# Automatic generation of virtual machine for security training

Manuel Sánchez Rubio[1], and Germán López Civera[2]

[1] Universidad Internacional de La Rioja (UNIR), La Rioja, Spain
manuel.sanchezrubio@unir.net

[2] Universidad de Alcalá, Alcalá de Henares, Spain
g.lopez@edu.uah.es

**Abstract.** This paper explores the applicability of configuration management tools to the design and development of practical content challenges in the field of cybersecurity. Using the tools Puppet and Packer, a series of templates have been developed to create a test scenario. Such scenario has been proposed to contain both vulnerable services and already implemented security measures. Based on this scenario, flexibility of the solution and time saving achieved have been assessed. Through this assessment, it has been determined that the use of these tools is a viable option in developing both small scale scenarios focused on teaching and big scenarios used in cybersecurity events.

**Keywords:** cyber exercise, configuration management, Puppet, virtualization.

## 1 Introduction

IT security has currently become one of the main concerns of companies and institutions. The practical component is extremely important as it prepares students to face real situations they will find in their jobs.
In teaching, the main problem is the time cost its development implies and, consequently, the scarce variety of scenarios that can be proposed. To this end, virtualized systems allow us to create believable environments where students can interact with the tools and face similar situations to those that may occur in a real environment. This can be achieved through gamification models or Capture The Flag (CTF) competitions where students must pass a series of tests.
The latter is added one more level of competitiveness since the different teams have to face each other. In this type of events, a series of vulnerable servers is delivered to each team. Each must find and fix these vulnerabilities, fending off other teams, while finding and exploiting failures in servers of opposing teams.
Attacking the problem from automation of the process of creating machines and services deployed in them, and preparing the necessary infrastructure to host a CTF type event or cyber exercise is a complex, multidisciplinary task that is to be addressed in this paper, in addition to preparing complex scenarios to install, configure, deploy and monitor a large number of machines from the world of virtualization to create different, dynamic scenarios in each event.

## 2 State of the art

This section presents first the two types of cybersecurity events used as models (cyber exercises and CTF competitions) and the use of the virtualization technique as an indispensable element to carry them out. Finally the DevOps concept that frames the configuration management tools used in this study is introduced.

### a. Cyber exercises

Based on the definitions provided by the glossaries of official bodies such as the Cooperative Cyber Defence Centre of Excellence (CCDCOE) or National Initiative for Cybersecurity Careers and

Studies (NICCS), we could say that a cyber exercise is an event that aims at evaluating and improving defense capabilities in the field of information technology of a particular community (either military or civilian) through recreation in a controlled environment of a confrontation situation within the scope of information networks.

According to cyber exercises taxonomies prepared by INCIBE (Razvan, Adrien, & Panagiotis, 2015) and (Jason Kick, 2014), the number of cyber exercises performed each year has experienced significant growth as shown by Fig. 1.

