# MI-LXC : Une plateforme pédagogique pour la sécurité réseau

François Lesueur
francois.lesueur@insa-lyon.fr

CITI - INSA Lyon

**Résumé.** MI-LXC, sous license AGPL, est un *framework* et une infrastructure pour pratiquer des exercices de sécurité réseau. Le *framework* permet de construire des infrastructures virtuelles utilisant LXC, typiquement pour l'enseignement de la sécurité, du réseau ou de l'administration système. Une infrastructure représentant un mini-internet (différents AS routés par BGP, une racine DNS, des services publics HTTP/SMTP, une organisation mêlant LDAP, filer, intranet et postes clients, ...) a été conçue sur ce *framework*. Cet article présente ces deux aspects de MI-LXC ainsi que l'usage possible pour des TP de sécurité.

L'enseignement pratique de la sécurité des réseaux et des systèmes se heurte à la complexité de proposer des systèmes d'étude adaptés. Sans un système suffisamment complexe, un simple TP *firewall*, par exemple, peut se résumer à appliquer la syntaxe iptables (voire manipuler l'interface d'une *appliance*) et analyser/constater les effets de ports bloqués. Hors, l'intérêt d'un TP firewall réside probablement d'abord, aujourd'hui, à réfléchir et proposer un cloisonnement adapté d'un système donné, plus fin que le périmétrique WAN/LAN/DMZ historique, puis à l'implémenter dans un langage quelconque (iptables ou autre). Cette réflexion ne peut se mener qu'au sein d'un système à la fois suffisamment complexe et suffisamment maîtrisé par les étudiants.

MI-LXC vise à répondre à ce type de problématique, à savoir fournir un *framework* permettant la conception de telles infrastructures, déployables sur de simples PC, ainsi qu'une infrastructure de référence représentant les éléments clés d'internet et d'un système d'information. MI-LXC est distribué sous licence AGPL à l'adresse `https://github.com/flesueur/mi-lxc`.

## 1 Le framework

MI-LXC suit le paradigme de l'*infrastructure-as-code* pour programmer la topologie réseau et le contenu des machines. Cette construction est ainsi

facile à modifier, partager, versionner, mettre à jour. Les machines vont de serveurs HTTP à des bureaux graphiques d'utilisateurs. Un mécanisme de *templates* permet de factoriser les similitudes entres différentes machines.

MI-LXC est écrit en Python3, l'infrastructure s'exécute ensuite sur LXC directement sur le poste client. LXC a l'avantage d'être très léger (plus léger que la virtualisation classique type VirtualBox) tout en offrant un système Linux complet (contrairement à Docker qui est conçu pour lancer un exécutable unique, et non un processus d'init). L'inconvénient principal est de limiter les systèmes générés au monde Linux ; l'intégration de postes Windows pourrait être réalisée en ajoutant un *backend* Virtualbox, l'ensemble pourra communiquer par les *bridges* ethernet de l'hôte Linux, mais cela n'est pas encore implémenté.

La figure 1 permet de voir la combinaison de machines *headless* et d'autres proposant un bureau graphique.

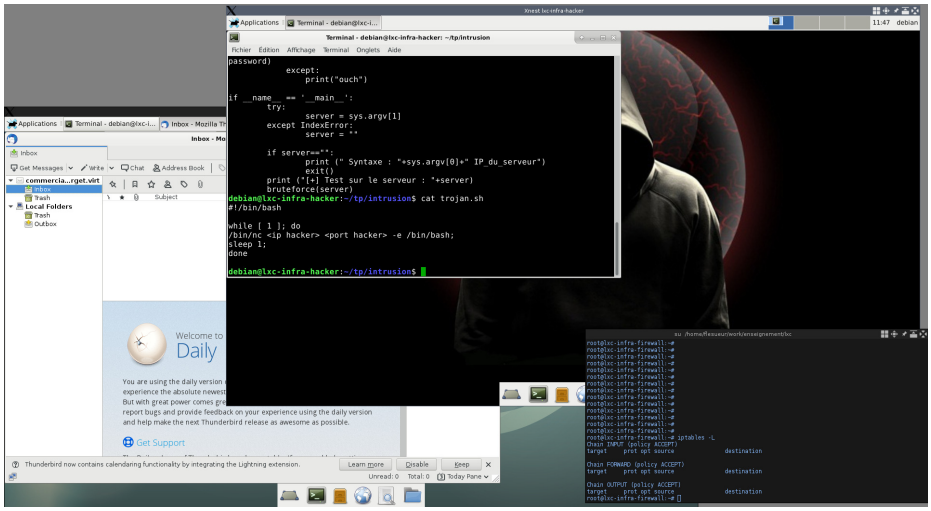

**Fig. 1.** Exemple d'infrastructure mêlant ligne de commande et accès X11

## 2 L'infrastructure actuelle

L'infrastructure actuellement proposée est illustrée figure 2. Les ovales verts correspondent aux switchs virtuels, les rectangles rouges aux machines. Le système est composé comme suit :

