des fonctions standardisées de manipulation du matériel tout en cachant les détails techniques de la mise en œuvre. En général, la couche HAL inclut le chargeur de démarrage, le package de support de carte (BSP), les pilotes de périphérique et d'autres composants.

6.3.2. Système d’exploitation

Un système d'exploitation est un système logiciel permettant de gérer les ressources matérielles de manière uniforme. Il résume de nombreuses fonctions matérielles et les fournit aux applications sous forme de services.

6.3.3. Couche de service système

La couche de service système est l'interface de service fournie par le système d'exploitation, pour que les applications puissent accéder à ces services à l’aide de cette interface. En effet, il joue le rôle de lien entre l’application et le système d’exploitation.

6.3.4. Couche d'application

L’application, située au plus haut niveau de la hiérarchie logicielle, elle implémente les fonctionnalités du système et la logique d’entreprise. D'un point de vue fonctionnel, tous les niveaux de modules de l'application visent à exécuter des fonctions du système. Du point de vue du système, chaque application est un processus de système d’exploitation distinct. [6]

7. Système d'exploitation pour l'embarqué

Un système d'exploitation est chargé d'assurer la liaison entre les ressources matérielles, l'utilisateur et les applications (traitement de texte, jeu vidéo, ...). Ainsi lorsqu'un programme désire accéder à une ressource matérielle, il ne lui est pas nécessaire d'envoyer des informations spécifiques au périphérique, il lui suffit d'envoyer les informations au système d'exploitation, qui se charge de les transmettre au périphérique concerné via son pilote. En l'absence de pilotes il faudrait que chaque programme reconnaisse et prenne en compte la communication avec chaque type de périphérique. [7]

Un système d'exploitation embarqué est un système d'exploitation pouvant être installé sur un système embarqué. Ce système d'exploitation est conçu avec des spécificités à gérer afin de répondre à des besoins spécifiques au type de système embarqué. [8]

Quelques types de systèmes d'exploitation embarqués :

- Windows CE: est une variation de Windows pour les systèmes embarqués et autres systèmes minimalistes, utilisée notamment dans les PC de poche . Il utilise un noyau distinct des autres Windows plutôt qu'une version allégée et supporte les architectures processeur Intel x86 et similaires, MIPS (jusqu'à CE 3.0), ARM .
- Firefox OS :est un système d'exploitation mobile libre et destiné aux objets connectés proposé et développé par la Mozilla Corporation.
- Windows phone : c'est un système d'exploitation pour smart phones que Microsoft a publié en octobre 2010. Il est venu remplacer Windows Mobile en introduisant une interface utilisateur totalement redessinée et pensée pour les terminaux à écran tactile. Celle-ci se compose de tuiles dynamiques dont l'affichage évolue en temps réel selon l'activité de l'application concernée.
- Palm OS : développé initialement par U.S Robotics Corporation Il est fourni avec une suite d'applications de base pour gestionnaire d'informations personnelles. Plus tard
- Android : est le système d'exploitation mobile de Google open-source qui équipe la majorité des Smartphones et tablettes du marché , Android était une startup dirigée par Andy Rubin et qui a été racheté par Google en 2005. Aujourd'hui, Android est le système d'exploitation mobile numéro un dans le monde.
- eCos : est un système d'exploitation temps réel. Il permet de réaliser des applications temps réel. Il est implémenté en C/C++ . [9]

8. Programmation

Un système embarqué doté d'un processeur a besoin de code pour faire quelque chose. Ce dernier est placé dans une mémoire accessible par le processeur afin de pouvoir être exécuté, instruction par instruction. Typiquement, le software d'un système embarqué sera présenté sous la forme d'une image ROM, définitive et non modifiable.

Étant donné un système embarqué, on connait la taille de la ROM qu'il possède et donc les adresses de tous les emplacements mémoire disponibles. Pour chacun de ces emplacements, il faut en définir le contenu binaire représentant des instructions à exécuter. Ces instructions sont écrites en respectant le jeu d'instructions du processeur utilisé, ce qui peut évidemment prendre beaucoup de temps à comprendre avant de commencer à programmer.

Pour éviter de mémoriser les codes machine de toutes les instructions d'un processeur donné, on peut programmer à l'aide d'un langage d'assemblage, représentation textuelle de plus haut niveau du code machine. Un programme écrit dans un langage d'assemblage est traduit en code machine par l'assembleur. Programmer à un tel niveau reste une tâche fortement consommatrice de temps, mais peut être plus efficace si l'on s'aide de librairies de code préexistant.

Enfin, pour gagner en temps de développement, il est possible d'utiliser un langage de haut niveau qui offre des constructions et abstractions plus pratiques et proches de la façon de penser du développeur et du problème qu'il veut résoudre. Quelques exemples d'un tel langage, adapté aux systèmes embarqués :

- le langage C. Langage de haut niveau le plus populaire dans les systèmes embarqués, compilé en code machine par le compilateur C, permet de développer des programmes concis et rapides.
- Le langage c++ : C'est une évolution orienté objet du langage C, Il constitue la base de plusieurs langages de programmation, le principal choix de programmation pour la plupart des applications de bureau les plus utilisées. un choix incontournable pour les pilotes de périphériques, les moteurs de jeux, les outils de traitement audio / image, les logiciels intégrés, etc.
- Java : Java est également utilisé dans de nombreux systèmes embarqués mais les programmes Java nécessitent une machine virtuelle Java (JVM) qui consomme beaucoup de ressources. C'est pourquoi ce langage n'est pas utilisé pour les appareils embarqués qui ne disposent que de ressources limitées.
- Langage Ada : langage de plus haut niveau que le C, mais conçu pour ce type de développement (permet de s'abstraire de l'OS)
- Pyton : Bien que n'étant pas traditionnellement associé aux systèmes embarqués, Python commence à être pris plus au sérieux dans les applications de systèmes embarqués, Python n'est peut-être pas aussi utile pour la programmation embarquée que C ou C ++, mais avec de nombreuses bibliothèques disponibles, il est facile d'implémenter des

fonctionnalités qui le rendent tout aussi utile. Il est excellent pour automatiser les tests, collecter et analyser les données

On pourra également employer d'autres langages de programmation ou langages de scripts célèbres dans le monde Unix, perl … [10]

9. Domaine d'application

Les systèmes embarqués sont introduits dans divers domaines, à savoir :

Le domaine grand public : Smart phone, console de jeux, appareil photos, lecteur audio

- Les moyens de transport Ordinateur de bord, GPS, système de navigation,

- Les équipements médicaux : Imagerie (rayon X, ultra-sons, IRM) endoscopie, caméra, monitoring, perfusion, lasers, chirurgie, stimulateur cardiaque.

- Les équipements de télécommunication : Station mobile, routeur, Gateway, satellite

- Les équipements industriels : Productions automatisées, systèmes de commande d'énergie, équipements de stockage

- Les équipements de bureautiques : Répondeurs, copieurs, imprimante

Conclusion

On a vu dans ce chapitre un aperçu du monde des systèmes embarqués en général, l'architecture matérielle et logicielle, ses catégories, ses domaines d'utilisation, les contraintes qu'il faut prendre en considération avant de concevoir un système embarqué.

Dans le chapitre suivant nous allons présenter le pare-feu netfilter qu'on utilisera dans notre projet et voir la notion de portail captif.

II. CHAPITRE 2

Pare-feu Netfilter et portail captif

1. Définition

Un pare-feu ou firewall en anglais est un équipement ou une application logicielle permettant de protéger une machine ou un réseau quelconque.

Chaque réseau dispose d'une politique d'accès à ses ressources pour se protéger de l'extérieur (internet par exemple), il est ainsi nécessaire de s'équiper d'un pare-feu pour définir, et faire respecter cette politique à travers des règles.

La figure suivante est une représentation simplifiée illustrant le rôle d'un pare-feu.

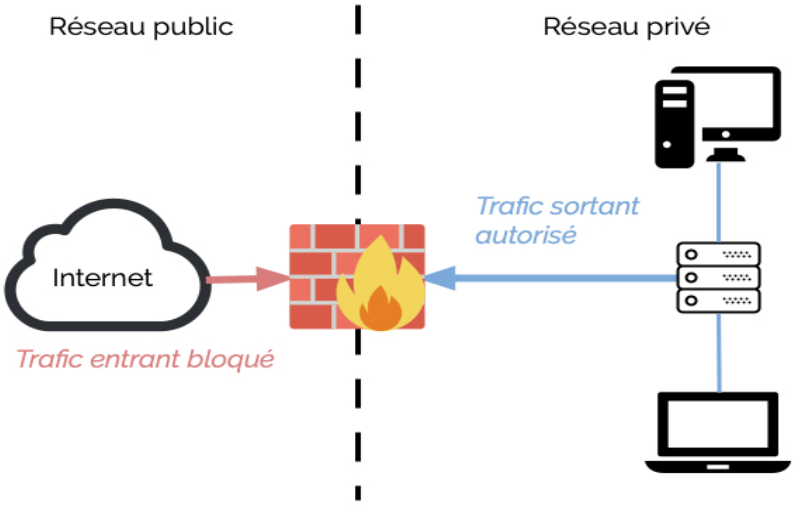


Figure II-1 rôle d'un pare-feu

2. Fonctionnement général d'un pare-feu :

Un pare-feu est constitué d'un ensemble de règles qui sont soumise à chaque paquet circulant dans le réseau qu'on souhaite protéger d'un autre réseau de confiance moindre.
En effet, un pare-feu divise ses communications en un ensemble de réseaux de confiance et définit ainsi sa politique et les droits d'accès à ses ressources selon chaque niveau de confiance, internet représente ainsi le premier échelon de confiance (confiance zéro).
Le filtrage de chaque paquet se fait à travers des informations contenues dans ce paquet, par exemple, le filtrage peut se faire selon :

- L'ordinateur d'origine (adresse IP).
- Destination prévue.
- Mode de traitement de donnée.
- Port de réception des paquets (port UDP, TCP…).

Nous illustrons un exemple de règles élémentaires de filtrage de pare-feu par la figure ci-après :

	src	port	dst	port	protocol	action
1	any	any	128.12.1.2	25	TCP	permit
2	128.12.1.2	25	any	any	TCP	permit
3	128.12.1.2	any	any	25	TCP	permit
4	any	25	128.12.1.2	any	TCP	permit
5	any	any	any	any	TCP	deny

Figure II-2exemple de règles de pare-feu

L'interprétation des règles suivante se fait comme suit :

- **Règle 1 et 2** : tous les paquets de n'importe quel IP source, port source de protocole de destination TCP sont autorisés à transiter vers la destination IP 128.12.1.2 et vice versa.
- **Règle 3 et 4** : tous les paquets de source IP 128.12.1.2, port source quelconque protocole de destination sont autorisés à transiter vers n'importe quel destination et vice versa.
- **Règle 5** : cette règle est particulaire et nécessaire à tous les pares-feux, c'est la règle par défaut cela veut dire qu'il faut interdire tout ce qu'on n'autorise pas.

⇒ Le monde actuel considère les pares-feux comme le moyen le plus vieux et de premier recours pour structurer la sécurité informatique de son ordinateur personnel, entreprise ... Etc. C'est pourquoi, à travers les générations plusieurs pare-feu se sont concurrencés selon l'évolution des besoins et de la technologie. On parlera alors des catégories de pare-feu.

3. Catégories de pare-feu

3.1.Les pares-feux sans état (stateless firewall)

Première génération de pare-feu, celui-ci tient compte uniquement des états statiques de connexion TCP, se basant sur les informations contenues dans l'entête du paquet (IP source, IP destination, port source, port destination).

Inconvénient :

- Étant donné qu'il ne garde pas l'état des connexion, l'administrateur doit spécifier beaucoup de règles correspondant aux différents types de connexion.
- Règles volumineuses ce qui oblige le firewall à les traiter à chaque fois (ralentissement).
- Le firewall peut laisser passer des paquet ACK sans vérification d'une existence d'un SYK préalable.

3.2.Pare-feu à état (state full firewall)

Ce type de pare-feu prend en compte l'état de connexion TCP de la pile protocolaire TCP/IP, ce type de pare-feu contient une table des connexions déjà établies fonctionnant comme une mémoire cache, si un paquet arrivant fait déjà parti d'une connexion ayant été établie (ayant déjà une entrée dans la table), alors celui-ci est autorisé à transiter.

si, par exemple, une machine tente de se connecter sans qu'il ait déjà une connexion TCP établie par exemple en envoyant un paquet ACK sans qu'aucun SYN d'établissement de la connexion n'a était envoyé, alors le paquet ACK sera rejeté automatiquement par le firewall sans même passer par les règles de sécurité de sa politique déjà établie par l'administrateur.

Inconvénient

•La table des connexions peut être saturée si un flux important de nouvelles connexions sont établies, ce qui ralenti considérablement le firewall ou ne plus marcher du tout (déni de service).

3.3.Pare-feu applicatif

Le pare-feu applicatif est l'incontournable pare feux de dernière génération, souvent utilisé pour prévenir les attaques visant le web.Toute les requêtes transitant du client vers le server passent par le pare-feu applicatif, il vise à filtrer le passage d'un protocole par port, par exemple il permet de vérifier que seul le Protocol HTTP passe par le port 80.

• Inconvénient : ce type de pare-feu nécessite un temps de calcul élevé dès que le débit d'information est important

3.4.Pare-feu identifiant

Ce type de pare-feu n'effectue plus de suivi de connexion à travers les adresses IP mais à travers l'utilisateur lui-même appelé l'identifiant, plusieurs techniques existent pour associer chaque utilisateur à son adresse IP afin de l'activité réseaux de l'utilisateur.

Exemple : identification par authpf, pare-feu entièrement basé sur l'identité prenant en compte l'identité de l'utilisateur, son post, niveau de sécurité informatique.

3.5.Pare-feu personnel

Les pares-feux personnels sont des logiciels directement installés sur ordinateur personnel agissant comme un pare-feu à état. [11]

4. Portail captif :

4.1.Définition :

Le portail captif est une méthode permettant de forcer les requêtes HTTP d'un client lors de sa tentative de connexion, et de lui afficher une page web particulière, (le plus souvent un formulaire d'authentification) avant d'accéder à la page voulue totalement. [12]

4.2.Principe de fonctionnement :

Etape1 : Le portail captif intercepte les paquets de Protocol HTTP/HTTPS quel que soient leurs destinations.

Etape2 : l'utilisateur est redirigé vers une page web afin de :

- S’authentifier.
- Effectuer un paiement en ligne.
- Récupérer l'approbation de l'utilisateurs concernant des conditions préalables ou collecter leurs données personnelles.
- Saisir des informations.
- Afficher des informations publicitaires

Etape3 : après avoir effectué les taches voulu du client http, si les conditions d'accès sont remplies, l'utilisateurs est enfin redirigé vers sa destination initiale.

