Complex Systems Informatics and Modeling Quarterly (CSIMQ)
eISSN: 2255-9922
Published online by RTU Press, https://csimq-journals.rtu.lv
Article 151, Issue 26, March/April 2021, Pages 55–77
https://doi.org/10.7250/csimq.2021-26.04

A Method for Assigning Probability Distributions in Attack Simulation Languages

Wenjun Xiong*, Simon Hacks, and Robert Lagerström

School of Electrical Engineering and Computer Science,
KTH Royal Institute of Technology, Brinellvägen 8, 114 28 Stockholm, Sweden

wenjx@kth.se, shacks@kth.se, robertl@kth.se

Abstract. Cyber attacks on IT and OT systems can have severe consequences for individuals and organizations, from water or energy distribution systems to online banking services. To respond to these threats, attack simulations can be used to assess the cyber security of systems to foster a higher degree of resilience against cyber attacks; the steps taken by an attacker to compromise sensitive system assets can be traced, and a time estimate can be computed from the initial step to the compromise of assets of interest.
Previously, the Meta Attack Language (MAL) was introduced as a framework to develop security-oriented domain-specific languages. It allows attack simulations on modeled systems and analyzes weaknesses related to known attacks. To produce more realistic simulation results, probability distributions can be assigned to attack steps and defenses to describe the efforts required for attackers to exploit certain attack steps. However, research on assessing such probability distributions is scarce, and we often rely on security experts to model attackers' efforts. To address this gap, we propose a method to assign probability distributions to the attack steps and defenses of MAL-based languages. We demonstrate the proposed method by assigning probability distributions to a MAL-based language. Finally, the resulting language is evaluated by modeling and simulating a known cyber attack.

Keywords: Attack Simulations, Threat Modeling, Domain-Specific Language, Cyber Security, Information Collection.

1 Introduction

Cyber security continues to be a key concern and a fundamental aspect of information technology (IT) and operational technology (OT) systems [1]. Recent years have witnessed some of the largest, most sophisticated, and most severe cyber attacks, such as the

* Corresponding author

Reference: W. Xiong, S. Hacks, and R. Lagerström, "A Method for Assigning Probability Distributions in Attack Simulation Languages," Complex Systems Informatics and Modeling Quarterly, CSIMQ, no. 26, pp. 55–77, 2021. Available: https://doi.org/10.7250/csimq.2021-26.04

Additional information. Author ORCID iD: W. Xiong – https://orcid.org/0000-0003-0434-4436, S. Hacks – https://orcid.org/0000-0003-0478-9347, and R. Lagerström – https://orcid.org/0000-0003-3089-3885. PII S225599222100151X. Received: 19 March 2021. Accepted: 16 April 2021. Available online: 30 April 2021.

SolarWinds attack[1], Florida water supply attack[2], and Facebook information leak[3], which affected millions of consumers and thousands of businesses. To conduct attacks against systems, attackers usually combine multiple vulnerabilities to compromise sensitive system assets [2] and penetrate the systems with damaging impact [3].

However, assessing the security level of systems is difficult. It is necessary, but challenging to identify all relevant system assets, their weaknesses in various cyber attacks, and possible mitigations. To proactively deal with security concerns, threat modeling as a solution for securing systems can make it more difficult for attackers to accomplish their malicious intent [4]. This includes holistic identification of the main assets within a system and threats to these assets. Threat modeling is used to assess the current state of a system and as a security-by-design tool for developing new systems. A recent improvement is to couple threat modeling with attack simulations [5], [6]. In such simulations, the steps taken by an attacker to compromise system assets are traced, and a time estimate is computed from the initial step to the compromise of assets of interest [2].

The Meta Attack Language (MAL) [2] was proposed as a framework that combines object-oriented modeling and attack simulations. Because MAL is a meta-language with no particular domain of interest, it is necessary to create a more concrete language that reflects the demands of a certain domain. This MAL-based language can then be used to automate the security analysis of instance models within the respective domain. We have designed enterpriseLang [7] as a modeling and simulation language for enterprise IT systems. However, only binary relations between attack steps and defenses were implemented, assuming that an attack step can be reached by an attacker instantly or cannot be reached because there is a defense in place. This is in contrast to real-world attack scenarios, in which many attack steps require a certain effort to be compromised, and defenses cannot achieve a 100% opportunity to block certain attacks.

To provide more realistic simulation results, some MAL-based languages (e.g., [8]) defined probability distributions on a few attack steps and defenses to express the attackers' efforts needed to exploit certain attack steps. Unfortunately, most of them lack a systematic approach to assign probability distributions to more than a few random steps, which would be needed for a robust and complete language. To bridge this gap, we propose a method for assigning probability distributions to attack steps and defenses of MAL-based languages (and potentially also useful for other attack-graph-based approaches). This will yield more realistic simulation results that can help stakeholders to investigate security settings, which, in turn, could be implemented to secure a system more effectively. Finally, we demonstrate our proposed approach by enriching enterpriseLang [7] with such probability distributions.

The remainder of this article is structured as follows. In Section 2, we review the current state-of-the-art. In Section 3, we present the background of this study. Section 4 presents the design methodology of our method. Section 5 describes the method in detail. In Section 6, we demonstrate the proposed method by assigning probability distributions to enterpriseLang. In Section 7, we present the first evaluation of the designed method. The results are discussed in Section 8, and finally, the article is concluded in Section 9.

2 Related Work

Our work relates to three domains of related work: model-driven security engineering, attack/defense graphs, and architecture modeling for system analysis.

[1] https://www.cnet.com/news/solarwinds-hack-officially-blamed-on-russia-what-you-need-to-know/

[2] https://www.industrialdefender.com/florida-water-treatment-plant-cyber-attack/

[3] https://www.cbsnews.com/news/millions-facebook-user-records-exposed-amazon-cloud-server/

The field of model-driven security engineering includes a large number of domain-specific languages. A well-known initiative for modeling from a system-wide perspective is UMLsec [9], which is an extension of the unified modeling language for developing security-critical systems. Cardenas *et al.* [10] provided a holistic view of the security requirements and threat models of sensor networks, focusing on high-level security goals. In these languages, it is possible to specify a system design in terms of components and their interactions, as well as security properties such as constraints, requirements, or threats.

Many model-driven approaches are based on attack trees. The concept of attack trees is commonly attributed to Bruce Schneier [11], [12]. Formal foundations of attack trees were laid by Mauw and Oostdijk [13], and the framework was further extended to include defenses by Kordy *et al.* [14]. Since then, many attack-graph-based methods (e.g., [15], [16], [17], [18]) have been presented. These theoretical descriptions led to the development of different tools using attack graphs, and these are mostly based on collecting information on existing systems and automatically creating attack graphs. For instance, MulVal [19] derived logical attack graphs by associating the vulnerabilities extracted from scans with a probability that expresses how likely an attacker is to exploit them successfully. k-Zero Day Safety [20] extended MulVAL to compute zero-day attack graphs. The TVA tool [21] modeled networks in terms of security conditions and used an information base of exploits as transitions between these security conditions. Similarly, NetSecuritas [22] composed scanner output and known exploits to generate attack graphs and corresponding security recommendations. In [23], a prototype was proposed for generating attack graphs from MITRE ATT&CK for ICS (industrial control system)[4] automatically for adversary behavior execution.

These attack graphs can be extended to probabilistic attack graphs. For instance, Frigault *et al.* [24] used the TVA-tool to generate attack graphs and transformed them into dynamic Bayesian networks. They also enriched them with probabilities using the common vulnerability scoring system (CVSS) scores[5]. Similarly, Wang *et al.* [25] proposed an attack-graph-based probabilistic metric for network security by combining the measurements of individual vulnerabilities obtained from existing metrics into an overall score of the network, where individual scores can be obtained by converting vulnerability scores provided by existing standards, e.g., CVSS scores to probabilities.

Architecture modeling can aid system analysis and help handling the increasing complexity of IT landscapes [26], [27]. The importance of creating models to support decision-making has previously been addressed in some studies. For instance, Lagerström *et al.* [28], [29] presented instantiated architectural models based on a metamodel for modifiability analysis of enterprise systems to support decision-making, where probabilistic relational models were used to combine regular entity-relationship modeling aspects to perform enterprise architecture analysis under uncertainty.

Previous work, including CySeMoL [30], P2CySeMoL [6], and pwnPr3d [31], also employed architectural modeling, in which attacks and defenses were coupled to objects of system architectures and were probabilistically related by Bayesian networks. Their design essentially involved creating its qualitative structure (assets, attacks, defenses, and associations) and populating the qualitative structure with quantitative data (how likely different attacks are to succeed, given the presence or absence of different defenses). However, these approaches did not follow an explicit meta language, making them difficult to change and re-use [2]. Therefore, MAL was proposed to create MAL-based languages. Till date, several MAL-based languages have been proposed, such as enterpriseLang [7] for modeling cyber attacks on enterprise IT systems, powerLang [8] for power-related IT and OT infrastructures, and coreLang [32] for common IT infrastructures.

