Automated CyberExercise and CyberScenario generation framework

Alejandro-David Cayuela-Tudela*, Javier Pastor-Galindo*, Pantaleone Nespoli*
*Department of Information and Communications Engineering, University of Murcia, 30100, Murcia, Spain
{alejandrodavid.cayuelat, javierpg , pantaleone.nespoli}@um.es

***Abstract*—The demand for cybersecurity and cyberdefence experts has increased to unprecedented levels. More qualified personnel are needed, better trained, with more tools, not only technical but also educational, to develop their skills and become the best possible technical corps. The full involvement of instructors in the design of cyberexercises is a very limiting factor. Automating the generation of cybersecurity exercises by reducing the instructor's workload is a key factor in improving the training, as well as the training platforms themselves. In light of the above, this article proposes a conceptual framework for automatic cyberexercise generation. This framework lays the foundation for such a generation from conception to post-execution results analysis, where each generation composed of devices, services, and configurations is unique with a wide range of customisation. In addition, it allows us to refine the generations with human feedback and to deploy cyberexercises in all types of environments.**

***Index Terms*—Scenarios Generation, Cyberexercise Generation, Hands-on Training, Cyber Ranges, Cybersecurity, Cyber-awarness**

Contribution type: *Original research*

I. Introduction

Cyberattack severity is escalating, with many tactics employed, and the resulting damage is increasingly catastrophic. The projections indicate a steep rise in the frequency of such attacks [1].

The effectiveness, quantity, damage, and widespread concern about cyberattacks present a clear reality: We live in an era where cybersecurity and cyberdefence are more important than ever. This alarming reality is a shared concern among users, companies, and countries striving to combat cybercriminals, organised criminal groups, terrorist organisations, and other nations rapidly enhancing their capabilities on the attack side [2].

In this context, the demand for cybersecurity and cyberdefence experts has surged to unprecedented levels, underscoring their indispensable role. While new professionals are brought in annually, their numbers fall short of the requirements [3]. The need of the moment is for more qualified personnel, better prepared and equipped with technical tools and comprehensive training to enhance their capabilities and form the backbone of our security operations [2], [1], [4]

Thus, educational training tools like the increasingly popular Cyber Ranges (CRs) [5], virtualised environments to perform hands-on exercises, are assuming a pivotal role in the cybersecurity landscape. Countries and companies are embracing the development of cybersecurity training platforms. The Sweden's national cyber training facility called CRATE [6] or KYPO Cyber Range Platform (KYPO CRP) developed by Masaryk University [7], [8] is a testament to their effectiveness and growing acceptance for training civilian, professional and military sectors.

On one the hand, the process of designing and deploying cyberscenarios (complete realistic network environments with some vulnerabilities) and cyberexercises (cyberscenario that it is generated to train specific cybersecurity competencies), involves designing a narrative, defining the topology, creating the virtualised devices, configuring them according to the needs of the cyberexercise, associating vulnerabilities, deploying the scenario and checking that everything has been set up correctly, all completely manual. A complex and time-consuming process, affected by various factors that limit its training potential. The scope and skills (competencies) trained depend on the knowledge and abilities of the instructors in charge of designing and manually deploying the cyberexercises [9].

Additionally, the technologies used to develop the CRs enable training for only specific competencies, which are aligned with the limitations of the technologies and the capabilities of the CRs [10]. Furthermore, test beds are often used to carry out cyberexercises, which are static proposals that do not allow students to train again [11]. Moreover, the high involvement of Instructors in the design, static cyberexercises, and the limitations of the CRs do not allow the training to be improved or novel with each iteration of the training deployment, a factor considered key in cybersecurity.

On the other hand, the automatic generation of cyberscenarios currently available is either unrealistic or, at a high level, far from allowing for technical exercises [12], [13]. Professionals should be trained in the most realistic conditions, bringing them closer to reality in controlled environments that allow them to develop the necessary competencies for their day-to-day work.

This article proposes a conceptual framework that lays the foundations for the automatic generation and deployment of cyberexercises, significantly reducing the time between the conception of a cyberexercise and its deployment. In particular, the contributions are as follows:

- The framework covers the automatic generation, deployment, monitoring, and analysis of cyberexercises.
- Each generation is unique, allowing for continuous reuse by students.
- Knowledge and value are extracted from the execution of the cyberexercises, allowing for continuous refinement of the succeeding generations.
- The proposal allows to deploy a cyberexercise into any deployment infrastructure.
- A first implementation of a part of the framework is

proposed by leveraging a novel generation methodology based on the network-as-puzzle concept.

The structure of the rest of the article is as follows: Section II contains an analysis of academic literature aligned with the proposal. The proposed framework is developed in Section III with an analysis of its different modules in their respective subsections. Then, one implementation based on the framework has been carried out in Section IV. Finally, Section V shows the conclusions drawn from this work and future directions for further research.

II. State of The Art

To develop this framework correctly a multidisciplinary analysis has been carried out to obtain a clear vision of the proposal's different contributions to the literature. Different study dimensions include modelling and cybersecurity scenario generation tools, CRs, and related frameworks.

