

# Mini-Internet using LXC (MI-LXC) : un générateur d'environnement réseau simulé

---

**François Lesueur**

francois.lesueur@insa-lyon.fr

@FLesueur

<https://github.com/flesueur/mi-lxc>

RESSI, 17 mai 2019

INSA Lyon, Département Télécommunications, Services et Usages

---



# Problème

# Ce qui me manque

## Firewall

- 1 Syntaxe iptables (INPUT/OUTPUT)
- 2 Intégration FORWARD entre 2/3 machines
- 3 WAN/LAN/DMZ
- 4 **Segmentation fine** (nécessite un SI fonctionnel suffisamment complexe)

## HTTPS

- 1 Syntaxe openssl (beurk !)
- 2 Déploiement d'un certificat sur une VM
- 3 Intégration avec un DNS
- 4 **Cycle de vie complet depuis la vérification par la CA en milieu hostile** (BGP, racine DNS, etc.)

⇒ Étudier la sécurité de systèmes digitaux ✓ complexes, fédérés, réalistes et sous des souverainetés différentes

# Et pour ça ? (1/2)

## Ce qu'il faut générer

- Un environnement au plus proche d'internet
- Des SI (avec services ouverts SMTP/HTTP, authentif centralisée, serveur de fichiers, sauvegarde, VPN, ...)
- L'interconnexion entre les SI (AS BGP)
- Des services communs (racine DNS)
- Un darknet✓ utilisant la blockchain✓

# Et pour ça ? (2/2)

## Comment il faut générer

- De manière souple/évolutive/historisable
- Scalable en SLOC (sur la taille de l'environnement généré)
- Implique une génération rapide. . .
- . . . et une prise en main acceptable

## Comment l'exécuter

- Sur des portables d'étudiants
- Facile à transférer
- Génération rapide, pas trop de RAM, pas trop de CPU

Quel contexte de développement ?

# Quel outil de "virtualisation" ?

## VMs (VirtualBox, VMWare)

- Similaire à une vraie machine
- Très bonne isolation
- Mais trop coûteux (RAM, CPU, temps de génération)

## Conteneurs (Docker, LXC)

- (Un peu) moins similaire à une vraie machine
- Beaucoup plus léger !

⇒ LXC qui propose une abstraction plus complète (démarrage d'un init plutôt que packaging d'une application)

# Quel outil de création/provisionnement ?

## Création

- Vagrant plutôt côté VM (plugin LXC non maintenu)
- LXC intègre des primitives de création directement

⇒ LXC via la bibliothèque python

## Provisionnement

- Puppet/Ansible résolvent des problèmes de masse/run qui ne se posent pas ici
- Se résumeraient à de simples scripts...

⇒ Des scripts bash

## Quels outils de lab ?

- Des outils de connectivité réseau sans aide à la création d'hôtes variés (Marionnet, Internet Simulator)
- Des outils basés sur Docker sans *init* (Dockernet, Kathara)
- Labtainers basé sur Docker, utilise une image avec *systemd* (non maintenue) et très compliqué à modifier (gros projet)

# Et donc ? MI-LXC ?

## Générateur disruptif ✓

- Crée des conteneurs LXC
- Spécification de la topologie (JSON)
- Provisionning individualisé
- *Templates*
- 390 SLOC dans `mi-lxc.py` (la complexité est en fait dans le provisionning)

## Infrastructure

- 17 conteneurs actuellement
- Plusieurs AS avec routage dynamique BGP
- Dont un SI complexe (firewall, DMZ, authentification centralisée, serveur de fichiers, intranet, SMTP ...)

Usage

# Séquence pédagogique (Bac+5)

## Intrusion

- Mise en situation dans le rôle du hacker
- Brute-force sur un wiki interne, dépôt d'un troyen revshell, incitation au DL par mail spoofé, mapping, rebond interne vers un serveur de prod

⇒ Découverte du concept d'attaque, rebond et de l'infrastructure

## Firewall puis IDS (et ELK ?)

- Passage côté défensif
- Segmentation (DMZ, authentif, admin, dev, prod) puis *implémentation* de cette conception
- NIDS/HIDS/SIEM sur ce qui reste ouvert

⇒ IPTables est un outil, la DMZ n'est qu'un cas particulier

# Séquence pédagogique (Bac+4)

## HTTPS cette année

- CA/Apache dans le SI simulé
- Connexion depuis l'extérieur, attaques sur le chemin
- (Préparé et déployé en 2h)

## HTTPS l'année prochaine

- Intégration d'un serveur de vérification ACME (Let's encrypt)
- Analyse d'une attaque BGP lors de la création

⇒ Le certificat ne prouve pas un état actuel mais un état lors de la vérification par la CA

# Comment l'installer ?

## Sous GNU/Linux (Debian, Ubuntu, Arch, Kali)

- `git clone https://github.com/flesueur/mi-lxc.git`
- `./mi-lxc create` (15-20 minutes)
- `./mi-lxc start`
- `./mi-lxc attach dmz ; ./mi-lxc display hacker`
- `./mi-lxc print`

## Sinon, pour les pommés

- `git clone https://github.com/flesueur/mi-lxc.git`
- `cd vagrant && vagrant up` (20-25 minutes)
- `./mi-lxc start` (dans la VM)
- `./mi-lxc attach dmz ; ./mi-lxc display hacker`
- `./mi-lxc print`

# Conclusion

## Bilan

- Un *framework*
- Les *sources* d'une infrastructure souveraine ✓
- Libre (AGPL), testé sur ~100aine de cobayes

## Et maintenant ?

- Amélioration des réagencements réseau
- Encore plus d'infra et de transitaires !
- Des attaques BGP
- Des bots actifs ? Des scénarios variés ? Du deep-learning ✓
- D'autres backends que LXC (VM ? HW ?)
- En support de projets étudiants cyber ✓ (conception de SI)

# Mini-Internet using LXC (MI-LXC) : un générateur d'environnement réseau simulé

---

**François Lesueur**

francois.lesueur@insa-lyon.fr

@FLesueur

<https://github.com/flesueur/mi-lxc>

RESSI, 17 mai 2019

INSA Lyon, Département Télécommunications, Services et Usages

---