[4] https://collaborate.mitre.org/attackics/

[5] https://www.first.org/cvss/

3 Background

3.1 Introduction to MAL

MAL is a modeling and simulation language framework that combines probabilistic attack and defense graphs with object-oriented modeling, which represents no particular domain of interest and can be used to create domain-specific languages (DSLs). A MAL-based DSL defines what information is required and specifies the generic attack logic of a domain under study. We refer to the original paper [2] for a detailed overview of MAL.

To create a MAL-based DSL, the first step is to identify all the relevant assets within a particular domain. Each asset contains multiple attack steps that represent actual attacks or threats to the asset. In addition, one successfully compromised attack step can lead to (represented by "–>") the next attack step, where each attack step is of type OR (represented by "|") or AND (represented by "&"). OR indicates that an attacker can start working on this attack step as soon as one of its parent attack steps is compromised, whereas AND indicates that all its parent attack steps are compromised for an attacker to reach this step. An asset may also feature defenses (represented by "#"). The sum of the attack paths is the attack/defense graph used for the attack simulation. Further, to model the possible transitions of an attacker among different assets, we connect the related ones by associations. Finally, assets can inherit from each other, meaning that an inherited asset inherits all attack steps and defenses of its parent asset.

In real-world attack scenarios, some attack steps can be compromised immediately, whereas most attack steps require a certain effort to be compromised, expressed by time, i.e., time to compromise (TTC) [2], [6]. The TTC value measures the security level of various assets in a modeled system in terms of attack resilience; the larger the value, the more resilient the system is against cyber attacks. Therefore, attack steps can be associated with a probability distribution that describes the effort required to perform each step. With binary relations implemented in a MAL-based language, each defense has a Boolean value to indicate their status, where "enabled" or "disabled" is represented by setting the defense value to True or False. For instance, if the defense has a value True, then its connected attack steps are not performed. With probabilistic relations, when we assign a probability distribution to a defense, it describes the probability that the defense is successful.

3.2 A MAL-based Language for Enterprise Systems

Based on the MAL framework, a MAL-based language for modeling enterprise IT systems, called enterpriseLang, was designed [7]. To design such a language, the information regarding the system assets (e.g., computer, service, OS, firewall, and internal and external networks), attack steps (e.g., spearphishing attachment, brute force, and automated exfiltration), and defenses (e.g., privileged account management, execution prevention, and network segmentation) were extracted from the MITRE Enterprise ATT&CK Matrix[6] and then converted and combined into enterpriseLang. The enterpriseLang metamodel, containing enterprise assets and associations, is shown in Figure 1.

In total, enterpriseLang contains 12 assets and 10 inherited assets. We organize all the assets into five categories: `Person`, `Account`, `Software`, `Hardware`, and `Network`.

The `Person` category includes one asset—`User`—and reflects the human aspect of cyber security.

The `Account` category includes two assets: `UserAccount` and `AdminAccount`. A `User` can log in to an `AdminAccount` or a `UserAccount`. By logging to an `AdminAccount`, one can change the security settings, install software, and access files on a `Computer`. The

[6] https://attack.mitre.org/

`AdminAccount` has two inherited assets, `WindowsAdmin` (for Windows) and `Root` (for Linux and macOS), depending on the operating system (`OS`) running on a certain `Computer`.

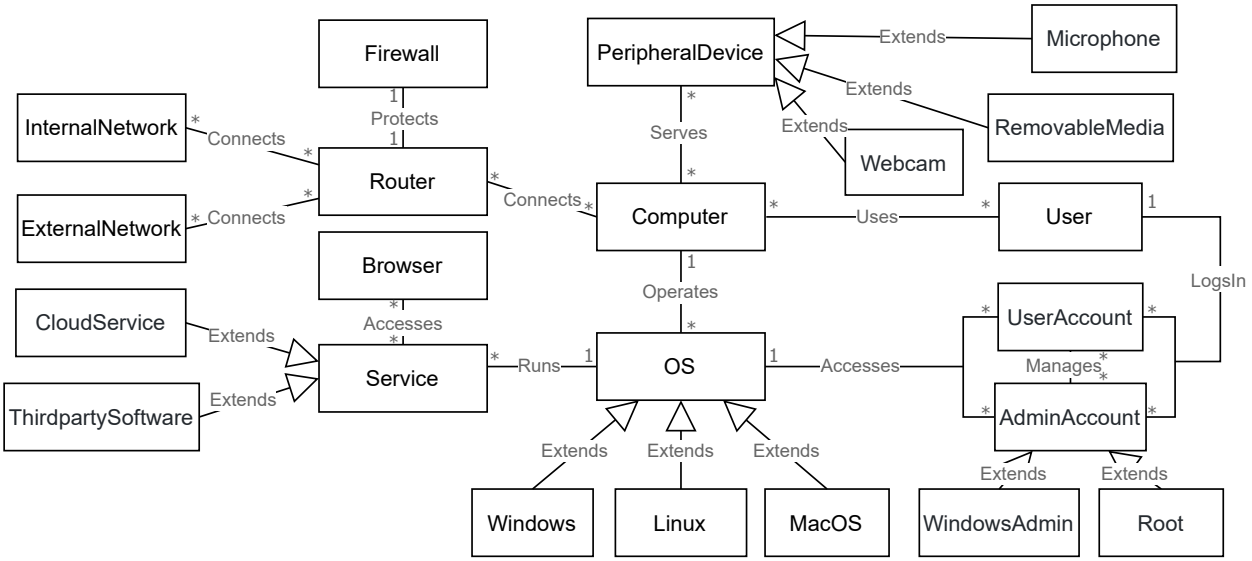


Figure 1. EnterpriseLang metamodel containing assets and associations

The `Software` category includes three assets: `OS`, `Service`, and `Browser`. When `OS`s boot up, they can run programs or applications called `Service`s to perform functions. One commonly accessed `Service` is `Browser`. In addition, `OS` has three inherited assets: `Windows`, `Linux`, and `macOS`, which represent the most commonly used operating systems. Further, `Service` has two inherited assets: `CloudService` and `ThirdpartySoftware`.

The `Network` category contains four assets: `InternalNetwork`, `ExternalNetwork`, `Router`, and `Firewall`, where a `Firewall` is connected to a `Router` asset and provides firewall rules to allow certain network activities.

The `Hardware` category contains two assets: `Computer` and `PeripheralDevice`. The `Computer` asset represents office PCs. Furthermore, `PeripheralDevice` has three inherited assets: `Microphone`, `RemovableMedia` (e.g., USB), and `Webcam`.

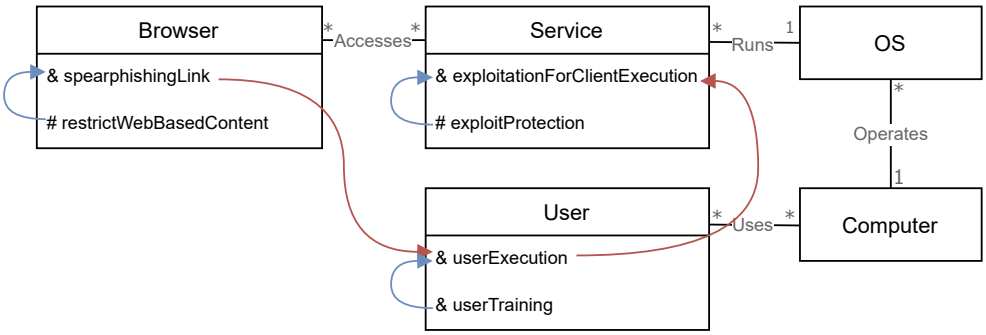


Figure 2. Graphical representation of attack steps and defenses relations

The attack steps and defenses are related to each other. In Figure 2, the red arrows represent the possible attack path that attackers can take to achieve their goals. First, attackers send spearphishing emails containing malicious links (i.e., *spearphishingLink*[7]), leveraging *userExecution*[8]. Users clicking on the links can lead to the exploitation of a service vulnerability via *exploitationForClientExecution*[9]. Defenses

[7] https://attack.mitre.org/techniques/T1566/002/
[8] https://attack.mitre.org/techniques/T1204/001/
[9] https://attack.mitre.org/techniques/T1203/

can be implemented to prevent certain attack steps, as represented by the blue arrows. In this example, *restrictWebBasedContent*[10] can be implemented to defend against the *spearphishingLink* attack, *userTraining*[11] can be performed as a way to raise awareness on common spearphishing techniques and prevent the *userExecution*, and *exploitProtection*[12] can be indicative of software exploitation occurring and defend against the *exploitationForClientExecution* attack.