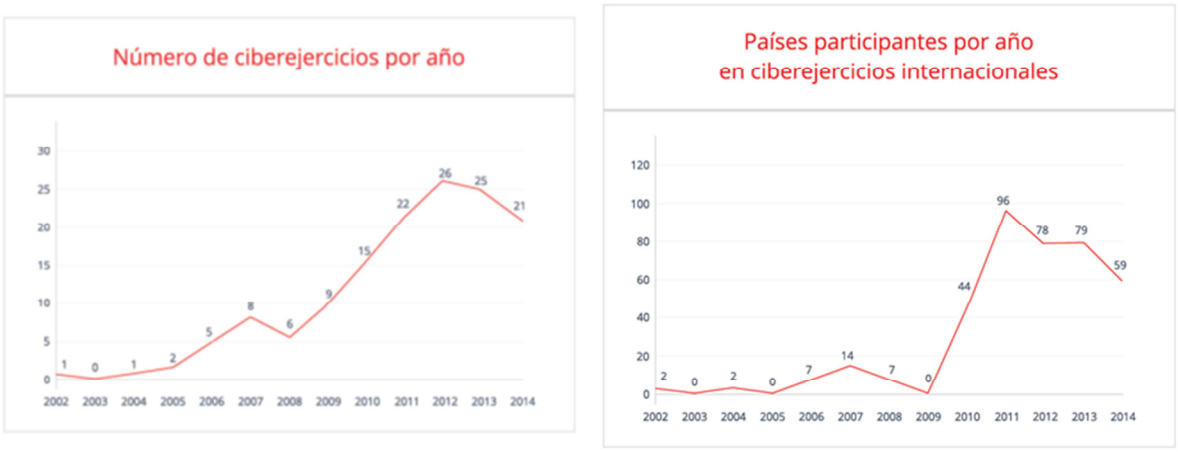


**Fig. 1.** Number of cyber exercises and participants by year (INCIBE, 2015)
[Traducción de la imagen: Number of cyber exercises per year / Countries involved in international cyber exercises per year]1

Most cyber exercises reflect the time when the incident occurs, while others also include before and after phases. The before phase may include securitization of a given infrastructure, while the after phase can focus on analyzing digital evidence that has been produced during the incident in the scenario and generate reports on events occurred and actions taken.

Lastly, a common component to most cyber exercises is the number and type of events (injects) introduced in the scenario during its development. These events come to represent the changing, unstable framework surrounding crisis situations in case of an attack.

Upon analyzing the main features that comprise the context surrounding the organization and execution of an event of this kind, we are going to frame the cyber exercises model to which this work intends to contribute, following the taxonomy described, as summarized in Fig. 2.

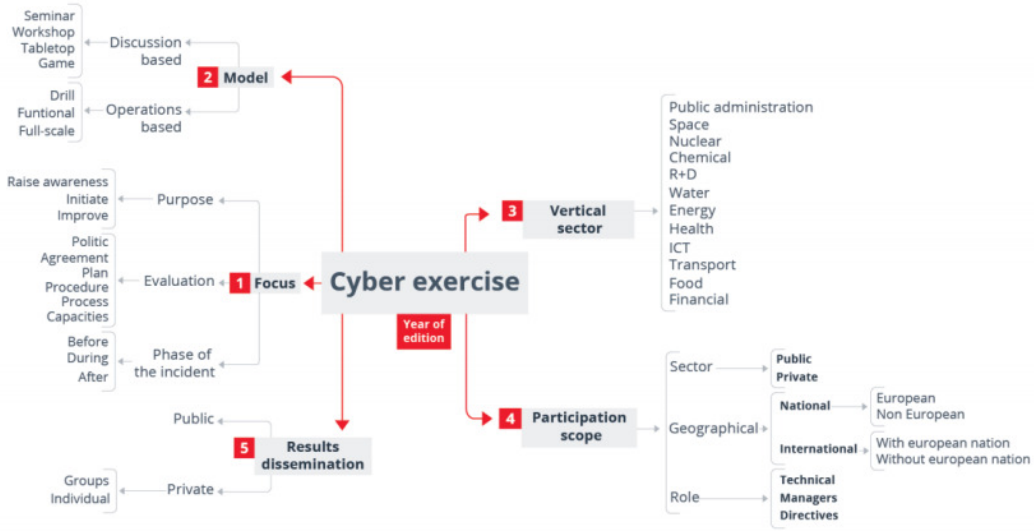


**Fig. 2.** Cyber exercise taxonomy schema (INCIBE, 2015)

### b. Capture the Flag competitions and their teaching applicability

Such events are characterized by being highly technical and lacking, in the vast majority of occasions, a script or story surrounding testing (unlike cyber exercises), focusing on the pursuit of participants' pure technical skill.

Conferences such as DEFCON have been organizing such competitions for more than two decades (DEFCON, 2016); a controlled environment in which participants can measure their knowledge is created to ensure a level playing field. There, techniques, exploits and tools are shared and it is mandatory to document and publicize the method for solving the challenges posed by the competition.

### 2.3 Virtualization

Virtualization is a technique that allows to create virtual versions of a software or hardware resource. It gives us the advantage of isolating the guest system (the virtual machine) from the host system, flexibility to define hardware features of the system (CPU, RAM, etc.), and the ability to restore through snapshot systems; for example, if we infect one system to analyze the behavior of a sample of malware, we can return the system to its initial state.

