

Automating Cyber-Range Virtual Networks Deployment Using Open-Source Technologies (Chef Software)

Ionuț LATEȘ

Military Technical Academy, Bucharest, Romania

latesionut.g@gmail.com

This paper consists in exemplifying the use of open-source technologies along with commercial solutions, to achieve an automated deployment of Cyber Range's virtual scenarios. The size and diversity of an IT network are two important factors affecting the complexity of the security solutions to be applied to secure the network against potential attackers. In addition to all these protection measures, both companies and public institutions having IT infrastructure exposed to the Internet, regularly perform simulations of cyber-attacks on virtual infrastructures, copies of their own networks. Replicating a computer network involves considerable effort, commensurate with its size and diversity. All these aspects being considered, the automation of the virtual network's creation and configuration process, becomes imperative. The issue of automating virtual network deployment has been addressed by many companies and organizations, but only some of these solutions offer open-source components that can be customized and enhanced to best fit the user's needs. A technology that offers the possibility of customizing and re-implementing the components used in the automation process is CHEF [3]. Integrating this technology, a network can be described using standard codes, making it possible to accurately replicate the production facilities.

Keywords: Cyber-Range; Chef DK; open-source network virtualization automation; virtual network infrastructure deployment automation; cyber defence exercise scenario deployment automation.

1. Introduction

The cyberspace continuous expansion causes, inevitably, the constant appearance of new attack vectors that can be used to compromise IT/OT/IoT infrastructures. Virtual networks are being adopted by more and more fields, presenting significant advantages both from an administrative and financial point of view. These networks can be deployed using their own hardware infrastructure or closed cloud infrastructure. The expansion of the computer usage areas determined a significant increase of computer networks, both in the number of connected devices and in the variety of technologies used. Virtual networks deployment represents an important step in the process of cyber-security training. Cyber Range concept represents the idea of a dual software-hardware system to serve the development of virtual test networks, used in cyber-security exercises [1]. It provides all the necessary tools used in cyber-security scenarios development. Automating the process of deploying and managing a virtual network is essential as the duration and volume of the involved resources

considerably decrease. The need for automation is felt even more when the topology and the components of the network are frequently changed (new devices are added or new software applications are installed). Cyber defence training scenarios represent the best example of frequently changed IT infrastructures. It is necessary to ensure a fast and secure process from the perspective of the integrity of the implemented scenarios [2]. One method to automate the process of implementing virtual networks is to use CHEF software technology. Chef is a network configuration management and automation platform, developed by Opscode and available both open-source and commercial. Chef has the basic idea of describing the infrastructure by code. An infrastructure described by code can be automated, tested and easily reproduced [3].

2. Objectives

The main objective consists in the implementation of a proof-of-concept system, a tool capable of performing automated deployment of a cyber-security small training network, using

CHEF software technology [3]. A complementary objective consists in the process of network analysis and detailed description of the installed devices. For each installed system, which is part of the network, a list of particularities will be created, such as: operating system type and version, installed applications/packages, security settings, users, etc. Next, this information will be transposed in descriptive code which will serve as input for the automation tools.

3. Chef Software Design

The Chef descriptive code consists of resources, each defining a single configuration task to be performed:

```
package "nginx" do
  action: install
end
```

Resources are declarative by design, which means that the information transmitted by the code describes the action to be taken on the infrastructure. Chef dynamically performs appropriate actions on the platform. Chef resources are also designed to be continuously running. Through the "test and repair" behaviour, Chef maintains the infrastructure's functionality, making predefined decisions, if the

situation requires it (when the operating parameters go beyond normal). If the configured and monitored systems behave in accordance with the default state definition, Chef only confirms that all configurations are up to date. Otherwise, Chef converges each system to the desired state, allowing the network administrator to continuously apply configuration updates, being confident that the Chef infrastructure will restore normal operating parameters even if updating errors occur [3]. Chef descriptive code is collected in artifact-specific objects called cookbooks [3]. Each cookbook defines a configuration scenario and all the components needed to support that scenario. Most cookbooks consist of:

- **Recipes** - Chef resource lists that describe an end-to-end configuration task.
- **Attributes** - user-defined variables that can be used to change the behaviour of a Chef recipe without having to change the base code.
- **Files and templates** - any static files or templates required for the activity's configuration process.
- **metadata.rb** - a metadata file used to ensure that each cookbook is correctly implemented in each node, providing support for the cookbook version.

```
1 #
2 # Cookbook:: foo
3 # Recipe:: default
4 #
5 package 'httpd' do
6   action :install
7 end
8
9
10 template '/etc/httpd/httpd.conf' do
11   source 'httpd.conf.erb'
12   owner 'httpd'
13   group 'httpd'
14   mode '0744'
15   action :create
16 end
17
18 service 'httpd' do
19   action [:enable, :start]
20 end
21
```

Fig. 1. The structure of a Chef cookbook [4]

Whether the network infrastructure is in the cloud, locally, or in a hybrid environment, Chef automates how the infrastructure is configured, implemented, and managed across the network, regardless of its size. There are three main components of a Chef infrastructure [5]:

- **Chef DK Workstation** - the workstation is used by the users (network administrators tasked to virtual deploy a network) to authorize and test cookbooks, using tools such as *TestKitchen*[6] and interact with the Chef server using the *Knife*[6] and

Chef command line tools.

- **Chef client nodes** - are machines managed by and through Chef. The Chef client is installed on each node and is used to configure the node to the desired state.
- **Chef Server** - acts as a hub for configuration data. The Chef server stores cookbooks, node policies, and metadata that describe each registered node that is managed by the Chef. Nodes use the Chef client to ask the Chef server for configuration details, such as recipes, templates, and file distributions.

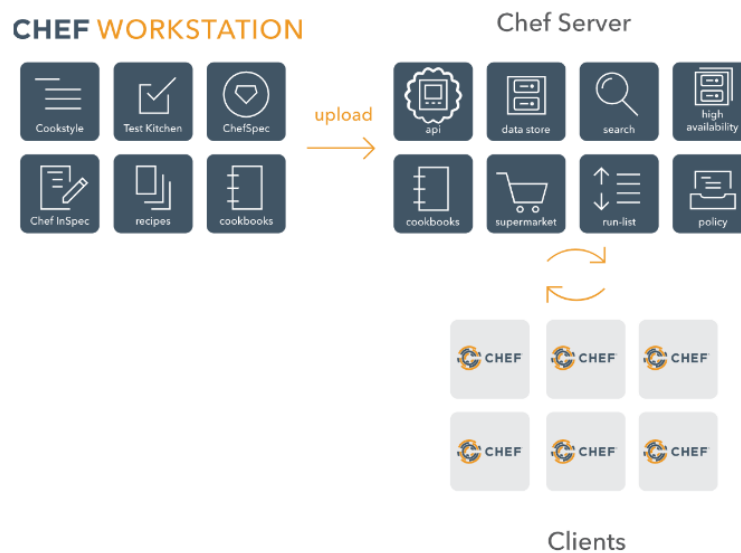


Fig. 2. The structure of a Chef cookbook [7]

These are just some important elements used for this paper's specific proof-of-concept implementation. CHEF offers much more tools and technologies which can be used to automatically deploy virtual networks, regardless of the purpose of the action [3].

4. Practical implementation

The purpose of this practical implementation is to configure, using the technologies described above, a virtual network used in a scenario that is part of a cyber-security exercise. The network consists of:

- a web server running a vulnerable version of *php*, connected to the exercise's simulated Internet,
- 2 workstations containing sensitive

information that an attacker can access after compromising the vulnerable server,

- an investigation station - a Kali Linux system running all the tools needed to investigate a cyber-attack, also used by those who act as investigators to replay the attack scenario.

The number of systems in the implemented network can be much higher. The current example serves as a proof-of-concept and it was performed using a single physical server, with limited hardware resources. Based on the description of the scenario, the roles of each system can be defined as follows:

- webserver - CHEF role: webserver
- workstation1 - CHEF role: workstation
- workstation2 - CHEF role: workstation
- investigator1 - CHEF role: investigator