In the code base of enterpriseLang, the aforementioned example is written as follows:

```
category Person {
    asset User {
        & userExecution
            ->  computer.os.service.exploitationForClientExecution
        # userTraining
            ->  userExecution
    }
}
category Software {
    asset Browser {
        & spearphishingLink
            ->  service.os.computer.user.userExecution
        # restrictWebBasedContent
            ->  spearphishingLink
    }
    asset Service {
        & exploitationForClientExecution
            ->  ...
        # exploitProtection
            ->  exploitationForClientExecution
    }
    asset OS {
    ...
    }
}
category Hardware {
    asset Computer {
    ...
    }
}
associations {
    Browser [browser] * <--Accesses--> * [service] Service
    OS [os] 1 <--Runs--> * [service] Service
    Computer [computer] 1 <--Operates--> * [os] OS
    User [user] * <--Uses--> * [computer] Computer
}
```

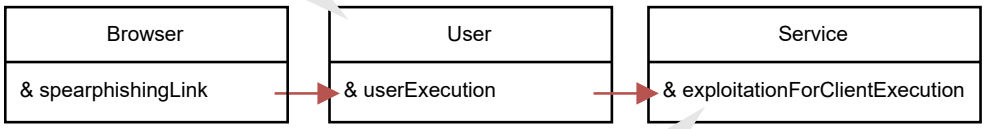
However, enterpriseLang only applied binary relations between different attack steps and defenses. In this case, all the accessible attack steps for an attacker can be compromised without any effort, and the defenses only have values True or False. In the aforementioned example, if the *exploitProtection* is implemented and enabled (i.e., set to value True), its connected attack step *exploitationForClientExecution* will be blocked. This is not realistic because security applications and antivirus software are not always up-to-date and cannot stop these attacks completely [33].

An example of the relations between attack steps is shown in Figure 3. With binary relations, when the *spearphishingLink* attack on the `Browser` is completed, the attack step, *userExecution*, can be reached immediately. If information sources [34], [35] state that *exploitationForClientExecution* can be performed within one day with a probability of 71.2%, we could assign a Bernoulli(0.712)*Exponential(1) distribution on the

[10] https://attack.mitre.org/mitigations/M1021/
[11] https://attack.mitre.org/mitigations/M1017/
[12] https://attack.mitre.org/mitigations/M1050/

exploitationForClientExecution step. Therefore, when the *userExecution* step is reached, the effort it requires for an attacker to perform the *exploitationForClientExecution* attack follows a Bernoulli(0.712)*Exponential(1) distribution. When conducting attack simulations, the local time to compromise (i.e., local TTC) of *exploitationForClientExecution* is sampled from its assigned probability distribution, describing the expected time required to perform the *exploitationForClientExecution* step, and thus can produce more realistic simulation results.

	User.userExecution: True	User.userExecution: False
Browser.spearphishingLink: True	1	0
Browser.spearphishingLink: False	0	1



	Service.exploitationForClientExecution: True	Service.exploitationForClientExecution: False
User.userExecution: True	Bernoulli(0.712)*Exponential(1)	1-Bernoulli(0.712)*Exponential(1)
User.userExecution: False	1-Bernoulli(0.712)*Exponential(1)	Bernoulli(0.712)*Exponential(1)

Figure 3. Example of binary and probabilistic relations between attack steps

A common issue for MAL-based languages is that only a few attack steps and defenses have probability distributions assigned that describe the efforts required for an attacker to compromise a certain attack step and the probability that a defense is effective. Therefore, it is essential to define probability distributions for most attack steps and defenses of MAL-based languages to provide more realistic simulation results for their system model instances. However, studies on assessing such probability distributions are scarce, and we often rely on security experts to model them [8].

4 Design Methodology

Design science research (DSR) is a widely applied and accepted means of developing artifacts in information systems research. It offers a systematic structure for developing artifacts, such as constructs, models, methods, or instances [36]. Thus, the application of DSR is appropriate here as it guides the development of our method. In this work, our method is designed according to the DSR guidelines of Peffers *et al.* [37], which include six steps:

- `Step 1: Identify Problem & Motivate`:
  To respond to threats on IT and OT systems, attack simulations can be used to assess the cyber security of systems to foster a higher degree of resilience against cyber attacks. Previously, MAL was introduced as a framework to develop security-oriented MAL-based DSLs, which allows attack simulations on modeled systems and analyzes weaknesses related to known attacks. To produce more realistic simulation results, probability distributions can be assigned to attack steps and defenses to describe the efforts required for attackers to exploit certain attack steps, while research on assessing such probability distributions is scarce.
- `Step 2: Define Objectives`:
  To produce more realistic simulation results, a method to assign probability distributions to the attack steps and defenses is needed for a robust and complete MAL-based DSL.

- `Step 3: Design & Development`:
  We situate our method to the risk and impact analysis stage of the Process for Attack Simulation and Threat Analysis (PASTA) process [38], which is presented in Figure 4. First, we collect information from different sources for the domain under assessment. Because the sources are of various types and include qualitative studies, we assess their quality by credibility assessment. We then interpret and convert information into probability distributions that can be applied to MAL-based languages.
- `Step 4 & 5: Demonstration & Evaluation`:
  We demonstrate the designed method by assigning probability distributions to enterpriseLang (cf. Section 6). The updated language is evaluated by comparing the simulation results using two different versions of it (i.e., binary relations and probability distributions) using a documented known cyber attack (cf. Section 7).
- `Step 6: Communication`:
  The research is communicated by the publication of this article. Also, MAL is an open-source project that is publicly available from the GitHub repository[13].

5 Method for Assigning Probability Distributions

As mentioned earlier, it is a common issue for MAL-based languages to have only a few attack steps and defenses with probability distributions (e.g., [8], [39]). Therefore, we propose a method that guides future developers to find such probabilities and assign them to MAL-based languages. However, our method is situated in a bigger picture of the entire threat modeling process. Because our method provides essential input for the risk and impact analysis in the form of the effort an attacker needs to spend on a certain attack step, and using our method could produce quantitative simulation results that are similar to the output of risk and impact analysis (i.e., quantitative risk analysis), it is reasonable to fit it into the PASTA [38].

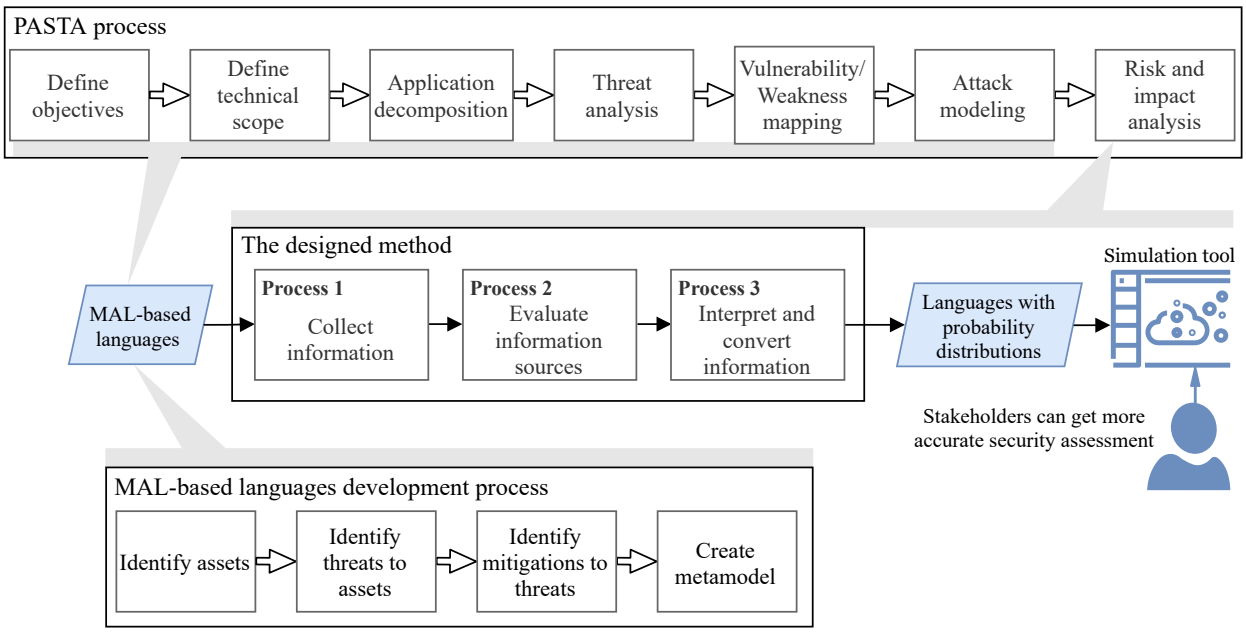


Figure 4. Overview process for attack simulation and threat (PASTA) analysis

PASTA is a seven-stage methodology that addresses the most viable threats to an application or system environment target. The inputs/outputs of the PASTA process include people, information sources, and artifacts to be created. Figure 4 shows the typical activities

[13] https://github.com/mal-lang/

of creating a MAL-based language within the first six stages of the PASTA process. For instance, the application decomposition action in PASTA is related to identifying assets in the development of MAL-based languages. However, in language development, we focus on the identification of reusable concepts, such as a Mac OS, while PASTA takes a step further and identifies concrete systems (e.g., Mac Big Sur 11.2.3) that realize these concepts. Thus, we recommend performing at least two iterations of the PASTA process when using MAL-based languages. In the first iteration, MAL-based languages are created or aligned with the needs of the organization. In the second iteration, MAL-based languages are used to create a concrete instance representing the organization.

