Generating Virtual Scenarios for Cyber Ranges from Feature-Based Context-Oriented Models: A Case Study

Pierre Martou
ICTEAM, UCLouvain, Belgium
pierre.martou@uclouvain.be

Kim Mens
ICTEAM, UCLouvain, Belgium
kim.mens@uclouvain.be

Benoît Duhoux
ICTEAM, UCLouvain, Belgium
benoit.duhoux@uclouvain.be

Axel Legay
ICTEAM, UCLouvain, Belgium
axel.legay@uclouvain.be

ABSTRACT

A cyber range is a virtual training ground for security experts. Trainees are separated into attacking and defending teams, whose roles are either to compromise or to protect some critical infrastructure. As reuse of a same scenario may significantly reduce training efficiency, recent research proposed to automate the process of defining and deploying arbitrarily complex cyber range scenarios through the use of a *virtual scenario description language* (VSDL). However, it remains a challenge to generate VSDL scenarios dynamically, i.e. in an adaptive manner, to avoid having to redefine new VSDL scenarios for each new situation. Moreover, existing VSDLs often consider limited contextual information (e.g., only the virtualization budget) and do not link explicitly the vulnerabilities of their scenarios together, which prevents from proposing scenarios with more advanced cyber security exploits. In this vision paper, we rely on feature-based context-oriented modelling to generate relevant cyber range scenarios from an explicit user profile and exploits described in attack-defence trees. This result has high industrial potential, as it could enable a kind of on-demand cyber range scenario generation service.

KEYWORDS

context-oriented modelling, generation of cyber range scenarios, virtual scenario description languages, attack-defence trees

ACM Reference Format:
Pierre Martou, Kim Mens, Benoît Duhoux, and Axel Legay. 2022. Generating Virtual Scenarios for Cyber Ranges from Feature-Based Context-Oriented Models: A Case Study. In *Proceedings of COP 2022: International Workshop on Context-Oriented Programming and Advanced Modularity (COP'22).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

A *cyber range* is a virtual training ground to train security experts. Typically, trainees are split into attack and defense teams, whose roles are respectively to compromise and to protect some valuable information or infrastructure. Such infrastructure is composed of various potentially vulnerable assets including computers, sensors, software systems and versions, and network topologies. Such a system, which represents the configuration scenario of a cyber range, is often made accessible via the Infrastructure as a Service (IaaS) virtualisation paradigm [MS14].

Existing cyber ranges such as the NATO lock shield range[1] host only a few and very specific virtual configuration scenarios. This limits the usage of the range as scenarios may not always match the needs of trainees. This situation can be explained because the first ranges did not use virtualisation technology but real hardware and therefore situations that are not easily reconfigurable or expandable.

Virtualisation principles made it possible to make a large number of assets accessible in a flexible and efficient way. Researchers built upon this technology to define *virtual scenario description languages* [CRA20]. VDSLs are domain-specific languages to specify interactions between virtualised assets and hence sets of scenarios. Such languages allow to represent sets of scenarios that share commonalities (topology, type of assets, range of IP addresses, type of computer), similar to how feature models would represent the commonalities of a product line. It is known that VSDL specifications can be reduced to first order logic formulas [BCD+11]. From such a specification, one can then instantiate a particular scenario that matches a specific user virtualisation budget (number of computers, software version, ...). Infrastructure-as-a-Service (IaaS) is finally used to deploy the range and the actual training can then start.

Existing VSDL languages still suffer from practical limitations, however. A first one is that they only take into account the virtualisation budget to extract particular scenarios. Ideally, we should also take into account other contextual information of the trainees such as their knowledge or their availability. This could trigger for example the deployment of an older version of a software system for which there is a more easily exploitable vulnerability than in the latest version. A second problem is that although VSDLs allow linking vulnerabilities to language components, they do not allow linking vulnerabilities together and therefore exploiting combinations of them to achieve more advanced cyber security exploits. Costa et al. [CRA20] suggested that the latter could be achieved by introducing *attack-defence trees*, i.e. a tree representation for describing such combinations of exploits. Leafs of such trees represent attack steps, i.e., vulnerability exploitation. Intermediary nodes represent combinations of such steps while the root represents an exploit. Attack trees can be translated to Boolean logic formulas.

[1] https://ccdcoe.org/exercises/locked-shields

COP'22, June 7-9 2022,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

To offer a solution to the two above limitations of current VSDLs, we propose a dynamic generation of virtual scenarios which takes inspiration from *feature-based context-oriented modelling* [DMDL19, DMD19]. Stemming from the domain of context-oriented programming, feature-based context-oriented modelling was conceived to address the problem of describing separately the contextual information (contexts) that could affect a dynamically adaptive software system, from the behavioural adaptations (features) triggered by such contexts. A *context model* and *feature model* express separately the contexts and features in terms of hierarchical tree structures. Both models are linked through a *context-feature mapping*, which describes what sets of contexts trigger what sets of features. All this can be reduced to a set of first-order logical formulas [MMDL21].

The main contribution and insight of this vision paper is that feature-based context-oriented modelling concepts can be exploited to extend VSDLs with user profile and attack trees. The idea being to let the context model represent the user profile and the attack tree, while the feature model is represented through a VSDL. The mapping can then be used to instantiate a specific VSDL scenario. As an example, the mapping could associate leaves of an attack tree to assets that contain such vulnerabilities. In case there is no mapping between the tree and the VSDL specification, one could conclude that the corresponding scenario is not appropriate for the user. Otherwise, a concrete scenario containing the necessary vulnerabilities is generated and the training can start.