In [14], a complete proposal is made focusing on the modelling and execution of a cybersecurity scenario in CRs. They present a workflow where the instructor must design and specify the cybersecurity scenario using a JSON-based on Domain Specific Languages (DSL). Afterward, the generation is verified, and if it is correct, it can be deployed in the virtualisation system. After the execution of the exercise, an evaluation of the exercise is carried out, and the results are obtained. This proposal aligns with the one presented in this paper, focusing on the modelling and deployment of the exercises, not on their automatic generation.

Additionally, Vulnerban [11] is a framework for modelling and generating cybersecurity scenarios using the "testbeds" previously defined in CyberVAN [15]. Based on Mitre's matrix[1], the network scenario (devices, vulnerabilities, configurations) and the attack steps that should appear in the scenario are defined in advance. In this way, a scenario is extracted using Prolog [16] (Prolog facts) and then validated. If the scenario is correctly structured, the topology, configurations, and an attack flow is generated.

Moreover, CRACK is a tool presented in [17] to model, verify, and test the scenarios of a CR. A Specification and Description Language based on TOSCA [18] is used to model and deploy the scenarios. For the verification of the scenarios, Datalog [19] is used to describe the objectives that must be satisfied for a correct scenario to be considered. CRACK allows the scenario to be automatically deployed and tested for consistency.

Furthermore, Secgen [20] is an open-source tool[2] focused on the generation of virtual machines. Based on a catalogue of vulnerabilities and the specifications of the machine's configuration file (written in a proprietary Extensible Markup Language (XML)), some of them can be randomly selected and then loaded into the virtual machine via Vagrant[3] and Puppet[4].

Also, AiCEF [12] is a framework whose objective is to generate a "cyber security exercise" (CSE) using machine learning techniques. Based on public cybersecurity articles, they use their ontology, Cyber Exercise Scenario Ontology (CESO), to generate graphs of cybersecurity scenarios that could be used as exercises. Similarly, in [13], two previously trained Large Language Models (LLMs) are used to act as a Chief Information Security Officer (CISO) to generate the scenario and a cybersecurity expert to evaluate the scenarios. It is a multi-step generation where the story is generated, the scripts, which are short stories that give realism to the scenario and difficulty to the exercises, the entities that will act in the scenario based on the scripts, the associated infrastructure, and the events that may occur. Once generated, the scenarios are evaluated in a supervised manner by cybersecurity experts. Both proposals obtain a high-level scenario without contemplating the deployment of virtualisation infrastructures or the objective of the training from a more non-technical point of view.

Concerning the analysis of the CRs, in [10], a deep analysis is performed about the generation of cybersecurity training scenarios of the most popular CRs based on 13 key aspects, including Infrastructure Technology, Topology Generation, and Scoring and Reporting. It discusses their strengths and weaknesses, where conclusions are drawn about the limitations of the different CRs. Among the problems discussed are how deployment technology affects the design and development of a cyberexercise, or the difficulty of designing a cyberexercise is partially solved by resorting to exercise catalogues.

After the analysis of the proposals, there are several factors that limit the training capabilities of the tools that are proposed. Specifically, the following key issues have been identified:

- Limitations in automation: Existing solutions struggle to automatically generate and deploy realistic cybersecurity scenarios. They often require pre-defined scenarios or involve high-level exercises that often lack realism.
- Human dependency: Many proposals rely on manual design and deployment by instructors, constrained by their theoretical and technical knowledge, which may limit extensibility and reusability once the exercises are completed.
- Technological constraints: The characteristics of technologies used to the deployment of the cyberexercises allow for a very restricted types of scenarios and cyberexercises.

To address these challenges, this article proposes a novel conceptual framework that enables the automatic generation and deployment of realistic cybersecurity scenarios and exercises with minimum instructor interaction. By leveraging various modules for automatic generation, the proposed solution significantly reduces the time needed for exercise conception and deployment compared to less automated methods. Additionally, the framework's extensibility allows for the deployment of scenarios tailored to specific training objectives in any virtualisation environment.

III. Framework

Creating cybersecurity training scenarios can be described as a very complex task, especially in educational platforms where preparing efficiently realistic environments for developing cybersecurity competencies or skills is mandatory. As

[1]https://attack.mitre.org/matrices/enterprise/
[2]https://github.com/cliffe/SecGen
[3]https://www.vagrantup.com/
[4]https://www.puppet.com/