However, the method presented here assumes that the activities mentioned above have already been performed and thus focuses on the identification of probability distributions, which should be attached to the identified attack steps. This directly relates to the risk and impact analysis stage of PASTA, as we perform a deeper analysis of the potential attack vectors and identify concrete efforts that a possible attacker needs to spend.

Our method is composed of three sub-processes: 1) collecting information for the domain in which the MAL-based language takes place, 2) evaluating the sources through credibility assessment, and 3) interpreting and converting information into probability distributions. By applying this method to MAL-based languages and conducting attack simulations, the MAL-based languages can provide more realistic simulation results of their system model instances. Therefore, stakeholders can assess the security of the system and investigate the security settings that can be implemented to secure the system more effectively.

5.1 Collect Information for Domain under Assessment

To collect information as input for the process, different sources are possible, such as performing systematic literature reviews (SLRs) (e.g., [40], [41]), interviewing experts [39], conducting studies [42], or making an estimation based on existing sources [43]. Considering SLRs, we expand their scope (see Figure 5) to include not only traditional academic publishing, such as journal papers and conference proceedings, but also online platforms such as security events (e.g., blackhat[14]), vulnerability databases (e.g., NVD[15]), blogs, and technical reports, since, oftentimes, the recent cyber threats are publicly disclosed online [44].

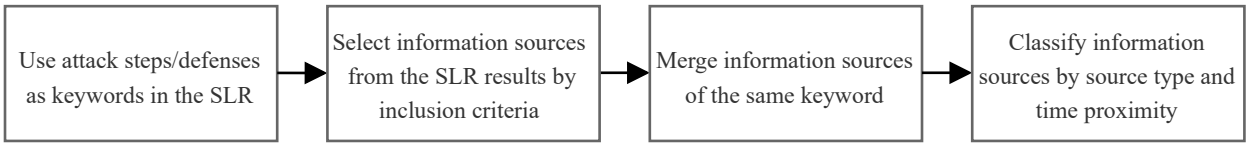


Figure 5. Overview of the information collection process

The information collection process based on SLRs includes four steps, as shown in Figure 5, and each step is detailed as follows:

1. For each MAL-based language, the list of system asset-related attack steps and defenses from the language as keywords is used to perform an SLR.
2. Information sources are selected by the following inclusion criteria:
   - The information source must focus on cyber security.
   - The information source must concentrate on the domain in which the MAL-based language takes place.

[14] https://www.blackhat.com/
[15] https://nvd.nist.gov/

- The information source must contain information describing the effort required (e.g., the time required, success probability) for an attacker to compromise one or more attack steps.

3. Information sources of the same keyword are merged for further analysis. The information sources collected for the same keyword contain quantitative information about the time required for an attacker to perform an attack step and/or the uncertainties of performing an attack step successfully.
4. All the collected information sources are stored with the metamodel shown in Figure 6. For each information source, we store its source link and classify the source by its source type (ST) and time proximity (TP). After all the collected sources are classified, they are prepared for further evaluation.

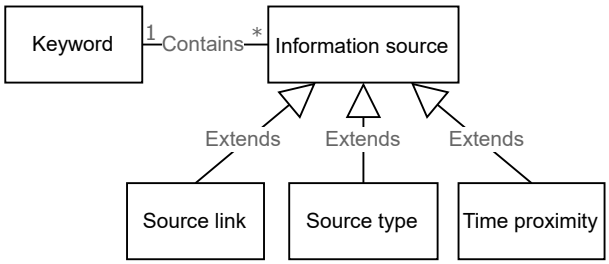


Figure 6. Metamodel of the collected information sources

5.2 Evaluate Information Sources by Credibility Assessment

Because the collected information sources are of various types and include qualitative studies, it is necessary to assess their quality [40]. During this process, we apply a set of credibility assessment heuristics [45] to evaluate the credibility of each information source, including ST and TP. They are presented in the form of these causal rules: "The more persistent the memory, the higher the credibility of the source" and "The shorter the time since the source was published, the higher the credibility of the source."

Then, we parameterize the applied individual heuristics, where a higher value on a heuristic represents a higher credibility of that type of source. We calculate the estimated credibility value for each information source by combining the parameterized heuristics and the overall credibility value C_{source} for a *source* can be calculated using the fundamental probability calculations:

$$C_{source} = 1 - (1 - P(ST))(1 - P(TP)) \quad (1)$$

where ST and TP represent the applied heuristics and the probability values $P(ST)$ and $P(TP)$ denote the parameterized heuristic values.

For instance, if a *source* is classified as *Database* by ST and *Less than 1 year* by TP, then we estimate its credibility as C_{source} = 1 - (1 - $P(ST = Database)$)(1 - $P(TP = Less\ than\ 1\ year)$)[16].

5.3 Interpret and Convert Information into Probability Distributions

An overview of the process of interpreting and transforming information into probability distributions is shown in Figure 7. For each keyword, we analyze the quantitative information contained in the collected information sources $source_1, ..., source_{m+n}$, and classify them into 1) the time needed (t) to perform the attack step $source_{t,1}, ..., source_{t,m}$, and 2) the success probability of performing the attack step or the defense is implemented as $source_{s,1}, ..., source_{s,n}$.

[16] For concrete values, refer to Section 6.2 and Table 1.

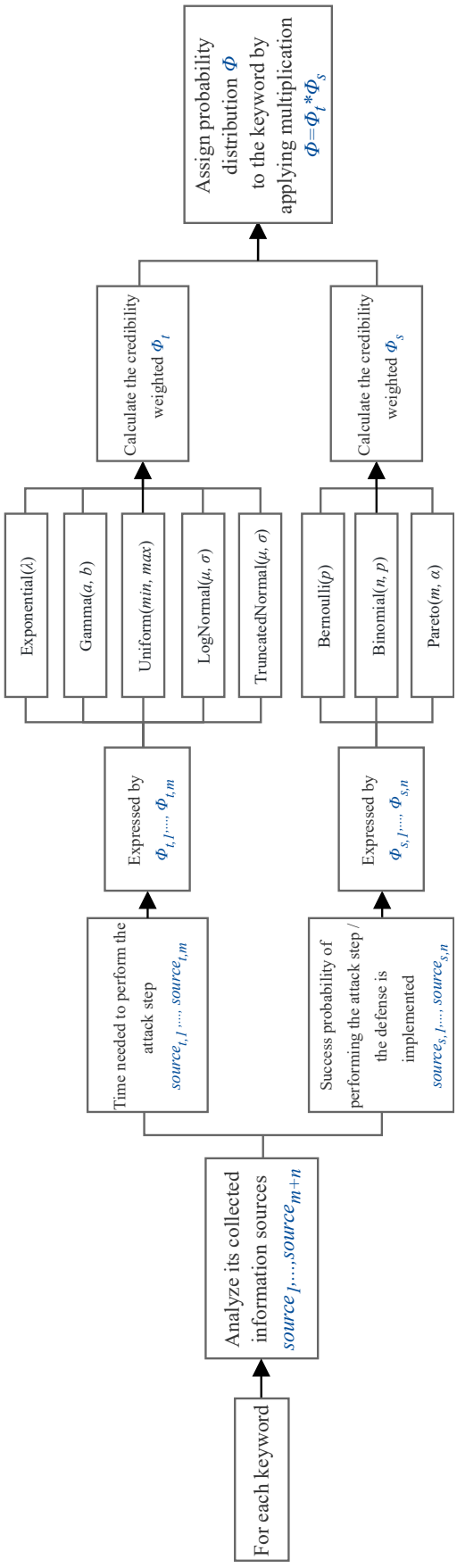


Figure 7. Overview of the interpreting and transforming information to probability distributions process

For $source_{t,1}, ..., source_{t,m}$, we convert each of them into a certain probability distribution $\Phi_{t,1}, ..., \Phi_{t,m}$ with parameters, specifying that the time it takes to perform the attack follows a certain probability distribution. Defined by the MAL framework[17], available distribution functions that express the time needed include exponential, gamma, uniform, log-normal, and truncated normal distributions. For instance, if a $source_{t,1}$ indicates the time required to perform an attack step *compromise* is $1/\lambda$ days, then we express it with an exponential distribution with parameter λ, i.e., $\Phi_{t,1} = \text{Exponential}(\lambda)$.