— le cœur de réseau est représenté par les deux opérateurs de transit *transit-a* et *transit-b*. L'opérateur *transit-a* donne de plus l'accès

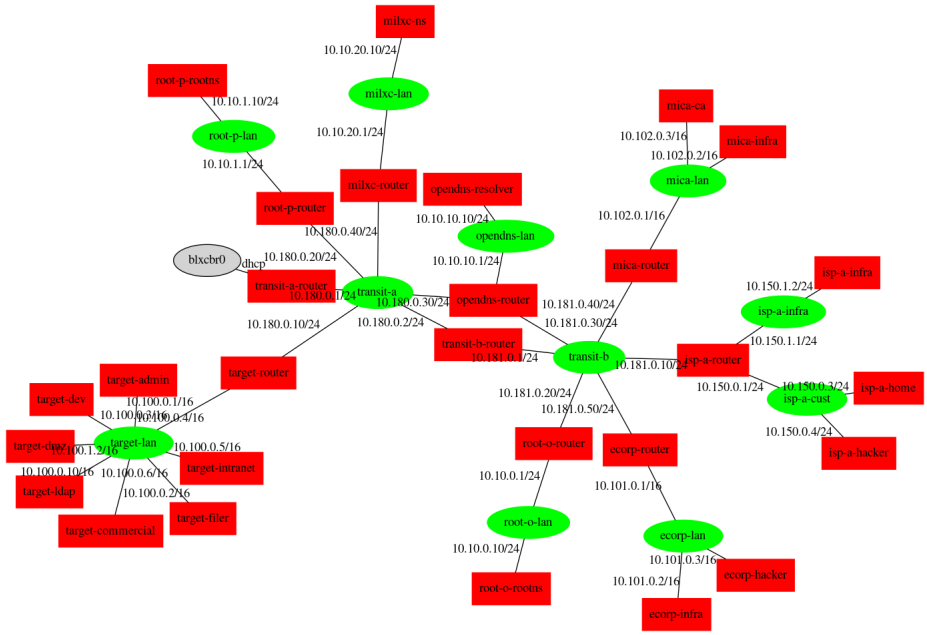


**Fig. 2.** Topologie actuelle

au *vrai* internet à l'ensemble de l'infrastructure, via le bridge LXC sur l'hôte
— le TLD `.milxc`, pour tous les domaines internes, est géré par l'AS *milxc*
— les 2 racines DNS (alternatives mais compatibles avec le vrai internet) sont *root-o* et *root-p*
— un résolveur DNS ouvert est proposé par *opendns*
— un FAI *isp-a* fournit l'accès à différents clients, honnêtes ou attaquants
— une autorité de certification utilisant le protocole ACME (comme *Let's Encrypt*) est proposée par *mica*
— l'entreprise *target* héberge un réseau (volontairement à plat sur ce point de départ, un TP consistant justement à le segmenter), fournissant des services externes SMTP, HTTP, DNS et internes LDAP, partage, intranet, ainsi que des postes client
— l'entreprise *ecorp* permet de mener des attaques BGP
— le protocole BGP est utilisé pour le routage entre ces différents AS

Tout ceci nécessite moins de 2Go de RAM et de 6Go d'espace disque (en plus de l'hôte Linux). Le dépôt git permettant de générer l'ensemble

fait moins d'1Mo et la recette de création de chaque machine représente quelques dizaines de lignes de script bash.

## 3 Exemples d'utilisation

Quatre exemples d'usages possibles sont proposés dans des sujets de TP (4h) disponibles sur la page du projet : un scénario d'intrusion, une segmentation par *firewall*, une surveillance par IDS et le déploiement d'une autorité de certification globale. Quelques perspectives, non encore approfondies, me paraissent intéressantes côté NSM/*hunting* ou collaboration au sein d'une fédération MISP.

### 3.1 Scénario d'intrusion

Ce TP précède ceux dédiés à la segmentation réseau et aux IDS. Il vise à appréhender la cinématique d'une attaque à travers plusieurs rebonds ainsi qu'à découvrir de quel type d'événements il faudra se protéger.