## 3 Working methodology

The general objective is to reduce the time needed and make the creation of virtual machines used in the practical section of information security teaching flexible. At the same time, the machine created in the VNX virtualization platform are intended to be integrated in such a way that scenarios can be easily deployed.

The specific objectives are to assess the service deployment platform on virtual machines (e.g. Chef, Puppet and Ansible), integration of the chosen tool with VNX platform, development of templates that allow to install vulnerable services or older versions of CMS e.g. Wordpress, Moodle, etc.), development of templates that

allow to install security services, being these firewalls and IDS as a minimum and, finally, development of a testing scenario that serves as demonstrator of developed templates. For instance, seeing how an IDS protects a machine with outdated Wordpress.

As methodology used to achieve the objectives specified in the preceding paragraph, it should be mentioned that there are two big different parts: the preliminary study of existing alternatives in the DevOps ecosystem and the assessment of its applicability within the teaching environments of a laboratory, as well as the completion of a cyber exercise / CTF.

## 4 Identification of requirements

To be able to identify the requirements of the software solution to be offered, we must be clear about the use cases to which we want it to be applied. To do this, we are going to differentiate three different use cases to which we want to apply the advantages of these tools.
4.1 Teaching laboratories: Small scale scenarios normally developed by a single person. They usually contain a moderate number of virtual machines (around 4 to 10) and machines with repeated roles are not common. Each machine plays a clearly differentiated role (attacking machine, victim machine, etc.) through the services they contain. In this case, customization of services and portability of the scenario are the most important features sought after. On the one hand, being able to customize scenarios provides the necessary flexibility to teaching, which as a rule is limited to offering scenarios pre-designed with machines containing vulnerable services. This approach does not contribute all the realism that would be desirable since its similarity with a real production environment in a company is poor. On the other hand, virtual machines are heavy resources, occupying several GB each. This represents an obstacle when sharing this scenario with students so that they can deploy it in the personal computers. Thus, configuration management tools set out the alternative to share only templates that are extremely light files so that the student is able to generate the scenario on their computer based on it.

4.2 Cyber exercises: In this case, we are facing scenarios of considerable size (from 100 to 1000 virtual machines) where the capacity to control the deployment prevails. Part of the scenario usually consists of exact copies delivered to different participating teams, thus faithful reproducibility of the scenario designed along those copies is critical. Due to the volume of machines to be managed, scalability is another aspect of high interest. The solution must be compatible with private cloud virtualization environments because these are usually chosen when deploying this type of events. The variety of host systems is another point to be highlighted; the solution must be able to configure systems on a wide range of operating systems. In this type of events, real industrial control or other infrequent systems in the IT ecosystem may be employed. For this reason, and despite the configuration management tools do not include support for this type of system, the greater the number of supported systems, the more flexibility we will obtain in order to minimize manual configuration of the event. Finally, the solution must not interfere with the proper operation of the exercise. This implies that vulnerabilities or information leaks that give advantage to the team discovering them must not be introduced therein.

4.3 CTF competitions: This use case is the most variable of the three because CTF competitions may vary greatly in scale from one event to another. As mentioned in Chapter 2.2, configuration management tools can only provide an improvement in Attack/Defense-type CTF since, in the Jeopardy type, many of the challenges do not require the use of servers. This type of CTF can be seen as a cyber exercise at a small scale. The number of machines to be handled is generally lower, but some of the requirements are shared such as the capacity to manage and monitor deployed machines, reproducibility of the scenario (copies of the scenario are usually delivered to each team too), or the need not to interfere in any way in the behavior of services deployed. In this case, flexibility in design the scenario is also important, but it is undeniable that in order to offer innovative challenges there will always be a craft component when designing them (although this can be incorporated into the workflow introduced by these tools).
After describing the different use cases, the requirements we can extract from them are as follows:

•Managed flexibility: The platform created must be capable of responding to different challenges but keeping control over the options offered. This means that it must be easy to modify the already designed scenarios to

include new elements or add new challenges to them. At the same time, control must be maintained over the infrastructure so that we can keep track of different versions created for each scenario, develop catalogs or use different variants without incurring added complexity.
•Scalability: The solution must be compatible with the creation of scenarios of various sizes. It must be useful for generating laboratories in workshops or subjects related to cybersecurity (4-10 virtual machines). Yet it must also be a significant help to server configuration in more complex architectures aimed at completing a cyber exercise or CTF-type competition.
•Minimization of tool trace: It must be avoided that the system used disclose information of network infrastructure or underlying services. At the same time, vulnerabilities that may be exploited in the security event and alter its good operation must not be introduced.
•Reproducibility: Performance evaluation of users (whether students or staff of the defense sector) should be conducted on exact copies of environments so as to ensure equality. It may also be interesting for forensic analysis as the exact steps taken by the attacker can be reproduced.
•Portability: The scenarios designed must be able to be deployed in different occasions and varied locations. This implies that a heavy system that entails big storage requirements for migration must not be developed.

# 5 Software development

This chapter will detail the process of developing the software solution to the problem presented. The main objective is to reduce the time for configuration of servers and services deployed therein. Thus, this section is divided in a first analysis of different tools that serves as a basis to identify their qualities and choose one of them. Using such tool, we will propose a series of templates to recreate a small scale scenario that serves as concept test and demonstrator.

5.1 Analysis of DevOps tools. Below we are going to highlight the principal features of the configuration management tools Puppet, Chef and Ansible as well as the virtual machine generator Packer. We will also comment on their main advantages with respect to generation of the scenarios proposed in the environment of cyber exercises / CTF in order to select one of them.

The three tools share most of their features as well as their mission. Consequently, we are dealing with tools that allow to describe in a template (called manifest, cookbook or playbooks, depending on the tool) the state in which we want to leave a machine at the time of deploying it. In addition to the syntax and semantics contributed by such description, the tool is capable of applying such changes on a variety of architectures and operating systems. In applying such configuration, idempotence is guaranteed to ensure that, regardless of configuration, the result will be the same.

5.2 Comparison of configuration management tools. On the one hand, Chef and Puppet need the installation of a software client agent in the host system. Agents play an important role and proper operation of the deployment architecture depends on them. For this reason, both the installation and management of the installed versions of these agents should be part of the action plan when implementing the use of these tools. See Puppet execution cycle in Fig. 3.

On the contrary, Ansible does not operate through a software agent. Instead, Ansible takes advantage of the fact that most of the times machines already have a remote login system (SSH generally). In this manner, commands to be executed by the machine are sent through such protocol, avoiding the use of any added software. This aspect represents a clear advantage with respect to the easy use of Ansible, minimizing the necessary preparation to use it and offering a low entry barrier.

In the case of Chef (see Fig. 4) and Puppet, architecture is composed of a master server that orchestrates configuration deployment to all servers registered against it. This server stores recipes and control the machine inventory it manages. The system administrator only interacts with the master server and never with end-computers. Thus, control can be maintained on a large volume of servers. Generally, recipes are not monolithic

blocks, but fed on other previously designed module recipes. These recipes are generally stored in the master server so that all servers use the same base templates.

