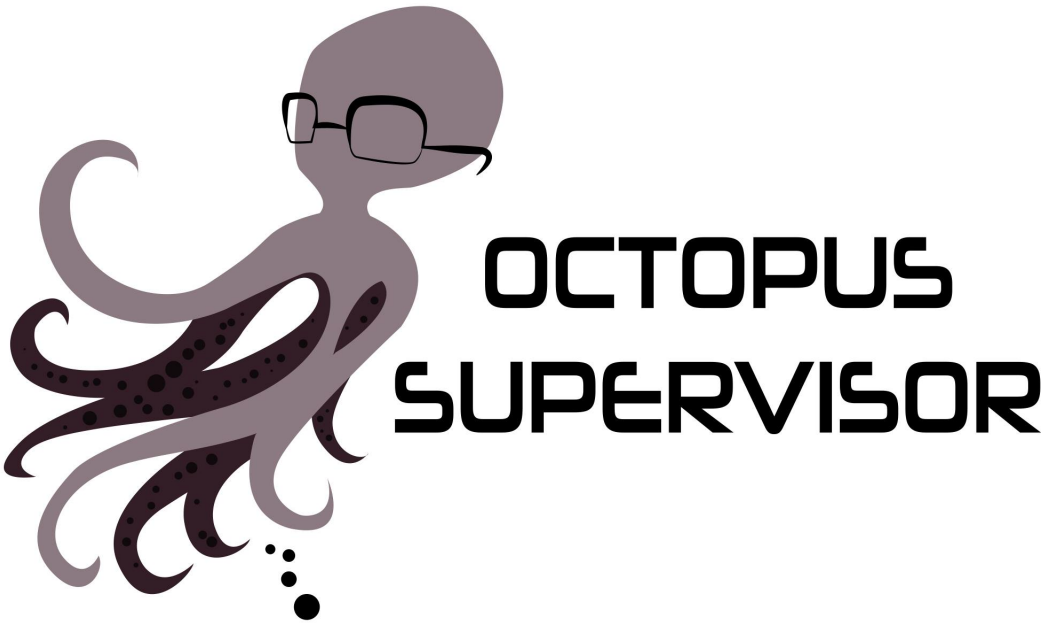
INSA Centre Val de Loire
Département Sécurité et Technologies Informatiques
Année 2014-2015

Responsables projet : Jérémy Briffaut et Martial Szpieg



Octopus Supervisor

Projet 4A - Supervision et audit de la sécurité système dans un réseau

Nicolas Cosnard et Cyril Segretain

Bourges, le 12 février 2015

Remerciements

Nous souhaitons remercier M. Jérémy Briffaut et M. Martial Szpieg pour leur aide lors du lancement de notre projet. Ils ont su répondre à nos questions et nous aider à nous orienter vers une solution réalisable.

Merci aussi à Charleen Plum, une artiste de talent qui a su prendre du temps pour nous proposer un superbe logo correspondant à nos attentes.
https ://www.behance.net/CharleenPlum

Nos remerciements vont également au Man de Linux pour son aide précieuse et sa disponibilité à toute heure. Les barbus ayant développés des plugins pour l'éditeur Vim nous ont été d'une grande aide dans ce projet qui a été quasi-intégralement développé sous Vim.

Sommaire

Remerciements **2**

Sommaire **3**

Introduction **5**

1 Fonctionnement **6**
1.1 Objectif 6
1.2 Algorithme 6

2 OctoBrain **9**
2.1 Algorithme 9
2.2 Maestro 10
2.3 WebServ 10
2.4 DbManager, Registerer et Commander 11

3 OctoTentacle **12**
3.1 Algorithme 12
3.2 Register 13
3.3 CommandWorker 13

4 Approfondissement technique sur la sécurité **14**
4.1 SSL 14
4.2 Handshake 14
4.3 Timeout web 16

5 Bilan **17**
5.1 Difficultés rencontrées 17
5.2 Axes d'améliorations 17
5.3 Bilan technique 18
5.4 Bilan humain 18

Conclusion **19**

A Installation des Machines Virtuelles **21**
A.1 Installation d'une Debian Minimale 21
A.2 Installation d'un Xubuntu 24

B Travailler avec Git **27**
B.1 Contexte 27
B.2 Les bases 27

C Installation d'Octopus Supervisor **28**

Table des figures **29**

Introduction

Dans le cadre de la deuxième année d'étude du cycle ingénieur dans la formation Sécurité et Technologies Informatique de l'INSA Centre Val de Loire, nous travaillons sur un projet en binome. Le sujet est "Supervision et audit de la sécurité système dans un réseau". Les professeurs tuteurs sur ce projet sont M. Briffaut et M. Szpieg.

L'objectif est donc de réaliser un logiciel maitre/esclave avec des fonctionnalités de supervision des esclaves et de gestion à distance de ceux-ci. La supervision et la gestion attendue concernent la sécurité des systèmes esclaves afin de pouvoir certifier la fiabilité des clients, la confidentialité des données, la tracabilité des communications et un système informatique sain pour l'entreprise.

Le cahier des charges demande donc un outil d'administration qui sera utilisé par des administrateurs réseaux ou des responsables sécurité d'entreprise. C'est un outil qui sera facilement déployable sur un parc informatique important et diversifié. Les fonctionnalités d'administration et de supervision des machines clientes seront avancées et pourront se faire de manière pratique via une interface Web.

Ce rapport vous présente de manière vulgarisée le fonctionnement de notre logiciel. Nous avons choisi de présenter le fonctionnement général du programme dans un premier temps. Ensuite, nous expliciterons les algorithmes de fonctionnement du serveur puis du client. Et enfin, nous présentons un peu plus en détail les points de sécurité de notre logiciel qui sont importants.

Nous proposons donc une solution légère mais offrant de nombreuses fonctionnalités permettant la supervision et l'audit de la sécurité des systèmes dans un réseau. Nous vous proposons une solution fiable, sécurisée et facile d'accès. Vous pourrez la découvrir dans ce rapport.

1 Fonctionnement

Ce projet est issu d'un besoin de supervision et d'audit sécurité système dans le réseau d'une entreprise d'aujourd'hui.

1.1 Objectif

Octopus a pour objectif d'être un outil d'administration réseau répondant à un besoin de gestion d'un parc informatique par un administrateur réseau.

Le nom du projet vient de la pieuvre qui a une tête qui contient le cerveau et qui dirige le reste de son corps, et les nombreuses tentacules de celle-ci qui ne sont que l'éxécution des commandes du cerveau (« brain ») et fait remonter des informations à ce « brain ». L'idée est donc d'avoir un point de contrôle central pour l'administrateur qui est le « brain » et ensuite chaque poste du parc informatique correspond à une tentacule (« tentacle »).