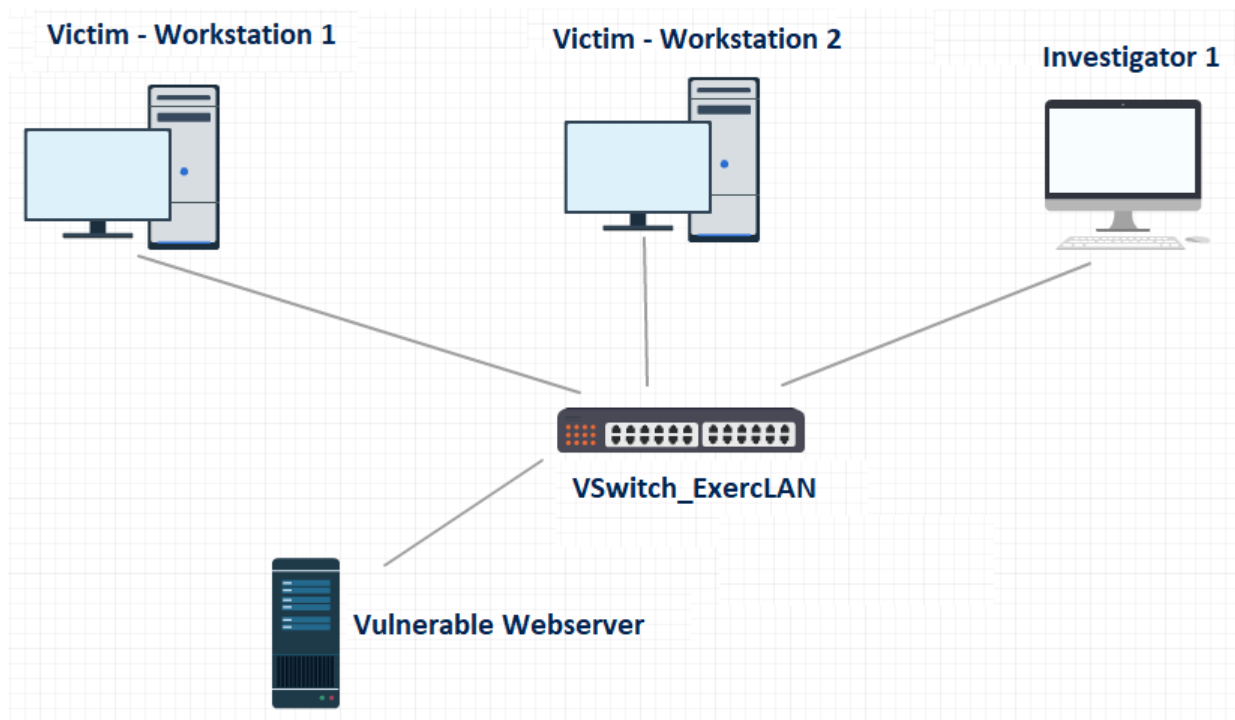


Fig. 3. The topology of the deployed virtual network

Having the roles thus defined, simultaneously specific actions can be taken on all the elements having a certain role.

For the implementation of the above-described scenario, there were used the following physical resources:

- A virtualization server running VMWARE ESXI 6.5. It has the following hardware features:
 - 40 x Logical CPU
 - 32 GB RAM Memory
 - 1 TB SSD Storage
- A workstation running VMware vSphere Client tool, used to manage the server and virtual components.

In addition to the ESXI component, VMware vCenter Server Appliance was installed and configured. This component is designed to integrate multiple physical servers running ESXI so that they are controlled as a single

entity. Also, for the integration with CHEF technology (more specifically, for integration with kitchen [7]) a complex plugin had been developed: kitchen-vcenter [9]. It allows the development of "CHEF based" systems on "private cloud" systems developed with VMware technologies.

From the virtualization component perspective, the following components were configured:

- the VCENTER instance that communicates with the kitchen utility via the kitchen-vcenter plugin (IP: 192.168.10.222)
- the Datacenter component within which the configured "Resource Pool" elements will be configured
- the Resource Pool component within which the elements of the current scenario will be configured.

