MI-LXC: a first step towards a free CyberRange ?

François Lesueur
INSA-Lyon, CITI
F-69621 France
firstname.lastname@insa-lyon.fr
https://github.com/flesueur/mi-lxc

Abstract

MI-LXC is a framework to construct virtual infrastructures on top of LXC to practice security on realistic infrastructures. MI-LXC follows the *infrastructure-as-code* paradigm to program the topology of the system and the provisioning of the different hosts. This construction is highly customizable, allowing to create hosts ranging from webservers to graphical user sessions. Provisioning of similar subsets of features on different hosts is attained through a template mechanism. MI-LXC is still in its early stages and I currently use it to generate some labs. In this paper, I wonder whether it could be used as a first step towards a free cyber-range. MI-LXC is licensed under AGPL.

1 Introduction

Teaching is an emancipation power which can take several forms. It is not reserved to schools and universities, as transmitting knowledge is also at the heart of collaboration or free software for instance. We can thus be particularly skeptical when large companies gain some momentum in the education space, such as Google or Microsoft teaching in some universities in France[1].

In the digital security area, a few large companies now propose what they call CyberRanges. A CyberRange is an integrated platform to train people, both professionals or students, by simulating an information system, an attacker, etc. During a cyberrange training, participants should collaborate in this exercise to fight the attacker. These platforms mix both general purpose PCs (running VMs) and dedicated hardware such as routers or security appliances. To the best of my knowledge, there are no FLOSS platforms to train on, and I see this as a risk.

A classical cyberrange seems to be composed of the following ingredients : a physical mobile platform, some racked servers running a large set of VMs to simulate a real environment, real dedicated hardware such as routers or security appliances, different scenarios and possibly a programmed attacker (who said AI ?). For a free cyberrange, we may need : no physical platform (everything should be easily deployable on commodity hardware), no dedicated hardware (we would only use free security tools (as a second step, interoperability with dedicated hardware might be discussed)), some tool to automate the conception, generation and deployment of the simulated infrastructure, some scenarios and some bots to play the attackers

In this paper, I briefly propose a global schema for a free cyberrange. Then, I discuss some related tools. I finally present MI-LXC[2], a software I develop (under AGPL) to programmatically generate a small-scale internet-like environment running as LXC containers.

Finally, the objective is to see if the community would like to join on this project since I think that MI-LXC may be used as a framework to propose different scenarios and attacker bots and, as such, may help providing a free counterpart to proprietary cyberranges.

2 Towards a free Cyberrange

What do we need to provide a free alternative to proprietary cyberranges ?

[1] https://affordance.typepad.com/mon_weblog/2018/02/facebook-google-universite-formation-et-merde.html
[2] https://github.com/flesueur/mi-lxc

1. Some realistic training scenarios. A simple scenario is currently included in MI-LXC, allowing to brute-force a wiki account, phish an internal user and then pivot to internal servers. The defense part consists in segmenting the network (iptables) and managing an IDS (OSSEC, suricata, prelude). More advanced scenarios would have to be specified.

2. A flexible virtual infrastructure, providing both applicative hosts, clients, security tools and network topology. This infrastructure must be easy to customize, regenerate and deploy. MI-LXC tackles this point by providing a framework to generate complex infrastructures based on LXC isolation.

3. Some background activity on the generated infrastructure, with both legitimate traffic and attacks. This could be done with scripts running on the different hosts, simulating activity. This part is not treated currently.

3 Some related tools

There are already a few tools to generate labs, possibly with vulnerable VMs. I explain here why they did not fulfilled my needs.

A set of recent tools uses docker to isolate the different parts of the system to simulate (Labtainers[3], Kathara[4], Dockernet[5]). Docker is quite popular but is centered on the paradigm of one process per container. Rather than launching an init process, docker containers run a single application which are then composed using docker-compose. It should be possible to run an init process in a docker (although there seems to be some incompatibilities with systemd at least) but it goes against the very basic philosophy of Docker, which is not a path I want to follow. And thus, complying with the docker philosophy does not allow to simulate realistic systems running on classical OS. In fact, these projects aim to teach network (services, routing) rather than security.