La figure suivante représente un exemple des agissements d’un portail captif

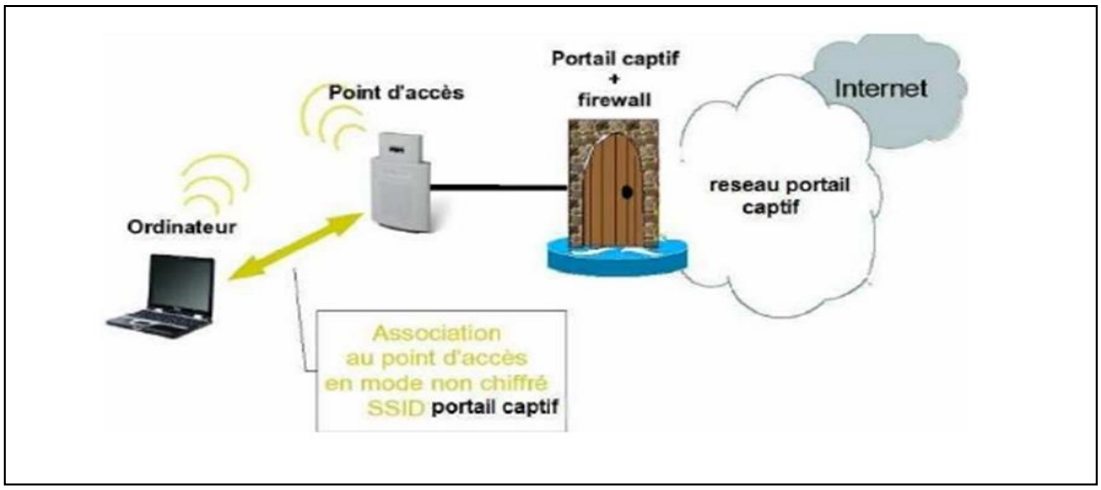


Figure II-3exemple des agissements d'un portail captif

Scénario : Le client se connecte au réseau par l'intermédiaire d'une connexion filaire ou au point d'accès wifi. Un serveur DHCP lui fournit une adresse IP ainsi que les paramètres de la configuration du réseau. A ce moment-là, le client a juste accès au réseau entre lui et la passerelle, cette dernière lui interdisant, pour l'instant, l'accès au reste du réseau. Lorsque le client va effectuer sa première requête de type web en HTTP ou HTTPS, la passerelle le redirige vers le portail captif.

Avantage

1. Écarter toute responsabilité d'une entreprise aux connexions internet dangereuses ou illégales et offre également une sécurité supplémentaire pour l'utilisateur.
2. Limiter la bande passante en limitant par exemple le temps de connexion de chaque utilisateur.
3. Sur le plan commercial, les portails captifs représentent une excellente opportunité pour un marketing transparent : ils facilitent l'engagement des utilisateurs à un moment critique de leur expérience Internet et constituent un moyen très puissant qui peut être utilisé pour répondre à divers besoins de l'entreprise. Utiliser un portail captif pour que les utilisateurs remplissent un sondage, voient une annonce sponsorisée ou mettent en évidence les promotions en cours. [13]

Remarque : en introduisant une redirection de trafic entre le paquet et sa destination, la stratégie du portail captif constitue une attaque de type Man In the Middle, il est donc possible à partir de 2015 d'avertir le client de l'existence d'un portail captif via le serveur DHCP (option 160) ou par le routeur. Le client aura donc la possibilité d'afficher le portail captif avant d'ouvrir d'autres tâches dont les paquets risqueraient d'être détournés.

Remarque : la redirection est effectuée grâce à un " DNS menteur " qui comme son nom l'indique, redirige le client vers une " fausse" page autre que celle demandé.

4.3.Exemple d'application de portail captif

- Dans les environnements privés, le portail captif peut être utilisé pour gérer l'entrée de visiteurs/employés dans le réseau d'entreprise. Il est à noter que l'utilisation du portail captif doit être associée à d'autres fonctionnalités de sécurité, permettant de contrôler la navigation des utilisateurs (définition de l'accès), la limitation de bande, l'isolation physique ou logique du réseau et d'autres ressources afin d'éviter toute compromission. Opérations de l'entreprise
- Une autre application assez courante du portail captif est associée à la gestion de l'accès Wi-Fi dans les événements. Pratiquement tous les événements, quelle que soit leur taille, fournissent une structure Internet aux participants, ce qui est indispensable pour garantir la sécurité et la disponibilité de cette ressource qui, dans certains cas, est directement associée au succès de l'événement.
- Les environnements publics tels que les hôtels, les hôpitaux, les cliniques, les centres commerciaux offrent également souvent un accès Internet à leurs clients ; Cependant, dans la plupart des cas, ces accès ne sont pas contrôlés, ce qui facilite l'action des utilisateurs malveillants. De plus, en raison de la survenue d'un incident de sécurité, l'absence d'enregistrement peut rendre difficile la détection des responsables de l'action.
- Ces dernières années, l'accès à Internet a cessé d'être une caractéristique exclusive des ordinateurs. La vulgarisation des smartphones, tablettes et autres appareils dotés d'une connexion Wi-Fi a permis de projeter de manière exponentielle les accès sur les réseaux publics et privés. Cette réalité exige le développement de contrôles de plus en plus perfectionnés sur la navigation de ces utilisateurs, garantissant la sécurité et le bon fonctionnement du service. [14]

4.4.Exemple de portail captif :

Il existe dans le monde du logiciel libre plusieurs solutions de portail captif dont voici les principaux :

4.4.1. Wifi Dog :

C'est un logiciel open source qui intègre un portail captif comme solution utiliséе pour construire des points d'accès sans fil. [15]

Caractéristiques :

- Fonctionne sur des plateformes embarquées ou autres
- Utilise le moins de ressources possibles
- Se compose de deux parties :
    - WifiDog Authentification Server : serveur d'authentification

- WifiDog Gateway : passerelle filtrante du système de portail captif
  Dépendances :
    - Un serveur web Apache mode SSL
    - Un serveur RADIUS
    - Un serveur DHCP
    - Un serveur DNS
    - Un pare-feu netfilter

4.4.2. Chili spot

ChilliSpot est un portail captif open source. Il est utilisé pour authentifier les utilisateurs d'un réseau local sans fil. Il prend en charge la connexion Web, qui est la norme actuelle pour les Hotspots publics.[16]

Caractéristiques :

- L'authentification, l'autorisation et la comptabilité.
- Connexion Internet.
- Wireless LAN Access point.
- Radius server.
- Web server.

4.4.3. NoDogsplash

NoDogsplash est un portail captif qui offre un moyen simple de fournir un accès restreint à Internet en affichant une page de démarrage à l'utilisateur avant que l'accès à Internet ne soit accordé.
Il a été dérivé à l'origine de la base de code du projet Wifi Guard Dog.[17]

Caractéristiques :

- Portail captif hautes performances.
- Faible encombrement.
- Il permet la création d'applications d'authentification sophistiquées.
- NoDogsplash supporte plusieurs moyens d'authentification :
    - Cliquez sur le bouton *Continuer* (par défaut)
    - Appelez un script externe pouvant accepter un nom d'utilisateur / mot de passe et définir les durées de session par utilisateur.
    - Transfert de l'authentification vers un service indépendant des NDS.

5. Pare-feu Netfliter :

5.1.Définition :

Netfilter est un module très performant de filtrage et de gestion de flux (les fonctions du pare-feu) au sein du noyau linux (à partir de la version 2.4), Il prévoit des accroches (Hook) dans le noyau pour l'interception et la manipulation des paquets, il est à la base de la plupart des principales solutions de pare-feu présentes dans le commerce que ça soit logicielles ou embarquées dans des boitiers matériels. Il permet de réaliser les fonctions suivantes :

- Réaliser des pares-feux à filtrage avec ou sans état.
- Utiliser le Nat pour masquer les identités ou pour partager des connexions internet.
- Déployer des proxys.
- Participer dans une architecture de QoS. [18]

5.2.Structure de netfilter :

Netfilter peut être vu comme une chaine de vérification, qui possède cinq points de vérification appelé Hook. Concrètement lorsqu'un paquet réseau arrive dans un de ces point de vérification, il est passé à netfilter par l'intermédiaire de sa fonction de rappel (*callback*). Il est alors examiné pour prendre une décision concernant son traitement futur, en effet, netfilter se comporte comme un automate qui compare le paquet successivement à plusieurs règles. Et selon le résultat du test, le paquet est traité ou transmis au test suivant.

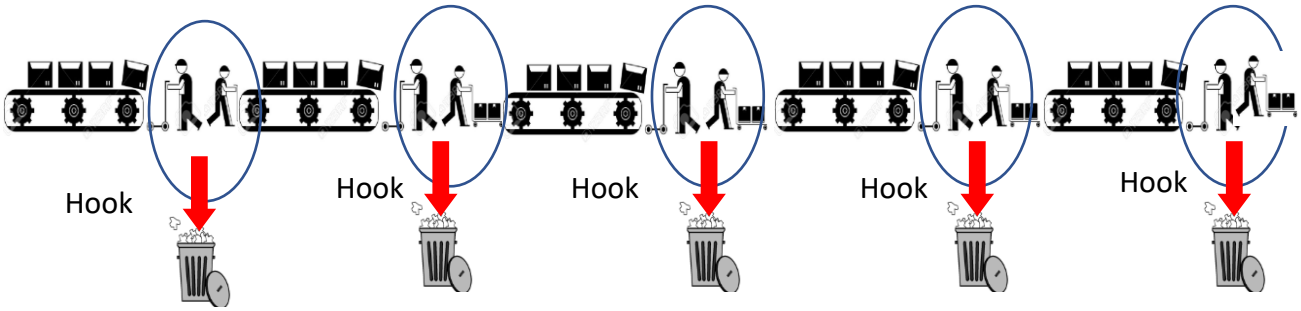


Figure II-4 structure de netfilter

Les chaines elles-mêmes sont formées d'un ensemble de règles de filtrage sauvegardées dans des tables particulière, ainsi, pour manipuler ces chaine l'administrateur réseau utilise un outil particulier de netfilter, en ligne de commande qui s'appelle **iptable**. [19]

5.3.Point d'accrochage (Hook) :

Les hooks ne sont que des points d'arrêt permettant d'examiner un paquet à un moment donné pendant sa transition sur le pare-feu. Il existe cinq points d'accrochage, comme c'est illustré dans la figure suivante :

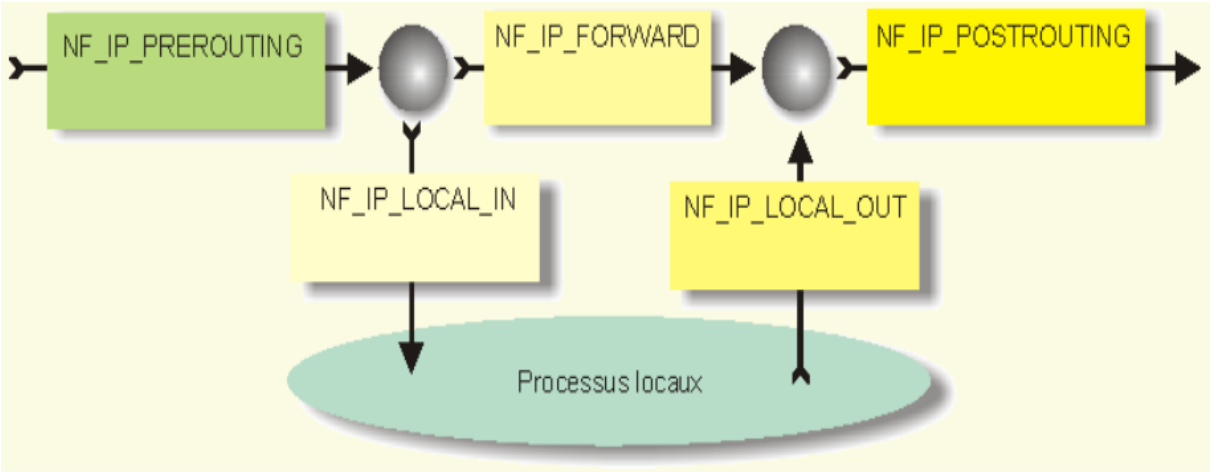


Figure II-5 point d'accrochage

- **NF_IP_PRE_ROUTING :** contient les paquets auxquels on n'a pas encore appliqué de décision de routage
- **NF_IP_LOCAL_IN :** contient les paquets qui ont été routés vers des processus locaux. Destinés aux serveurs implémentés sur l'hôte ou réside Netfilter (Web, FTP, …etc)
- **NF_IP_LOCAL_OUT** contient les paquets qui sortent des processus locaux
- **NF_IP_FORWARD** contient les paquets routés vers les réseaux (interne et externes et qui ne sont pas issus des processus locaux
- **NF_IP_POSTROUTING** contient tous les paquets destinés vers l'extérieur du pare-feu, qu'ils soient issus des processus locaux ou du réseau.[19]

5.4.Les tables, chaînes et cibles :

Netfilter possède trois tables prédéfinies : FILTER, NAT et MANGLE, chaque table contient des chaînes prédéfinies. L'administrateur peut modifier les chaînes prédéfinies, et créer ou détruire des chaînes personnelles dans n'importe quelle table. Au départ, toutes les chaînes sont vides et ont une politique par défaut accepter ou rejeter le paquet.

5.4.1. Table Filter

La table Filter sert principalement à filtrer les paquets. C'est l'endroit prévu pour intervenir sur les paquets et analyser leur contenu, c'est-à-dire les détruire (avec la cible **DROP**) ou les accepter (avec **ACCEPT**) suivant leur contenu. Chaque paquet passe à travers cette table, et traverse une et une seule des chaînes prédéfinies suivantes :

- **Chaîne INPUT** : Tous les paquets *destinés* à cet ordinateur passent dans cette chaîne.
- **Chaîne OUTPUT** : Tous les paquets *créés* par cet ordinateur passent dans cette chaîne.

- **Chaîne FORWARD** : Tous les paquets traversant cet ordinateur passent dans cette chaîne. [20]

5.4.2. Table NAT

La table NAT est responsable de la traduction d'adresses (modification des adresses et/ou des ports des paquets) sur les différents paquets. Le premier paquet de chaque connexion passe à travers cette table, Ensuite, les autres paquets subiront automatiquement le même sort que le premier. Voici les cibles actuelles capables d'accomplir ce genre de choses :

- La cible **DNAT** est généralement utile dans le cas où on dispose d'une adresse IP publique et qu'on désire rediriger les accès vers un pare-feu localisé sur un autre hôte (par exemple, dans une zone démilitarisée ou DMZ). Concrètement, on change l'adresse de destination du paquet avant de le router à nouveau vers l'hôte désigné.
- La cible **SNAT** est quant à elle employée pour changer l'adresse de source des paquets. La plupart du temps, elle est utilisée pour cacher le réseau local ou la DMZ, etc. par exemple, un pare-feu pour lequel l'adresse externe est connue, mais qui nécessite de remplacer les adresses IP du réseau local avec celle du pare-feu. Avec cette cible, le pare-feu effectuera automatiquement sur les paquets du **SNAT** dans un sens et du **SNAT inverse** dans l'autre, rendant possible les connexions d'un réseau local sur Internet.
- La cible **MASQUERADE** s'utilise exactement de la même façon que la cible **SNAT**, mais la cible **MASQUERADE** demande un peu plus de ressources pour s'exécuter. L'explication vient du fait que chaque fois qu'un paquet atteint la cible **MASQUERADE**, il vérifie automatiquement l'adresse IP à utiliser, au lieu de se comporter comme la cible **SNAT** qui se réfère simplement à l'unique adresse IP configurée. Par conséquent, la cible **MASQUERADE** permet de faire fonctionner un système d'adressage IP dynamique sous DHCP, que les FAI devrait les fournir pour des connexions à Internet de type PPP, PPPoE ou SLIP.