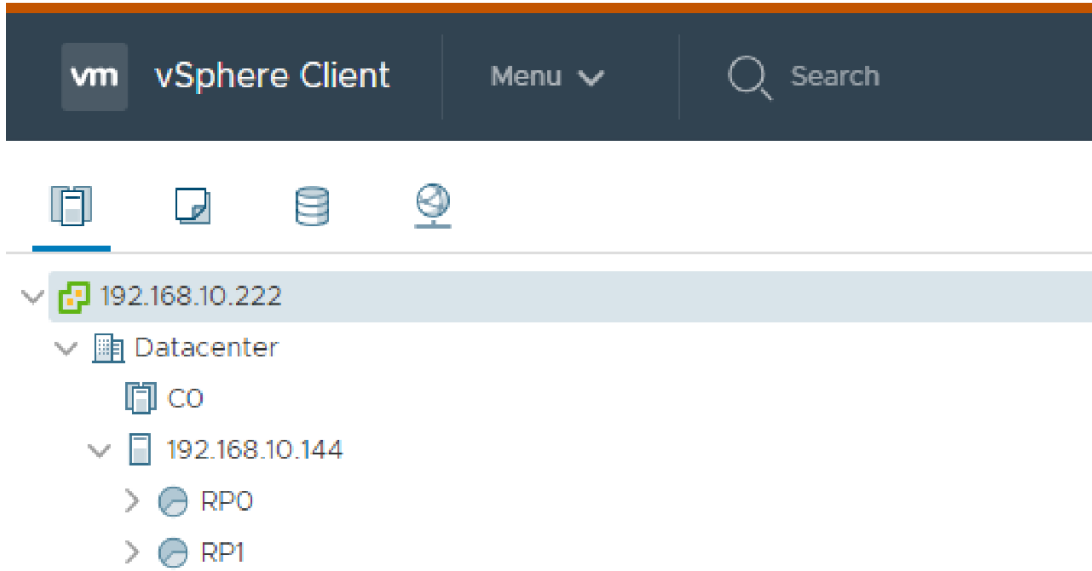


Fig. 4. Resource pool exemplification

Each element of the virtual network will be cloned, configured and customized, starting

from a previously defined template, managed by the VCENTER system:

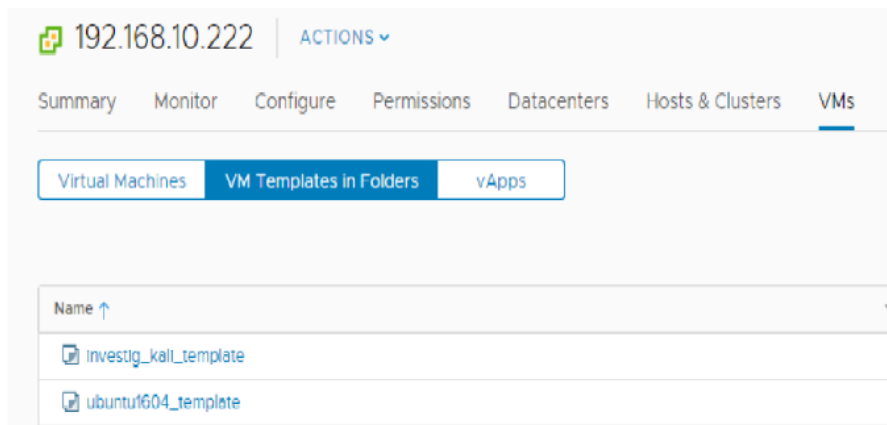


Fig. 5. VMware Vcenter templates exemplification

The vulnerable PHP component is configured through a cookbook referenced in the .kitchen.yml specification file. This is the

main configuration element of the entire virtual network. Each component of this descriptive configuration file is explained below.

```

---
driver:
name: vcenter
vcenter_username:
  'administrator@vcenter.vhost'
vcenter_password: 'vcenter.password'
vcenter_host: '192.168.10.222'
vcenter_disable_ssl_verify: true

provisioner:
name: chef_zero
sudo_command: sudo
deprecations_as_errors: true
retry_on_exit_code:
  - 35
rebooting:
  max_retries: 2
  wait_for_retry: 90

platforms:
- name: webserver1
  driver_config:
    targethost: 192.168.10.144
    template: ubuntu1604_template
    datacenter: "Datacenter"
  transport:
    username: "root"
    password: "password"
- name: workstation1
  driver_config:
    targethost: 192.168.10.144
    template: ubuntu1604_template
    datacenter: "Datacenter"
  transport:
    username: "root"
    password: "password"
- name: workstation2
  driver_config:
    targethost: 192.168.10.144
    template: ubuntu1604_template
    datacenter: "Datacenter"
  transport:
    username: "root"
    password: "password"
- name: investigator1
  driver_config:
    targethost: 192.168.10.144
    template: investig_kali_template
    datacenter: "Datacenter"
  transport:
    username: "root"
    password: "password"

suites:
- name: project_disrt
  driver:
    vm_hostname: net.local
  run_list:
    - recipe[phpapp::default]
  members:
    ["investigator1",
     "workstation1",
     "workstation2",
     "webserver1"]

```

Then, after the full implementation of the configuration (the previously code descriptive section), the specified platforms have already been registered by the kitchen utility, with the

initial status of "<Not Created>" (because the virtual machines related to each platform have not yet been created).