Next, we calculate the credibility weighted Φ_t of the above sources. For i sources that are expressed by the same distribution function, we calculate the credibility weighted parameter θ of their distribution functions:

$$\theta = \frac{\sum_{source=1}^{i} C_{source} * \theta_{source}}{\sum_{source=1}^{i} C_{source}} \tag{2}$$

where θ_{source} is the parameter of its corresponding distribution derived from each source.

Thus, the resulting aggregated probability distribution Φ_t describing the time required is expressed by

$$\begin{aligned}\Phi_t(\theta) = (&\sum_{source=1}^{m1} C_{t,source} * Exponential(\lambda) + \sum_{source=1}^{m2} C_{t,source} * Gamma(a,b) + \\ &\sum_{source=1}^{m3} C_{t,source} * Uniform(min,max) + \sum_{source=1}^{m4} C_{t,source} * LogNormal(\mu,\sigma) + \\ &\sum_{source=1}^{m5} C_{t,source} * TruncatedNormal(\mu,\sigma))/\sum_{source=1}^{m} C_{t,source}\end{aligned} \tag{3}$$

where $m1, ..., m5$ are the number of sources that correspond to a specific distribution function, and m is their sum.

Similarly, for $source_{s,1}, ..., source_{s,n}$, we convert the quantitative information into probability distributions $\Phi_{s,1}, ..., \Phi_{s,n}$ with parameters. Defined by the MAL framework, available distribution functions that can express the uncertainties include Bernoulli, binomial, and Pareto distributions. For instance, if a $source_{s,1}$ indicates that an attack step *compromise* can be performed immediately with a probability of p, then we express it with a probability distribution $\Phi_{t,1}$=Bernoulli(p).

Then, we calculate the credibility weighted parameter θ of the distribution function according to Equation (2) and the aggregated probability distribution $\Phi_s(\theta)$ describing the uncertainties of performing an attack step or a defense is implemented as follows:

$$\begin{aligned}\Phi_s(\theta) = (&\sum_{source=1}^{n1} C_{s,source} * Bernoulli(p) + \sum_{source=1}^{n2} C_{s,source} * Binomial(n,p) + \\ &\sum_{source=1}^{n3} C_{s,source} * Pareto(m,\alpha))/\sum_{source=1}^{n} C_{s,source}\end{aligned} \tag{4}$$

where $n1, ..., n3$ are the number of sources that correspond to a specific distribution function, and n is their sum.

Therefore, the probability distribution Φ is associated with the keyword by applying multiplication $\Phi = \Phi_t * \Phi_s$, specifying the time required and uncertainties in performing it. In addition, if we have either Φ_t or Φ_s (i.e., no information sources are collected for the

[17] https://github.com/mal-lang/malcompiler/wiki/Supported-distribution-functions

uncertainties or the time needed), we assume the other is equal to 1, so that $\Phi = \Phi_t$ or $\Phi = \Phi_s$.

By following the above processes, we can assign probability distributions to the corresponding attack steps and defenses needed for a MAL-based language.

6 Demonstration

In this section, we demonstrate the proposed method by assigning probability distributions to enterpriseLang.

6.1 Collect Information for Enterprise Systems

To collect information sources for enterprise systems, the existing attack steps and defenses in enterpriseLang were used as keywords to perform an SLR. In total, 266 unique attack steps and 41 defences were used as keywords for our SLR. We searched for these keywords on Google Scholar[18] and expanded our search on Google[19] to include online platform resources. We assessed all relevant information sources and based on the given inclusion criteria (cf. Section 5.1), 120 unique information sources were selected for further analysis.

The collected 120 information sources contain information needed for 198 attack steps and 12 defenses. Then, we merged the information sources of the same keyword (i.e., attack step and defense) in one markdown file for further interpretation. A list of all the markdown files is available on Github[20].

Next, we stored the collected information sources (cf. Figure 8) and classified the information sources according to their ST and TP for further credibility assessment. According to the ST, the collected sources were classified into online platforms and traditional academic publishing, where the plurality (82.5%) of the information sources are online platforms, which include *Blogs*, *Reports*, *Net stats*, *Analyst papers*, *Surveys*, and *Databases*. Only 17.5% of the sources are collected from traditional publishing, including *Journal papers*, *Conference papers*, *Books*, and *Student theses*. According to the TP, we classified the collected sources into *Less than 1 year*, *1 to 5 years*, and *More than 5 years*. Overall, the classification of all collected information sources is shown in Figure 9.

	A	B	C
1	**Information source**	**Source Type (ST)**	**Time Proximity (TP)**
2	https://www.recordedfuture.com/mitre-attack-tactics/	Analyst papers	Less than 1 year
3	https://www.varonis.com/blog/data-breach-statistics/	Blogs	Less than 1 year
4	The Art of Memory Forensics: Detecting Malware and Threats in W	Books	More than 5 years
5	A View on Current Malware Behaviors	Conference papers	More than 5 years
6	https://capec.mitre.org/data/definitions/646.html	Databases	1 to 5 years
7	Breaching the Human Firewall: Social engineering in Phishing and	Journal papers	More than 5 years
8	https://www.statista.com/statistics/271037/distribution-of-most-con	Net stats	1 to 5 years
9	http://www.storiprotection.fr/EagleEye.pdf	Reports	1 to 5 years
10	IoT-lang: Threat modeling for Internet of Things	Student theses	Less than 1 year
11	https://www.checkpoint.com/downloads/products/ransomware-defe	Surveys	1 to 5 years

Figure 8. Screenshot of the collected information sources

[18] https://scholar.google.com/
[19] https://www.google.com/
[20] https://github.com/mal-lang/enterpriseLang/tree/master/enterpriselang%20probabilities

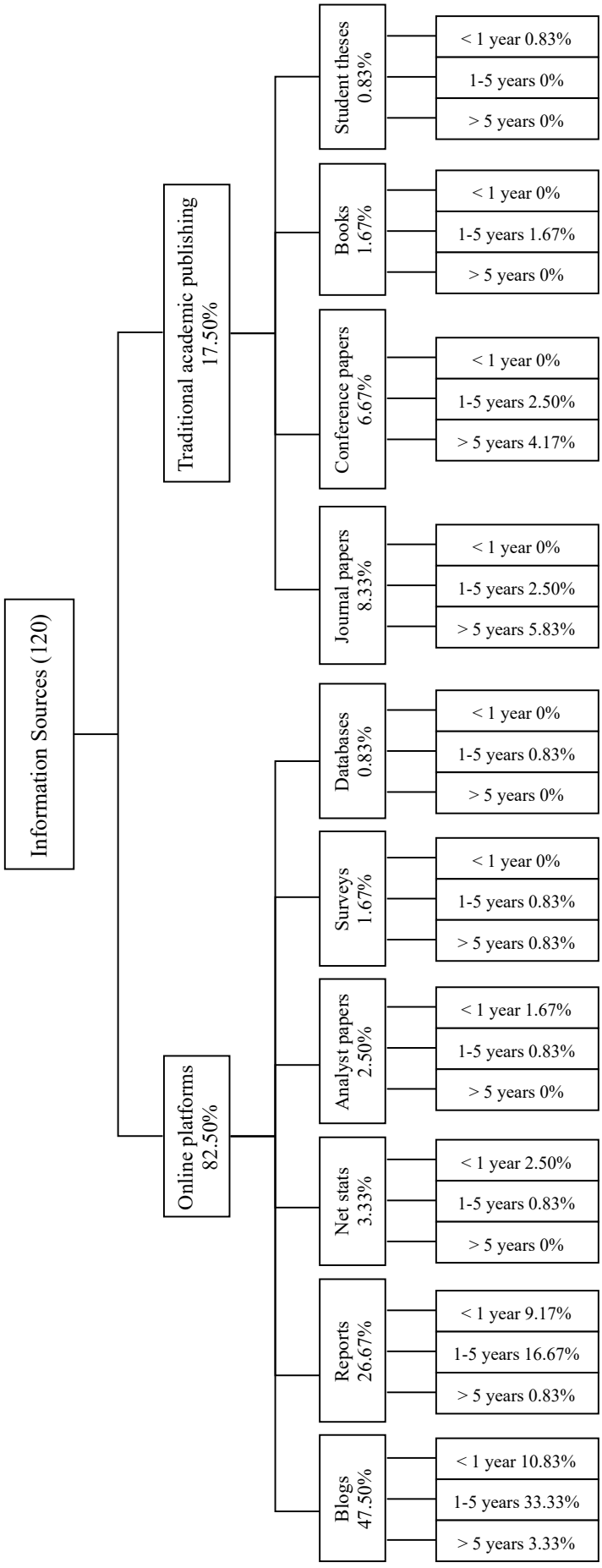


Figure 9. Classification of the collected information sources

6.2 Evaluate Information Sources by Credibility Assessment

The collected information sources are of various types, and thus, of various quality. During this process, we evaluated the credibility of each source by applying parameterized heuristics. We set the probability values for the individual heuristics (e.g., $P(ST = Blogs)$), ranging from 0% to 100%, where a higher probability value on a heuristic equals a higher credibility of that type of source. The overall parameterized heuristics applied in this study are listed in Table 1.