L'outil d'administration que nous proposons permet de connaître en temps réel le statut de connexion au réseau des postes distants et de pouvoir lancer des scripts sur ces machines distantes. Les résultats des scripts sont disponibles sur le poste d'administration. Nous pouvons alors connaître à un moment donné l'état d'utilisation du disque dur d'un poste, l'état des processus lancé sur un autre poste, ... La gestion du serveur est faite à travers une interface Web accessible avec des identifiants pour l'administrateur. Depuis cette interface, nous pouvons gérer le parc, ajouter de nouvelles machines, voir l'état du parc, commander l'éxécution de scripts sur les hôtes distants.

Le serveur est multi-client et permet donc la gestion d'un parc informatique important. L'administrateur peut créer des scripts afin de ne gérer que les informations dont il juge importante pour son parc. Un échantillon de script de base est installé par défaut dans le package client.

1.2 Algorithme

Dans un premier temps, nous avons travaillé sur un algorithme de notre solution afin de montrer le fonctionnement et les points-clés de celle-ci.

Les fonctionnalités maitre/esclave que présente Octopus Supervisor sont :

— Ajout de nouveaux esclaves
— Gestion des esclaves
— Envoi de scripts aux esclaves
— Réception et stockage de résultats de scripts sur le serveur

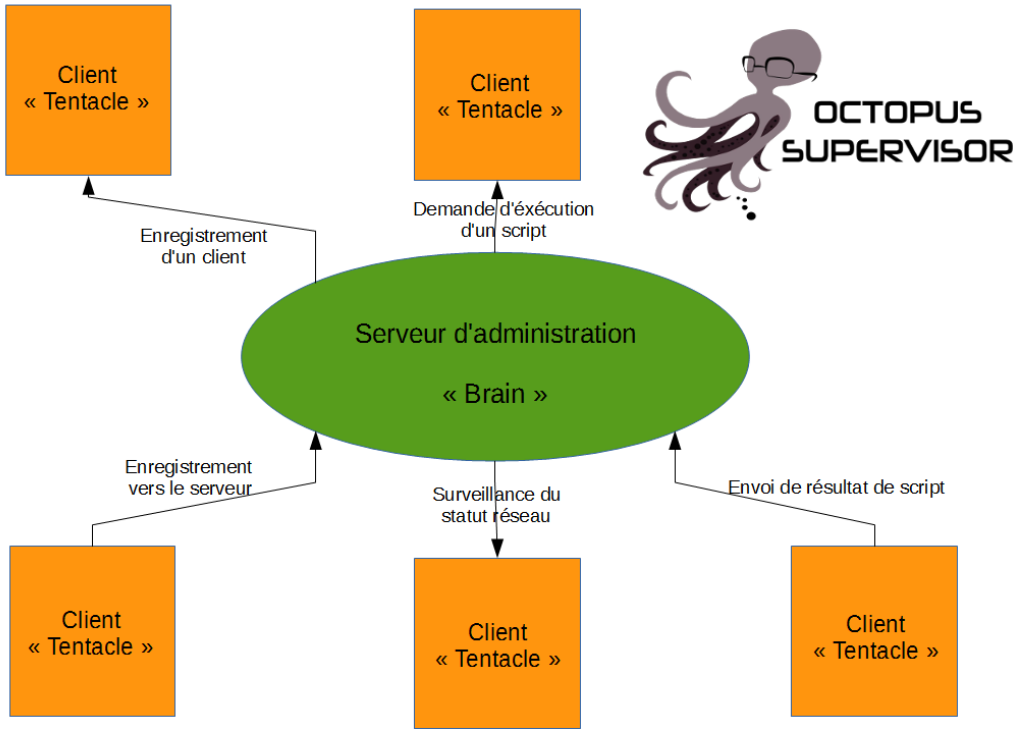


FIGURE 1 – Algorithme général de fonctionnement de Octopus Supervisor

Ci-dessous, un algorithme explicite techniquement le fonctionnement général de Octopus Supervisor.

Les deux entités maitre et esclave communiquent sur le réseau grâce à leur adresse IP et un port défini suivant le type de communication. Lors de l'enregistrement d'un client, le serveur est réceptif sur le port 6000. Si le client n'a pas de serveur spécifié lors de son lancement, il va attendre une communication sur le port 7000 de la part d'un serveur.

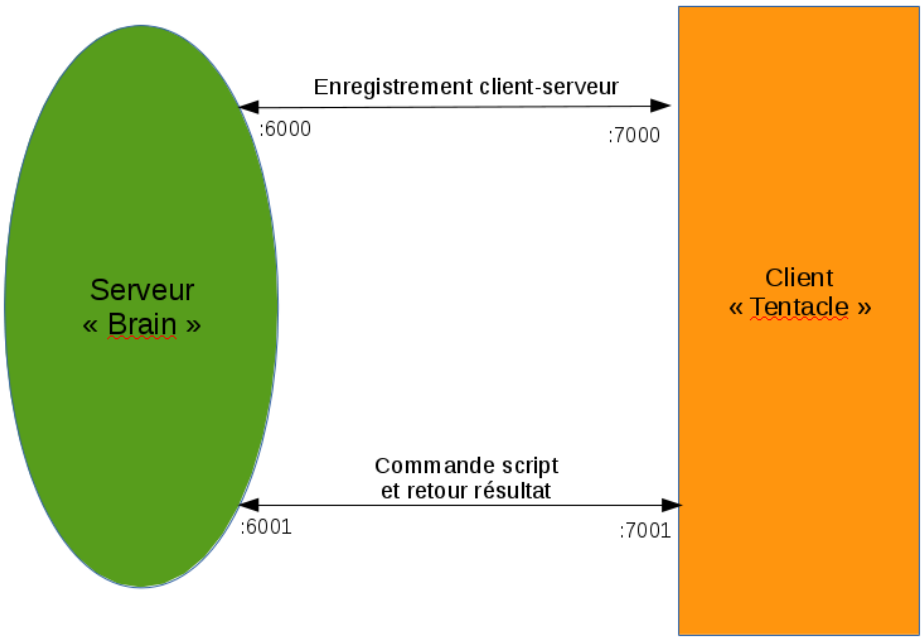


FIGURE 2 – Fonctionnement

2 OctoBrain

La partie de notre projet appelé OctoBrain est en fait la partie serveur d'administration. C'est le cerveau du logiciel qui contrôle les hôtes et permet la supervision et l'audit du parc.