	Driver	Provisioner	Verifier	Transport	Last Action	Last Error
webserver1	Vcenter	ChefZero	Inspe	Ssh	<Not Created>	<None>
workstation1	Vcenter	ChefZero	Inspe	Ssh	<Not Created>	<None>
workstation2	Vcenter	ChefZero	Inspe	Ssh	<Not Created>	<None>
investigator1	Vcenter	ChefZero	Inspe	Ssh	<Not Created>	<None>

Fig. 6. The initial state (0-state)

Next, kitchen tool is used to start the virtual machines creation process:

```
$ kitchen create -c 4
```

Note: kitchen "create" actions can be parallelized - multiple virtual machines can be

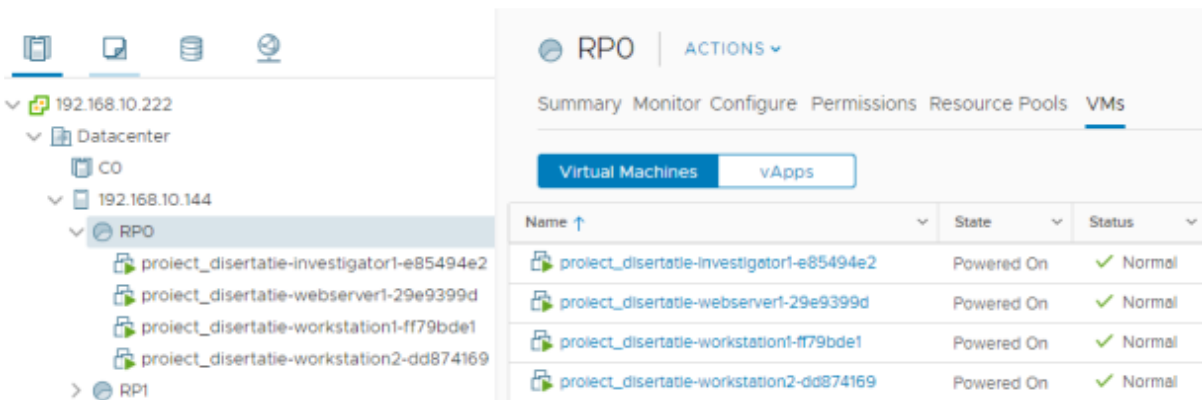
simultaneously created. Current situation describes the usage of 4 threads, as in the specification file had been described only 4 platforms – all platforms will simultaneously be deployed. The speed of creating virtual machines is strictly related to the hardware capabilities of the physical systems used.

```

root@disertatie:/home/chefserver/Desktop/ap2# kitchen create -c 4
-----> Starting Kitchen
-----> Creating <webserver1>...
-----> Creating <workstation2>...
-----> Creating <workstation1>...
-----> Creating <investigator1>...
Cloning the template 'ubuntu1604_template' to create the VM...
Cloning the template 'investig_kali_template' to create the VM...
Cloning the template 'ubuntu1604_template' to create the VM...
Cloning the template 'ubuntu1604_template' to create the VM...

```

Fig. 7. Templates cloning to virtual machines



The screenshot shows the vCenter interface. On the left, a tree view shows the hierarchy: 192.168.10.222 > Datacenter > CO > 192.168.10.144 > RPO. Under RPO, four VMs are listed: project_disertatie-investigator1-e85494e2, project_disertatie-webserver1-29e9399d, project_disertatie-workstation1-ff79bde1, and project_disertatie-workstation2-dd874169. On the right, the 'VMs' tab is active, showing a table with columns: Name, State, and Status. All VMs are in 'Powered On' state with a 'Normal' status.