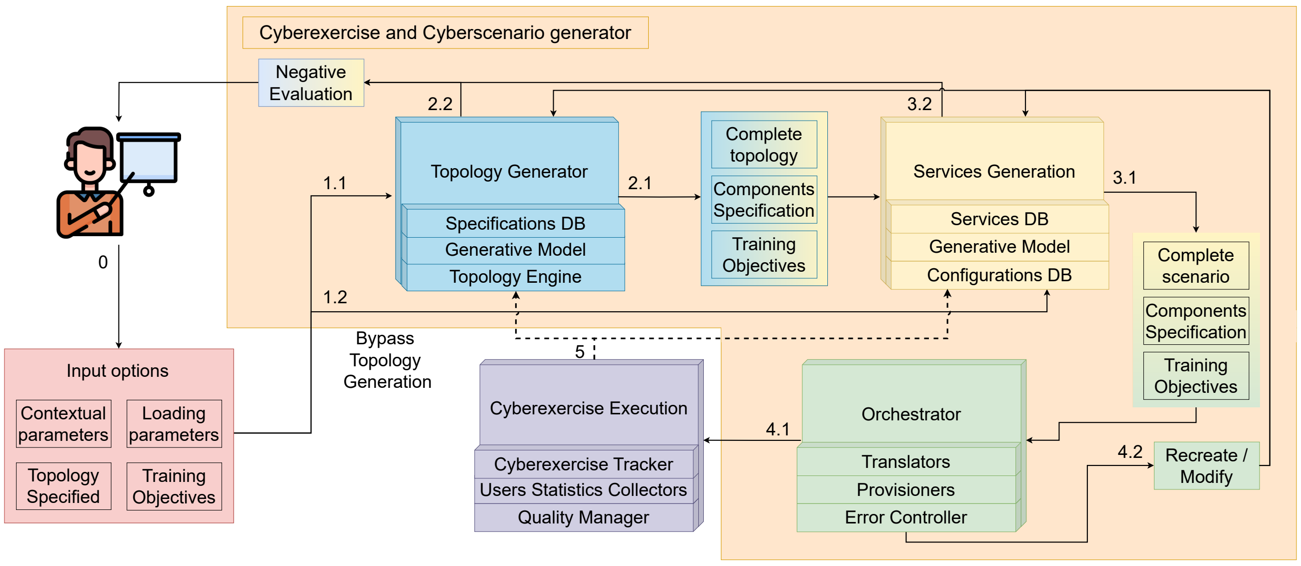

Fig. 1. Framework for automatic generation of Cyberexercise and Cyberscenario

mentioned in the previous Section II, several dimensions and approaches exist to create these cybersecurity scenarios. The framework proposed in this paper is an evolution of this analysis, focusing on bringing a solution by collecting these approaches and providing a unified proposal.

The Fig. 1 shows the proposed conceptual framework. Specifically, it covers the process of creating a cyberscenario and cyberexercise, from conception to post-mortem analysis. As we will see next, the Cyberexercise Execution module could update the framework's various components using the analysis to refine the generations.

A. *Input options*

The Instructor is responsible for designing the cybersecurity training and can indicate various entry options (step 0) that affect the generation to suit the desired type of cyberexercise better. The framework provides the facilities to rapidly generate a complete cyberexercise with minimum effort and a high degree of personalisation.

- **Contextual parameters**: with this option, it is possible to configure the environment of the generations. Values such as topology type (for example, Smart-Home or Smart-Industry), generation complexity (for example, how many network segments are in the topology or more advanced devices as intrusion detection systems), or device density (how many devices are in the topology).
- **Loading parameters**: devices specifications that will be loaded in the Specifications, Services, and Configuration Databases (DBs) within the Topology Generation and Services Generation module and used during the generation. With this options is possible to, for example, define several types of servers with different operation systems and software, which could be used during a web cyberxercise. The Loading parameters allow to inclusion of new possibilities to enrich the generations.
- **Topology Specified**: this feature enables the Instructor to perform and generate a cyberexercise loading a previously specified network topology.
- **Training Objectives**: the instructor could define training objectives. These training objectives will be used to guide the cyberexercises generation. For example, the Instructor wants to prepare a red team cyberexercise where the students take the role of an attacker who has to perform cyberattacks and exfiltrate some information. So the Instructor indicates that the desired cyberexercise should have parts to train, for example "Scanning Networks", "Brute Force Attack", "Privilege escalation" and "Information Exfiltration". With these indications, the cyberexercise generated would allow us to train all these competencies. Also, it is possible not to indicate any objectives. With this option, the output of this option is a generated realistic cybersecurity scenario with some vulnerabilities.

There are two possible flows with the specification of a topology; on the one hand, the Topology Generation updates or modifies the network topology specified (step 1.1); on the other hand, bypassing this phase goes directly to the Services Generation (step 1.2). This second flow is for work around the services using the topology specified.

B. *Topology Generator*

The Topology Generation corresponds to the network topology generation with different devices based on the specification in the Input Options. For instance, the Instructor may require a Smart-Home topology that contains a Network-attached storage (NAS) server, IoT sensors, and smart TVs. The Topology Generator module must include these devices and can also additionally add dynamically other devices, such as laptops, IP cameras, and more, for a better and more realistic cyberscenario.

The Topology Generator comprises the three following components:

- The **Specifications DB** contains the devices that could be used in the network topology and their configurations. In this case, the configurations correspond with the deployment specifications as the type of virtualisation or the operation system.
- The **Generative Model** provides the "intelligence" used during the generation process, such as an algorithm, artificial intelligence, or other methods. It uses the Specification DB to extract devices and their configurations and generates the topology.
- The **Topology Engine** acts as a controller to create and manipulate the network using specific software (NetworkX[5]) and network logic, such as device IPs management.

The output of the Topology Generation is a network topology consisting of multiple devices, each device's complete specification and basic configuration, and the objectives of the cyberexercise (step 2.1), if indicated. If the Instructor deems the generation invalid, a Negative Evaluation can be used to start again generating a new network topology (step 2.2). To improve the generation's quality, the Instructor can send a negative or positive evaluation, if desired, to the generation model for further refinement of the generations.