If a *source* collected for a keyword is classified as *Blogs* by ST and *Less than 1 year* by TP, then we applied their parameterized heuristic values $P(ST = Blogs)$ and $P(TP = Less\ than\ 1\ year)$ from Table 1. Next, we calculated the overall credibility C_{source} according to Equation (1). For instance, the source [46] was collected for the keyword *resourceHijacking* and was classified into *Analyst papers* and *Less than 1 year*. Therefore, we estimated the overall credibility of the source C_{source} = 1 - (1 - $P(ST = Analyst\ papers)$)(1 - $P(TP = Less\ than\ 1\ year)$) = 1 - (1 - 0.6)(1 - 0.75) = 90%.

Table 1. Parameterized heuristics applied in this work

Heuristics	Var	Parameters	Probability Values
Source Type	ST	$P(ST = Books)$	90%
		$P(ST = Journal\ papers)$	85%
		$P(ST = Conference\ papers)$	80%
		$P(ST = Student\ theses)$	75%
		$P(ST = Reports)$	70%
		$P(ST = Surveys)$	65%
		$P(ST = Analyst\ papers)$	60%
		$P(ST = Databases)$	55%
		$P(ST = Net\ stats)$	40%
		$P(ST = Blogs)$	20%
Time Proximity	TP	$P(TP = Less\ than\ 1\ year)$	75%
		$P(TP = 1\ to\ 5\ years)$	50%
		$P(TP = More\ than\ 5\ years)$	25%

6.3 Interpret and Convert Information into Probability Distributions

During the process, we analyzed all the quantitative information collected for each keyword in its markdown file.

For instance, three sources [47], [48], [49] were collected for the keyword *processInjection*. First, we analyzed the quantitative information of the sources. Two of them were classified into the time required to perform the attack step, where $source_{t,1}$ [47] states that it takes an attacker 11 min to trigger a process injection, and $source_{t,2}$ [48] states that the process injection can be triggered in three min. However, [49] was classified into the success probability of compromising the step (i.e., $source_{s,1}$) because it states that 35% of organizations are affected by process injection.

During the previous process (cf. Section 6.2), $source_{t,1}$ was classified as *Blogs* by ST and *1 to 5 years* by TP. Therefore, we calculated its credibility value according to Equation (1) and Table 1, i.e., $C_{t,1} = 60\%$. Similarly, we calculated that the credibility value of $source_{t,2}$ is $C_{t,2} = 60\%$, and $source_{s,1}$ is $C_{s,1} = 80\%$.

According to the MAL framework, both $source_{t,1}$ and $source_{t,2}$ were expressed with an exponential distribution function. According to Equation (2), we calculated $\theta = 205.7$. Then, we calculated that Φ_t = Exponential(205.7) according to Equation (3).

Similarly, $source_{s,1}$ was expressed by the Bernoulli distribution function. Because it was the only source collected for the success probability, we calculated Φ_s = Bernoulli(0.35) according to Equations (2) and (4), meaning that it can be performed immediately with a probability of 35%, and not at all with a probability of 65%.

By applying multiplication, we assigned Φ = Exponential(205.7) * Bernoulli(0.35) to the *processInjection* attack step. Therefore, in the code base of enterpriseLang, we assigned the probability distribution to the related attack steps and defenses as follows:

```
asset OS {
    & processInjection [Exponential(205.7) * Bernoulli(0.35)]
        ->  threadExecutionHijacking,
            ...

    | threadExecutionHijacking

    # behaviorPreventionOnEndpoint
        ->  processInjection,
            ...
}
```

7 First Evaluation

In this section, we evaluated the designed method for simulating known cyber attacks on system model instances using two different versions of enterpriseLang. Therefore, we conducted attack simulations and compared the simulation results provided by enterpriseLang against binary relations and probability distributions by following the designed method to determine whether the latter could provide a more realistic security assessment by comparing the simulation results to a documented attack.

According to a security alert[21], attackers are increasingly using password spraying as a brute force against organizations around the world. These attacks can have severe impacts on networked systems, including the loss of sensitive information and disruption to regular operations. We illustrate this example in the following.

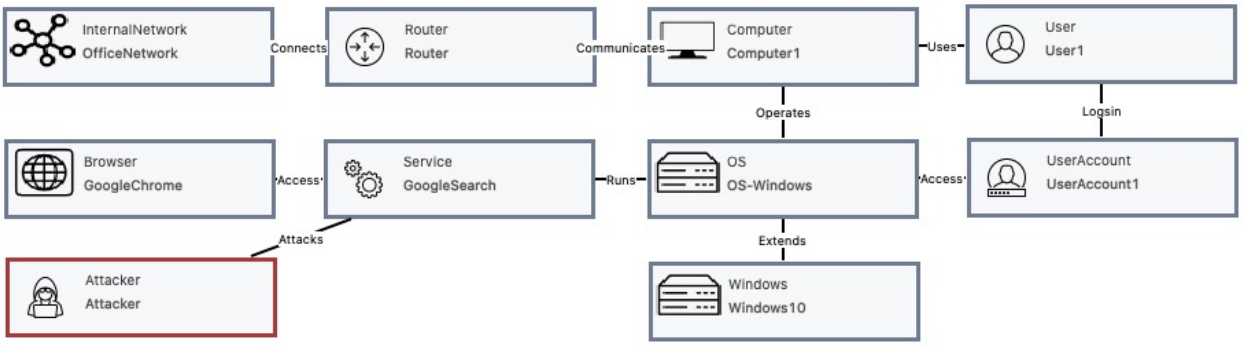


Figure 10. Simplified system model

Figure 10 shows the simplified system model created with enterpriseLang to simulate the behavior of the attackers, where each asset (e.g., `OS-Windows`) contains multiple attack steps (e.g., *passwordSpraying*) and defenses (e.g., *multiFactorAuthentication*) that are connected. The attackers first performed online research (i.e., `GoogleSearch`) and sent spearphishing messages to target organizations and user accounts (e.g., `UserAccount1`). Then, they used a sub-technique of *bruteForce* known as *passwordSpraying* against these accounts to gain *userCredentials*. By leveraging the *userCredentials*, the attackers performed

[21] https://us-cert.cisa.gov/ncas/alerts/TA18-086A

an *emailCollection* from the `UserAccount1` and a larger *passwordSpraying* against them. The attackers then attempted to expand laterally through *remoteDesktopProtocol* within the `OfficeNetwork`, and performed mass *dataExfiltration* using *fileTransferProtocols*. A successful compromised system can have severe impacts, including temporary or permanent loss of sensitive or proprietary information, and financial losses incurred to restore systems and files.

Given the system model (cf. Figure 10), we performed attack simulations in a simulation tool called securiCAD [5], which transforms MAL-based languages into attack graphs. We specified the entry point of the attack as *spearphishingViaService*. Figure 11 shows one of the shortest attack paths that results in *dataExfiltration* from the simulation results, where each node represents an attack step reachable by attackers and green circles represent possible defenses that can be implemented by stakeholders within the system to prevent certain attack steps.