The remainder of this paper is structured as follows. Section 2 introduces and illustrates the notions of feature-based context-oriented modelling, cyber ranges and attack-defence trees. Section 3 presents and explains our adapted modelling approach in detail, through each contribution and our case study. The case study that will be used to illustrate our approach is that of an SDN-based network. Section 4 shows the overall workflow of how to create a cyber range configuration scenario using our approach. Section 5 concludes the paper and summarises its main insights.

2 BACKGROUND

In this background section we introduce the underlying principles and concepts of feature-based context-oriented modelling (2.1), cyber ranges (2.2) and attack-defence trees (2.3).

2.1 Feature-based context-oriented modelling

Context-Oriented Programming (COP) languages [CH05, HCH08, GCM+11, SGP11] help developers build context-aware applications. Based on contextual information sensed from the surrounding environment, COP programs adapt to their most appropriate behaviour at runtime. *Feature-Based Context-Oriented Programming* (FBCOP) [DMD19, DMDL19] is a particular class of COP. It combines Context-Oriented Programming, Feature Modelling [KCH+90] and (Dynamic) Software Product Lines (DSPL) [HT08, ACF+09, COH14, MCHK17] into a single and unified software development approach [Duh22].

Both at the modelling and implementation level, FBCOP programs separate explicitly the different contexts to which the context-oriented program can adapt, from the behavioural adaptations it exhibits depending on these contexts. A context model describes the various *contexts* (noise level, localisation, weather) that represent the surrounding environment in which the program runs. A feature model describes the possible behaviours (features) of the program. Finally, a context-feature mapping expressed in terms of a set of relations from the context model to the feature model describes what contexts trigger what features, in order to adapt dynamically the behaviour of the running system to those contexts.

Figure 1 illustrates the context, feature and mapping model of a simple FBCOP program, i.e. a smart messaging application that can adapt or refine some of its behaviour at runtime according to some contexts. The system's core functionality consists of *Sending* or *Receiving* messages to and from other users.

Depending on the *User Availability* and the ambient *Noise* level, different *Notification* mechanisms can be used when receiving a message. E.g., the messaging application will stay *Mute* if the user is *Occupied*, or when in a *Quiet* environment like a library. If the ambient environment is *Loud* and the user is *Available*, then the Notification mechanism is switched to *Vibration*, as the user would not hear an *Alarm*.

In order for an FBCOP application to adapt its behaviour at runtime to detected context changes, Duhoux et al. proposed a dedicated software architecture [DMD19]. Figure 2 depicts the workflow of such an architecture, describing how the system is deployed for each new situation. Let us explain this workflow based on the example of our smart messaging application.

First, from contextual information gathered about the surrounding environment (*context data*) and taking into account the constraints imposed by the *context model*, the *context activation* produces a particular *context model configuration* (i.e., an instantiation of the context model where each context is set to be either active or inactive). An example of such a valid configuration is one where the *User Availability* is set to *Available*, the *Noise* level is *Normal*, and the device used has a *AudioCard* but no *VideoCard*.

The second step is finding a corresponding *feature model configuration*. While there are multiple ways to achieve this, the one closest to the workflow that we will present in section 4, would be to make use of a SAT(isfiability) solver. SAT solving can be used to verify the validity of a partial configuration of a feature model (i.e. where part of the features are set to active or not, but not all).

Starting from a partial configuration, i.e., the context model configuration, the *feature selection* will make use of a SAT solver to find a complete and valid configuration of the entire feature-based context-oriented model described by the context model, feature model and context-feature mapping. This was shown to be possible in our previous work [MMDL21]. Continuing our example, the SAT solver would find that we need to activate the *Alarm* feature, as well as the *Vocal* message type. The *Photo* feature will be left inactive, as the device does not have a *VideoCard*.

Next, the *Feature activation* will activate and deactivate the features of the new configuration. This step will also aim to ensure the configuration of the feature model is valid. This step is not needed when using a SAT solver since the solver is able to guarantee the configuration of the feature model. In case the model is invalid, or if the SAT solver would be unable to find a valid configuration, we would need to modify the context model configuration so that it could lead to a valid feature configuration.