La table NAT possède 3 chaines prédéfinies :

- **Chaîne PREROUTING** : Permet la traduction d'adresses de destination uniquement pour les paquets entrant dans l'ordinateur. Le nom de la chaîne vient du fait que les paquets passent cette chaîne avant de passer dans les tables de routage.
- **Chaîne POSTROUTING** : Permet la traduction d'adresses source uniquement pour les *paquets sortant* de l'ordinateur. Le nom de la chaîne vient du fait que les paquets passent cette chaîne après avoir passé les tables de routage.
- **Chaîne OUTPUT** : Permet la *traduction d'adresses de destination* uniquement pour les *paquets créés* par l'ordinateur.

5.4.3. Table Mangle

Cette table est responsable de la transformation des options des paquets, comme la qualité de service. Chaque paquet passe à travers cette table. Son but n'est pas le filtrage de paquet Comme cette table est prévue pour des options avancées, elle contient toutes les chaînes prédéfinies :

- **Chaîne PREROUTING** : Tous les paquets *entrant* sur l'ordinateur passent dans cette chaîne *avant d'avoir passé le routage*.
- **Chaîne INPUT** : Tous les paquets *destinés* à l'ordinateur passent dans cette chaîne.
- **Chaîne OUTPUT** : Tous les paquets *créés* par l'ordinateur passent dans cette chaîne.
- **Chaîne FORWARD** : Tous les paquets *traversant* l'ordinateur passent dans cette chaîne.
- **Chaîne POSTROUTING** : Tous les paquets *quittant* l'ordinateur passent dans cette chaîne *après avoir passé le routage*.

Les cibles suivantes sont valides uniquement dans la table mangle, elles ne doivent pas être utilisées en dehors de cette table.

- La cible **TOS** permet de définir et/ou modifier le champ de Type de Service d'un paquet. C'est utile pour définir des stratégies réseau concernant le choix de routage des paquets.
- La cible **TTL** permet de modifier le champ duré de vie ou TTL (Time To Live) d'un paquet. Il est possible par exemple de spécifier aux paquets d'avoir un champ TTL spécifique. Ceci peut se justifier lorsque vous ne souhaitez pas être rejeté par certains Fournisseurs d'Accès à Internet (FAI) trop indiscrets.
- La cible **MARK** permet d'associer des valeurs de marquage particulières aux paquets. Elles peuvent ensuite être identifiées par les programmes **iproute2** pour appliquer un routage différent en fonction de l'existence ou de l'absence de telle ou telle marque. On peut ainsi réaliser de la restriction de bande passante et de la gestion de priorité.
- La cible **SECMARK** peut être utilisée pour placer des marques dans un contexte de sécurité sur des paquets dans SELinux ou tout autre système de sécurité capable de gérer ces marques.
- **CONNSECMARK** sert à copier un contexte de sécurité vers ou depuis un simple paquet ou vers une connexion complète. Elle est utilisée par SELinux ou autre système de sécurité pour affiner cette sécurité au niveau connexion.[21]

Conclusion

Dans ce chapitre nous avons présenté en premier lieu les pares-feux, leur principe de fonctionnement et les différentes catégories qui existais, ensuite,

Dans le chapitre suivant nous allons aborder l'analyse et la conception de notre solution

III. CHAPITRE 3

CONCEPTION

Introduction

Dans ce chapitre nous aborderons une description générale de notre projet ensuite nous mettons en évidence le côté conceptuel de notre projet qui constitue une étape fondamentale qui précède la réalisation, permet de détailler les différents diagrammes et scénarios à implémenter dans la phase suivante. Ceci permettra de mieux comprendre notre projet.

1. Présentation de notre application

Notre projet consiste à réaliser un système embarqué intégrant un point d'accès wifi pour l'utilisation public (un restaurant, coffee shop, bibliothèque), Ce point d'accès sera doté des fonctionnalités suivantes :

1.1. Un portail captif

Dès qu'un utilisateur se connecte à notre point d'accès, et avant d'avoir accès à internet, le client est redirigé vers notre portail captif qui est une page web affichant une publicité est un bouton accéder à internet.

1.2. Un pare-feu

Qui est implémenté sur le point d'accès, et géré par l'administrateur via une interface web accessible par le réseau local du Point d'accès, cette interface web est protégé par un mot de passe.

Nous avons choisi la Raspberry pi 3 model b+ avec le système d'exploitation Raspbian pour réaliser cette solution. Cette carte a deux interfaces réseaux une interface wifi et l'autre Ethernet.

L'interface wifi est utilisée pour réaliser le point d'accès et l'interface Ethernet est utilisée pour fournir l'accès internet à la carte.

2. Conception du portail captif

Tout d'abord expliquons le principe avec lequel les périphériques des clients détectent l'existence d'un portail captif.

Le mécanisme de la détection automatique du portail captif est basé sur une simple vérification effectuée par le système opératoire du périphérique du client (smart-phone, tablette, laptop).

Chaque périphérique, après connexion au point d'accès, dans l'arrière-plan, va générer une requête http de test de connectivité ou l'URL de la requête est spécifié par le fabriquant du périphérique.

1.s'il n'y a pas de portail captif, la réponse de requête HTTP sera HTTP 204 indiquant l'accès internet.

2. Si la réponse de la requête est HTTP 302 qui le code de redirection, le périphérique réalise l'existence d'un portail captif et sans terminer le test de connexion le périphérique ouvre une nouvelle fenêtre du navigateur avec le lien de redirection et indiquant aux utilisateurs qu'il y a un portail captif.

Les différents liens de test de connectivité pour chaque fabriquant :

Android Captive Portal Détection

- clients3.google.com
- connectivitycheck.gstatic.com
- connectivitycheck.android.com

Apple iPhone, iPad with iOS 6 Captive Portal Detection

- gsp1.apple.com
- *.akamaitechnologies.com
- www.apple.com
- apple.com

Apple iPhone, iPad with iOS 7, 8, 9 and recent versions of OS X

- www.appleiphonecell.com
- *.apple.com
- www.itools.info
- www.ibook.info
- www.airport.us
- www.thinkdifferent.us
- *.apple.com.edgekey.net
- *.akamaiedge.net
- *.akamaitechnologies.com

Windows

- ipv6.msftncsi.com
- ipv6.msftncsi.com.edgesuite.net
- www.msftncsi.com
- www.msftncsi.com.edgesuite.net
- teredo.ipv6.microsoft.com
- teredo.ipv6.microsoft.com.nsatc.net

Pour réaliser le portail captif nous avons besoin de trois composants logiciels :

1.l'outil netfilter.

2.un serveur reverse proxy

3.un serveur web pour la page du portail captif.

L'algorithme suivi dans la conception du portail captif et comme suit:

Ajouter les règles suivantes au pare-feu netfilter.

```
$ Iptables −t mangle −N internet
$ Iptables −t mangle −A internet −j MARK −−set −mark 99
$ Iptables −t mangle −A PREROUTING −i eth1 −p tcp −m  tcp −−d port 80 −j internet
$ Iptables −t nat −A PREROUTING −i eth1 −p tcp −m mark −−mark 99 −m tcp
−−dport 80 −j DNAT −−to−destination 192.168.4.1
$ Iptables −t mangle −A PREROUTING −i eth1 −p tcp −m tcp −−dport 443 −j internet
$ Iptables −t nat −A PREROUTING −i eth1 −p tcp −m mark −−mark 99 −m tcp
−−dport 443 −j DNAT −−to−destination 192.168.4.1
```

1. Cette configuration du pare-feu nous permet de marquer tout le traffic HTTP et HTTPS avec un numéro, dans notre cas c'est 99, après, nous redirigeons les paquets marqués à l'aide de la table PREROUTING vers notre serveur reverse proxy, qui est à l'adresse192.168.4.1. En conséquence toutes les requêtes de test de connectivité sont redirigées vers le serveur reverse proxy.

2. Le serveur reverse proxy va répondre aux requêtes de test de connectivité par un HTTP 302 redirigent les périphériques connectés aux points d'accès vers notre serveur web contenant la page du portail captif.
3. Une fois le client arrivé à la page du portail captif, il clique sur le bouton accéder à internet qui va démarquer les paquets HTTP venant de son périphérique à l'aide de son adresse MAC, en conséquence les paquets venant de ce périphérique ne passeront plus par la table PREROUTING.

4. Après le démarquage des paquet HTTP le périphérique du client peut terminer le test de connectivité et il aura accès à internet.

Cet algorithme est illustré dans le schéma suivant :

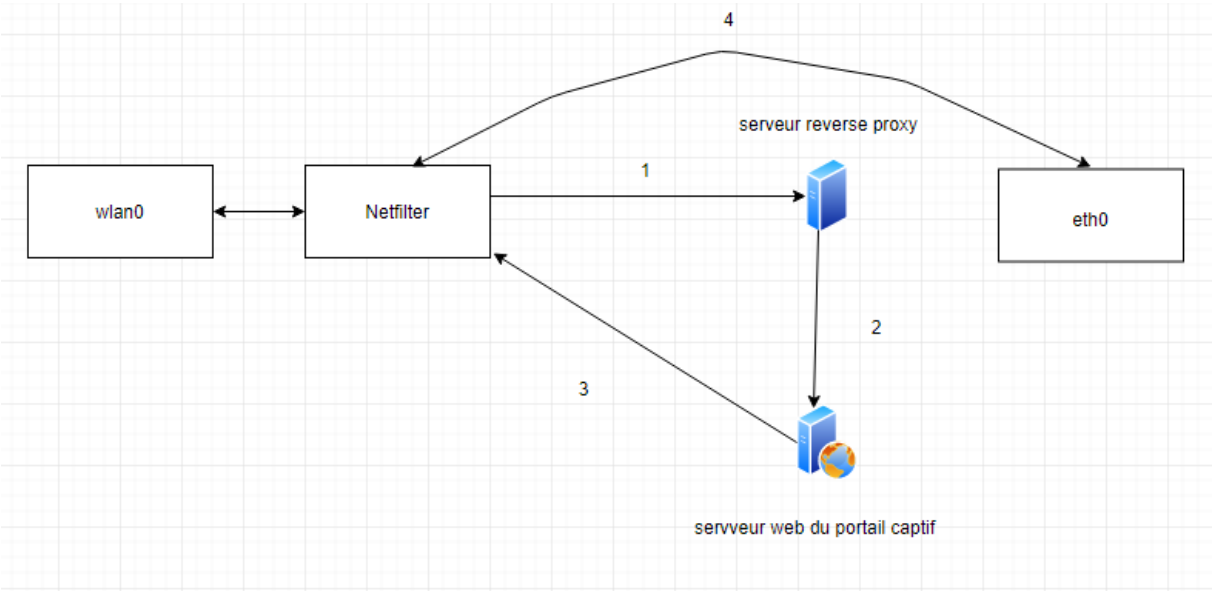


Figure III-1 Schéma logiciel de notre portail captif

3. Conception de l'interface web de la configuration du pare-feu :

Afin de concevoir cette interface nous avons besoin d'un serveur web et l'outil Netfilter.

Le serveur web fournit une interface repartie en deux sections :

Première section : un ensemble de champs inputs pour récupérer les arguments de la nouvelle règle à ajouter dans le pare-feu (adresse ip source, ip destination, port source, port destination, Protocol), et un bouton pour exécuter la règle.

Deuxième section : un tableau affichant les règles déjà insérées par l'administrateur avec un bouton supprimer pour chaque règle.

La liste des règles déjà ajoutées au pare-feu est enregistrée dans un fichier Json qui est mis à jour par l'application à chaque fois l'admin ajoute ou supprime une règle.

4. Fonctionnement de l'interface graphique

Nous allons illustrer le fonctionnement avec un schéma

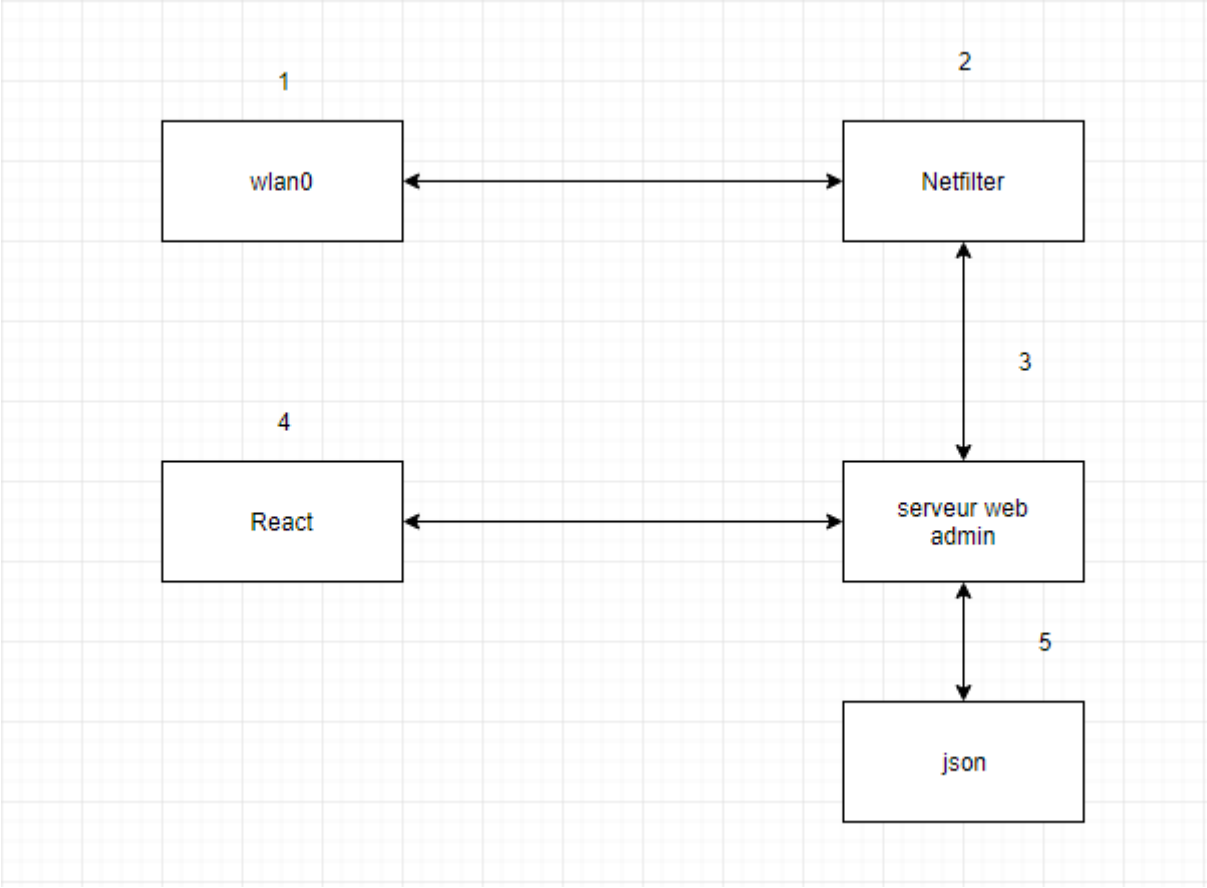


Figure III-2 Schéma logicielle de l'interface graphique de configuration

1. l'admin connecte au point d'accès
2. l'admin passe par le portail et effectue la connexion.
3. l'admin saisie l'adresse du serveur web avec le port.
4. récupérer les arguments de la nouvelle règle a ajouter ou la commande de suppression d'une ancienne règle.
5. exécution de la commande choisie par l'admin.
6. ajout de la nouvelle règle dans un fichier Json ou supprimer une règle déjà existante.