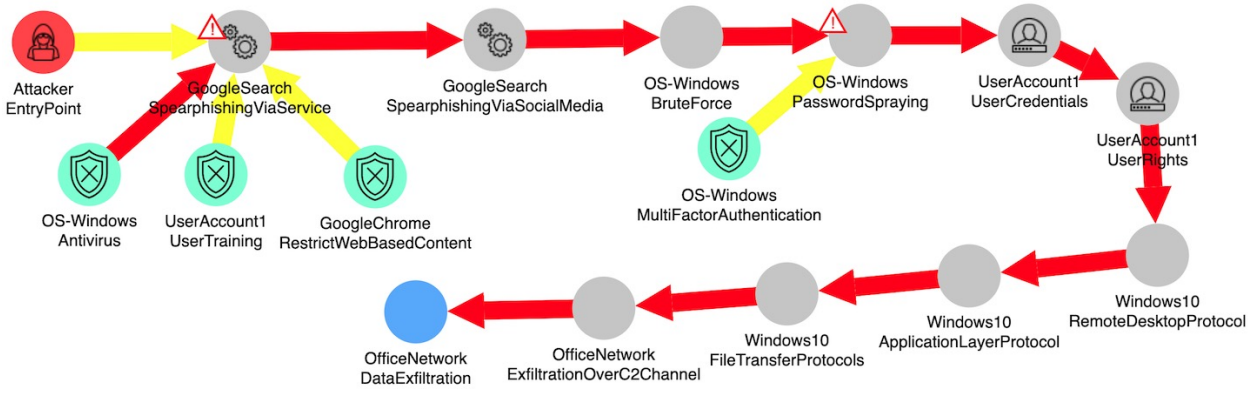
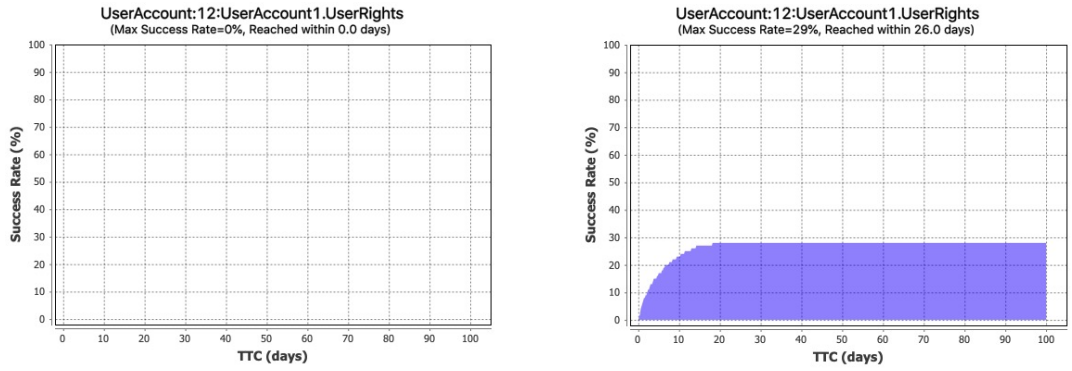


Figure 11. Attack path to perform data exfiltration

Here, we compare the simulation results provided by enterpriseLang with binary relations to the version with probability distributions by calculating the global time to compromise (i.e., global TTC) of *userRights*. According to the MAL framework [2], rational attackers would select the shortest path to reach various attack steps. Therefore, the global TTC of *userRights* is the shortest time required for attackers to reach *userRights* by attempting various avaliable attack steps and compromising individual attack steps from the entry point *spearphishingViaService* (see Figure 11). The comparison results are shown in Figure 12.



(a) enterpriseLang with binary relations (b) enterpriseLang with distributions

Figure 12. Simulation results comparison of the global TTC to compromise a certain attack step

As shown in Figure 12(a), when we use enterpriseLang with binary relations, the global TTC of compromising *userRights* has a maximum success rate of 0% when

multiFactorAuthentication is enabled. When we use enterpriseLang with probability distributions (cf. Figure 12(b)), the global TTC of compromising *userRights* has a maximum success rate of 29% when *multiFactorAuthentication* is enabled, and the attack step can be reached within 26 days.

While it is difficult to verify the global TTC value with real-world attacks following the same attack sequence, the source[22] identified that 29% of users submitted their credentials after clicking the spearphishing link, which is the same as the result produced by enterpriseLang with distributions (cf. Figure 12(b)). Therefore, enterpriseLang with probability distributions could provide more realistic global TTC values than with binary relations, where an attack is either always and immediately effective or cannot be performed at all.

8 Discussion

In this article, we proposed a method to assign probability distributions to attack steps and defenses for MAL-based languages. We further applied the method to assign probability distributions for a MAL-based language called enterpriseLang, where the MITRE ATT&CK Matrix has been transformed into an attack/defense graph with probabilities of the relations. Although our approach produces more realistic simulation results than previous approaches, our method has some limitations.

First, we solely considered the literature sources in our demonstration for setting our probabilities. The advantage of relying on existing literature is that this information can be collected relatively easily and it is traceable. The downside is that some sources could be outdated. To address this issue, we emphasize more recent literature by considering the publication date. However, there are still other sources that could be considered, such as vulnerability scores and databases, vulnerability scanners, expert knowledge, logs and alerts, and system state information [50]. Considering additional sources can increase the accuracy of the simulation results, but it comes with more effort (e.g., conducting several interviews with experts), or will create organization-specific outcomes (e.g., using scanners). In particular, the latter is not generalizable to the language level.

Second, the quantitative information collected that describes the efforts needed for attackers to exploit certain attack steps can be diverse. At the moment, we compute a weighted average based on a calculated score that represents the trustworthiness of the different sources. A common problem with average calculations is that extreme values distort the result and, hence, may produce unrealistic simulation results in our case. Such extreme values might be compensated by a large number of values. Unfortunately, our probabilities are usually determined by three sources on average, and thus, our simulation results might be influenced by extreme values. To balance this, the consideration of further information sources can be helpful, e.g., an expert can help to decide on the value.

Further, the parameterized probability values shown in Table 1 were set without any further research. Thus, changing these parameters will lead to other simulation results. However, this does not harm our approach per se, since our demonstration was successful, and our first evaluation showed that the simulation results could be improved even without perfect parameterization. Nonetheless, future efforts should be made to find better parameterizations, leading to more realistic simulation results.

Finally, the method was designed and tested by one team of MAL language developers and reviewed by two other MAL language developers. The main feedback received was considering the applicability of the method in other domains, e.g., industrial control systems, where

[22] https://www.prnewswire.com/news-releases/cyber-security-report-reveals-factors-that-contribute-to-high-click-rates-and-high-risk-of-credential-theft-300995105.html

there are fewer published sources available. This is mainly due to confidentiality in critical infrastructure. The proposed method can be extended to other types of sources that are more suitable for a certain domain. When including expert opinions, the suggested determination of probability values can still be used (with modifications).

9 Conclusion

In this article, we propose a method to assign probability distributions to attack steps and defenses for MAL-based DSLs. The proposed method fits into the PASTA process and contains three sub-processes. We demonstrated the proposed method by assigning probability distributions to a MAL-based DSL called enterpriseLang. By conducting attack simulations on a system model instance using two different versions (i.e., binary relations and probability distributions) of enterpriseLang, the one with probability distributions shows more realistic simulation results than the one with binary relations. Therefore, stakeholders can assess the security of the system and investigate the security settings that can be implemented to secure the system more effectively.

However, there is still some work that needs to be done. First, we solely considered literature-based sources for our probabilities, and other information sources could also be included so that the missing probabilities for attack steps and defenses can be added and the existing ones can potentially be revised and updated. Second, the quantitative information collected that describes the efforts needed for attackers to exploit certain attack steps can be very diverse (i.e., far from each other); thus, further information sources should be included to become more certain about the result. Our future work will also include evaluating the method with other developers to reveal shortcomings in the method and misinterpretations from our side.

Apart from the MAL-based DSLs, the proposed method could also be used for more general attack/defense graph/tree-based approaches, because they have the same need to interpret and convert numeral values from many existing sources to probabilities to analyze the security of a system or to harden the system for better security [25]. However, further research is needed to generalize our proposal or to confirm that it works as is.

In order to evaluate MAL-based languages ensuring their correct functionality, our current and future work also includes conducting qualitative assessment by investigating if the language development process follows the principles of good language design [51], [52], [53], and quantitative assessment by creating test cases and assessing the test coverage [54].

Acknowledgements

The authors would like to thank Quentin Biharé and Love Wessman for their contributions to the project and helpful discussions.

This work has received funding from European Union's H2020 research and innovation programme under Grant Agreement no. 832907, the Swedish Energy Agency, Vinnova (Sweden's innovation agency), and the STandUP for Energy programme.

References

[1] J. P. Shim, R. Sharda, A. M. French, R. A. Syler, and K. P. Patten, "The internet of things: Multi-faceted research perspectives," *Communications of the Association for Information Systems*, pp. 511–536, 2020. [Online]. Available: https://doi.org/10.17705/1CAIS.04621

[2] P. Johnson, R. Lagerström, and M. Ekstedt, "A meta language for threat modeling and attack simulations," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*. ACM, 2018, pp. 1–8. [Online]. Available: https://doi.org/10.1145/3230833.3232799

[3] A. Singhal and X. Ou, "Security risk analysis of enterprise networks using probabilistic attack graphs," in *Network Security Metrics*. Springer, 2017, pp. 53–73. [Online]. Available: https://doi.org/10.1007/978-3-319-66505-4_3

[4] W. Xiong and R. Lagerström, "Threat modeling - a systematic literature review," *Computers & Security*, vol. 84, pp. 53–69, 2019. [Online]. Available: https://doi.org/10.1016/j.cose.2019.03.010

[5] M. Ekstedt, P. Johnson, R. Lagerström, D. Gorton, J. Nydrén, and K. Shahzad, "Securi cad by foreseeti: A cad tool for enterprise cyber security management," in *2015 IEEE 19th International Enterprise Distributed Object Computing Workshop (EDOCW)*. IEEE, 2015, pp. 152–155. [Online]. Available: https://doi.org/10.1109/EDOCW.2015.40

[6] H. Holm, K. Shahzad, M. Buschle, and M. Ekstedt, "P^2CySeMoL: Predictive, probabilistic cyber security modeling language," *IEEE Transactions On Dependable And Secure Computing*, vol. 12, no. 6, pp. 626–639, 2015. [Online]. Available: https://doi.org/10.1109/TDSC.2014.2382574

[7] W. Xiong, E. Legrand, O. Åberg, and R. Lagerström, "Cyber security threat modeling based on the MITRE Enterprise ATT&CK Matrix," *submitted*, 2020.