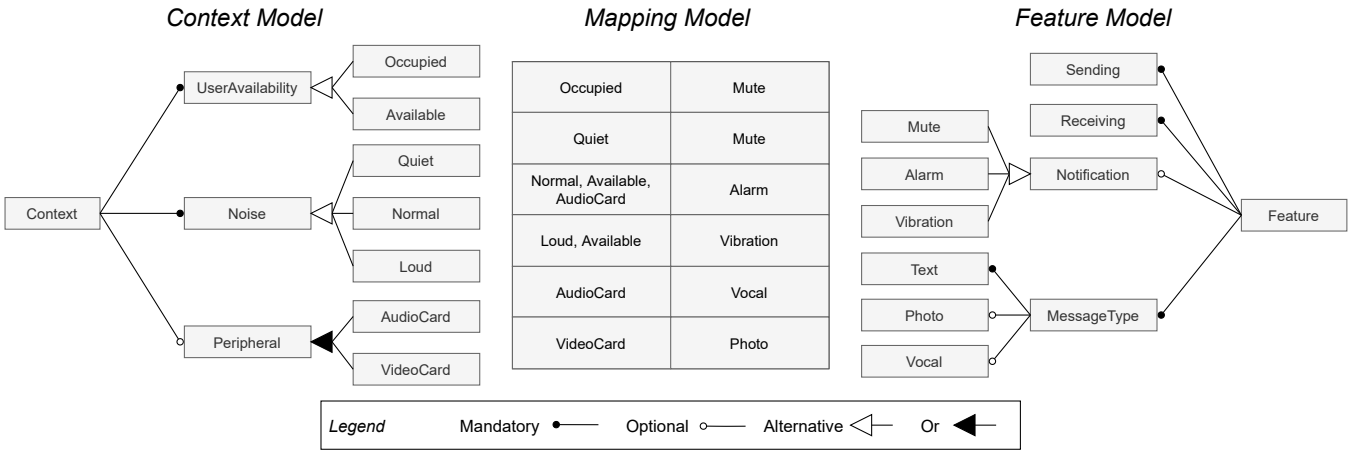

Figure 1: Context-feature model of a simplified messaging application (full version in [MMDL21])

Finally, once a valid feature model configuration is obtained, it will be deployed in the context-oriented system (by adding all the activated features to the running system and removing the deactivated ones). How this deployment process is implemented is out of the scope of this paper; for more information see [DMD19].

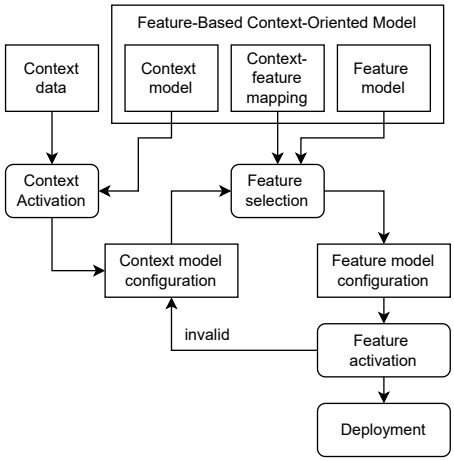

Figure 2: Workflow of the deployment of a feature-based context-oriented system.

2.2 Cyber range configuration scenarios

The goal of cyber ranges is to train future security experts through attack/defense strategies on virtualised infrastructure called configuration scenarios. Such scenarios contain elements such as servers, computers, smartphones, etc. Typically, the features needed to create a complete cyber range configuration scenario belong to the following categories [CRA20]:

- Networking: existing channels and connections, active firewalls;
- Hardware: hardware capabilities of the elements, roles;
- Software: operating system running on the nodes, applications and libraries installed;
- Data: information stored on the nodes, relevant files (e.g. sensitive information);
- Users and privileges: access to the nodes for users, permissions on the file system;
- Time: change of the infrastructure over time, nomadic nodes, node or network failures.

Multiple specification languages have been proposed to describe cyber range configuration scenarios. For example, there is the Infrastructure and Network Description Language (INDL) introduced by Ghijsen et al. [GVDHGDL12] through two virtual infrastructures. The description language VXDL [KPC08] can define the latency requirements that the infrastructure must satisfy to achieve its goal. In recent work, Costa et al. [CRA20] proposed a *virtual scenario description language* (VSDL) allowing one to write a set of statements that fully describes a set of configuration scenarios. Such a set can then be instantiated and deployed as a virtualised infrastructure, depending on the available virtualization budget. The virtualization budget typically refers to hardware requirements: as a virtual cyber range scenario is simulated on a machine with limited CPU frequency and limited storage, the scenario can not allocate more than this limit to the network's elements.

A VSDL scenario completely defines a cyber range's virtual scenario and is sufficient to instantiate it, as is shown by the simplified workflow depicted in Figure 3. This process is to be read top-down. Processes are represented by rounded rectangles, and data by straight rectangles. In short, the goal of this workflow is to deploy a virtual infrastructure from a VSDL scenario. A VSDL scenario is first translated into a first-order logic formula with sets of constraints that include the virtualization budget (see Costa et al. [CRA20] for details). An SMT solver is then used to instantiate a concrete scenario that satisfies those constraints. If the VSDL scenario is *unsat* (not satisfiable), it must be modified. Otherwise, a complete scenario is deployed via state-of-the-art technologies like

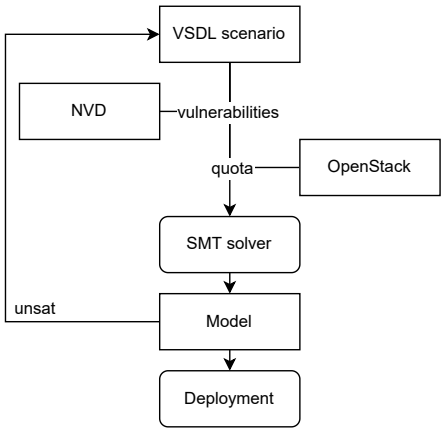

Figure 3: VSDL-based workflow for deploying a cyber range configuration scenario.

OpenStack, Terraform[2] and Packer[3]. Note that some VSDL components may contain vulnerability identifiers from the National Vulnerability Database (NVD)[4]; if an element contains such identifier, it means that it hosts a vulnerable configuration described in the NVD (e.g., a specific software or OS version).

Note that these nodes can have software constraints themselves on some attributes (e.g. the computer does not need more than 128 GB). The values are decided by the SMT solver, which can take into account all constraints, notably the budget. These constraints justify the use of an SMT solver, as arithmetic constraints cannot be handled by a simpler SAT solver.