2.1 Algorithme

Le maestro est le chef d'orcheste de notre logiciel, il coordonne le serveur et ses différents threads. Nous avons choisi de diviser les actions du serveurs en plusieurs threads :

- le serveur Web qui interagit avec l'administrateur via une interface Web. Le serveur est entièrement codé en C, il permet d'afficher des pages HTML et du Javascript.
- un registerer, qui signifie « enregistreur ». Celui-ci permet d'écouter sur le port 6000 si un client souhaite s'enregistrer sur le serveur, et il permet aussi d'ajouter manuellement des clients grâce à leur adresse IP.
- un commander qui permet la gestion des commandes sur le serveur. Il peut envoyer des commandes aux clients et recevoir les résultats.

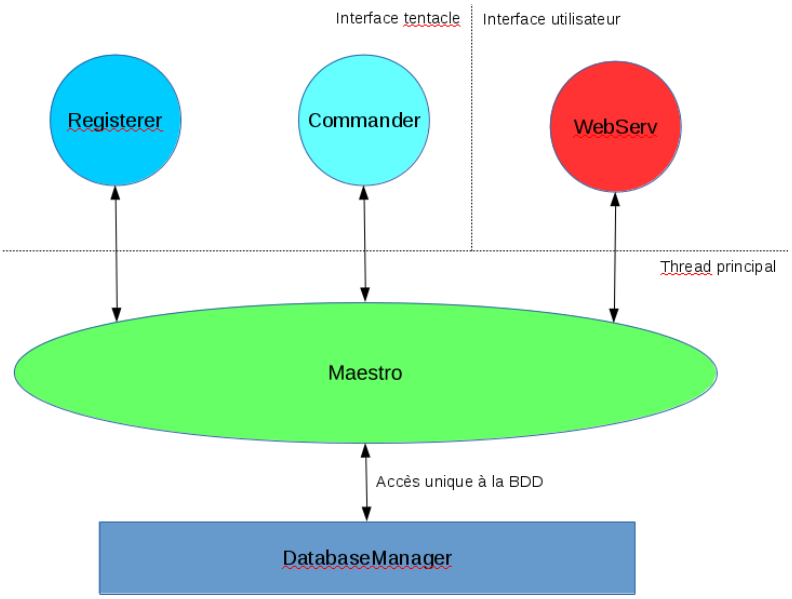


FIGURE 3 – Fonctionnement technique du Brain

L'architecture que nous avons développé pour le Brain reprend une architecture MVC (Modèle-Vue-Contrôleur). Ce type de fonctionnement permet de bien organiser son code source et de séparer facilement les rôles entre les processus du serveur. Le côté Modèle se retrouve dans le dbManager, la partie Vue est représenté par les différentes interfaces et le Contrôleur correspond au Maestro.

2.2 Maestro

Le logiciel coté serveur est orchestré autour d'un processus appelé « Maestro ». Son rôle est de coordonner les différents processus que peut lancer le serveur. Ces processus sont des forks permettant de communiquer de manière indépendante avec les clients. Le maestro joue un rôle indispensable dans notre logiciel car il est l'interface de communication avec la base de données du serveur. C'est le seul accès permettant de lire et écrire en base de données. En effet, généralement, les bases de données peuvent accepter les accès multiples en lecture mais un accès unique en écriture. Nous avons donc créé ce maestro afin de coordonner tous les liens avec la base de données. Le maestro est en écoute constante sur des pipes reliés aux différentes parties du serveur. Le contrôle d'accès peut se faire en un point unique grâce à ce Maestro.

2.3 WebServ

Le WebServ de Octopus permet l'administration du serveur à distance par l'administrateur. Ce serveur web permet de voir et d'ajouter les clients, d'envoyer des instructions de scripts aux clients et aussi d'afficher les résultats des scripts éxécutés. Le WebServ est l'interface de manipulation des données contenues en base de données.

Le serveur Web est un daemon écrit en C qui permet de lire et d'envoyer des fichiers (HTML, CSS, JavaScript, images) et du texte JSON. Le site est formé d'une unique page HTML, dont le contenu est modifié dynamiquement grâce à du javascript et des appels AJAX. Les interactions avec des boutons et liens déclenchent des requêtes GET avec différents arguments, et le serveur Web retourne du JSON avec les données demandées. Cela permet un contenu dynamique, tout en minimisant les échanges réseau.

2.4 DbManager, Registerer et Commander

Le DatabaseManager est le gestionnaire de fonctions de lecture et d'écriture dans la base de données. Le logiciel utilise Sqlite3 comme base de données. L'avantage est la facilité d'utilisation de Sqlite3 et il ne demande que peu de ressources matérielles. Afin d'accéder en lecture et écriture sur les différentes tables de la base de données, nous avons recréé des fonctions spécifiques.

Le Registerer est le daemon d'enregistrement des clients. Il est composé de deux fonctions principales. Une première qui est un socket en écoute permanente sur le port 6000 dans le cas ou un Tentacle rejoindrait le réseau et aurait l'adresse IP du Brain. Lorsqu'un nouveau Tentacle tente de contacter le Brain sur le port 6000, celui-ci lance l'enregistrement du Tentacle. La seconde fonction est appelé lorsque l'administrateur du réseau ajoute manuellement une nouvelle machine sur le réseau. L'ajout de la nouvelle machine se fait en fournissant l'adresse IP, ensuite le Brain prend en charge de contacter le client sur le port 7000 pour avoir les informations complémentaires permettant l'enregistrement.

Le Commander est le troisième fork qui est en fonctionnement sur le serveur. Celui-ci permet d'envoyer des commandes aux clients et de recevoir les résultats des scripts. Ce service se lance lorsque l'administrateur demande à lancer un script sur un Tentacle à travers le WebServ ou quand un Tentacle a terminé l'éxécution d'un script et veut envoyer le résultat de celui-ci au serveur. Le Commander peut donc ouvrir une socket vers le client sur le port 7001 pour l'envoi d'une commande, il se déconnecte ensuite. Le Commander est également en écoute sur le port 6001 du Brain et quand un Tentacle se connecte pour envoyer des données, il sauvegarde le rapport en base de données.

3 OctoTentacle

La partie OctoTentacle de notre logiciel est le côté client. C'est un module qui est installé sur un hôte afin de pouvoir être superviser et auditer par le serveur.

3.1 Algorithme

Le client est divisé en deux parties principales : une partie pour l'enregistrement du client auprès du serveur et une partie qui permet d'interagir avec le serveur pour l'éxécution des scripts et l'envoi des rapports.