SecGen[6] creates full VMs but is dedicated to CTF-style challenges. It creates voluntarily vulnerable images and plants flags inside. It is rather a tool to practice offense with artificial vulnerabilities, where I rather aim at practicing defense of up-to-date systems.

VM building tools such as Vagrant and provisioning tools such as puppet or ansible are of course also in the scope. But in fact, they tackle slightly different problems. Vagrant automates the VM creation but the biggest work in our case is, in fact, the recipes of VM creation. Vagrant moreover does not support well LXC (only unofficial plugin) and mostly targets heavy virtualization such as VirtualBox. Puppet or ansible target the provisioning but not the creation process and network topology aspects.

With all that in mind, and given LXC provides official Python3 bindings, I developed MI-LXC to suit my needs. Anyway, most of the work lies in the recipes to configure the different hosts in the simulated system.

4 MI-LXC

MI-LXC is a python tool to program the generation of an infrastructure. This infrastructure then runs on LXC and is connected through virtual ethernet bridges. The topology is described in the `setup.json` file and the provisioning of the different hosts is described in the `files/` subfolder. Provisioning scripts can be host-specific or templates.

Each container is described by a name, a set of configured network interfaces, a set of templates and a provisioning script. All of this is specified in `setup.json` and, in my current setup, I manage 11 containers with 83 lines of description.

Most of the work lies in the `files` subfolder. For each container, a `<container>/provision.sh` script is run during the creation to configure applications, copy files, initialize home directories. I favor a programmatic approach, installing and configuring in-situ with sed for instance, rather than blindly overwriting files.

My example infrastructure is composed of Debian hosts : a firewall, a DMZ, an internal server, a filer, a centralized authentication, 3 internal hosts for employees, 2 external hosts and a routing backbone.

[3] `https://my.nps.edu/web/c3o/labtainers`
[4] `http://www.kathara.org/`
[5] `https://github.com/gmiotto/dockernet`
[6] `https://github.com/cliffe/SecGen`

All of that needs less than 1GB of RAM and less than 4GB of disk space. This allows to practice an intrusion, firewall and IDS. It is illustrated running on Figure 1.

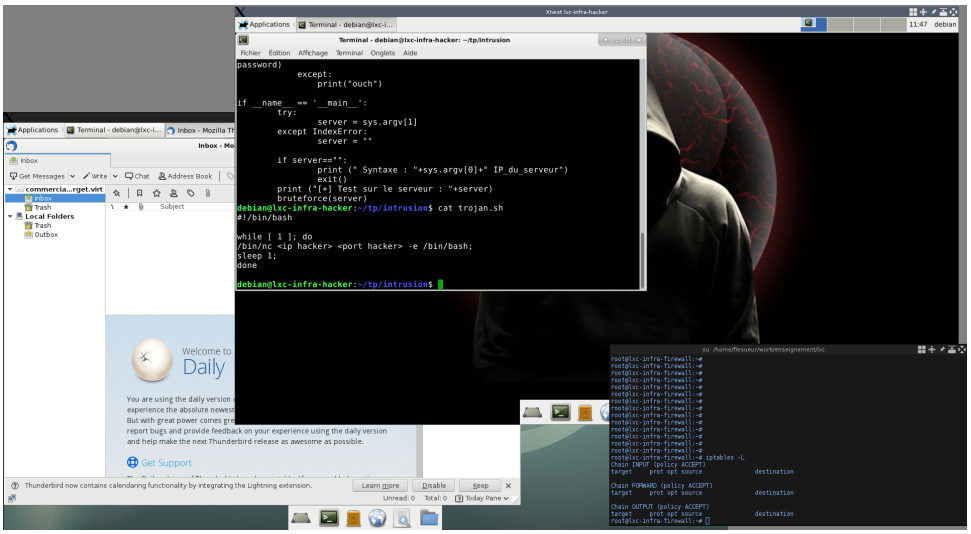

Figure 1: Example of a running infrastructure, showing both command-line access and X11 access

5 Conclusion

MI-LXC allows to specify, program and deploy virtual infrastructures for training. I think it may be a step towards a free counterpart to commercial cyberranges. I would thus be particularly interested by some feedbacks as well as potential discussions on the missing aspects, mainly different scenarios and bots to generate legitimate traffic or adaptive attackers.