2.3 Attack-defence trees

Attack-defence trees are hierarchical representations for sets of exploits [FMDC09, Yes14]. Each leaf represents an attack step (exploitation of a given vulnerability) while intermediary nodes represent Boolean combinations of such steps. The root of the tree localises the set of exploits. For example, consider a volumetric DDoS, that is a massive set of connections to take down a server. To perform such an attack, one should first build several botnets to perform the connections or rent them [HPB18]. The DDoS attack is itself an attack step participating to the overall objective of making the target unreachable. Observe that attack trees could be regarded as Feature Diagrams whose features are the attack steps.

An attack-defence tree that represents a set of exploits on a simple *software-defined networking* (SDN) based network [KRV+14] is given in [HPB18] and depicted in Figure 4. The network is composed of one SDN controller and several SDN switches. A SDN-based network centralizes the network management through this SDN controller. In this example, the network is an IT infrastructure of a public authority which stores critical datasets in its servers. These servers cannot be accessed from outside, but linked web servers can be accessed from the Internet. The goal of the tree is to compromise the network, through three possible approaches: (1) information disclosure (i.e. have access to critical data), (2) DDoS attack, or (3) compromising the SDN controller itself.

[2]https://www.terraform.io/docs
[3]https://www.packer.io/docs
[4]https://nvd.nist.gov

3 MODELLING CYBER RANGES

The main feature of a cyber range is its ability to virtualise scenarios on which trainees can exploit vulnerabilities to achieve cyber security exploits. Cyber range configuration scenarios can be described via configuration languages such as VSDL. Such languages are powerful enough to describe sets of configuration scenarios. However, they do not take the profile of their users into account, nor do they allow to model explicitly advanced cyber security exploits. Our objective is to resolve both issues by following a feature-based context-oriented modelling approach.

We propose to define a context model consisting of the user profile of the trainees and the attack-defence trees, and a feature model consisting of a VSDL specification. In such a modelling approach, a configuration of the feature model would be equivalent to a virtual network (its elements, with the vulnerabilities injected). Thanks to a context-feature mapping, this feature model configuration would be adapted to the needs of the trainees, as defined by a configuration of the context model. This section presents our vision of how to build such context, feature and mapping models for cyber ranges. For each of them, we show how to create the model and illustrate it with an example based on our SDN case study.

3.1 Context model

The context model will consist of two parts:

User profile. The user profile contains the name of the tenant and its teams (a more mature context model might store the user profiles of many different teams that come and train with the cyber range). Other data of interest are the budget or quota of the trainees (which may constrain the resources to put in the virtual network, e.g. maximum CPU speed, disk sizes, ...), their skills in cyber security, as well as the expected training time. The user profile could also include learning preferences to better choose which attack patterns should be enabled. This user profile is not exhaustive, and could be enhanced over time to include additional information relevant to the trainees. For example, in the future we expect that popular skill frameworks could be integrated into the user profile, to better adapt the cyber range scenario to the skills of its trainees. Such a skills framework is currently being developed by the European Union Agency for Cybersecurity (ENISA).

Attack-defence trees. The second part of the context model consists of attack-defence trees. An attack-defence tree has one top-most node that represents its exploit, and several internal nodes that describe the attack steps necessary to achieve this goal. In a configuration of the context model, an attack step could either be *active* or *inactive*. When an attack step is active, then this attack step should be possible to execute on the corresponding virtual network. If such a configuration (i.e. a set of *active* attack steps) is valid according to the attack-defence tree, then this configuration represents a coherent set of attack steps and its top-most goal will be achievable in the corresponding cyber range configuration scenario.

This means that not all attack steps will always be possible, and the virtual network may be different according to the selected attack