5. Architecture logicielle globale :

La figure suivante présente l'architecture logicielle globale, que nous avons expliqué précédemment

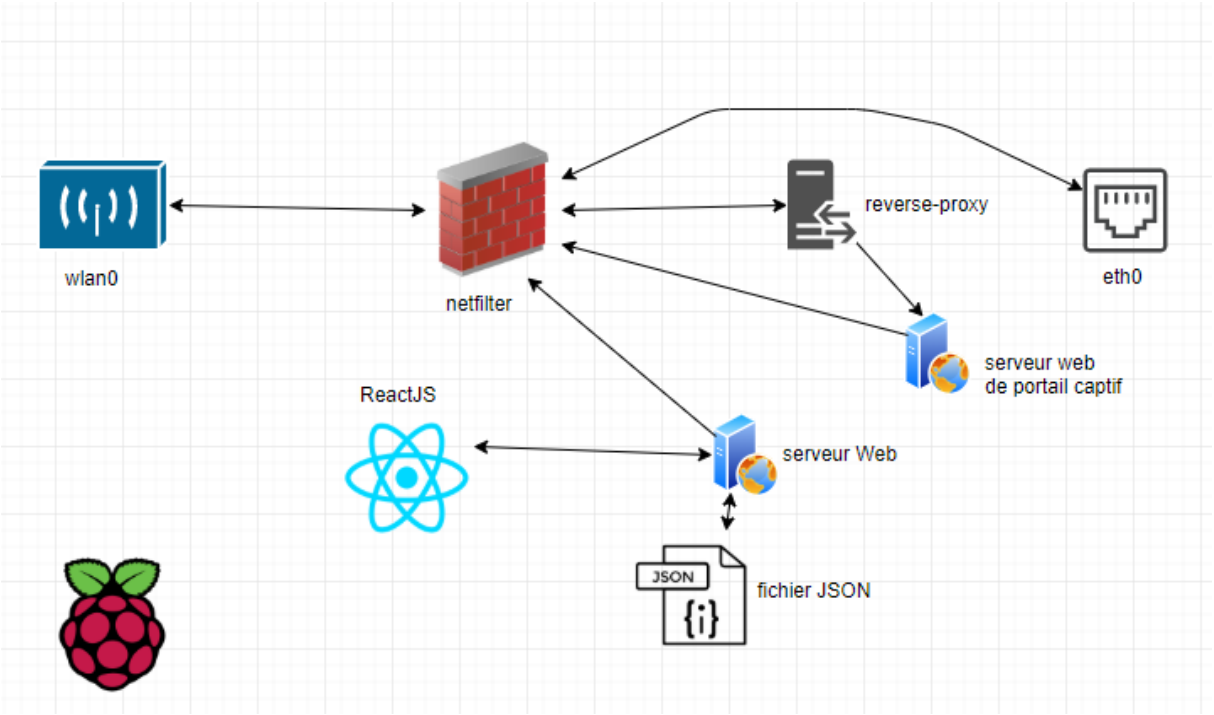


Figure III-3 schéma logiciel global de notre système

6. Architecture matérielle

Afin de réaliser le système décrit précédemment, nous avons choisi d'utiliser une Raspberry pi 3 model B+ qui est équipée de deux interfaces réseau, un wifi utilisé comme point d'accès et une autre interface Ethernet connectée à un modem qui possède une connexion internet. Nous illustrons cette architecture dans la figure suivante :

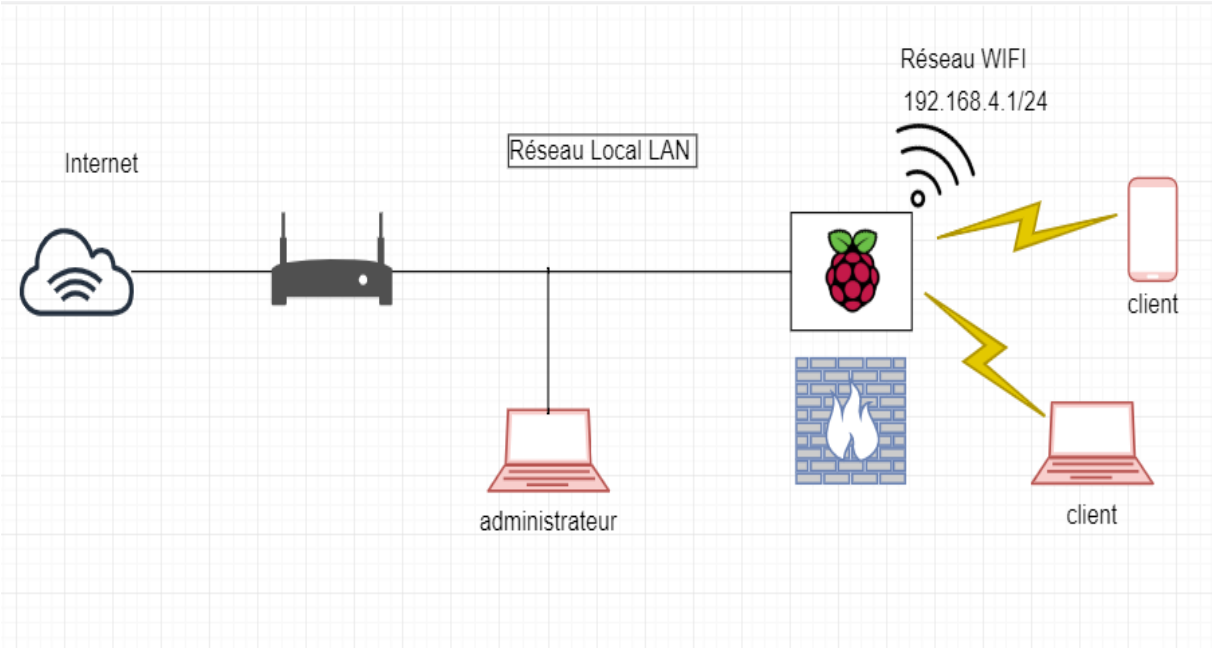


Figure III-4 architecture matérielle de notre système

Conclusion

Tout au long de ce chapitre, nous avons exposé les différents besoins auxquels doit répondre le système ainsi que l'étude des différents concepts et moyens utilisés pour la réalisation de ce système.

Dans le chapitre suivant, nous allons aborder la réalisation de notre système en pratique.

IV. CHAPITRE 4

RÉALISATION

Introduction

La réalisation est la phase la plus importante après celle de la conception. Le choix des outils de développement détermine énormément le cout en temps de programmation, ainsi que la flexibilité du produit à réaliser. Cette phase consiste à transformer le modèle conceptuel établi en des composants logiciels formant notre système. Dans un premier temps, nous allons procéder à la spécification de l'environnement matériel et logiciel utilisé dans notre projet. Ensuite nous nous intéresserons à décrire les différentes étapes de réalisation de l'application « portail captif et interface netfilter ».

1. Description du matériel

1.1.Raspberry

Raspberry est une carte mère d'un nano-ordinateur. C'est un ordinateur embarqué qui peut être branchée à n'importe quel périphérique (souris, clavier...). Cette carte est fabriquée pour aider à étudier les ordinateurs et pour représenter un moyen d'apprentissage de la programmation informatique en plusieurs langages (python, scratch...). Elle est aussi capable de lire les vidéos à haute définition et même à installer des jeux vidéo. L'intérêt d'utiliser la Raspberry Pi est sa capacité d'interaction avec le monde extérieur et d'exécuter plusieurs variantes du système d'exploitation libre (GNU/Linux, Raspbian Debian ...) et d'autres logiciels compatibles.

1.2.Les modèles de Raspberry Pi

1.2.1. Raspberry Pi 1

Cette première génération est arrivée en 2012 avec deux versions, les versions A et B puis l'année suivante les versions A+ et B+, la différence entre ces modèles est la taille des GPIO, le nombre de ports USB et la présence ou non d'Ethernet, le support de la carte MicroSD et la taille de la RAM.

Modèle A

Figure IV-1 Raspberry pi 1 model A

Model B

Figure IV-2 Raspberry pi 1 model B

	MODEL A	MODEL B
PROCESSEUR	ARM1176JZF-S (ARMv6) 700 MHz Broadcom 2835	ARM1176JZF-S (ARMv6) 700 MHz Broadcom 2835
RAM	256 Mo	512 Mo
SORTIE VIDEO	Composite et HDMI	Composite et HDMI
SORTIE AUDIO	Sortie Jack 3.5	Sortie Jack 3.5
USB	1 Port USB 2.0	2 Port USB 2.0
ETHERNET	N / A	1 Port Fast Ethernet (10/100 Mbit/s)

Figure IV-3 comparaison entre Raspberry pi 1 model A et B

Différence modèle A+ avec le A:

-Plus petit : 65 mm de long (contre 86 mm)

-Lecteur de carte microSD à la place du lecteur de carte SD

-GPIO 40 broches

-Nouveau chipset audio

-Consommation électrique moindre

Figure IV-4 Raspberry pi 1 model A+

Différence modèle B+ avec le B:

-GPIO 40 broches

-4 ports USB 2.0 au lieu de 2

-Réduction de consommation de 3,5 W à 3 W

-Nouveau chipset audio

Figure IV-5Raspberry pi 1 model B+

1.2.2. Raspberry Pi 2

Pour la version 2, il y a qu'un seul modèle, le B. Ce modèle est sorti en Février 2015, il est équipé du processeur Broadcom BCM2836 avec quatre cœurs ARMv7 à 900 MHz et1 Go de RAM. Concernant les connectiques il n'y a pas de différence avec le modèle B+.

Figure IV-6Raspberry pi 2

1.2.3. Raspberry Pi 3

Le Raspberry de la troisième génération présente une puissance de traitement encore plus grande ainsi qu'une connectivité sans fils embarqué et le Bluetooth.

1.2.3.1. Raspberry Pi 3 modèle 3B

Elle repose sur un processeur à quadruple cœur ARM Cortex-A53, le BCM2837 de Broadcom, cadencé à 1,2 GHz. Cela signifie qu'il est 50 à 60% plus rapide que le Raspberry Pi 2 B.La communication par Wi-Fi 802.11n et du Bluetooth 4.1 est maintenant intégrée au RPi 3 ; ce nouveau modèle est toujours rétro compatible avec les modèles précédents. [17]

Les principales caractéristiques de cette carte sont :

- Processeur BCM2837 de Broadcom, cadencé à 1,2 GHz
- Quadruple cœur ARM Cortex-A53 à 64 bits
- Wi-Fi 802.11 b/g/n intégré
- Bluetooth 4.1 (Classic& Low Energy) intégré
- Coprocesseur multimédia à double cœur Videocore IV®
- Mémoire LPDDR2 de 1 Go
- Supporte les distributions GNU/Linux ARM les plus récentes et Windows 10 IoT
- Connecteur micro-USB pour alimentation 2,5 A
- 1 port Ethernet 10/100
- 1 connecteur audio/vidéo HDMI
- 1 connecteur audio/vidéo RCA
- 4 port USB 2.0
- 40 broches d'E/S à usage général
- Antenne sous forme de puce
- Connecteur DSI pour écran
- Lecteur de carte micro-SD
- Dimensions : 85 x 56 x 17 mm

1.2.3.2. Raspberry Pi 3 modèle 3B+

Le Raspberry Pi 3 Model B + est le dernier produit de la gamme Raspberry Pi 3, avec un processeur quad core 64 bits cadencé à 1,4 GHz, un WiFIbibande 2,4 GHz et 5 GHz, du Bluetooth 4.2 / BLE, un Ethernet plus rapide.

Le Raspberry Pi 3 Model B + conserve la même empreinte mécanique que le Raspberry Pi 2 Model B et le Raspberry Pi 3 Model B.

Figure IV-7Raspberry pi 3 model B+

En regardant les caractéristiques des cartes Raspberry cité, nous allons remarquer que la Raspberry pi 3 B+ est le meilleur choix pour notre système, car cette dernière est plus performante que les précédente en particulier la vitesse de calcul.

2. Coté software :

2.1.Systèmes d'exploitation pour la Raspberry Pi

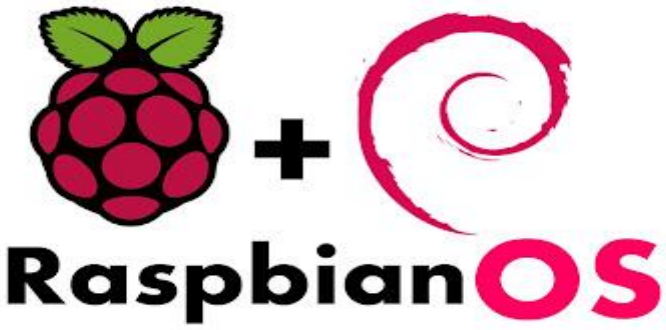
Les Raspberry Pi sont livrés par défaut vierges de tout système d'exploitation et de tout dispositif de stockage. C'est à l'utilisateur de choisir et d'installer le système d'exploitation qu'il veut (la plus grande majorité sont des systèmes Linux) et de choisir la carte de stockage de type micro SD avec la capacité qu'il veut (16 Go minimum conseillés).

Il y a beaucoup de différents systèmes d'exploitation disponibles pour le Raspberry pi.

Presque tous les systèmes d'exploitation qui fonctionne sur un processeur ARM peuvent normalement être installé sur la Raspberry po, nous présentons quelques-uns des plus populaires systèmes d'exploitation personnalisés pour le Raspberry pi .

2.1.1. Raspbian

Raspbian, est un système d'exploitation libre et gratuit basé sur Debian GNU/Linux optimisé pour fonctionner sur un Raspberry pi. Il représente un ensemble de programme et d'utilitaires s'assure que votre appareil fonctionne correctement. Raspbian n'est pas qu'un système d'exploitation, il vient également avec plus de 35000 paquets dans un format adapté pour une installation rapide. Raspbian s'améliore chaque jour et est toujours en développement. Ce système est livré avec un environnement de bureau conçu pour ressembler aux systèmes d'exploitation que nous utilisons tous les jours. Utilisation de l'environnement de bureau est parfait pour les utilisateurs qui ne maitrisent pas bien la console linux.