L'attaquant est externe et dispose uniquement de la machine *isp-a-hacker*. Son but est de voler et supprimer des informations hébergées sur un serveur interne de l'entreprise *target*. L'attaque proposée se déroule en plusieurs étapes :

- Attaque par brute-force d'un accès administrateur sur une application web accessible depuis l'extérieur, hébergé sur *target-dmz*. Il s'agit d'un wiki sur lequel l'attaquant peut ensuite déposer un cheval de troie, sous la forme d'un simple reverse-shell
- Envoi d'un mail spoofé à un commercial de l'entreprise, lui demandant de télécharger et exécuter le script déposé précédemment. Les étudiants doivent exploiter de l'ingénieurie sociale et peuvent profiter du fait que le script à faire exécuter est accessible par une URL interne à l'entreprise, puisque déposé précédemment sur le wiki.
- L'attaquant obtient un reverse shell sur la machine du commercial
- L'attaquant peut scanner le réseau avec nmap (qui est à transférer vers la machine du commercial). À ce moment, les étudiants doivent établir une cartographie complète du réseau, avec les numéros de version des services, les bannières, les index des serveurs web, *etc.*
- Un serveur web expose clairement les données recherchées
- Le compte du commercial peut se connecter à ce serveur web par SSH, il faut pour cela récupérer le mot de passe du commercial. Deux approches sont proposées, une par un *keylogger*, l'autre en

utilisant un logiciel type *Lazagne* pour récupérer le mot de passe dans la configuration du client mail
— Le hacker, via le compte et la machine du commercial, peut ainsi se connecter en SSH au serveur interne. Une recherche exhaustive des droits permet de constater que les droits sur les fichiers importants sont trop laxistes
— Un autre chemin d'attaque existe par le poste du développeur, dont le code est déployé de manière continue et qui peut donc être modifié par l'attaquant (puis automatiquement déployé)

### 3.2 Segmentation réseau

Ce TP se joue sur la machine *target-admin*, pour administrer le firewall de *target-router*. Une première partie est centrée sur l'usage de la commande iptables. Ensuite, ayant compris le principe de ce que l'on peut filtrer, le type de mouvement problématique et la structure du système d'information, les étudiants doivent proposer une segmentation sur papier. Cette étape intermédiaire porte à discussion, jusqu'à converger vers une proposition cohérente. Enfin, la troisième étape consiste à implémenter cette politique par segmentation (rajout d'interfaces réseau et modification de la topologie) puis filtrage.

À la fin de ce TP, les étudiants obtiennent un système dans lequel les zones fonctionnelles sont clairement isolées et les déplacements latéraux contraints. L'accès à l'ensemble du réseau depuis le poste du commercial n'est par exemple plus possible. Il ne reste qu'un ensemble de flux nécessaires, qui sont ceux qui devront être surveillés lors du TP IDS.

### 3.3 IDS

Ce TP est toujours basé sur le scénario d'intrusion vu précédemment, à partir duquel les étudiants manipulent HIDS, NIDS et éventuellement corrélation. Les outils utilisés sont OSSEC, Suricata, Prelude et Prewikka. Les étudiants doivent proposer les phénomènes à observer puis les réaliser. Il est par exemple possible de surveiller le brute-force sur le wiki (OSSEC et Suricata), l'apparition de fichiers sur le wiki (OSSEC), l'ouverture du reverse-shell (Suricata), les altérations de fichiers sur le serveur interne (OSSEC), *etc.*

Ce travail permet d'évaluer les différents points de contrôle, leur pertinence, mais aussi la simplicité d'évasion de ces contrôles. Il serait intéressant, à partir de là, de creuser dans un autre TP des approches plus

NSM ou la collaboration entre plusieurs organisations, à travers MISP, pour par exemple partager des règles Suricata.

### 3.4 Autorité de certification

Ce TP est dissocié des précédents et est réalisé dans une autre matière. Il s'agit de comprendre le fonctionnement et les limites d'une autorité de certification. L'objectif est de permettre à un navigateur sur la machine *isp-a-home* de se connecter de manière sécurisée au site hébergé sur *target-dmz*.

Dans la première partie, les étudiants analysent le problème de sécurité en attaquant une connexion http simple, via une attaque DNS (modification de la zone) ou BGP (altération par l'AS *ecorp*). Ensuite, ils créent l'autorité sur la machine *mica-ca*, signent un certificat pour *www.target.milxc* et configurent le navigateur client pour reconnaître cette autorité. À ce moment, l'attaque initiale (DNS ou BGP) ne fonctionne plus. Enfin, ils peuvent retenter ces attaques DNS ou BGP lors de la phase de la signature du certificat de *www.target.milxc* pour obtenir, avec succès, un certificat pour ce site qui sera bien signé mais indu : une des limites de l'approche CA est ainsi pointée, dans cette phase de vérification initiale par la CA.

## 4 Comment tester ?

MI-LXC est disponible à l'adresse `https://github.com/flesueur/mi-lxc`. Il y a plusieurs façons de le tester, telles que documentées sur la page du projet :

- Sous Linux, les dépendances pour quelques distributions sont indiquées et il est possible d'exécuter directement MI-LXC ensuite (quelques dizaines de minutes pour générer l'infrastructure)
- Sous Windows/OSX/Linux, il est possible de générer avec Vagrant une VM VirtualBox dans laquelle MI-LXC sera automatiement installé et les conteneurs LXC générés (quelques dizaines de minutes pour générer l'infrastructure, connexion root/root)
- Pour la dernière release, une VM Virtualbox exportée (.ova) est disponible directement au téléchargement sur github (connexion root/root)