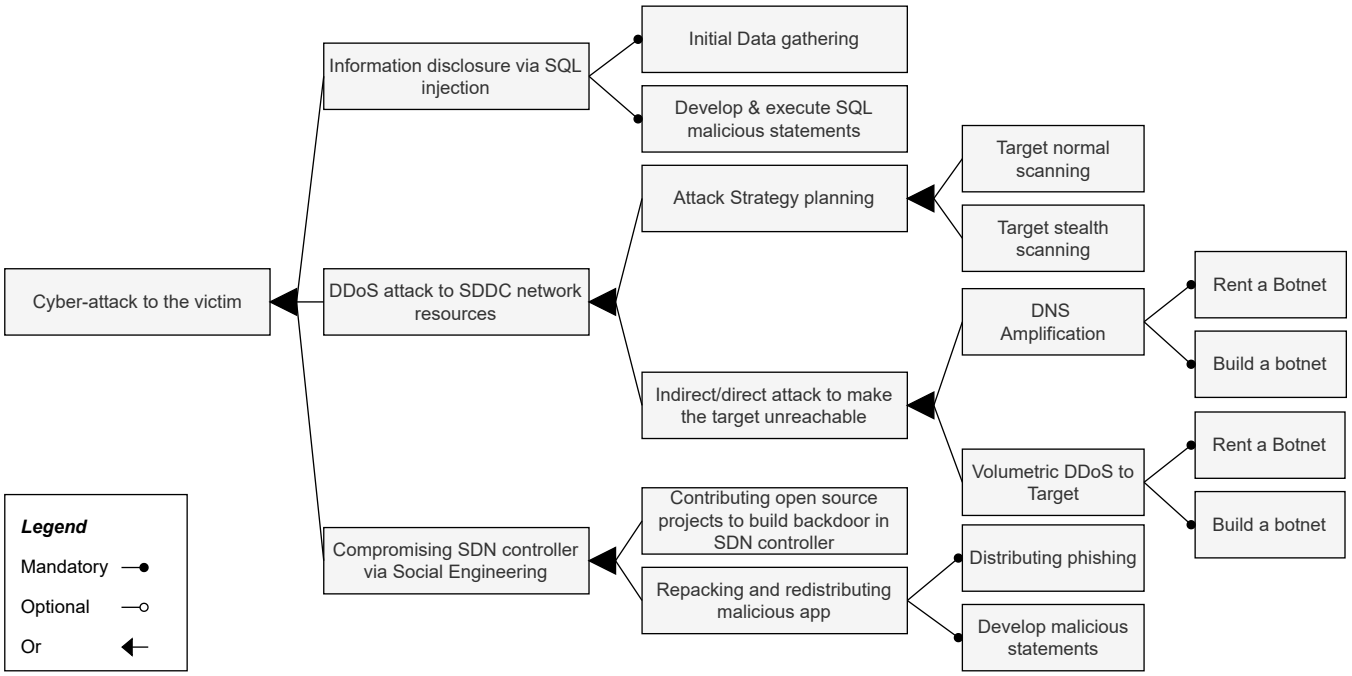

Figure 4: An attack-defence tree for an SDN-based network, inspired by [HPB18].

steps. Trainees or their supervisor could select their preferences to ask for specific attack patterns, in order to vary their experience and improve reusability of the context model. In the same spirit, as the attack-defence tree is now explicitly involved in the generation of a cyber range scenario, we could now consider the skills or time requirements of the trainees. Indeed, each attack step could specify the skills or time needed to complete it. Taking the example of the attack-defence tree depicted in Figure 4, we could set the difficulty/time required for distributing phishing to easy and fast, whereas developing and executing malicious statements inside a protected network could be set to longer and more difficult. This will be further discussed in Section 3.3.

Lastly, using attack-defence trees as part of the context model may improve the quality of the cyber range scenario. Indeed, the virtual network of a cyber range should be realistic (i.e. the virtual network should contain plausible vulnerabilities, and these vulnerabilities can be exploited to create attacks). However, this criterion is often implicit and hard to check. We solve this problem at its source by making attack-defence trees, which consist of coherent attack patterns, an integral part of the modelling and generation of scenarios. At the best of our knowledge, using attack-defence trees as contextual information is a novel approach for cyber range configuration scenario generation, but also for feature-based context-oriented modelling.

3.2 Feature model

A virtual network is composed of elements such as servers, computers, smartphones, and so on, as well as their relations. Such a virtual network can be described using a language like VSDL [CRA20], in which each element would correspond to a set of VSDL statements. For example, a computer in the network would be described by a set of VSDL statements to specify its OS, CPU frequency, disk sizes, etc. Each of these elements, i.e. each of these sets of VSDL statements, could be considered as a separate feature. Another configuration of that computer would be described by another set of VSDL statements and thus constitute another alternative feature. Whenever a feature is selected, the element described by that feature will be included in the generated VSDL scenario, whereas unselected features would not become part of that scenario.

Note that a feature could either be an entire asset (e.g., the definition of an entire computer), or a smaller part of the definition of a bigger asset. For example, just setting the OS of a computer to Windows 7 in order to allow for a certain vulnerability, while the OS could be set to Windows 10 if the vulnerability is not to be included. This example also illustrates that not all features can be selected together (we should select either Windows 7 or Windows 10 on that computer), and that some features require other to be selected (the features which sets the OS of a computer needs the computer itself to be deployed to have a meaning). Such constraints can be expressed in terms of feature-feature constraints as we will see in Subsection 3.3.

Let us consider the example of a simple SDN-based network that handles a public healthcare service such as a hospital. The core part of this network is the SDN controller, which runs on a computer, that is in charge of managing the hospital's entire network. In the connected private servers, sensible data on the health consultations of its patients is stored. Finally, a website is made available to the patients to make appointments or to receive diagnostics. Whereas these three elements are the foundations of the network, other elements can be added to make the scenario more realistic, or to

inject more vulnerabilities for the sake of training. For example, the web site could be hosted on multiple servers, with different types of services. The hospital could also own an email domain reserved for professional mails of its employees; in the cyber range configuration scenario, it would mean that phishing could now be used against these employees. We now illustrate some of these features with simple VSDL statements. As stated above, the *Core* feature is the network of the hospital.

```
\\ Core Feature
network Hospital {
    gateway has direct access to the Internet;
    addresses range from 4.4.4.1 to 4.4.4.64; }
```

The *Computer* and *Website* features each create a new node and connect it to the hospital network.

```
\\ Computer Feature
node Computer {
    flavour is computer;
    not (disk is larger than 128 GB);
    not (cpu is faster than 3 GHz);
    OS is Windows7 or Windows10; }
network Hospital {
    node Computer has IP 4.4.4.3; }

\\ Website Feature
node Website {
    flavour is webserver;
    disk is larger than 100 GB;
    cpu is faster than 8 GHz;
    OS is Debian-8; }
network Hospital {
    node Website has IP 4.4.4.10; }
```