Avantage :

- Facile a utilisé.
- Excellent support.

Inconvénients :

- Mises à jour logiciels retardées

2.1.2. Kali linux :

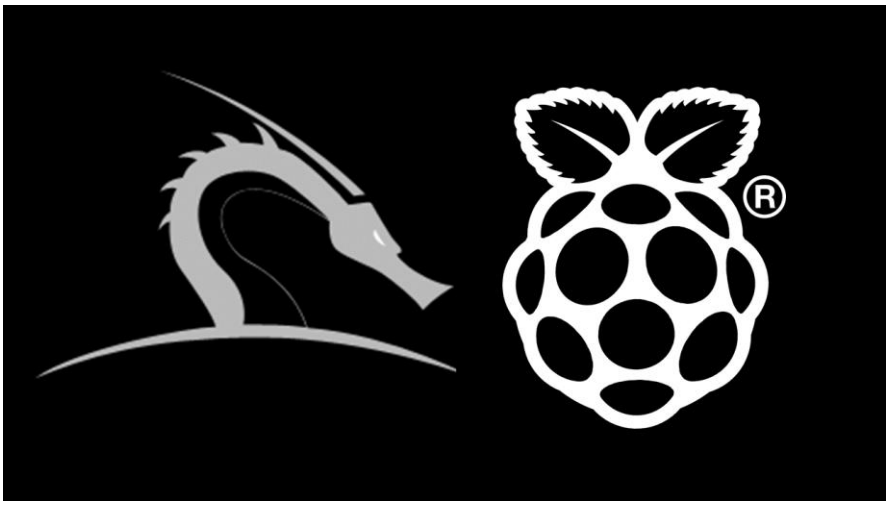
Est un autre dérivé de Debian qui peut être utilisé dans son édition arm comme distribution pour Raspberry pi, la distribution est open source et peut être utilisé comme un système d'exploitation ordinaire, mais, elle est principalement utilisée pour des tests de sécurité et des tests d'intrusion étendus.

Avantages :

- Divers outils de sécurité intégrés.
- Permet de tester la sécurité du réseau.

Inconvénient :

- Inapproprié pour les débutants sur linux
- Exigences relativement élevées en matière de RAM



2.1.3. Windows IoT :

Windows IoT est le premier système de Microsoft destiné aux appareils IoT comme la Raspberry pi 3, Windows IoT s'appuie sur l'API interne « Universal Windows platform », qui permet aux développeurs d'écrire des applications pour leur propres appareils.

Avantages :

- Très facile a utilisé.

Inconvénient :

- Il n'y a pas de possibilité d'intégrés Iptables.
- Parfois très gourmand en ressources.

Apres comparaison entre ces systèmes d'exploitation, nous avons choisi d'utiliser le système d'exploitation Raspbian.

2.2.Installation de Raspbian

Avant de commencer à installer le système d'exploitation sur Raspberry nous aurons besoin de :

- Carte SD : il existe plusieurs types des Carte SD, il est recommandé d'utiliser une carte SD de 8Go au minimum de classe 10 ou classe 4
- Cable d'affichage et de connectivité : Tout moniteur ou téléviseur HDMI / DVI doit fonctionner comme un affichage pour le Pi. Pour de meilleurs résultats, nous avons utilisé un écran avec entrée HDMI pour la première utilisation de la carte.
- Clavier et souris : Tous les claviers et souris USB standard fonctionneront avec votre Raspberry Pi.
- Une source du courant : Le Pi est alimenté par une micro-alimentation USB.

Une fois nous avons tous ces besoins, nous allons passer au côté logiciel, nous avons suivi les étapes suivantes afin de bien installer les système Raspbian :

Nous avons utilisé l'outil NOOBS, ce dernier est un programme d'installation simple de système d'exploitation, il contient Raspbian et libreELEC, NOOBS peut être télécharger à partir de site officiel de Raspberry.

Une fois NOOBS est téléchargé, il suffit de :

- Extraire tous les fichiers.
- Copier tous les fichiers obtenus dans carte SD.
- Insérer la carte SD dans le Raspberry.
- Allumer le Raspberry.

Le menu de démarrage de Noobs nous permet de choisir la distribution que nous souhaitons installer, Raspbian, Pidora, RaspBMC, RiscOS, Arch Linux ou OpenELEC.

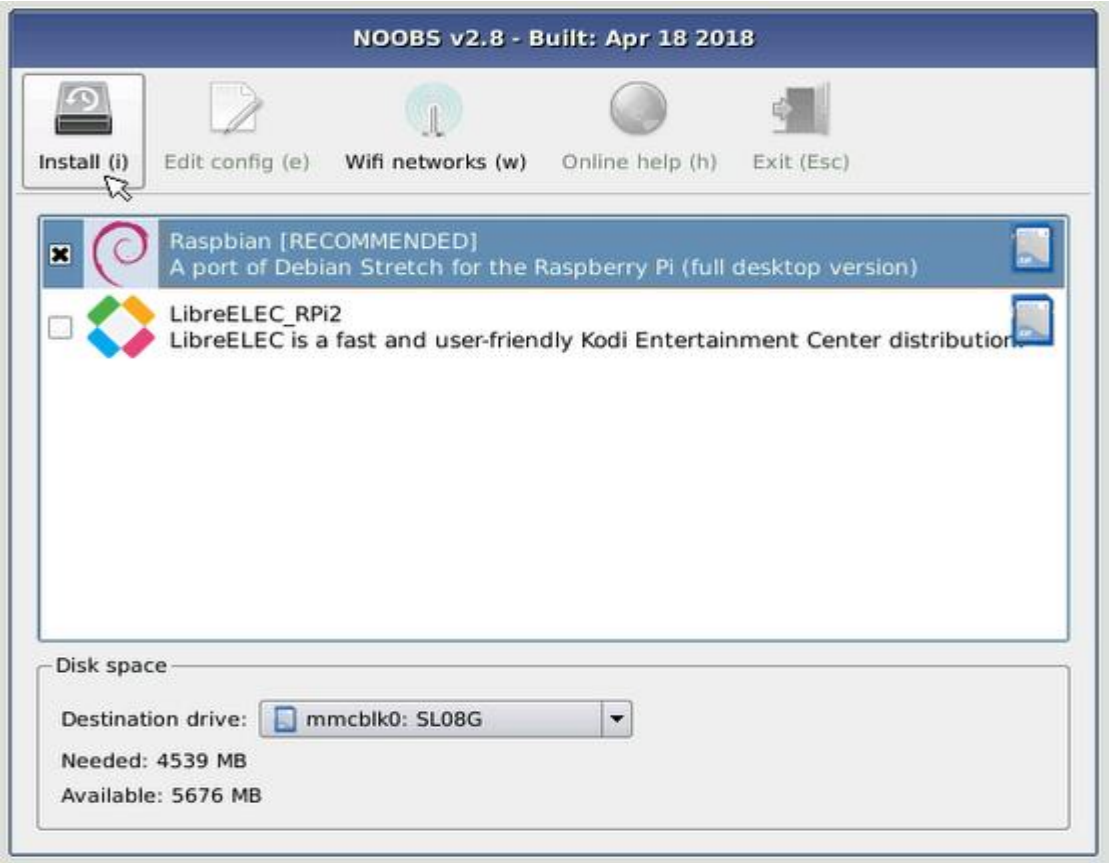
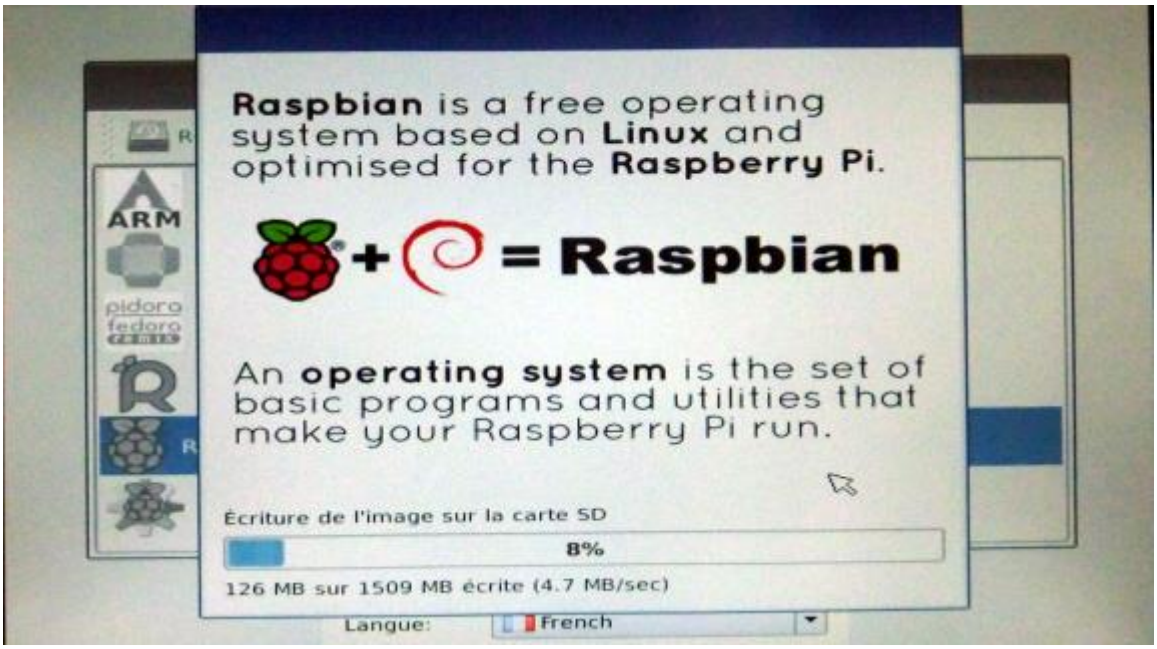


Figure IV-8 interface d'installation de Raspbian

Il suffit juste de cocher le système à installer, puis cliquer sur installe et Noobs va tous faire



Une fois l'installation est terminé, le système Raspbian démarre et on arrive sur l'ecran de configuration Raspi-conf

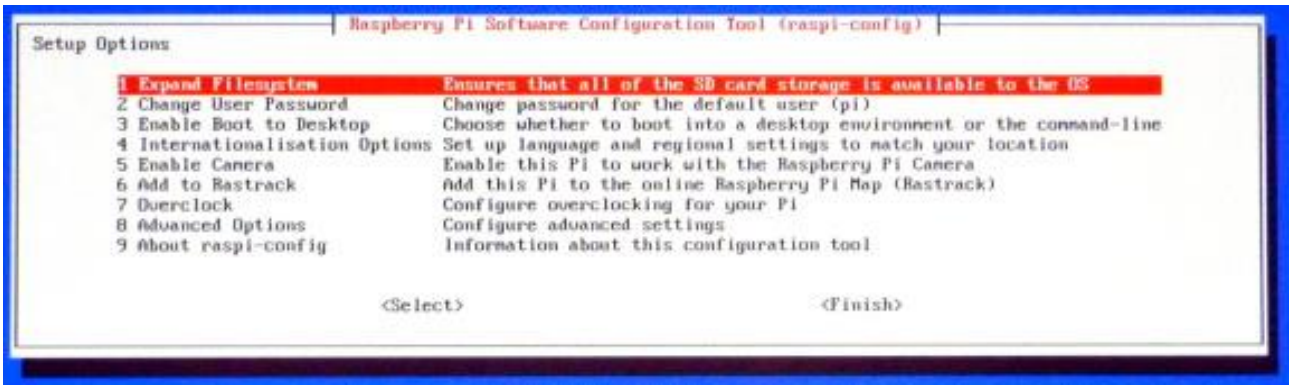


Figure IV-9 interface de configuration de Raspberry

Raspi-config est un outil de configuration écrit a l'origine par Alex Bradbury, il vise Raspbian.

Pour pouvoir utiliser cet outil après le premier démarrage, il suffit d'exécuter l'opération suivante dans la ligne de commande :

```
$ Sudo Raspi-conf.
```

Nous trouvons les opérations suivantes :

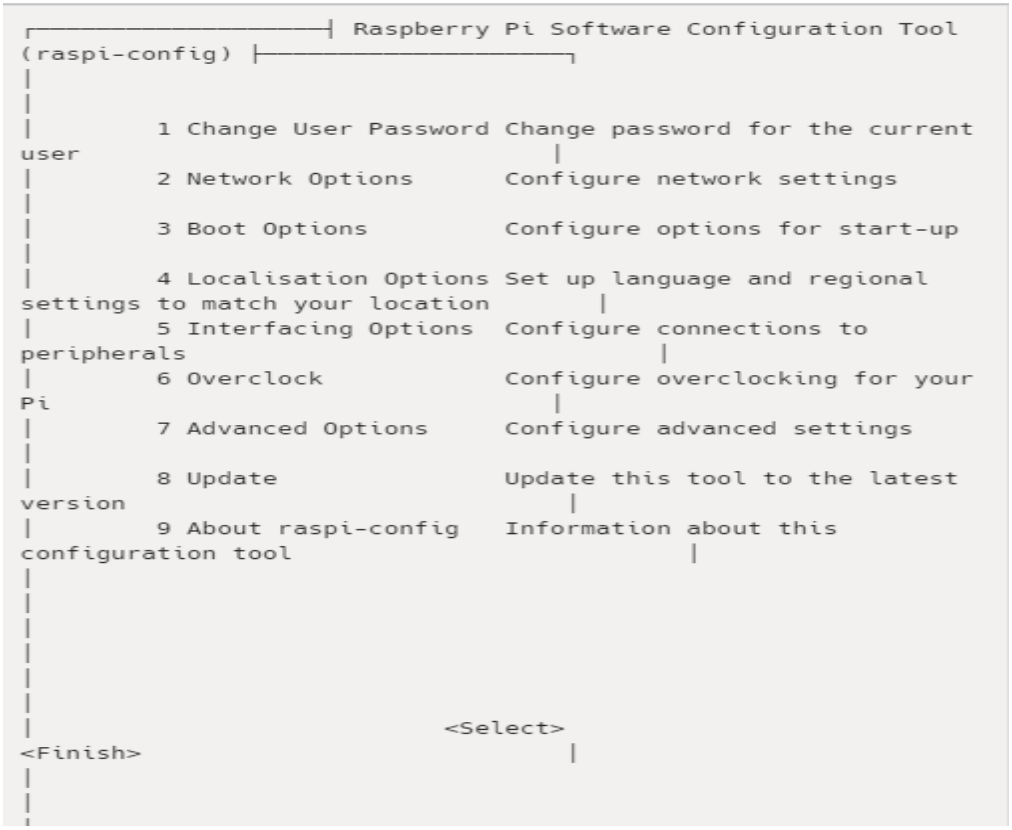


Figure IV-10 les opération disponible dans l'outil Raspi-conf

Création et configuration de point d'accès

Le Raspberry Pi peut être utile comme point d'accès sans fil sur un réseau autonome.