C. Services Generator

The objective of this module is to endow the realistic network topology with elements that make it a real network environment, such as services, vulnerabilities (either through vulnerable services, misconfigurations, etc.), traffic generators, or network agents, bots that mimic human behaviour on the different devices, and more. The Services Generator also comprises three modules: Services DB, Generative Models DB, and Configurations DB.

- The **Services DB** acts as a repository to extract the services, versions, possible vulnerabilities (and associated cybersecurity competencies and scores), traffic generators, and agents.
- The **Generative Model** guides the inclusion of each pair device service following the objectives (if indicated) to generate a cyberexercise. In other cases, services will be selected according to the method used. It is worth noting that these generative models could be afresh an algorithm, artificial intelligence, or other methods that will provide the "intelligence" to the generation. Service selections will be made by making the best possible association between the Training Objectives and scores within the Services DB.
- The **Configurations DB** contains the configuration of the services. Three different approaches are available: maintaining the configurations of each service in a large database, using configuration templates, and filling them with the desired configuration (such as Jinja[6]), or generating configurations on demand. All these approaches are very useful and extremely complex, especially considering several devices and services.

The output of the Services Generation (step 3.1) is a complete scenario composed of the network topology with devices, their services, and configurations.

After generating the topology and services, the Instructor may express favourable (step 3.1) or unfavourable (step 3.2) opinions. If necessary, the Instructor can review these outputs and start from scratch with a new network topology, work with a specific topology, add specific devices, or modify the generated services. Providing the Instructor with this capability is critical to refining the generation and increasing efficiency, greatly reducing the time required to design a complete cyberexercise. The reduction of the design time and the great flexibility it provides allows the creation of more cyberscenarios, more cyberexercises, and, in short, more training sessions, greatly increasing the educational value of the proposal.

D. Orchestrator

Once the Instructor has given the green light to the cyberexercise or cyberescenario, the Orchestrator steps in with its array of tools, and concretely, these tools are designed to streamline and manage the deployment process, making it easy to deploy the scenario in the desired infrastructure and recover from an error.

It is worth noting that a cybersecurity scenario can be fully virtualised on a virtualisation platform, fully physical in a physical environment, and hybrid if it is a mix of both.

Deploying scenarios can be challenging, particularly in hybrid environments or with diverse devices. To address this issue, the Orchestrator uses Translators and Provisioners to facilitate communication between the scenario and deployment environment. The error controller also ensures that the deployment finishes successfully (step 4.1).

- **Translator**: it prepares the scenario information and settings for the desired Provisioner and deployment infrastructure.
- **Provisioner**: it is used to communicate with the deployment infrastructure and coordinated and managed by the Orchestrator. Particularly, each requires a precise and standardised specification as input, which is then processed to enable the deployment and configuration of devices.
- **Error Controller**: it verifies that the scenario is correctly deployed and configured. This component generates several error checks that the deployment should pass to ensure it is correctly deployed and configured. Also, it monitors the deployment, and in case of an error, it will try to fix it (step 4.2), for example, by redeploying a certain number of times or reconfiguring a device. Additionally, it notifies the Instructor whether it is possible to recover from the error.

To clarify those concepts, let us take two examples to see the Orchestrator in action. In the first example, the scenario and device settings are stored in a JSON file. The Translator takes this unformatted file and generates a new one in a specific format that Terraform[7] (Provisioner) can process. Terraform is an infrastructure-as-code software tool to define and provide data center infrastructure using a declarative configuration language. This new file contains the configuration

[5]https://networkx.org/
[6]https://jinja.palletsprojects.com/en/3.1.x/
[7]https://www.terraform.io/

for the deployment in Proxmox Virtual Environment[8] (PVE) (Deployment infrastructure). PVE is a complete, open-source server management platform for enterprise virtualisation. In the second example, the Instructor wants to deploy a scenario in a hybrid environment with two physical IoT devices and VMware ESXi[9] as a Deployment infrastructure. The Translator analyses the JSON file that contains the scenario specification and generates the necessary files for each Provisioner, in this case, Ansible[10] for the IoT devices and Terraform for the VMware ESXi[11].

As can be seen, these two components play a fundamental role in the scalability of any implementation based on the framework. They allow numerous Provisioners to be brought together through the Translator figure. Adding a new Provisioner is as simple as generating a Translator that produces the formalised entry of the new Provisioner.

E. Cyberexercise Execution Module

The Cyberexercise Execution module is in charge of monitoring the status of the cyberexercise and enhancing the quality of cyberexercise generations through its different components.

- **Cyberexercise Tracker**: it monitors the status of the cyberexercise for each student. For example, if the Cyberexercise comprises five steps, in which step is currently the student, how much time they have consumed during each step, or how many hints has it consumed.
- **Users Statistics Collector**: it records the actions performed by the students during the execution of the cyberexercises, for example, what command the students have used, at what time, clicks, and other desired details.
- **Quality Manager** leverages the statistics and monitoring data to enhance the quality of cyberexercise generations. It reviews topologies, services, and configurations and combines this data with student statistics. Additionally, incorporating an external feedback factor, such as instructor and student satisfaction surveys, adds value by linking performance metrics to the human factor in education systems. Once it has analysed the data it extracts, it assigns marks across objectives, devices, services, and configurations, thereby improving the value of the generation and aligning it with the Instructor's objectives. Finally, if the results are successful, it could update the different DBs with the score that aids in the generation of the scenarios to fit better with the desired training objectives (step 5).