The *PrivateServers* feature adds a new server node and a private network to which only the Computer node is connected. Observe how the *PrivateServers* feature incrementally refines the previously declared *Computer* feature by adding an additional property to the Computer node declared by that feature.

```
\\ PrivateServers Feature
node PrivateServer {
    flavour is server;
    disk is larger than 200 GB;
    contains (./SensibleData) path;
    mounts software SQL Server; }
node Computer {
    mounts software SQL Server; }
network ServersNetwork {
    addresses range from 1.1.1.1 to 1.1.1.64;
    node PrivateServer has IP 1.1.1.5;
    node Computer has IP 1.1.1.6; }
```

3.3 Mapping model

In the previous two subsections we described the contexts and features that make up the context and feature models, respectively. The mapping model is meant to link both models, and to allow finding a valid and relevant feature model configuration from a given context model configuration. In this section, we list and illustrate all constraints that imply the contexts or features, including the ones mentioned in the two previous sections.

In this endeavor, we largely extend the definition of a mapping model from traditional feature-based context-oriented modelling since it will not only contain *context-feature constraints* that determine what contexts will trigger what features, but also *context-context constraints* that describe the dependencies between contexts inside the context model, and *feature-feature constraints* which describe intra feature model dependencies. Our mapping model is thus essentially a set of constraints between contexts and features (or between themselves) that can be expressed in first-order temporal logic. One reason for combining these three types of constraints all into a single mapping model is that we will use an SMT solver to resolve the constraints, which allows for more complex constraints than a SAT solver and will be able to handle the arithmetic equations.

Context-feature constraints. The main goal of this mapping model, and closest to the role of the mapping model in the original feature-based context-oriented modelling approach (cf. Subsection 2.1), is to declare the context-feature constraints. These define how the activation of features should be triggered when some contexts are activated. For example, the vulnerabilities required to execute certain attack steps should be included in the virtual network. Secondly, we should not exceed the budget or quota set by the users.

In the following example, we follow NVD[5] and use dummy identifiers for vulnerabilities (*V1*, *V2*, *V3*). Each vulnerability can be seen as a feature as explained in the previous Subsection, i.e. a set of VSDL statements linked to the properties of a device that should eventually be included in the configuration scenario; its precise specifications can be found in the NVD. Let us consider again the public healthcare service example and the attack-defence tree depicted in Figure 4. Some context-feature constraints from that example are defined hereafter:

$$Develop\&ExecSQLMaliciousStatements \Rightarrow V1\_Website \tag{1}$$

$$TargetStealthScanning \Rightarrow V2\_Computer \tag{2}$$

$$ContributingOpenSourceProjects \Rightarrow V3\_Computer \vee V4\_Computer \tag{3}$$

$$DistributingPhishing \Rightarrow EmailDomain \tag{4}$$

$$quota_{CPU\_frequency} \geq \sum_{device \in VirtualNetwork} device(CPU\_frequency) \tag{5}$$

Whereas the first three constraints will enable vulnerabilities in the basics elements of our public healthcare service (which can be specific versions of software, e.g. *V1* would be the VSDL statement `node Computer software version SQLServer is less than 14`), the fourth enables an entirely new element that is the email domain.

[5] https://nvd.nist.gov

Constraint (3) permits us to choose one vulnerability among two, and this choice can be influenced by the other constraints in the mapping model. For example, vulnerability *V3* could require a CPU frequency of at least 2GHz on the device hosting the SDN controller, while *V4* could require 3GHz. This choice could be sorted out by the constraint (5), which states that the CPU frequency of all activated devices should not exceed a given quota set by the users. In practice, this shall be automatized via a translation to first-order logic and the use of an SMT solver.

Context-context constraints. In addition to defining the constraints inside and between attack-defence trees and on the user profiles, context-context constraints can also link the user profiles and attack-defence trees through skill and time requirements. Below is an example of such constraints stating that the average skill requirement (resp. total time) of the activated attack steps should not exceed the trainees' skills (resp. time).

$$TraineesSkills \geq \frac{1}{N_{ActivatedAttackSteps}} \sum_{step \in ActivatedAttackSteps} step(skill) \quad (6)$$

$$TraineesTime \geq \sum_{step \in ActivatedAttackSteps} step(time) \quad (7)$$

Feature-feature constraints. Many constraints on and in between features are already expressed through VSDL statements (e.g. that the Computer feature must use one among two specific OS). And just like in the VSDL-based workflow (Figure 3), constraints on the vulnerabilities can be expressed and retrieved through the NVD.

Additional feature-feature constraints worthwhile expressing would be hierarchical constraints akin to constraints in feature modelling, such as requirements relationships (e.g., a feature that defines the type of OS of a computer requires this computer to be in the virtual network) or mutual exclusion (e.g., between two features that set the version of the same software, on the same device). Typically, such constraints allow to express relevant variations in the virtual network enabling different configurations to lead to different cyber range configuration scenarios.

4 EXTENDED WORKFLOW

Section 3 sketched a feature-based context-oriented model adapted to the generation of configuration scenarios for cyber ranges. We now present a workflow for configuration scenario instantiation and virtualization that takes this modelling into account. Our workflow can be seen as an improvement of Costa's [CRA20], obtained by modifying the initialisation steps before deployment. The workflow of our approach is shown in Figure 5.