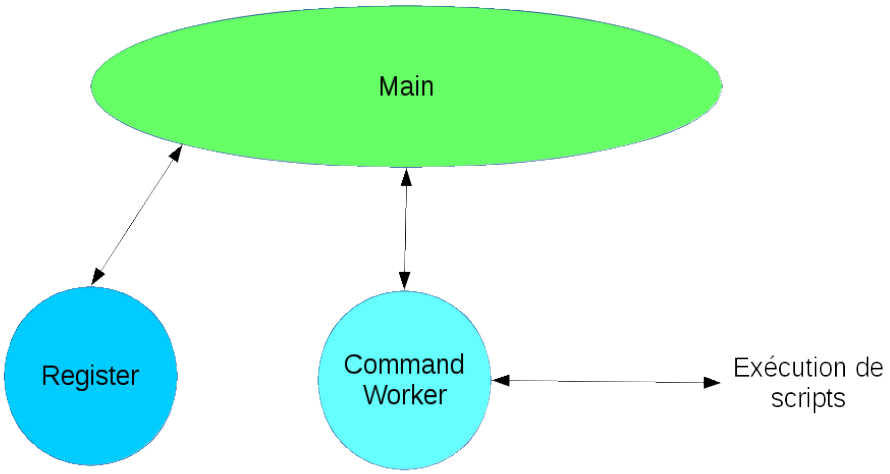


FIGURE 4 – Fonctionnement technique du Tentacle

3.2 Register

La partie enregistrement du client possède deux fonctions : une première fonction qui est un daemon et attend qu'un serveur d'administration le contacte pour proposer un enregistrement sur le réseau, une seconde fonction qui est à l'initiative de l'enregistrement auprès du brain en connaissant son adresse IP. Le daemon d'enregistrement écoute sur le port 7000. Lorsqu'un serveur souhaite enregistrer le client sur son réseau, il y a un échange d'informations. Le client donne son hostname, le serveur lui fournit un ID et une clé générée par le brain et qui permettra de sécuriser l'authentification entre le client et le serveur. Une fois l'enregistrement terminé, le thread se termine et un nouveau thread se lance, le daemon de scripts. Si lors de son lancement le client a en paramètre l'adresse IP d'un serveur avec l'option -b, le client va alors contacter directement le serveur pour demander à être enregistrer. L'enregistrement se fait ensuite de la même manière.

3.3 CommandWorker

Le CommandWorker est un processus du Tentacle qui permet la réception des demandes d'éxécution de script de la part du Brain, l'éxécution de ces commandes et l'envoi des résultats vers le Brain.

Dans un premier temps, le Tentacle se place en écoute sur le port 7001. Quand le Brain auquel il est affilié se connecte sur ce socket, il y a authentification des parties et le Brain demande au Tentacle d'éxécuter un script qu'il possède. La connexion se ferme ensuite. Le Tentacle éxécute alors le script demandé en local et sauvegarde le retour du script dans un fichier en local. Une fois le fichier sauvegardé en local, le Tentacle va initier une connexion avec le Brain sur le port 6001 en demandant la sauvegarde du résultat du script en base de données. Le contenu du fichier est donc envoyé au Brain et le résultat sera ensuite accessible pour l'administrateur via l'interface Web.

4 Approfondissement technique sur la sécurité

Cette section du rapport rassemble des points plus technique concernant Octopus Supervisor. Elle met en avant les points d'honneur fait sur la sécurité du logiciel.

4.1 SSL

Dès le lancement du projet, nous avions décidé que le client et le serveur échangerait des données chiffrées par SSL. Toutes les communications à travers des sockets utilisent le protocole SSL (Secure Socket Layer). Nous avons donc auto-signé un certificat pour le serveur. Une authentification SSL est effectuée à chaque ouverture de socket entre les extrémités serveur et client.

Les échanges du serveur Web se font aussi avec le protocole SSL, nous avons donc une sécurisation des commandes de l'administrateur depuis l'interface Web. Ceci nous permet d'assurer la confidentialité des données de bout en bout lors des communication entre des entités.

4.2 Handshake

Octopus Supervisor travaille avec un serveur et plusieurs clients. Afin de savoir avec qui l'on communique, il y a une phase d'authentification au début d'un échange entre deux entités. Cette authentification est appelée "handshake", signifiant poignée de main. Elle commence par un message initiant la communication puis un acquittement. Ensuite il y a un échange d'ID pour identifier le Tentacle. La dernière étape du Handshake est de vérifier la clé correspondant à l'ID qui a été générée lors de l'enregistrement du Tentacle sur le Brain.

La Figure 5 représente l'initialisation de la communication entre le Brain et un Tentacle lors de l'enregistrement du dernier sur le Brain. Quand le Brain reçoit une adresse IP à ajouter à sa base de données, la communication suivante est effectuée.

La Figure 6 présente le handshake qui est fait lorsqu'un Tentacle contacte le Brain pour envoyer un résultat de script.