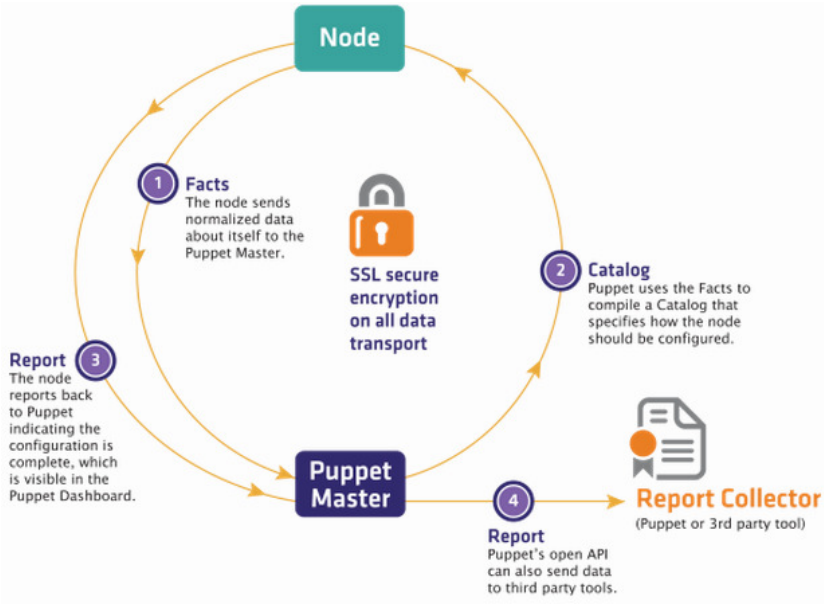


**Fig. 3.** Puppet execution cycle

In this way, it is avoided that different system administrators use different templates for common tasks, maintaining homogeneity in the deployment. Inside of or next to the master server usually is the database that contains information of architecture supplementing templates. This database contains the information that can be used by templates to complete system configuration (e.g. IP address distribution, credentials, etc.) so that its use can be managed centrally.

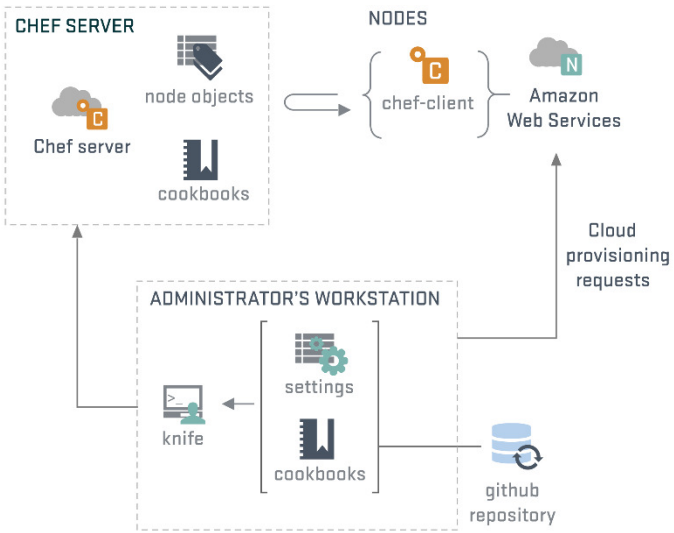


**Fig. 4.** Chef architecture

Regardless of architecture, all tools employ safe protocols when exchanging information among nodes.
•Ansible: Deploys pairs of public and private keys for each node so that they can be mutually authenticated and uses the SSH protocol to maintain confidentiality.

•Puppet: Makes use of a public key infrastructure, employing certificates to protect requests made among the architecture components.
•Chef: Uses the HTTPS protocol to protect messages and employs pre-shared secrets between agent and master server.

Supported platforms
•Chef maintains an updated list of compatible systems, dividing the compatibility of different operating systems into Tier 1 and Tier 2 depending on the degree of support offered. In both Tiers, we find all the most common operating systems (Debian, CentOS, Red Hat, Windows, Ubuntu...) so that incompatible systems are limited to older versions of different operating systems.
•Puppet also documents extensively the platforms it supports and the dependencies it introduces in the system. Like Chef, it supports most of UNIX systems as well as MAC OS X and Windows. Dependencies revolve around the language interpreter Ruby where the tool is developed and a series of binaries that allow it to perform its function properly (base64, md5sum, openssl, syslog, etc.)
•Ansible does not have an exhaustive list of compatible systems, but limits the requirements to the software it needs to operate. Specifically, it needs Python installed and the python-simplejson module. It also needs a way to communicate with the managed node, using ssh, scp or sftp. Windows is also supported from the 1.7 version and has similar requirements to the UNIX environment. The difference is that, in this operating system, communication is achieved through PowerShell remoting.

## 6 Assessment

As a final phase of the project, we will design and create a scenario with the tools we have analyzed in order to assess the level of compliance with the objectives proposed. The objective of this scenario is to capture the result of generating different virtual machines.