Pour créer un point d'accès, le Raspberry doit avoir un logiciel de point d'accès, ainsi qu'un logiciel de serveur DHCP, permettant aux périphériques de connexion d'accéder à une adresse réseau. Avant de commencer l'installation, il faut s'assurer que le système est à jour avec la commande suivante :

```
$ Sudo apt-get update
```

Une fois le système est à jour, nous passons à l'installation des logiciel requis Hostapd et Dnsmasq avec la commande suivante :

```
$ Sudo apt install dnsmasq hostapd
```

Hostapd : c'est un logiciel démon qui permet aux cartes d'interface réseau d'agir comme un point d'accès.

Dnsmasq : DnsMasq est un petit serveur DNS (cache DNS) qui intègre un serveur DHCP. Peu gourmand en ressources et très simple à configurer, il est bien adapté à une installation sur une solution embarquée.

Configuration d'adresse ip statique

Pour commencer a créé notre point d'accès, nous devons attribuer une adresse IP statique à l'interface réseau sans fil « wlan0 », pour ce faire, nous allons ouvrir le fichier de configuration et le modifier comme suit :

```
$ sudo nano /etc/dhcpcd.conf
```

Ensuite modifier le contenu de fichier, après modification et enregistrement, nous avons redémarré dhcpcd demon avec la commande suivante :

```
$ sudo systemctl restart dhcpcd
```

Configuration de serveur DHCP

Le service DHCP est fourni par dnsmasq. Par défaut, le fichier de configuration contient beaucoup d'informations inutiles et il est plus facile de recommencer à zéro. Nous avons renommé ce fichier de configuration et modifions-en un nouveau avec les commandes suivantes :

```
$sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
$ sudo nano /etc/dnsmasq.conf
```

Ensuite nous saisissons les instructions suivantes et nous enregistrons le fichier :

```
Interface=wlan0
dhcp-range=192.168.4.2,192.168.4.254,255.255.255.0,24h
```

Donc ici notre point d'accès wifi va fournir des adresses ip dynamiquement, comprise entre 192.168.4.2 et 192.168.4.254 avec une durée du bail de 24h.

Pour que la configuration va être mise a jour nous avons rechargé le service dnsmasq avec la commande suivante :

```
$ sudosystemctlreloaddnsmasq
```

Configuration du hostapd

Nous devons éditer le fichier de configuration hostapd, situé dans /etc/hostapd/hostapd.conf, pour ajouter les différents paramètres de Notre réseau sans fil. Après l'installation initiale, ce sera un nouveau fichier / vide.

```
$ sudo nano /etc/hostapd/hostapd.conf
```

nous avons ajouter les informations ci-dessous au fichier de configuration.

```
interface=wlan0
driver=nl80211
ssid=RaspiSH
hw_mode=g
channel=7
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
```

Nous devons maintenant indiquer au système où trouver ce fichier de configuration, comme suit :

```
$ DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Nous allons remplacer la ligne #DAEMON_CONF par :

```
$ sudo nano /etc/default/hostapd
```

Ensuite nous allons activer et démarrer le hostapd en exécutant les commandes suivantes :

```
$ sudosystemctlunmask hostapd
$ sudosystemctl enable hostapd
$ sudosystemctl start hostapd
```

Maintenant nous allons s'intéresser au partage de connexion Internet, de sorte que toutes personnes connectées au point d'accès puissent accéder à Internet, à condition que l'Ethernet câblé du Pi soit déjà connecté à Internet, via un modem, donc, nous allons ajouter quelques instructions à la configuration précédente.

Pour ce faire, un « pont » doit être mis en place entre le périphérique sans fil et le périphérique Ethernet sur le point d'accès Raspberry Pi. Ce pont passera tout le trafic entre les deux interfaces. Nous allons Installé le packages suivant pour activer la configuration et le pontage du point d'accès :

```
$ sudo apt install bridge-utils
```

Le pontage crée une construction de plus haut niveau sur les deux ports en cours de pontage. C'est le pont qui constitue le périphérique réseau. Nous devons donc empêcher les adresses IP attribuées aux ports eth0et wlan0attribuées par le client DHCP du Raspberry Pi aux ports.

Nous allons ajouter les deux commandes suivante à la fin du fichier « dhcpcd.conf »

```
denyinterfaces wlan0
denyinterfaces eth0
```

nous enregistrons le fichier, ensuite, le fichier d'interface doit maintenant être modifié pour que les différents périphériques puissent fonctionner avec un pontage.

Nous ouvrons le fichier :

```
sudo nano /etc/network/interfaces
```

Puis nous ajoutons les informations suivantes dans le fichier :

```
# Bridge setup
auto br0
iface br0 inetmanual
bridge_ports eth0 wlan0
```

Maintenant il faut redémarrer le service dhcpcd pour que les modifications prennent effet :

```
Sudo ifup br0
sudosystemctl restart dhcpcd
```

Puis, nous allons ajouter l'instruction suivante au fichier « hostapd.conf » :

```
bridge=br0
```

Et mettre la commenter l'instruction « driver=nl80211 »

Elle doit être juste au-dessous de l'instruction « interface=wlan0 » comme suit :

```
interface=wlan0
bridge=br0
#driver=nl80211
ssid=RaspiSH
hw_mode=g
channel=7
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
```

Ensuite il suffit d'activer et de démarrer le hostapd.

Ajouter le routage et le masque rade

Pour ajouter le routage nous devons modifier l'instruction qui permet le routage au sein de fichier « etc/sysctl.conf » nous aurons le résultat suivant :

```
net.ipv4.ip_forward=1
```

Ensuite pour activer le masquerade, nous avons besoin d'installer d'abord Iptables qui est installé d'origine sur le RPi, sinon nous l'installons.

Installation de iptables :

L'installation de iptables se fait pas la commande suivante :

```
$ sudo apt-get install iptables
```

Une fois le pare-feu installé nous vérifions que les tables sont vides avec la commande suivante :

```
$ sudo iptables -L
```

Nous devons voir le résultat suivant :

```
Chain INPUT (policy DROP)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     protopt source               destination

Chain OUTPUT (policy ACCEPT)
target     protopt source               destination
```

Maintenant, nous ajoutons une mascarade pour le trafic sortant sur eth0 comme suit

```
$ sudo iptables -t nat -A  POSTROUTING -o eth0 -j MASQUERADE
```

Après, nous enregistrons cette règle, en exécutant la commande suivante :

```
sudo sh -c "iptables-save> /etc/iptables.ipv4.nat"
```

Maintenant, pour installer cette règle au démarrage, nous allons modifier le fichier « /etc/rc.local », et nous allons ajouter la commande suivante juste au-dessus de « exit 0 » :

```
iptables-restore < /etc/iptables.ipv4.nat
```

2.3.Accès à distance au Raspberry Pi

2.3.1. Connexion à distance via SSH

Secure Shell est un protocole de réseau crypté pour initier des sessions shell textuelles sur des machines distantes de manière sécurisée. Cela permet à un utilisateur d'exécuter des commandes sur l'invite de commande d'une machine sans qu'il soit physiquement présent à proximité de la machine.

SSH a été créé en 1995 pour le principal but de permettre la prise de contrôle à

Distance d'une machine à travers une interface en lignes de commande.

Pour faire le contrôle à distance sur le Raspberry Pi on doit installer le SSH du coté serveur sur notre Raspberry pi et d'un autre client SSH sur notre ordinateur. Le serveur SSH est déjà installé par défaut et activé par défaut sur la Raspberry Pi, ce qui reste seulement d'installer un client SSH sur notre ordinateur ; pour cela on va installer le logiciel Putty sous Windows qui est un client

SSH. La figure présente la fenêtre de démarrage de Putty.

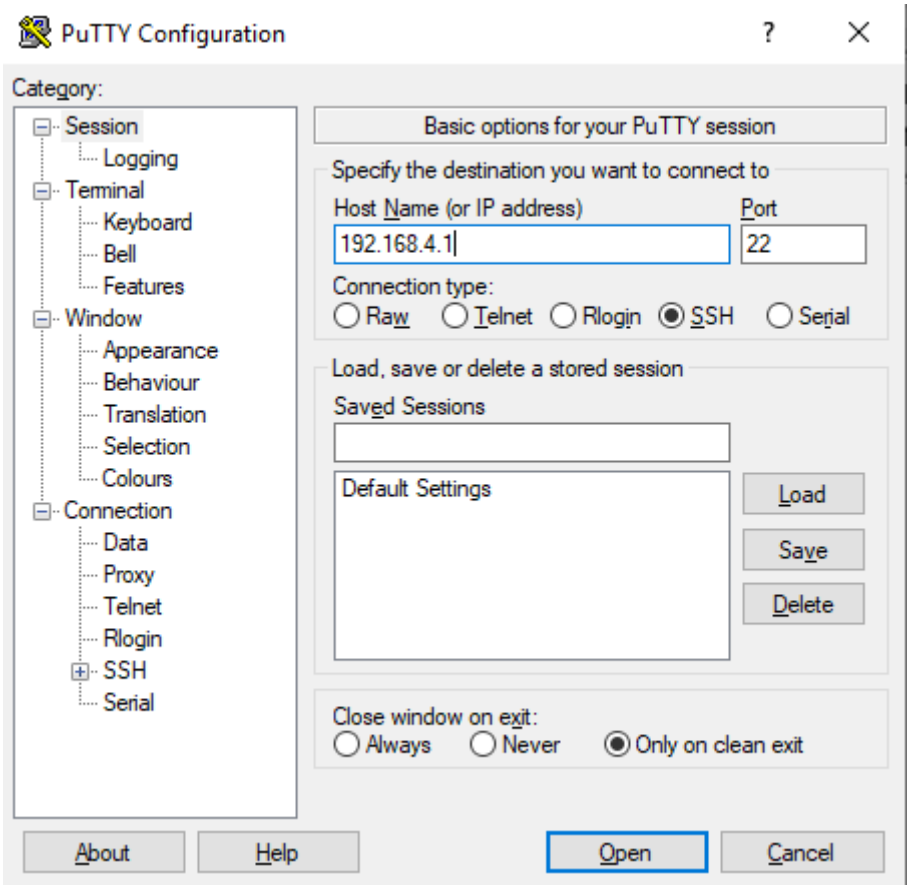


Figure IV-11 logiciel PuTTY

2.3.2. Connexion à distance via VNC

Virtual Network Computing est un système de visualisation et de contrôle d'un ordinateur distant. Il permet au logiciel client VNC de transmettre les informations de saisie du clavier et de la souris à l'ordinateur distant, possédant un logiciel serveur VNC à travers un réseau informatique. Il utilise le protocole RFB qui est un protocole simple pour l'accès à distance aux interfaces graphiques des utilisateurs. Comme pour le SSH, le contrôle à distance sur le Raspberry Pi nécessite une installation du VNC du côté d'un serveur sur notre Raspberry Pi et d'un autre d'un client VNC sur notre ordinateur, la distribution de Dexter Industries « Raspbian For Robots » contient déjà un serveur VNC, il ne reste seulement que d'installer un client VNC sur notre ordinateur, le logiciel qui va jouer le rôle d'un client VNC est VNC VIEWER. La connexion est très simple en entrant l'adresse IP, puis en cliquant sur « entrer »

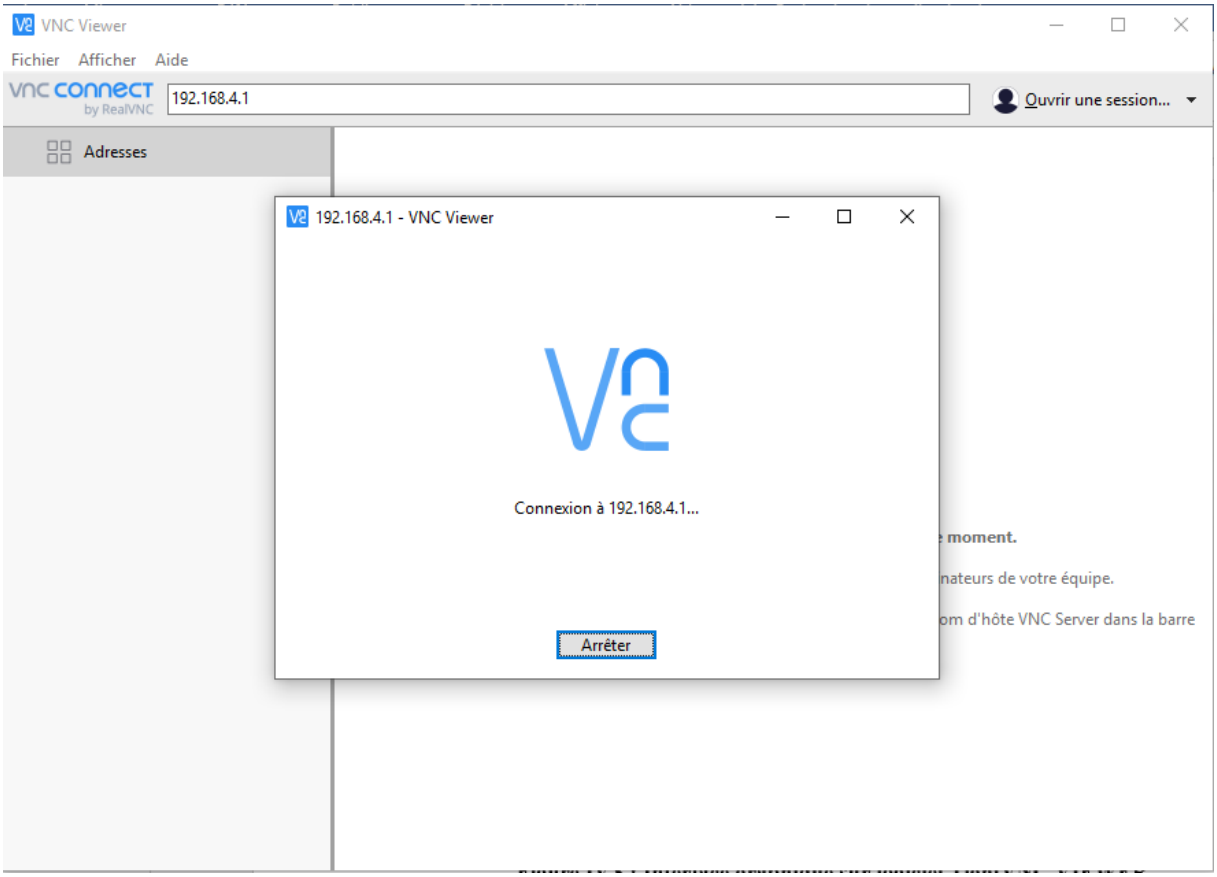


Figure IV-12 logiciel VNC viewer

Après l'authentification (en entrant notre mot de passe) le logiciel VNC VIEWER affiche l'interface graphique du système d'exploitation, donc à ce moment on a tous les droits et toute la liberté d'utiliser l'interface graphique de notre système d'exploitation sur notre ordinateur en bénéficiant du clavier, souris et écrans de notre propre ordinateur, comme si le Raspberry est branché à un écran par le câble HDMI et utilise un clavier et souris via les ports USB.

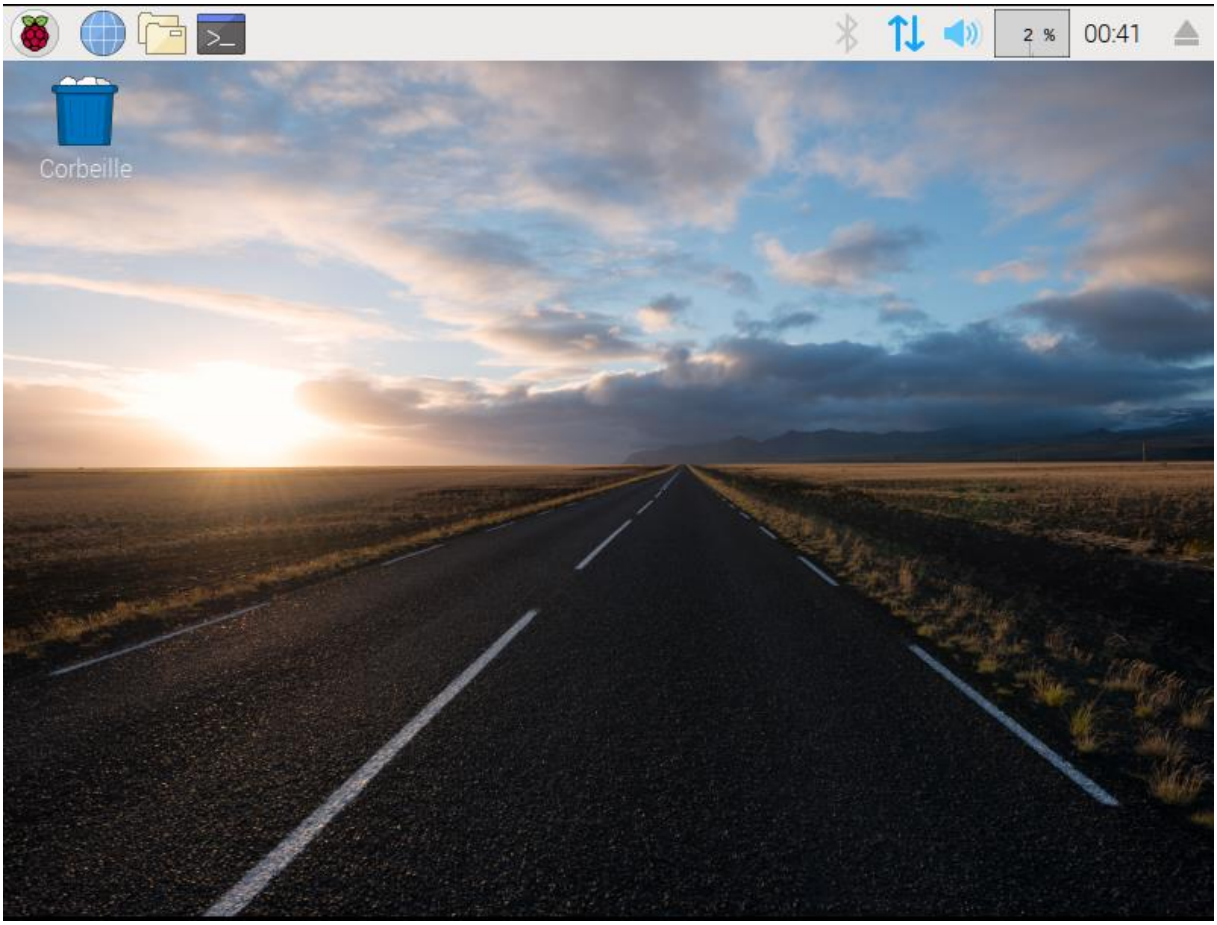


Figure IV-13 Interface graphique sur logiciel VNC VIEWER

2.4.Les outils de développement :

Pour développer notre système, nous avons eu recours à divers éléments notamment les outils logiciels, les langages de programmation qui nous ont permis la mise en œuvre de notre système.

2.4.1. nodeJS :

Node.js estune plateformelogicielle libre et événementielle en JavaScript orientée vers les applications réseau qui doivent pouvoir monter en charge.

Elle utilise la machine virtuelle V8 et implémente sous licence MIT les spécifications CommonJS.

Parmi les modules natifs de Node.js, on retrouve http qui permet le développement de serveur HTTP. Il est donc possible de se passer de serveurs web tels que Nginx ou Apache lors du déploiement de sites et d'applications web développés avec Node.js.

Concrètement, Node.js est un environnement bas niveau permettant l'exécution de JavaScript côté serveur.

Node.js est utilisé notamment comme plateforme de serveur Web.

2.4.2. Nginx :

est un logiciel libre de serveur Web (ou HTTP) ainsi qu'un proxy inverse écrit par Igor Sysoev, dont le développement a débuté en 2002 pour les besoins d'un site russe à très fort trafic (Rambler).

2.4.3. Reactjs :

est une bibliothèque JavaScript libre développée par Facebook depuis 2013. Le but principal de cette bibliothèque est de faciliter la création d'application web monopage, via la création de composants dépendant d'un état et générant une page (ou portion) HTML à chaque changement d'état.

2.5. Création de portail Captif :

pour pouvoir crée notre portail captif qui intercepte les paquet http et les rediriger vers une page publicitaire, nous avons besoin des outils suivant : iptables , nginx(reverse proxy), nodejs (serveur web), reactjs.

2.5.1. Installation de Nginx :

Pour installer Nginx, il suffit de saisir la commande suivante sur l'invite de commande :