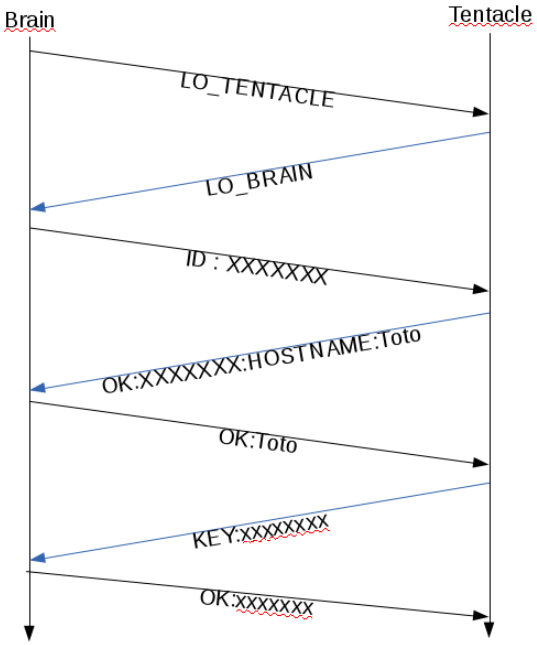


FIGURE 5 – Handshake enregistrement

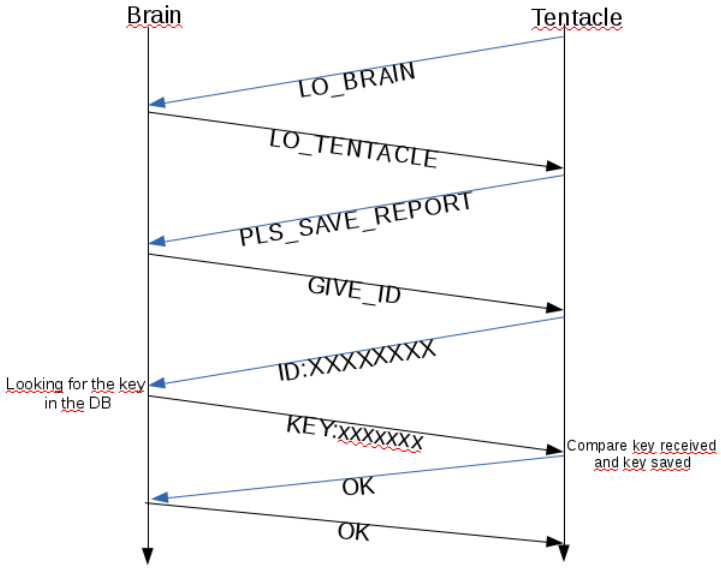


FIGURE 6 – Handshake envoi résultat

4.3 Timeout web

L'utilisation de Octopus Supervisor se fait grâce à une interface Web. Celle-ci nécessite une connexion avec un login et un mot de passe pour pouvoir identifier les administrateurs du logiciel. Lorsqu'un utilisateur du logiciel se connecte à l'interface Web, une clé de session est créée et un timer est lancé. Si au bout de 300 secondes, l'utilisateur n'a pas interagi avec le WebServ, la clé de session ne sera plus valide et l'utilisateur sera automatiquement déconnecté. Cela permet d'éviter une connexion permanente qui expose aux risques d'usurpation d'identité par session d'utilisateur.

5 Bilan

5.1 Difficultés rencontrées

Au cours de ce projet, nous avons rencontrées plusieurs difficultés, certaines humaines et d'autres techniques.

D'une part, la gestion de projet a été faite de manière approximative car nous n'étions que deux sur le projet. Cela a été un avantage mais aussi un inconvénient qui était de ne pas avoir de suivi régulier sur les tâches à effectuer.

D'autre part, sur le point de vue technique, nous nous sommes heurtés à des problèmes d'interprétation de commande SSL. Lorsqu'un message commençait par "R", OpenSSL considérait que c'était une commande qui lui est propre. Nous avons donc modifié nos messages.

L'envoi des résultats de scripts a été compliqué à mettre en place puisque nous avons souhaité mettre le contenu du fichier dans un buffer de taille dynamique pour l'envoi, la réception et la sauvegarde en BDD. La gestion des buffers a été complexe mais fonctionne correctement aujourd'hui.

5.2 Axes d'améliorations

Octopus Supervisor est un logiciel fonctionnel mais auquel nous pourrions rajouter beaucoup de fonctionnalités. Parmi celles qui seraient intéressantes, nous avons retenus :

- Gestion des utilisateurs depuis l'interface Web
- Utilisation de fichier de configuration pour le serveur et les clients
- Gestion d'un "KeepAlive" sur les Tentacles grâce à un ping Brain vers Tentacle
- Modification et suppression des scripts via l'interface Web
- Optimisation du WebServ et sécurisation avancée de celui-ci

Toutes ces fonctionnalités seraient une vraie valeur ajoutée pour notre logiciel. Elles sont dans l'idées d'approfondir la supervision des clients et de faciliter la gestion pour l'administrateur.

5.3 Bilan technique

Sur le plan des compétences technique, nous avons beaucoup appris durant ce projet. Nous avons approfondi nos connaissances dans les langages C, Javascript et JSON surtout. Git a été un outil très important dans la réalisation de notre projet. Nous avons également pu faire une utilisation poussée de l'éditeur Vim et de ses capacités de développement.

5.4 Bilan humain

Sur le plan humain, ce projet nous a permis d'améliorer nos compétences de travail en groupe. Nous avons également pu mettre en pratique des connaissances de gestion de projet acquises au cours de notre formation.

Conclusion

Octopus Supervisor est donc un outil de supervision et d'audit de la sécurité système dans un réseau qui est fonctionnel et prêt au déploiement. L'outil a les fonctionnalités correspondant au cahier des charges établi au départ avec des aspects en plus tel que la portabilité du logiciel.

C'est un outil permettant à des utilisateurs (administrateurs réseaux le plus souvent ou responsable sécurité des SI) de superviser en temps réel un parc de machines clientes. Cette supervision passe par l'éxécution à distance de script sur les clients afin de déceler des failles systèmes dans le parc.

Notre solution rassemble la fiabilité du logiciel qui possède un code robuste, la confidentialité car le logiciel est sécurisé sur de multiples points, la tracabilité puisque la base de données sauvegarde tous les résultats de script et des logs sont sauvegardés et l'intégrité des données car celles-ci sont chiffrées de bout en bout lors des communications. Octopus Supervisor est une solution légère et facilement implémentable sur des systèmes à faibles performances. Ainsi, les administrateurs pourront superviser une grande diversité d'hôtes à travers leur parc informatique.

Le logiciel que nous présentons est complètement fonctionnel mais peut être améliorer afin de correspondre à des besoins plus précis suivant les entreprises. Un administrateur pourrait souhaiter une interface différente d'une interface Web et notre code est prêt à recevoir une nouvelle interface avec peu de modifications.

ANNEXES

A Installation des Machines Virtuelles

A.1 Installation d'une Debian Minimale

Ce chapitre va présenter l'installation d'une machine virtuelle sous le système d'exploitation Debian. Nous vous proposons de suivre une installation minimale, sans connexion réseau.