With this module, the status of the cyberexercise can be correctly monitored, and the Instructor's analysis capacity is improved, allowing the creation of new cyberexercises of higher quality and better fitting to the training objectives.

Finally, it is important to note that the privacy concerns of students' data represent a crucial aspects during cyberexercises lifecycle, where sensitive information may be generated and utilised. Although the data generated will depend on the specific implementations and the statistics collected, certain aspects deserves particular attention. Encryption protocols, access controls, and secure storage mechanisms should be employed to protect data generated during cyberexercises. Additionally, anonymisation techniques must be applied to remove personally identifiable information from collected user statistics while retaining their analytical value. Anonymised data can help refine the models used in generations of cyberexercises by providing valuable information on learner behaviour and performance trends without compromising individual privacy. This iterative refinement process ensures that future generations of exercises are continuously optimised to meet evolving training objectives, while maintaining strict data protection and privacy standards.

[8]https://www.proxmox.com/en/proxmox-virtual-environment/overview
[9]https://www.vmware.com/products/esxi-and-esx.html
[10]https://www.ansible.com/
[11]https://www.vmware.com/products/esxi-and-esx.html

IV. Implementation

Once the general framework for generating security scenarios for training has been proposed, one partial implementation is carried out using it, demonstrating that it is possible to develop solutions that produce cybersecurity scenarios. Currently, it is the first step towards full implementation, thus covering the part focused on the realistic automatic generation of network topologies and their subsequent deployment in a virtualisation environment.

A. Generation using Piece-Based Methodology

Network architects generally design network structures, which is a complex process. However, after an analysis of topologies, one could say, to the best of our knowledge, that there is a high degree of repetition of structures and devices in some network environments. To take advantage of this repetition, a new methodology is proposed based on network topologies as puzzles, where these repeated structures are pieces of the complete puzzles.

This methodology, together with our JSON-based modeling, allows us to use an algorithm based on this methodology to act as the Generative Model and easily load content to the different databases defined in the framework. This algorithm will first use the interconnection rules between the pieces to generate an environment and then select each piece, generating first the structure and then devices and services. It also allows new cybersecurity environments to be added easily by identifying the pieces that usually appear in that environment and modeling them to be used by the generation algorithm.

To illustrate this idea, Fig. 2 shows a network topology with different environments and pieces. This figure, has three environments corresponding to Cloud Infrastructure, Smart Home, and Smart Office. Normally, they comprise several interconnected pieces (this can be seen in the Smart Office environment). However, smaller environments such as Smart Home can overlap pieces and environments. Generations using this methodology will iteratively generate first the environments and then the pieces of each environment. In the case of Smart Home, a network topology will be generated with different devices such as laptops, smart TVs, smartphones, and IoT devices. Regarding the Smart Office environment, first, the External DMZ piece will be generated where different public access servers are located, and then the other pieces, such as the Secured Zone piece with the organisation's critical

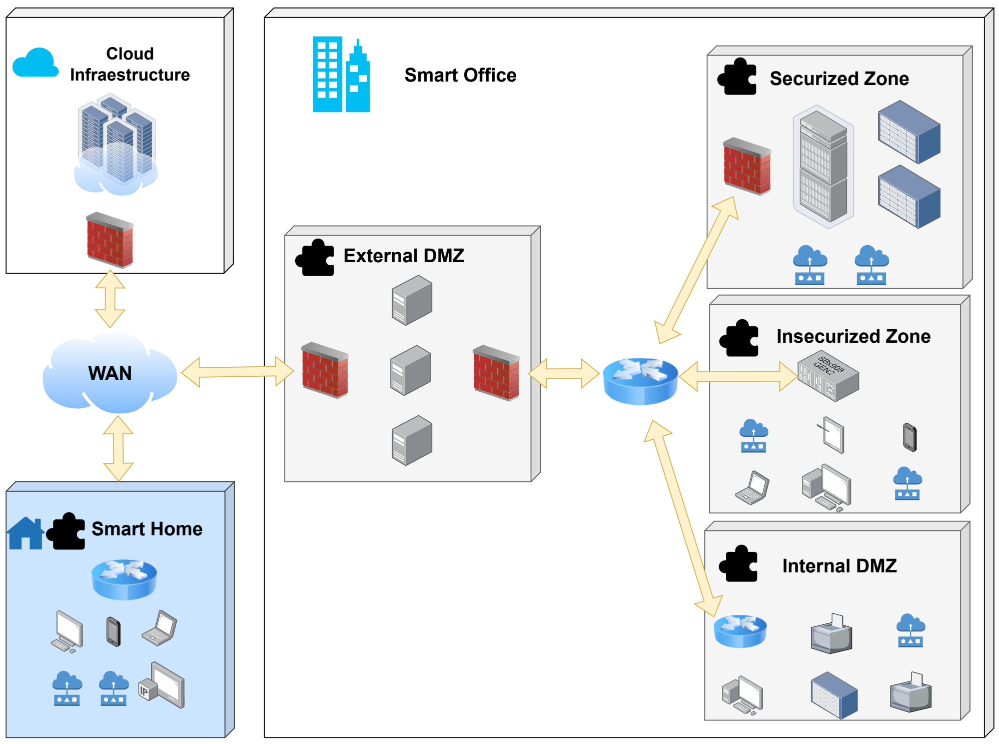

Fig. 2. Example of a high-level topology using the piece-based methodology with three environments and multiple pieces

components, and the Internal DMZ piece where the workers or personal access zone of the organisation's users are located.