Name	State	Status
project_disertatie-investigator1-e85494e2	Powered On	Normal
project_disertatie-webserver1-29e9399d	Powered On	Normal
project_disertatie-workstation1-ff79bde1	Powered On	Normal
project_disertatie-workstation2-dd874169	Powered On	Normal

Fig. 8. Created virtual machines

Finally, kitchen tool is used to start the “convergence” process:

```
$ kitchen converge -c 4
```

The "converge" actions can be parallelized so that connections can be simultaneously made with multiple virtual machines. While in the

creation process the parallelization was directly influenced by the hardware component, at this stage the use of simultaneous connections has fewer limitations.

After completing the converging process, the status of the platforms is changed at the level of the kitchen manager (the new status - converged):

```

root@disertatie:/home/chefserver/Desktop/ap2# kitchen list

```

	Driver	Provisioner	Verifier	Transport	Last Action	Last Error
webserver1	Vcenter	ChefZero	Busser	Ssh	Converged	<None>
workstation1	Vcenter	ChefZero	Busser	Ssh	Converged	<None>
workstation2	Vcenter	ChefZero	Busser	Ssh	Converged	<None>
investigator1	Vcenter	ChefZero	Busser	Ssh	Converged	<None>

Fig. 9. Created virtual machines

Starting from this point, the network is fully functional, and the cyber security exercise can begin.

5. Observations, problems and solutions

For the kitchen-vcenter plugin to connect to the newly created virtual machine, the cloning template needs to contain the VMware-tools utility already installed and configured. Failure to meet this condition will lead to the

blocking of the "kitchen create" process, the VMs being created but without being detected by the kitchen utility to take over the new status. This problem is manifested by blocking the process with the message:

"Waiting for network interfaces to become available ..."

Each template used in cloning the platforms described in the kitchen specifications, must have ssh server (Linux platforms) or winrm

server installed (Microsoft Windows platforms)

When creating the kitchen specifications, the root/Administrator credentials must be provided for each platform. Otherwise, the "kitchen converge" process will generate an error of insufficient privileges.

6. Results and conclusions

As the main objective has been met, it can be stated that CHEF technology can be successfully used in creating virtual IT infrastructures, part of cyber security exercises. Practical implementation of the proof-of-concept system was performed on a virtualization infrastructure implemented with VMware technologies (ESXI, vSphere, VCENTER). Considering this aspect, it is important to note that the use of other virtualization technologies involves adapting the solution to the interface component provided by the technology (in this practical implementation, the interface technology is represented by the kitchen-vcenter plugin). The conclusion drawn from the tests is that the open-source CHEF technology fully satisfies the need to automate the implementation of the software component in Cyber Range systems. The implementation methods vary depending on the hardware component and the virtualization technology used.

7. Future work

As future work, the implementation can be tested at a larger scale, including multiple physical virtualization servers, and developing fully complex cyber security exercise scenarios. Also, the solution can be implemented

using different virtualization platforms (other than VMWare). Then, different implementations result can be compared to elaborate a new performant solution.

References

- [1] T. Debatty and W. Mees, "Building a Cyber Range for training CyberDefense Situation Awareness," 2019 International Conference on Military Communications and Information Systems (ICMCIS), 2019, pp. 1-6, doi: 10.1109/ICMCIS.2019.8842802.
- [2] M. Karjalainen and T. Kokkonen, "Comprehensive Cyber Arena; The Next Generation Cyber Range," 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), 2020, pp. 11-16, doi: 10.1109/EuroSPW51379.2020.00011.
- [3] Progress, Chef Documentation, Online, <https://docs.chef.io/>
- [4] Progress, Chef Infra – Cookbook Reference, Online, <https://docs.chef.io/cookbooks/>
- [5] Progress, Chef Infra - metadata.rb, Online, https://docs.chef.io/config_rb_metadata/
- [6] Progress, Chef Infra Overview, Online, https://docs.chef.io/chef_overview/
- [7] Progress, Chef Workstation – Test Kitchen, <https://docs.chef.io/workstation/kitchen/>
- [8] Progress, Chef Infra Images – start_chef.svg, https://docs.chef.io/images/start_chef.svg
- [9] progress, Chef Infra – Chef and VMware, <https://docs.chef.io/vmware>



Ionuț LATEȘ has graduated The Faculty of Communication Integrated Systems and Information Technology, from Military Technical Academy in 2016. He has a master's degree in Information Technology Security (2018) and multiple certifications related to cyber security field. His main fields of interest are cybersecurity, embedded software programming, software programming and machine learning.