FIGURE 7 – Ecran d'accueil pour l'installation de Debian

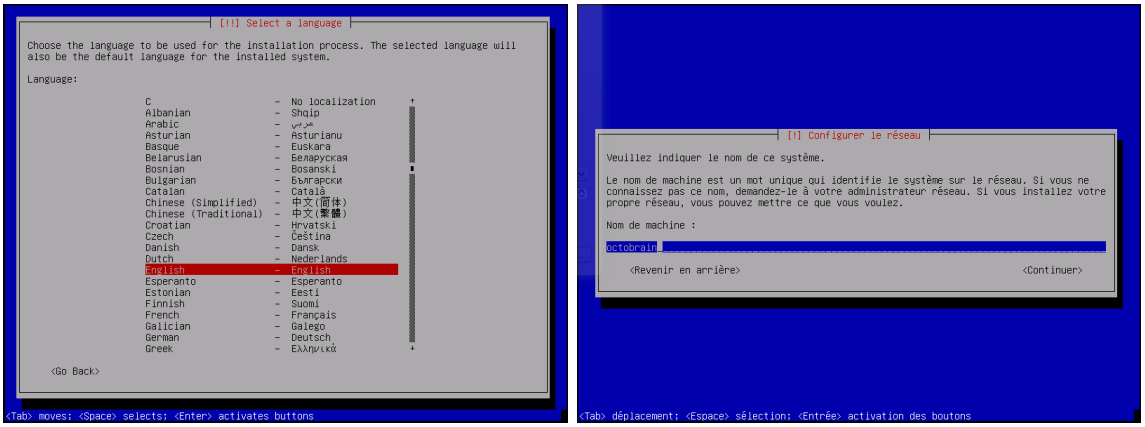


FIGURE 8 – Choix de la langue et configuration réseau

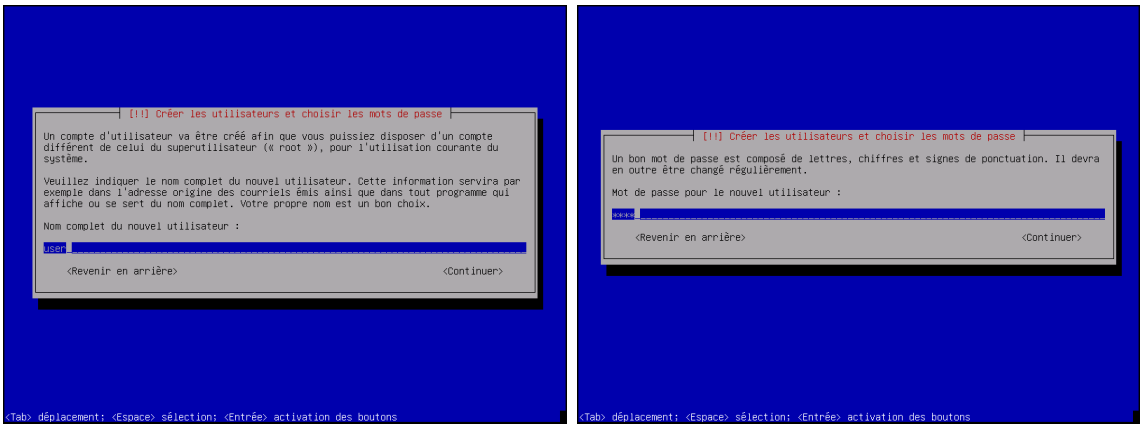


FIGURE 9 – Création d'un utilisateur avec son mot de passe

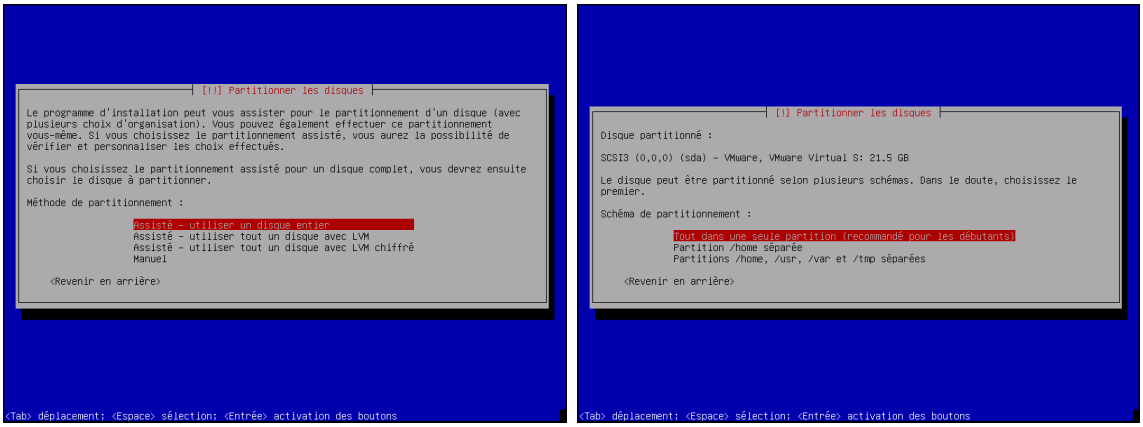


FIGURE 10 – Partitionnement du disque dur et choix d'emplacement d'installation

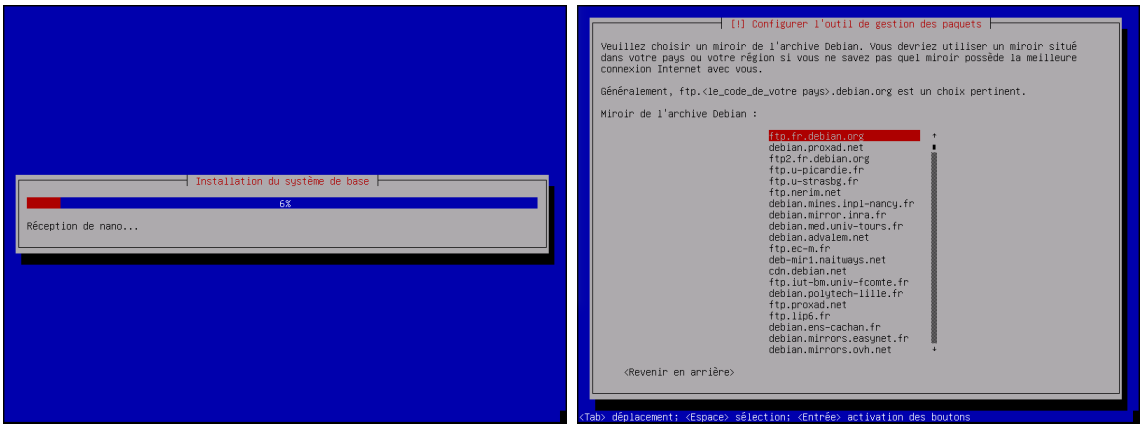


FIGURE 11 – Installation du système et configuration des miroirs de packets

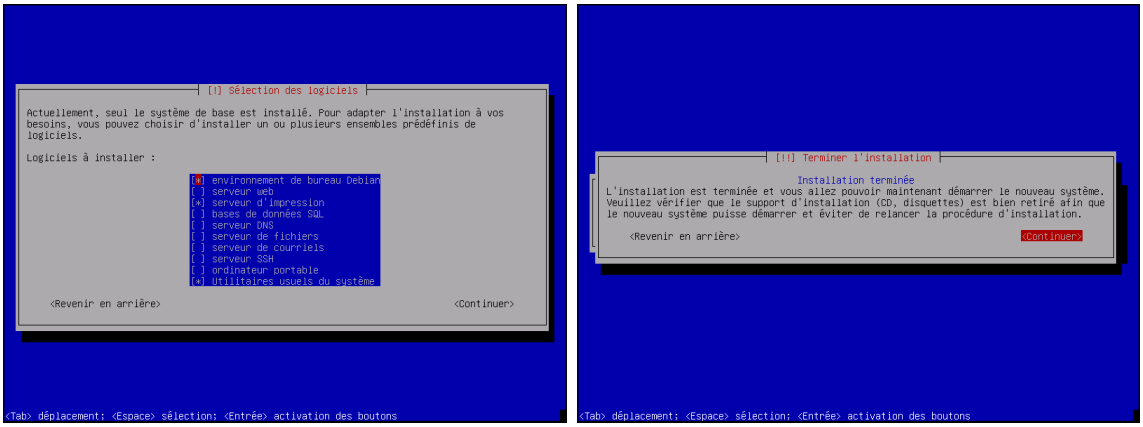


FIGURE 12 – Choix des logiciels importants et finalisation de l'installation