A workflow is developed to deploy automatically generated cybersecurity scenarios on the vSphere virtualisation platform using the piece-based topologies and the proposed framework. This workflow comprises three main actors based on the modules described in Section III: Topology Generator, Services Generator, and the logic to deploy the cyberscenarios (that corresponds with the Orchestrator).

The topology is created by the Topology Generator, where, given five input options (defined above), a realistic pseudo-random topology is automatically generated. It serves as input to the Service Generator, where the services and configurations will be selected. From these selections, two JSON files are generated with the topology and services configurations that the Translator will use to prepare the specifications necessary for the Provisioners to deploy topology and services. In this case the Provisioner for the deployment of the devices is Terraform[12] which are compatible with the virtualiser ESXi[13] and its management server vCenter[14]. Corresponding to the provisioning of services, a new Provider has been developed that will use the vCenter API to communicate with the deployed VMs on which services and their configurations will be loaded, started, and executed remotely via Docker containers[15]. Additionally, specific virtual machine templates containing services can increase efficiency and ease of generation. The orchestration of the deployment process is currently done through Terraform, which maintains continuous monitoring of the deployment process and partial error recovery.

It is worth noting that, in our case, we have decided to opt for virtual machine templates and containers to facilitate a partial deployment. However, this approach is not mandatory; other provisioning methods could be considered with a more mature process.

For the topology generation, five input parameters are defined. With these parameters, generating several network topologies with a high degree of particularisation is possible. It is worth noting that the first three are related to the scenario characteristics. These parameters have been selected mainly because of the flexibility and characterisation capacity In this case, both Complexity and Environment will define the characteristics and context of the cybersecurity scenarios. Environment Specification, Tile Specification, and Device Specification will be loaded into the Specifications and Services DBs and will also be used by the generation algorithms (connection rules of the Environment Specification). These last three parameters allow us to add new components and rules to the generation easily; they do not need to be indicated in each new generation, only if the instructor wants to add new devices or rules.

- **Complexity**: it defines density and variety of devices in the topology.
- **Environment**: it indicates the generation's background. In this case, Smart-Home or Smart-Office.
- **Environment Specification**: it contains the available pieces per environment and the rules for their combination, namely, how the different pieces are connected during the generation. Table I shows an example of the Smart-Home and Smart-Office environment specification.

[12]https://www.terraform.io/
[13]https://www.vmware.com/es/products/esxi-and-esx.html
[14]https://www.vmware.com/es/products/vcenter.html
[15]https://www.docker.com/

- **Tile specification**: it contains the devices that may appear in the piece and describes its internal structure, i.e., the order in which the components could b generated. The instructor can indicate new pieces or use previously available pieces. This is used if new pieces are to be added.
- **Device specification**: it categorises the devices' characteristics and their operational dependencies with other devices, such as an IP camera and its control display monitor. Like the tile specification, the instructor can indicate new types of devices or use previously available ones. This is used if new devices are to be added.

It is worth noting that training objectives have been left out of the input parameters in this first partial version mainly because of the complexity of the decision-making in the generation algorithms. Moreover, the lack of standardisation of cybersecurity competencies does not help to develop an efficient algorithm or model for generating cyberexercises.

Environment	**Complexity**	**Available pieces**	**Combinatory**	**Rules**
Smart-Home	Easy	Smart-Home	EP ->SH	1. Entry Point
		Entry point		2. Smart Home
Smart Office	Normal	Entry Point	EP ->DMZ	1. Entry Point
		Demilitarised Zone	DMZ ->OZ	2. Demilitarised Zone
		Office Zone	DMZ ->SZ	3. DMZ ->aleatory
		Securized Zone		4. DMZ ->aleatory

TABLE I
ENVIRONMENT SPECIFICATION EXAMPLE OF SMART-HOME AND SMART-OFFICE

Three components, responsible for network management, network logic (network software), and generation corresponding to the framework's definition, have been developed for implementing the topology generator module.

- **Topology Engine component**: it obtains the networks and subnetworks of the routers and the IP addresses for the devices. In addition, it maintains control so that no network or IP address is repeated. It is also in charge of the management of the network software (in this case, NetworkX[16] is used).
- **Specifications component**: it allows loading devices specified in a JSON file to be used during generation. It also contains the representation of devices that may appear within the pieces in JSON format using the structure of the piece and device specification tables.
- **Generation Algorithm component**: it contains all generation-related functionality. It selects the devices read from the input JSON file, processes the specification by extracting the devices by piece and complexity, and generates a list of the devices that will appear in the topology.

Fig. 3 shows an example of a smart-home topology generation using this methodology. The output of the generation is a realistic topology specified in JSON. It contains the definition of each device used to deploy the topology in the VMware ESXi virtualisation platform. The JSON specifications will be translated into Terraform format to able the Terraform technology (Provisioner) to connect and correctly deploy the topology. The developed Provisioner will connect to each VM to load the configurations and deploy the Docker containers through the vCenter API.