Fig. 5 shows a diagram of the proposed scenario.

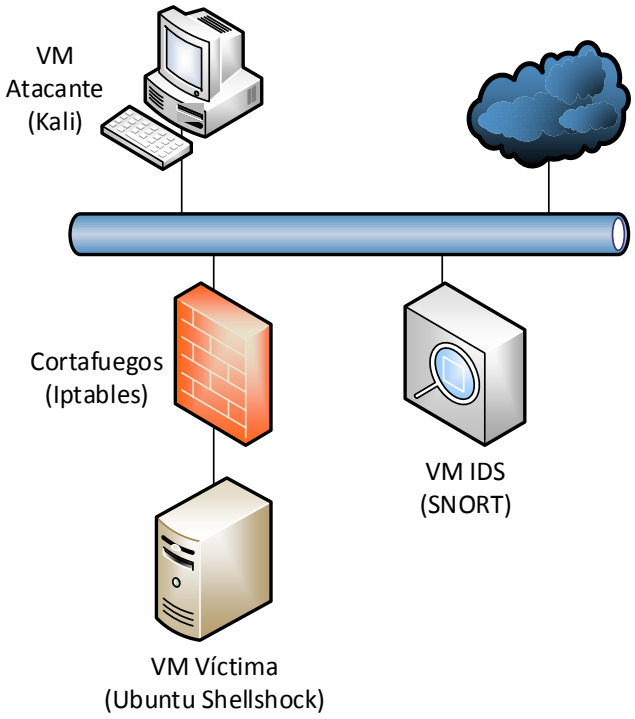


**Fig. 5.** Diagram of the assessment scenario

[Traducción imagen: Attacking VM (Kali) / Firewall (Iptables) / IDS VM (SNORT) / Victim VM (Ubuntu Shellshock)]

In this scenario, we will simulate an attack from the "Attacking VM" to the "Victim VM" using Shellshock vulnerability. Then we will see how we can mitigate this attack, creating a new scenario containing filtering rules on the firewall that reduce the impact of the vulnerability as well as the detection of such attack from the "IDS VM".

The scenario will be deployed using the virtualization platform VirtualBox, using the NAT mode to connect the machines we launch. In using this connection mode, we do not need to perform an extra configuration of network interfaces for the machines included in the scenario. In case of deploying the scenario in a laboratory having a range of IP addresses assigned.

Below we describe the process of generating each machine:
•Victim VM: We will use the Packet template and the manifest described in the section dedicated to vulnerable services.
•IDS VM: We will use the method described in the previous section, being the Packer template identical to that of the victim machine, except for the Ubuntu version used (14.04.03) and that we will allow system updates.
•Attacking VM: In this case, by using a Kali machine that needs no further configuration, we can directly use an image already downloaded from the Internet or use a basic Packer template that simply installs the system without any provisioning.

As we have mentioned, most of the time is spent in installing the operating system; despite this, generation of machines can be parallel (if the computer has enough resources to do so) so we can say that the net overall time is 12 minutes. This measure denotes a clear improvement in the use of this type of tools. The manual process of installing the victim machine conducted for preparing the templates took more than two hours. However, once we have the templates prepared, deployment reduces dramatically as we have seen. At the same time, the configuration has been recorded in a versionable text file in such a way that the possibility of making a mistake during configuration is reduced. The students may receive these files from the teacher and generate the scenario in their own computers in a short time, allowing to improve the learning process.
Continuing the evaluation process, if we launch the machines generated in our virtualization environment and we log in the Kali machine, we can start the simulated attack. To do this, we can use Metasploit with a module dedicated to exploit Shellshock but, because of its easy exploitation, we can run a test using basic tools like curl or wget (or even the browser itself with a plugin that allows us to modify headers).