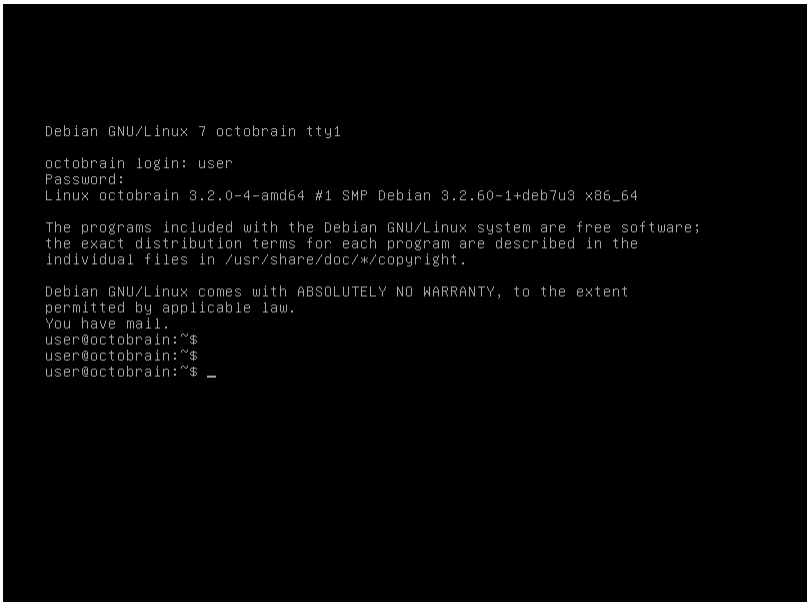


FIGURE 13 – Ecran du système fonctionnel

Nous avons maintenant une machine virtuelle fonctionnant avec Debian 7 en installation minimale. C'est-à-dire que seul les packages essentiels sont installés. Afin de faire fonctionner correctement Octopus Supervisor, il faut ajouter les paquets suivants : gcc, make, libssl-dev, libsqlite3-dev.

A.2 Installation d'un Xubuntu

Nous allons vous montrer ci-dessous les différentes étapes d'une installation graphique du système Xubuntu.

Maintenant, nous avons un client opérationnel sous Xubuntu. Et nous pouvons ensuite installer le paquet client d'Octopus Supervisor.