[16]https://networkx.org/

V. Conclusions and Future Work

Over the last two decades, the importance of cyber security has been gaining global prominence. All sectors of society are becoming increasingly concerned about the rise in attacks and damage caused by cybercriminals [21].

It is crucial to have adequate cybersecurity training tools to simulate realistic network configurations and real situations of attacks and vulnerabilities in a controlled environment to provide a clear vision and quality training regarding different environments and cybersecurity competencies [22].

These training tools have shortcomings, especially in their ability to automatically create and deploy realistic exercises. Some proposals are automated; however, the cyberexercises must be defined in advance, the generations are poor and unrealistic, or the generation scope does not allow technical cyberexercises to be deployed. In addition, some factors limit the capabilities of these training tools. Either the trainer has to manually design and deploy the scenarios (a time-consuming and error-prone task), or the pre-designed cyberexercises do not allow for re-use once a learner knows the scenario or the technology of the tools themselves limits the range of possibilities for deploying the cyberexercises.

This article proposes a framework that lays the foundations for the automatic generation and deployment of cyberexercises. The definition of various input parameters, where, for example, the factors of the cyberscenario environment or the specification for loading devices and preconfigurations are defined, allows for a high degree of customisation and realism of the generations. The generation modules and their components have content repositories used by the generative models to make the scenarios realistic, greatly reducing the instructor's involvement. Deployment, error control, and monitoring in different infrastructures are allowed, achieving great extensibility in the proposal and overcoming the constraints of technology in the training capabilities. Finally, a quality system allows extracting knowledge from the generations and refining the databases and generation models, providing a basis for continuous improvement for each training. Additionally, one framework implementation has been presented, using a completely new methodology based on topologies as puzzles and a partial implementation based on the framework achieving a deployment of a network topology generated automatically and deployed on a virtualisation platform.

From this proposal, several lines of research were generated. We are performing a test about the generation of cyberescenarios using LLMs; however, further refining is necessary. Another line is to generate cyberexercises using specific trained LLMs. Also, the methodology based on the puzzles must be further developed; it is necessary to explore new generative models. Finally, once an implementation has been completed, a study will be conducted through real tests with students.

Acknowledgment

This work has been partially funded by the strategic project CDL-TALENTUM from the Spanish National Institute

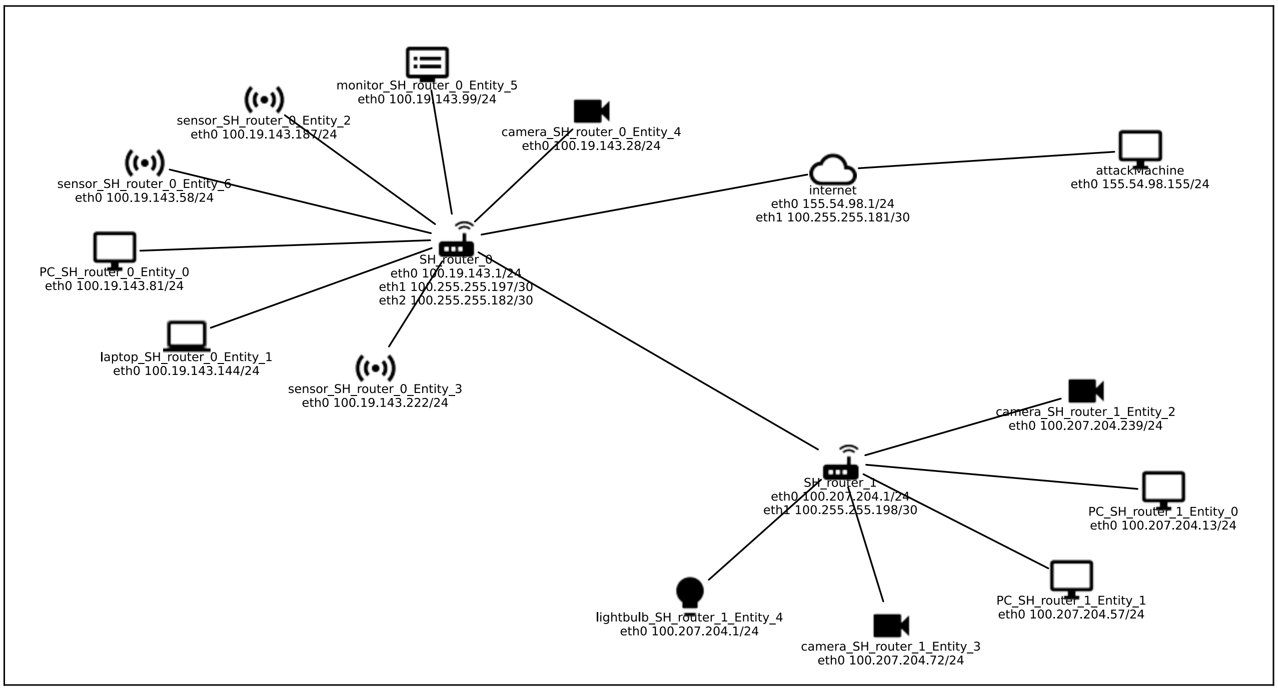

Fig. 3. Example of the output of a generated Smart-Home using the piece-based methodology

of Cybersecurity (INCIBE) and by the Recovery, Transformation, and Resilience Plan, Next Generation EU, from the postdoctoral grant Margarita Salas (172/MSJD/22), as well as project PID2021-122466OB-I00 (Spain), funded by MCIN/AEI/10.13039/501100011033/FEDER (Spain)..