The first step is the definition of the feature-based context-oriented model, such as defined in the previous section. The proposed modelling approach is supported by different concepts and technologies, different resources, that are summarized in the topmost box. Once the model defined, we can start using it to generate concrete scenarios for trainees. Such scenarios are generated in several steps. First, a partial configuration (i.e. a configuration which contains some context activations, but not all) is obtained by collecting the user profile data of a team of trainees. This user profile

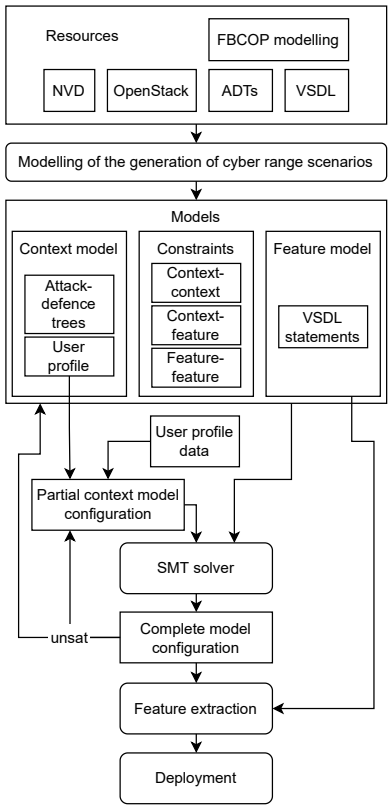

Figure 5: Workflow of the generation of a cyber range scenario

is usually provided by a *trainer* that supervises a team of trainees and who wishes to train them. This data is contained in the context model (see Section 3.1).

An SMT solver is then used to check the *satisfiability* of the partial configuration. If the partial configuration is satisfiable, the SMT solver will generate a valid complete configuration (i.e. that satisfies all constraints, and hence all the requirements put forward by the user profile). Our use of an SMT solver is close to the one in the VSDL-based workflow as presented by Costa et al. [CRA20] (i.e. we feed it constraints from a VSDL scenario, through the feature model), but also to the use of a SAT solver in the deployment of a feature-based context-oriented software system (we also feed it a partial configuration to find a complete one). In theory, using a combination of both approaches is sound with an SMT solver, as it simply increases the number of constraints it has to handle.

Once a complete configuration is obtained from the SMT solver, we need to extract the features useful for the virtual network deployment, the virtual scenario definition. Indeed, while the SMT solver chooses what contexts to activate as well (e.g. which attack steps were selected), we only need the activated features and hence the resulting VSDL scenario, to instantiate a virtual network. The deployment from a model generated by passing a VSDL scenario to an SMT solver was shown in detail in Costa's VSDL-based approach [CRA20]; hence, from this point onwards, the deployment step is the same as in their workflow. Being able to connect to their

workflow implies that our proposed approach will likewise be able to use state-of-the-art technologies (OpenStack, Terraform, Packer) for the virtual network instantation.

When using the SMT solver, the partial configuration can be found to be *unsatisfiable*. There might be two causes: first, the trainer asked for invalid or too harsh requirements. For example, the trainer asked for two vulnerabilities to be present in the virtual network; however, both vulnerabilities are incompatible (e.g. they impose different versions of the same software on a device). Another likely scenario would be an insufficient budget for their requirements (e.g. specific attack patterns that require large servers); the model might not find a suitable scenario. The second cause might be the model itself, which can of course be faulty. For example, a trainer rightfully asked for a specific attack pattern with sufficient budget; however, the defined model doesn't contain any combination of features that allows it. In both case, either the user profile data is modified to suit the current model, or the cyber range creators modify their models to correct it or to accept new requirements.

5 CONCLUSION

VSDL is an established language to describe cyber range configuration scenarios. It can be seen as a way to describe sets of configuration scenarios on which cybersecurity specialists can learn how to protect or attack some infrastructure or information system. One of VSDL's current drawbacks, however, is that it doesn't take into account user skills and possibilities to accomplish exploits.

In this vision paper we showed how a feature-based context-oriented modelling approach stemming from COP could be used to extend VSDL with user profiles (used to take user skills into account) and attack-defence trees (used to represent security exploits). Those two extensions are described within a context model, while the features to be included in a cyber range configuration scenario are declared using VSDL. The two models are connected via a mapping. In addition to this modelling approach, we sketch a full workflow for the deployment and virtualisation of the configuration scenario from the model with the help of an SMT solver.

As this is a vision paper, whose main objective is to open up feature-based context-oriented modelling to a new application area, we mostly focused on discussing the modelling approach and how to apply it to a simple case study. Many directions for future work remain, the first one being to develop this approach as a working proof of concept. Considering huge case studies may lead to intriguing combinations of constraints which could make the use of SMT solving problematic. To overcome this problem we could consider approximation solvers such as those introduced in [BLM20].

One may also observe that the proposed generation is dynamic, but the generated scenarios are not. Indeed, while the scenario plays out, the network's elements will remain the same. Hence, another line of future work would consist of adding dynamic components, such as time waves [CRA20], to capture the timing evolution of the configuration scenario (e.g., an element only appears after two hours). Another possible dynamic component would be agents [YK22] that emulate attacker (or defender) behaviour during a scenario in order to propose a more realistic experience.

Finally, observe that combining these two different research domains is beneficial for both. First, it further legitimates the development of a feature-based context-oriented modelling approach that separates the modelling of contexts and features as first-class citizens. Second, it helps formalizing the relationship between the various training elements that constitute a cyber range experience.