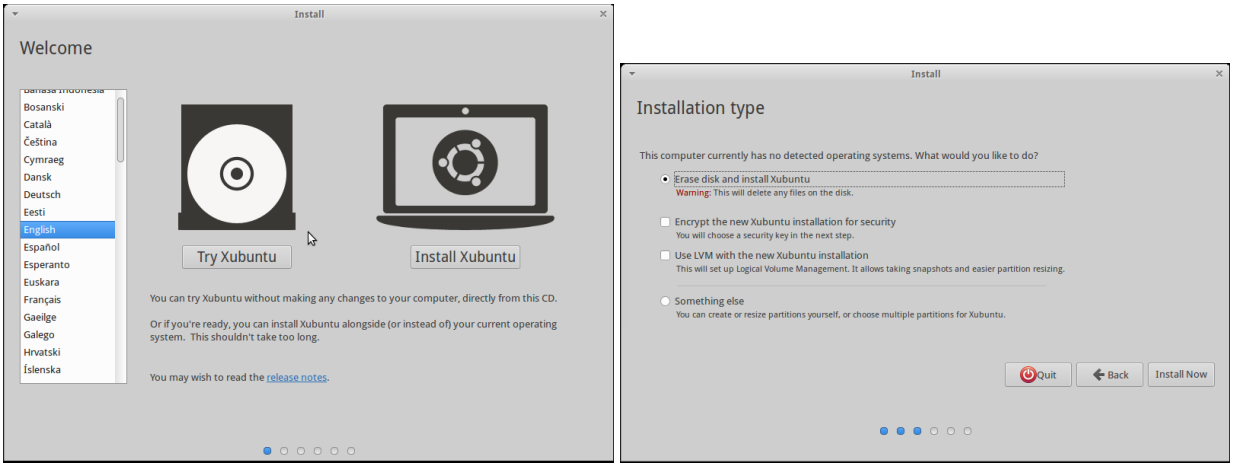


FIGURE 14 – Choix de la langue et du type d'installation

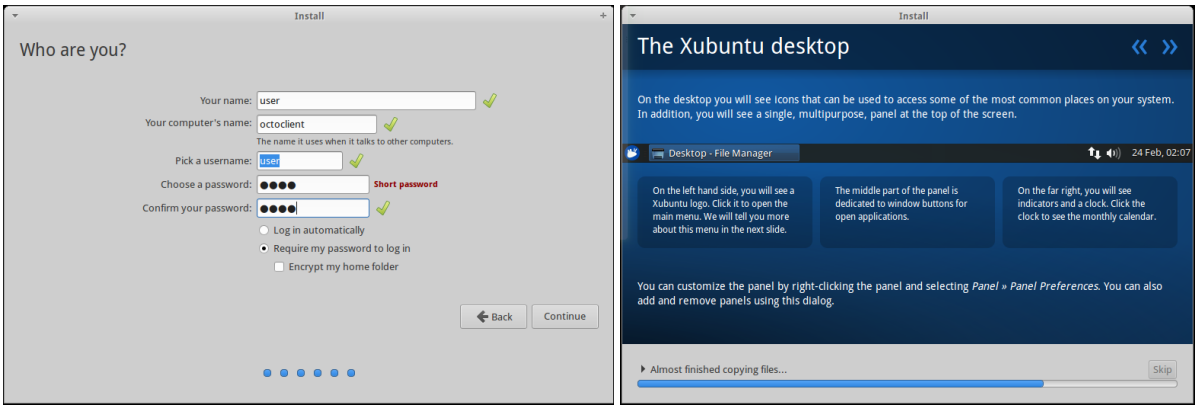


FIGURE 15 – Définition d'un utilisateur et installation du système

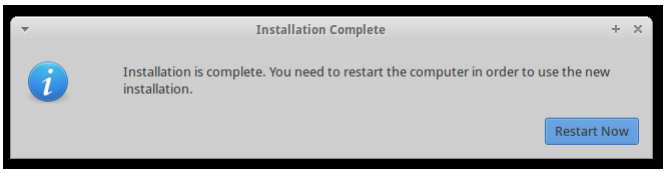


FIGURE 16 – Fin de l'installation

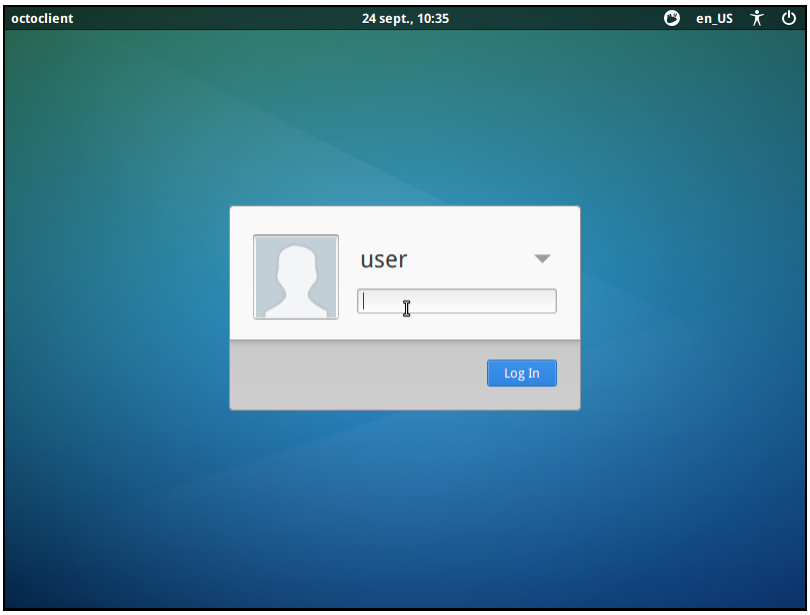


FIGURE 17 – Ecran de connexion utilisateur

B Travailler avec Git

B.1 Contexte

Git est un outil de développement très pratique lors de projet en équipe. C'est un outil de sauvegarde permettant de faire du versionning. Nous avons utilisé la forge "Gitlab" sur le serveur personnel de Nicolas pour travailler dans les meilleures conditions possibles sur ce projet.

B.2 Les bases

Pour présenter les bases de Git, nous allons vous expliquer les commandes indispensables.

Tout d'abord, il faut travailler dans un dossier vierge afin de ne pas parasiter votre projet. Ensuite, on initialise le répertoire en tant que dépot Git avec la commande "git init". Afin de lié notre répertoire local avec un dépot distant sur un serveur Git, il faut utiliser la commande "git clone user@url/depot/git :projet.git". Celle-ci permet de sauvegarder les versions de votre travail sur le serveur "url/depot/git" dans le projet se nommant "projet". Ensuite, vous pouvez commencer à travailler le développement de notre projet en créant des fichiers et répertoires.

Lorsque vous avez fini de développer une fonctionnalité de votre projet ou que vous souhaitez créé un point de sauvegarde, vous allez devoir commiter avec la commander "git commit". Vous pouvez commiter tout votre projet ou seulement certains fichiers. A ce moment, vous devez mettre une phrase courte pour expliquer ce que vous venez de développer.

Une fois que vous avez fini votre session de codage et que vous avez commit votre projet, vous allez donc le sauvegarder sur le serveur Git afin d'avoir un backup et de permettre à vos collègues de travailler sur vos modifications. Pour faire cela, vous devez d'abord récupérer les modifications que vos collègues ont mis sur le serveur avec la commande "git pull". Vous utiliserez cette commande pour mettre à jour votre dossier local. Ensuite pour mettre votre code sur le serveur, vous allez utiliser la commande "git push" qui va pousser vos derniers commits.

Les bases de Git sont maintenant établies. Une utilisation avancée en fait un outil très puissant et c'est pour cela que notre choix s'est porté dessus.

C Installation d'Octopus Supervisor

L'installation du logiciel Octopus Supervisor doit être faite par un administrateur de la machine. Il faut d'abord compiler le programme avec la commande "make", ensuite il faut exécuter l'installation en root avec "make install". Voila, maintenant Octopus Supervisor est installé et prêt à fonctionner !

README

```
/*************************************************************
*   Octopus Supervisor                  *
*   Written and owned by Nicolas Cosnard and Cyril Segretain*
*************************************************************/

What you need before installing Octopus Supervisor is :
   - gcc
   - make
   - libssl-dev
   - libsqlite3-dev

Use "sudo apt-get install gcc make libssl-dev libsqlite3-dev"
if packages are missing.

How to install Octopus Supervisor ?
   > First, you need to get the package for the server or the client.
   > Second, go into the Octopus directory.
   > Then, use the command "make" to compile the software.
   > The last part is to use "make install" in root mode to install
   files in your system.

How to launch Octopus Supervisor ?
   > You need to be root.
   > Use "octoBrain" for the server and "octoTentacle" for the client.
   > If you want that the client connect itself to the server, use the
   option "-b [ip server]" with the server IP.
   > If you want to have logs on STDOUT and debug logs, use the option
   "--no-daemon".
```

Table des figures

1 Algorithme général de fonctionnement de Octopus Supervisor 7
2 Fonctionnement 8
3 Fonctionnement technique du Brain 9
4 Fonctionnement technique du Tentacle 12
5 Handshake enregistrement 15
6 Handshake envoi résultat 15
7 Ecran d'accueil pour l'installation de Debian 21
8 Choix de la langue et configuration réseau 21
9 Création d'un utilisateur avec son mot de passe 22
10 Partitionnement du disque dur et choix d'emplacement d'installation 22
11 Installation du système et configuration des miroirs de packets 23
12 Choix des logiciels importants et finalisation de l'installation 23
13 Ecran du système fonctionnel 24
14 Choix de la langue et du type d'installation 25
15 Définition d'un utilisateur et installation du système 25
16 Fin de l'installation 25
17 Ecran de connexion utilisateur 26