## 7 Conclusions and future work

In this document, we have analyzed the applicability of tools framed within the DevOps philosophy to the generation of security scenarios. We have also evaluated the potential impact of their use in improving the design, management and deployment of CTF competitions and cyber exercises, identifying their strengths and how we can further contribute to their development.
After conducting the assessment process, we have observed that we are capable of generating reproducible scenarios from light files that can be shared. Times for generation and modification of scenarios allow them to be used by students in their computers, thus improving the quality of practical content that can be offered in workshops related to cybersecurity.
We have also managed to integrate development with the virtualization platform VNX in order to create scenarios compatible with such tool. A disadvantage with respect to the use of VNX that has been set out during the performance of this work is the capacity to control the scenario upon deployment. The solution developed on VNX allowed certain interaction with the scenario from the host machine so that the student can introduce events in it. In our case, these tools focus solely on the generation and configuration processes, so we depend on the opportunities provided by the deployment platforms we use.
The project in turn gives rise to multiple future lines of work, given its heterogeneity. The development of templates represents one of these lines, being able to create specific repositories focused on security teaching in order to share a catalog of scenarios. Some particularly interesting templates would be dedicated to operating

systems that emulate the behavior of routers and switches (Vyos / pfSense) that would provide the scenario with greater flexibility in describing complex network topologies.
In the generation of vulnerable services, there is also room for improvement. One of the initial proposals at the beginning of the project was automatic installation of vulnerable versions of applications such as Wordpress or Joomla. Research conducted during the performance of the project has shown that there are templates dedicated to it, but only the process of installing the application is automated. This implies that there is still a manual process when introducing content in the blog (in the case of Wordpress) or installing plugins. Projects focused on deployment management automation in the Wordpress application such as wp-cli, which allow management from a command line have been discovered. The creation of templates that allow to install this type of projects and manage the installation of plugins would add a side to the variety of vulnerable services.
Another line to be followed is the use of light virtualization technologies (LXC, Docker) that would reduce the hardware requirements necessary to deploy scenarios with a large number of nodes. This approach would be useful in teaching, even though the isolation capabilities this type of virtualization has should be analyzed in detail to avoid the introduction of vulnerabilities in the scenarios.
Finally, the lack of control over the scenario after deployed is another critical contribution. The possibility to instantiate a master server within the scenario to make hot modifications is a very interesting line to introduce interaction with the scenario.

## References


[1] CCDCOE. (2012). Cyber Defence Exercise Locked Shields 2012 - After Action Report. Tallin.

[2] CCDCOE. (2013). Cyber Defence Exercise Locked Shields 2013 - After Action Report. Tallin.

[3] DEFCON. (2016, January 15). DEF CON; Hacking Conference - CTF History. Retrieved from https://www.defcon.org/html/links/dc-ctf-history.html

[4] EMAD. (n.d.). Reseña Histórica - EMAD. Retrieved from http://www.emad.mde.es/CIBERDEFENSA/historia/

[5] HashiCorp. (2015, 12 20). Packer by HashiCorp. Retrieved from https://www.packer.io/

[6] HashiCorp. (2015, 12 21). QEMU Builder - Packer by HashiCorp. Retrieved from https://www.packer.io/docs/builders/qemu.html

[7]iCTF. (2016, January 16). iCTF 2015 Scoreboard. Retrieved from https://ictf.cs.ucsb.edu/ictfdata/2015/scoreboard

[8] INCIBE. (2016, January 10). Retos Individuales Cybercamp 2015. Retrieved from https://cybercamp.es/retos/CTF_individual

[9] Legitimate Business Syndicate. (2016, January 16). Announcing DEF CON CTF 2016 Qualifying Contests. Retrieved from https://blog.legitbs.net/2015/12/announcing-def-con-ctf-2016-qualifying.html

[10] National Initiative for Cybersecurity Careers and Studies (NICCS). (2016, January). Cyber Glossary. Retrieved from https://niccs.us-cert.gov/glossary
reddit. (2016, January 10). /r/netsec's Q3 2015 Academic Program Thread. Retrieved from https://www.reddit.com/r/netsec/comments/3mktwy/rnetsecs_q3_2015_academic_program_thread/

[11] Wiki-like CTF write-ups repository. (2016, January 15). Retrieved from https://github.com/ctfs