REFERENCES

[ACF+09] Mathieu Acher, Philippe Collet, Franck Fleurey, Philippe Lahire, Sabine Moisan, and Jean-Paul Rigault. Modeling context and dynamic adaptations with feature models. In *4th International Workshop Models@ run. time at Models 2009 (MRT'09)*, page 10, 2009.

[BCD+11] Clark Barrett, Christopher L Conway, Morgan Deters, Liana Hadarean, Dejan Jovanović, Tim King, Andrew Reynolds, and Cesare Tinelli. Cvc4. In *International Conference on Computer Aided Verification*, pages 171–177. Springer, 2011.

[BLM20] Eduard Baranov, Axel Legay, and Kuldeep S. Meel. Baital: an adaptive weighted sampling approach for improved t-wise coverage. In Prem Devanbu, Myra B. Cohen, and Thomas Zimmermann, editors, *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, pages 1114–1126. ACM, 2020.

[CH05] Pascal Costanza and Robert Hirschfeld. Language constructs for context-oriented programming: An overview of contextl. In *Proceedings of the 2005 Symposium on Dynamic Languages (DLS '05)*, page 1–10. ACM, 2005.

[COH14] Rafael Capilla, Óscar Ortiz, and Mike Hinchey. Context variability for context-aware systems. *Computer*, 47(2):85–87, 2014.

[CRA20] Gabriele Costa, Enrico Russo, and Alessandro Armando. Automating the generation of cyber range virtual scenarios with VSDL. *arXiv preprint arXiv:2001.06681*, January 2020.

[DMD19] Benoît Duhoux, Kim Mens, and Bruno Dumas. Implementation of a feature-based context-oriented programming language. In *Proceedings of the Workshop on Context-oriented Programming*, COP '19, pages 9–16. ACM, 2019.

[DMDL19] Benoît Duhoux, Kim Mens, Bruno Dumas, and Hoo Sing Leung. A context and feature visualisation tool for a feature-based context-oriented programming language. In *Proceedings of the Seminar Series on Advanced Techniques & Tools for Software Evolution (SATTOSE '19)*, volume 2510 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.

[Duh22] Benoît Duhoux. *Feature-Based Context-Oriented Software Development*. PhD thesis, 2022.

[FMDC09] Igor Nai Fovino, Marcelo Masera, and Alessio De Cian. Integrating cyber attacks within fault trees. *Reliability Engineering & System Safety*, 94(9):1394–1402, 2009.

[GCM+11] Sebastián González, Nicolás Cardozo, Kim Mens, Alfredo Cádiz, Jean-Christophe Libbrecht, and Julien Goffaux. Subjective-c: Bringing context to mobile platform programming. In *Proceedings of 3rd International Conference on Software Language Engineering*, SLE '10, pages 246–265. Springer, 2011.

[GVDHGDL12] Mattijs Ghijsen, Jeroen Van Der Ham, Paola Grosso, and Cees De Laat. Towards an infrastructure description language for modeling computing infrastructures. In *2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications*, pages 207–214. IEEE, 2012.

[HCH08] Robert Hirschfeld, Pascal Costanza, and Michael Haupt. Generative and transformational techniques in software engineering ii. chapter An Introduction to Context-Oriented Programming with ContextS, pages 396–407. Springer, 2008.

[HPB18] Mehrdad Hajizadeh, Trung V Phan, and Thomas Bauschert. Probability analysis of successful cyber attacks in sdn-based networks. In *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–6. IEEE, 2018.

[HT08] Herman Hartmann and Tim Trew. Using feature diagrams with context variability to model multiple product lines for software supply chains. In *Proceedings of 12th International Software Product Line Conference*, SPLC '08, pages 12–21. IEEE, 2008.

[KCH+90] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, and A. Spencer Peterson. Feature-oriented domain analysis (foda) feasibility study. Technical report, Carnegie-Mellon University Software Engineering Institute, November 1990.

[KPC08] Guilherme Piegas Koslovski, Pascale Vicat-Blanc Primet, and Andrea Schwertner Charao. Vxdl: Virtual resources and interconnection networks description language. In *International Conference on Networks for Grid Applications*, pages 138–154. Springer, 2008.

[KRV+14] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2014.

[MCHK17] Kim Mens, Rafael Capilla, Herman Hartmann, and Thomas Kropf. Modeling and managing context-aware systems' variability. *IEEE Software*, 34(6):58–63, 2017.

[MMDL21] Pierre Martou, Kim Mens, Benoît Duhoux, and Axel Legay. Test scenario generation for context-oriented programs. *arXiv preprint arXiv:2109.11950*, 2021.

[MS14] Sunilkumar S Manvi and Gopal Krishna Shyam. Resource management for infrastructure as a service (iaas) in cloud computing: A survey. *Journal of network and computer applications*, 41:424–440, 2014.

[SGP11] Guido Salvaneschi, Carlo Ghezzi, and Matteo Pradella. Javactx: Seamless toolchain integration for context-oriented programming. In *Proceedings of 3rd International Workshop on Context-Oriented Programming*, COP '11, pages 4:1–4:6. ACM, 2011.

[Yes14] Ahmed S. Yesuf. *Context-based attack tree modeling for software development: A framework for computer-aided and context-aware attack tree modeling approach for software development.* LAP LAMBERT Academic Publishing, 2014.

[YK22] Muhammad Mudassar Yamin and Basel Katt. Modeling and executing cyber security exercise scenarios in cyber ranges. *Computers & Security*, 116:102635, 2022.