```
$ Sudo apt-get install nginx
```

2.5.2. configuration nginx :

nous devons configurer le serveur nginx pour qu'il puisse intercepter les paquet http et leur envoyer une repone avec en-tete 302 avec un lien de la page denotre portail captif, la configuration est illustrée dans la figure suivante :

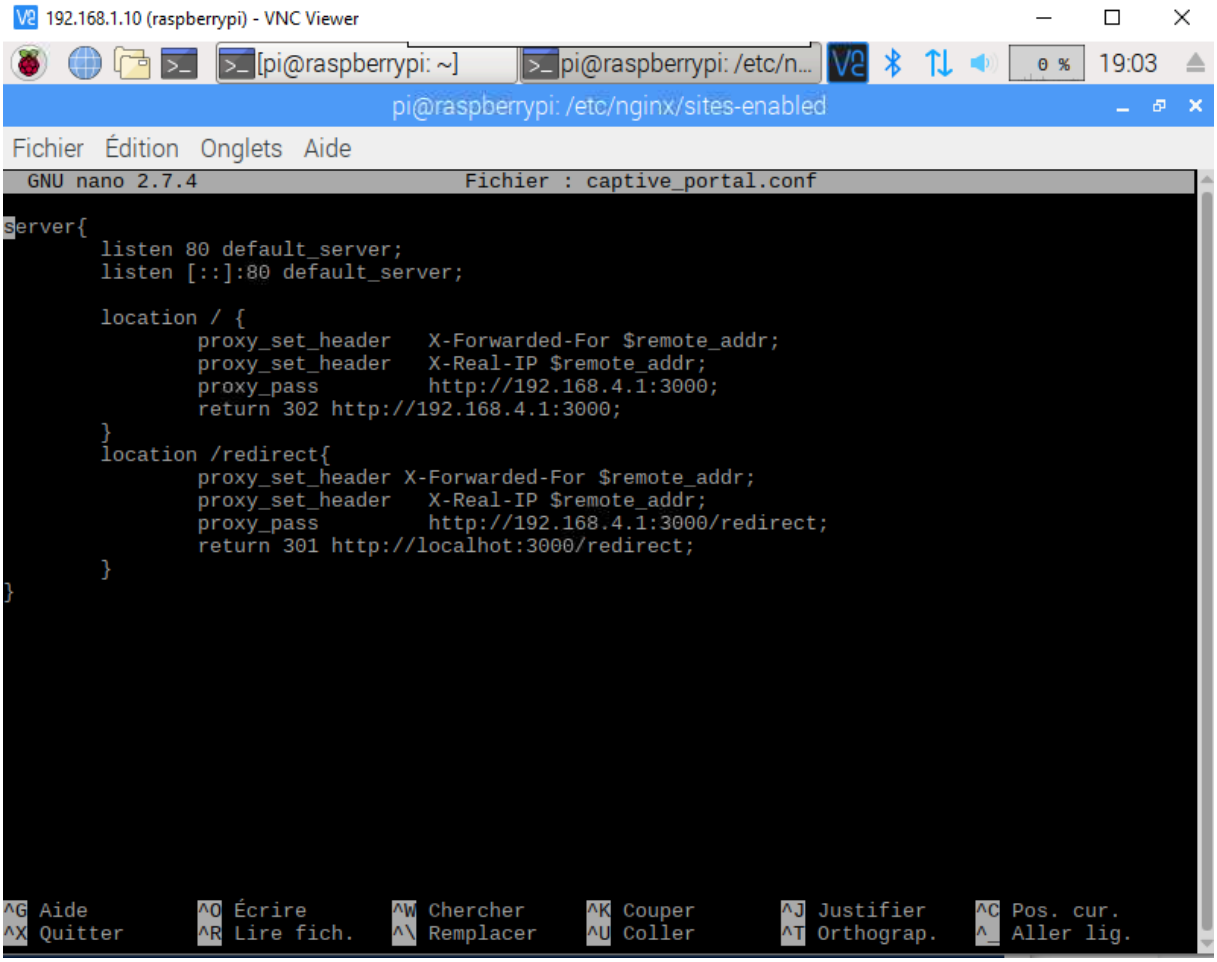


Figure IV-14 script de configuration de Nginx

2.5.3. configuration de iptable

la redirection des paquet vers le serveurs nginx se fait par le pare-feu iptable avec les regles suivante :

```
$ Iptables –t mangle –N internet
$ Iptables –t mangle –A internet –j MARK ––set –mark 99
$ Iptables –t mangle –A PREROUTING –i eth1 –p tcp –m  tcp ––d port 80 –j internet
$ Iptables –t nat –A PREROUTING –i eth1 –p tcp –m mark ––mark 99 –m tcp ––dport 80 –j DNAT ––to–destination 192.168.4.1
$ Iptables –t mangle –A PREROUTING –i eth 1 –p tcp –m tcp ––dport 443 –j internet
$ Iptables –t nat –A PREROUTING –i eth 1 –p tcp –m mark ––mark 99 –m tcp ––dport 443 –j DNAT ––to–destination 192.168.4.1
```

Installation de nodejs et npm

Nous avons exécuté la commande suivante :

```
$ Sudo apt-get install nodejs npm
```

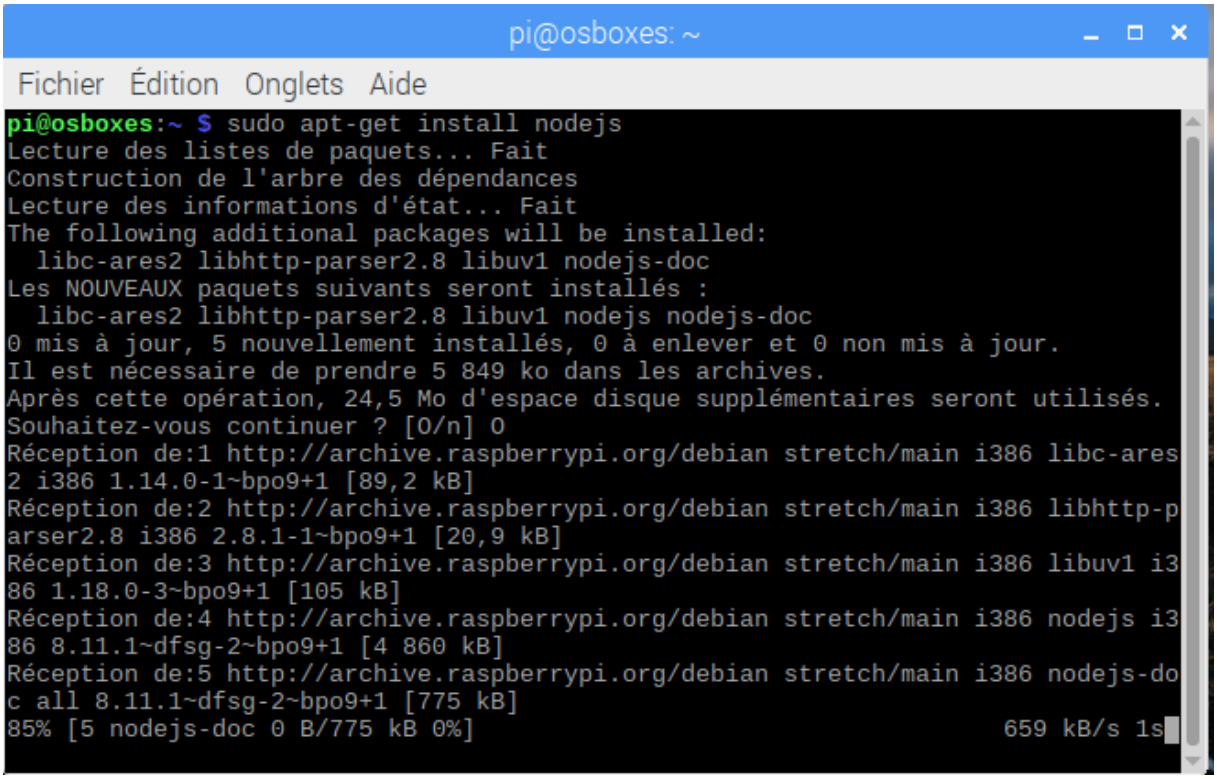


Figure IV-15 installation de nodejs et npm

Création de serveur web avec nodejs :

Nodejs possède plusieurs bibliothèques qui facilitent la création des serveurs et gérer les différentes requêtes, la bibliothèque la plus utilisée pour créer un serveur web sous nodejs est ExpressJS.

Création de projet

Il suffit de saisir les commandes suivantes dans le terminal pour crée un dossier vide

```
$ mkdir projet
$ cd projet
```

Ajout de la bibliothèque expressJS

```
$ npm install express
```

Programmation du serveur

Nous avons créé un serveur qui répond au client par la page de portail captif avec le code suivant :

```
JS portal.js
JS portal.js ▸ ...
1   const express = require('express');
2   const app = express();
3   const sh = require('shelljs');
4   const path = require('path');
5   var cors = require("cors");
6   var bodyParser = require('body-parser');
7
8
9   // Serve any static files built by React
10  app.use(express.static(path.join(__dirname, "dist")));
11  app.use(cors());
12  app.use(bodyParser.urlencoded({ extended: false }));
13  app.use(bodyParser.json());
14  app.set('trust proxy', true);
15
16  app.get("/", function(req, res) {
17    console.log('client request');
18    var ip = req.headers['x-forwarded-for'] || req.connection.remoteAddress;
19    console.log(ip);
20    res.sendFile(path.join(__dirname, "dist", "index.html"));
21
22  });
23
24
25  app.get("/redirect", function(req, res) {
26     console.log('client request');
27     var ip = req.headers['x-forwarded-for'] || req.connection.remoteAddress;
28     var rawIp = ip.slice(7);
29     var command =`sudo arp -a ${rawIp}`;
30     console.log(command);
31     const outp = sh.exec(command);
32     console.log('command successful');
33     var mac = outp.slice(43,60);
```