References

[1] I. N. d. C. Incibe, "(en) 2022 cybersecurity review infographic," Mar 2022. [Online]. Available: https://www.incibe.es/sites/default/files/paginas/que-hacemos/cybersecurity_balance_2022_incibe.pdf

[2] Google and T. cocktail analysis, "Panorama actual de la ciberseguridad en españa," online, 2019.

[3] I. N. d. C. Incibe, "Análisis y diagnóstico del talento de ciberseguridad en españa," Apr 2022. [Online]. Available: https://files.incibe.es/incibe/talento/INCIBE_InformeCompleto_DIAG.pdf

[4] Deloitte, "Global future of cyber survey 2023," online, 2021.

[5] R. Petersen, D. Santos, M. C. Smith, K. A. Wetzel, and G. Witte, "Workforce Framework for Cybersecurity (NICE Framework)," NIST, Tech. Rep., 2020. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-181r1.pdf

[6] T. Gustafsson and J. Almroth, "Cyber range automation overview with a case study of crate," in *Secure IT Systems*, M. Asplund and S. Nadjm-Tehrani, Eds. Cham: Springer International Publishing, 2021, pp. 192–209.

[7] J. Vykopal, P. Čeleda, P. Seda, V. Švábenský, and D. Tovarňák, "Scalable learning environments for teaching cybersecurity hands-on," in *2021 IEEE Frontiers in Education Conference (FIE)*, 2021, pp. 1–9.

[8] J. Vykopal, R. Ošlejšek, P. Celeda, M. Vizváry, and D. Tovarňák, "Kypo cyber range: Design and use cases," in *International Conference on Software and Data Technologies*, 01 2017, pp. 310–321.

[9] M. Karjalainen, T. Kokkonen, and S. Puuska, "Pedagogical aspects of cyber security exercises," in *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2019, pp. 103–108.

[10] A. Grimaldi, J. Ribiollet, P. Nespoli, and J. Garcia-Alfaro, "Toward next-generation cyber range: A comparative study of training platforms," in *European Symposium on Research in Computer Security*. Springer, 2023, pp. 271–290.

[11] S. Venkatesan, E. A. Newcomb, B. Hoffman, N. Buchler, J. A. Youzwak, S. Sugrim, C.-Y. J. Chiang, A. Poylisher, M. Witkowski, G. Walther, M. Wolberg, and R. Chadha, "Vulnervan: A vulnerable network generation tool," in *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*. IEEE Press, 2019, p. 1–6.

[12] A. Zacharis and C. Patsakis, "Aicef: an ai-assisted cyber exercise content generation framework using named entity recognition," *International Journal of Information Security*, vol. 22, no. 5, pp. 1333–1354, 2023.

[13] M. M. Yamin, E. Hashmi, M. Ullah, and B. Katt, "Applications of llms for generating cyber security exercise scenarios," 2024.

[14] M. M. Yamin and B. Katt, "Modeling and executing cyber security exercise scenarios in cyber ranges," *Computers & Security*, vol. 116, p. 102635, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404822000347

[15] R. Chadha, T. Bowen, C.-Y. J. Chiang, Y. M. Gottlieb, A. Poylisher, A. Sapello, C. Serban, S. Sugrim, G. Walther, L. M. Marvel, E. A. Newcomb, and J. Santos, "Cybervan: A cyber security virtual assured network testbed," in *MILCOM 2016 - 2016 IEEE Military Communications Conference*, 2016, pp. 1125–1130.

[16] J. Wielemaker, T. Schrijvers, M. Triska, and T. Lager, "Swi-prolog," *Theory and Practice of Logic Programming*, vol. 12, no. 1-2, pp. 67–96, 2012.

[17] E. Russo, G. Costa, and A. Armando, "Building next generation cyber ranges with crack," *Computers & Security*, vol. 95, p. 101837, 2020.

[18] OASIS, "Oasis topology and orchestration specification for cloud applications (tosca) tc." [Online]. Available: https://www.oasis-open.org/committees/tosca/

[19] S. Ceri, G. Gottlob, L. Tanca, S. Ceri, G. Gottlob, and L. Tanca, "Syntax and semantics of datalog," *Logic Programming and Databases*, pp. 77–93, 1990.

[20] Z. C. Schreuders, T. Shaw, M. S.-A. Khuda, G. Ravichandran, J. Keighley, and M. Ordean, "Security scenario generator (secgen): A framework for generating randomly vulnerable rich-scenario vms for learning computer security and hosting ctf events." in *ASE@ USENIX Security Symposium*, 2017.

[21] W. E. Forum and Accenture, "Global cybersecurity outlook 2024," Jan 2024. [Online]. Available: https://www.weforum.org/publications/global-cybersecurity-outlook-2024/

[22] M. Zwilling, G. Klien, D. Lesjak, Ł. Wiechetek, F. Cetin, and H. N. Basim, "Cyber security awareness, knowledge and behavior: A comparative study," *Journal of Computer Information Systems*, vol. 62, no. 1, pp. 82–97, 2022.