```
32     console.log('command successful');
33     var mac = outp.slice(43,60);
34     console.log(mac);
35     var rmCommand = `sudo iptables -I internet -t mangle -m mac --mac-source ${mac} -j RETURN`;
36     console.log(rmCommand);
37     if (sh.exec(rmCommand).code == 0){
38        console.log('command success');
39     }
40
41   });
42
43   app.listen(3000, console.log('App Listening to port 3000'));
```

Figure IV-16 programme de serveur du portail captif

Ce code permet de créer un serveur web qui écoute les requêtes sur le port 3000 et répond par une page web, celle du portail captif.

Ensuit nous avons programmé un autre serveur, qui écoute sur un autre port, afin de répondre aux requêtes de l'administrateur et exécuter ces commandes. Nous montrons le code dans les figures suivantes :

app.js ▸ app.post('/createDir') callback ▸ fs.readFile() callback

```
1   const express = require("express");
2   const path = require("path");
3   const fs = require('fs');
4   const sh = require('shelljs');
5   const DATA_FILE = path.join(__dirname, '/data.json');
6   const RULE_FILE = path.join(__dirname, '/rules.json');
7   var cors = require("cors");
8   var bodyParser = require('body-parser');
9
10
11
12  const app = express();
13  const port = process.env.PORT || 3000;
14
15  // Serve any static files built by React
16  app.use(express.static(path.join(__dirname, "dist")));
17  app.use(cors());
18  app.use(bodyParser.urlencoded({ extended: false }));
19  app.use(bodyParser.json());
20
21  app.get("/", function(req, res) {
22    res.sendFile(path.join(__dirname, "dist", "index.html"));
23  });
24
25  app.get('/validation', (req, res) => {
26    fs.readFile(DATA_FILE, (err, data) => {
27      res.setHeader('Cache-Control', 'no-cache');
28      res.json(JSON.parse(data));
29    });
30  });
31
32  app.get('/getRules', (req, res) => {
33    fs.readFile(RULE_FILE, (err, data) => {
34      const rules = JSON.parse(data);
35      netRules = rules.slice(1);
36      res.setHeader('Cache-Control', 'no-cache');
37      res.json(netRules);
38    });
39  });
40
```

```
app.post('/createDir', (req , res) => {
  const ch = req.body.ch;
  const ip = req.body.ip;
  const ipd = req.body.ipd;
  const port = req.body.port;
  const dport = req.body.dport;
  const protocol1 = req.body.protocol;

  var sip = (ip != '') ? ` -s ${ip}` : ``;
  var dip = (ipd != '') ? ` -d ${ipd}` : ``;
  var ports = (port != '') ? ` -sport ${port}` : ``;
  var portd = (dport != '') ? ` -dport ${dport}` : ``;
  var proto = (protocol1 != '') ? ` -p ${protocol1}` : ``;

  var command = `sudo iptables -A ${ch}${sip}${dip}${ports}${portd}${proto} -j DROP`;
  console.log(command);
  fs.readFile(RULE_FILE, (err, data) => {
    const rules = JSON.parse(data);
    rules[0].number = rules[0].number + 1;
    console.log(rules[0].number)
    const newRule = {
      number : rules[0].number,
      chaine : ch,
      ips : ip,
      ipd : ipd,
      sport: port,
      dport : dport,
      protocol : protocol1,
    };
    rules.push(newRule);

      netRules = rules.slice(1);
    fs.writeFile(RULE_FILE, JSON.stringify(rules, null, 4), () => {
      res.setHeader('Cache-Control', 'no-cache');
      res.json(netRules);
    });
  });


```

```
app.delete('/delete',(req,res) => {
  const num = req.body.rNum;
  console.log('rule '+num);
  fs.readFile(RULE_FILE,(err,data) => {
    const rules = JSON.parse(data);
    rules.splice(num,1);
    rules[0].number = rules[0].number - 1;
    for (var i = num; i< rules.length;i++){
      rules[i].number = rules[i].number - 1;
    }
    netRules = rules.slice(1);
    fs.writeFile(RULE_FILE,JSON.stringify(rules,null,4),() => {
      res.setHeader('Cache-Control','no-cache');
      res.json(netRules);
    });
  });
});

app.listen(port, () => console.log(`Listening on port ${port}`));
```

Figure IV-17 script de serveur de configuration netfilter

3. Présentation de l'application (notre système)

Page d'authentification de l'administrateur :

Pour que l'administrateur puisse ajouter des règles au pare-feu netfilter, il doit accéder a l'interface graphique, pour cela, il faut s'assurer que seule l'admin qui y vas accéder,

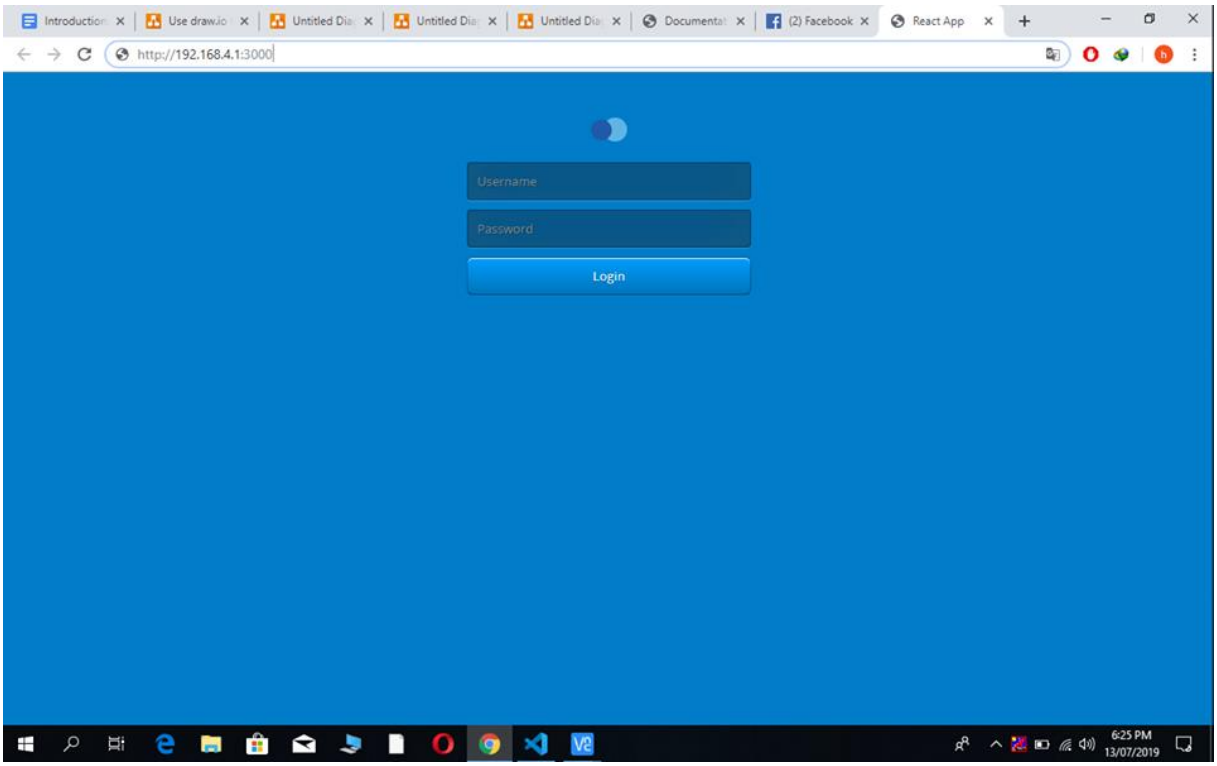


Figure IV-18 page d'authentification de l'administrateur

Page de configuration du pare-feu :

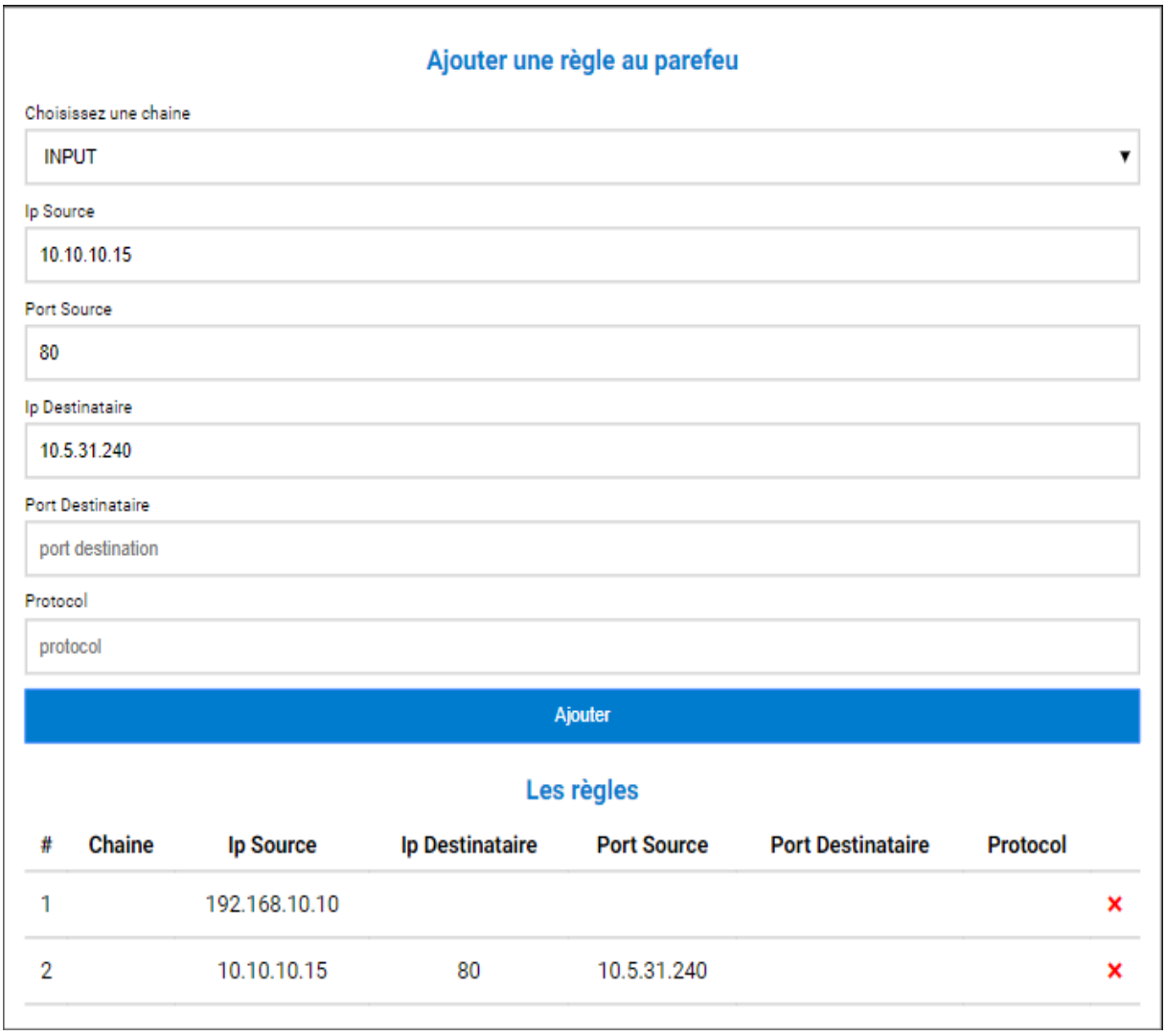


Figure IV-19 page de configuration de netfilter

Page du portail captif

Figure IV-20 page de portail captif

Conclusion

Au cours de ce de dernier chapitre, nous avons décrit les étapes d'installation, de configuration et les outils de programmation nécessaires à là de réalisation de notre système. En premier lieu, nous avons présenté le matériel utilisé-et ses caractéristiques ensuite, nous avons présenté tous ce qui est software et les étapes nécessaires, et les scripts des configurations, et à la fin nous avons présenté quelques tests de notre application ainsi que les résultats obtenus illustrés par quelques figures.

On peut conclure que le système mise en place répond bien aux objectifs

Conclusion Général

L’objectif étant de créer un pare-feu basé sur Netfilter jumelé à un portail captif, embarqués dans un Raspberry PI. Ce système peut servir à plusieurs fonctions. A titre d'exemple, nous l'avons configuré pour afficher un spot publicitaire quand un client se connecte au réseau d'une cafeteria. L'objectif est atteint, l’échafaudage de notre travail est tel qu’on peut désormais proposer cette solution à tout établissement publique intégrant une connexion wifi : restaurant, coffee-shop, aéroport et même espace publique ouvert. Notre projet nous a permis de mettre en œuvre les connaissances élémentaires et techniques acquises tout au long de notre cursus, de les approfondir et d’en apprendre davantage. Cette continuité d’apprentissage nous offre des perspectives nouvelles afin d’améliorer notre produit et le perfectionner pour atteindre des buts de plus en plus innovants tels que : ajouter une possibilité à l’administrateur de modifier la page du portail captif, lui ajouter une autre possibilité qui lui permet de voir les logs des connexions, et tant d’autres.

Bibliographie

[1] https://fr.wikipedia.org/wiki/Pare-feu_(informatique)

[2] https://www.memoireonline.com/05/12/5830/m_Les-systemes-embarques4.html

[3] https://www.fafiec.fr/images/contenu/menuhaut/observatoire/etudes/2013/systemes embarqu%C3%A9s/SE-Ecosysteme_des_Systemes_Embarques_-_20140606.pdf

[4] https://www.ukonline.be/cours/embeddedsystems/programming/chapitre1-1

[5] https://fr.wikipedia.org/wiki/Syst%C3%A8me_embarqu%C3%A9#Contraintes

[6] **Book Android Application Development for the Intel Platform, by Tao Wang, Ryan Cohen , *September 2014***

[7] Introduction à la notion de système d'exploitation , www.Introduction à la notion de système d'exploitation.pdf

[8] https://fr.wikipedia.org/wiki/Syst%C3%A8me_d%27exploitation_embarqu%C3%A9

[9] Systèmes d'exploitation embarqués - ELECINF344 - rfc1149.net

[10] Système embarqué : Aspect software - UKO , https://www.ukonline.be/cours/embeddedsystems/programming/chapitre1-3

[11] https://fr.wikipedia.org/wiki/Pare-feu_(informatique)#Cat%C3%A9gories_de_pare-feu

[12] https://fr.wikipedia.org/wiki/Portail_captif

[13] https://www.linksys.com/us/r/resource-center/captive-portal/

[14] https://ostec.blog/en/perimeter/captive-portal-what-is-it

[15] https://en.wikipedia.org/wiki/WiFiDog_Captive_Portal

[16] http://www.chillispot.org/

[17] https://openwrt.org/docs/guide-user/services/captive-portal/wireless.hotspot.nodogsplash

[18] http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-fr-4/index.html

[19] **cours Mr daoui chapitre 7 Netfilter**

[20] https://www.malekal.com/tutoriel-iptables/#Les_tables

[21] https://inetdoc.net/guides/iptables-tutorial/traversingoftables.html