[8] S. Hacks, S. Katsikeas, E. Ling, R. Lagerström, and M. Ekstedt, "powerlang: a probabilistic attack simulation language for the power domain," *Energy Informatics*, vol. 3, no. 1, pp. 1–17, 2020. [Online]. Available: https://doi.org/10.1186/s42162-020-00134-4

[9] J. Jürjens, "Umlsec: Extending uml for secure systems development," in *UML 2002: «UML» 2002 - The Unified Modeling Language*. Springer, 2002, pp. 412–425. [Online]. Available: https://doi.org/10.1007/3-540-45800-X_32

[10] A. A. Cardenas, T. Roosta, and S. Sastry, "Rethinking security properties, threat models, and the design space in sensor networks: A case study in scada systems," *Ad Hoc Networks*, vol. 7, no. 8, pp. 1434–1447, 2009. [Online]. Available: https://doi.org/10.1016/j.adhoc.2009.04.012

[11] B. Schneier, "Attack trees," *Dr. Dobb's Journal of Software Tools*, vol. 24, no. 12, pp. 21–29, 1999.

[12] B. Schneier, *Secrets and Lies: Digital Security in a Networked World*. New York: John Wiley & Sons, 2000.

[13] S. Mauw and M. Oostdijk, "Foundations of attack trees," in *International Conference on Information Security and Cryptology*. Springer, 2005, pp. 186–198. [Online]. Available: https://doi.org/10.1007/11734727_17

[14] B. Kordy, S. Mauw, S. Radomirović, and P. Schweitzer, "Foundations of attack–defense trees," in *International Workshop on Formal Aspects in Security and Trust*. Springer, 2010, pp. 80–95. [Online]. Available: https://doi.org/10.1007/978-3-642-19751-2_6

[15] K. Ingols, M. Chu, R. Lippmann, S. Webster, and S. Boyer, "Modeling modern network attacks and countermeasures using attack graphs," in *Computer Security Applications Conference*. IEEE, 2009, pp. 117–126. [Online]. Available: https://doi.org/10.1109/ACSAC.2009.21

[16] I. Kotenko and E. Doynikova, "Evaluation of computer network security based on attack graphs and security event processing," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 5, no. 3, pp. 14–29, 2014.

[17] X. Ou and A. Singhal, "Attack graph techniques," *Quantitative Security Risk Assessment of Enterprise Networks*, pp. 5–8, 2011. [Online]. Available: https://doi.org/10.1007/978-1-4614-1860-3_2

[18] L. Williams, R. Lippmann, and K. Ingols, "Garnet: A graphical attack graph and reachability network evaluation tool," in *VizSec 2008: Visualization for Computer Security*. Springer, 2008, pp. 44–59. [Online]. Available: https://doi.org/10.1007/978-3-540-85933-8_5

[19] J. Homer, S. Zhang, X. Ou, D. Schmidt, Y. Du, S. R. Rajagopalan, and A. Singhal, "Aggregating vulnerability metrics in enterprise networks using attack graphs." *Journal of Computer Security*, vol. 21, no. 4, pp. 561–597, 2013. [Online]. Available: https://doi.org/10.3233/JCS-130475

[20] L. Wang, S. Jajodia, A. Singhal, P. Cheng, and S. Noel, "k-zero day safety: A network security metric for measuring the risk of unknown vulnerabilities," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 1, pp. 30–44, 2014. [Online]. Available: https://doi.org/10.1109/TDSC.2013.24

[21] S. Noel, M. Elder, S. Jajodia, P. Kalapa, S. O'Hare, and K. Prole, "Advances in topological vulnerability analysis," in *Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications Technology*, 2009, pp. 124–129. [Online]. Available: https://doi.org/10.1109/CATCH.2009.19

[22] N. Ghosh, I. Chokshi, M. Sarkar, S. K. Ghosh, A. K. Kaushik, and S. K. Das, "NetSecuritas: An integrated attack graph-based security assessment tool for enterprise networks," in *Proc. of the 2015 Int. Conf. on Distributed Computing and Networking*. ACM, 2015, p. 30. [Online]. Available: https://doi.org/10.1145/2684464.2684494

[23] J. Hoff, "Creating attack graphs for adversary emulation, simulation and purple teaming in industrial control system (ics) environments," Master's Thesis, University of Hagen, Germany, 2021.

[24] M. Frigault, L. Wang, A. Singhal, and S. Jajodia, "Measuring network security using dynamic bayesian network," in *Proc. of the 4th ACM workshop on Quality of protection*. ACM, 2008, pp. 23–30. [Online]. Available: https://doi.org/10.1145/1456362.1456368

[25] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, "An attack graph-based probabilistic security metric," in *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 2008, pp. 283–296. [Online]. Available: https://doi.org/10.1007/978-3-540-70567-3_22

[26] M. Välja, F. Heiding, U. Franke, and R. Lagerström, "Automating threat modeling using an ontology framework," *Cybersecurity*, vol. 2, pp. 1–20, 2020. [Online]. Available: https://doi.org/10.1186/s42400-020-00060-8

[27] P. Närman, P. Johnson, R. Lagerström, U. Franke, and M. Ekstedt, "Data collection prioritization for system quality analysis," *Electronic Notes in Theoretical Computer Science*, vol. 233, pp. 29–42, 2009. [Online]. Available: https://doi.org/10.1016/j.entcs.2009.02.059

[28] R. Lagerström, U. Franke, P. Johnson, and J. Ullberg, "A method for creating enterprise architecture metamodels - applied to systems modifiability analysis," *International Journal of Computer Science and Applications*, vol. 6, no. 5, pp. 89–120, 2009.

[29] M. Ekstedt, U. Franke, P. Johnson, R. Lagerström, T. Sommestad, J. Ullberg, and M. Buschle, "A tool for enterprise architecture analysis of maintainability," in *2009 13th European Conference on Software Maintenance and Reengineering*. IEEE, 2009, pp. 327–328. [Online]. Available: https://doi.org/10.1109/CSMR.2009.44

[30] T. Sommestad, M. Ekstedt, and H. Holm, "The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures," *IEEE Systems Journal*, vol. 7, no. 3, pp. 363–373, 2013. [Online]. Available: https://doi.org/10.1109/JSYST.2012.2221853

[31] P. Johnson, A. Vernotte, M. Ekstedt, and R. Lagerström, "pwnpr3d: an attack-graph-driven probabilistic threat-modeling approach," in *Availability, Reliability and Security (ARES), 2016 11th International Conference on*. IEEE, 2016, pp. 278–283. [Online]. Available: https://doi.org/10.1109/ARES.2016.77

[32] S. Katsikeas, S. Hacks, P. Johnson, M. Ekstedt, R. Lagerström, J. Jacobsson, M. Wällstedt, and P. Eliasson, "An attack simulation language for the it domain," in *Graphical Models for Security*. Springer International Publishing, 2020, pp. 67–86. [Online]. Available: https://doi.org/10.1007/978-3-030-62230-5_4

[33] I. Gashi, V. Stankovic, C. Leita, and O. Thonnard, "An experimental study of diversity with off-the-shelf antivirus engines," in *2009 Eighth IEEE International Symposium on Network Computing and Applications*, 2009, pp. 4–11. [Online]. Available: https://doi.org/10.1109/NCA.2009.14

[34] M. Butavicius, K. Parsons, M. Pattinson, and A. McCormac, "Breaching the human firewall: Social engineering in phishing and spear-phishing emails," in *Australasian Conference on Information Systems*, 2015, pp. 1–10.

[35] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer, "Social phishing," *Communications of the ACM*, vol. 50, no. 10, p. 94–100, 2007. [Online]. Available: https://doi.org/10.1145/1290958.1290968

[36] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004. [Online]. Available: https://doi.org/10.2307/25148625

[37] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007. [Online]. Available: https://doi.org/10.2753/MIS0742-1222240302

[38] T. UcedaVélez and M. M. Morana, "Intro to pasta," in *Risk Centric Threat Modeling*. John Wiley & Sons, Inc, 2015, pp. 317–342.

[39] S. Katsikeas, P. Johnson, S. Hacks, and R. Lagerström, "Probabilistic modeling and simulation of vehicular cyber attacks: An application of the meta attack language," in *Proceedings of the 5th International Conference on Information Systems Security and Privacy (ICISSP)*, 2019. [Online]. Available: https://doi.org/10.5220/0007247901750182

[40] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," School of Computer Science and Mathematics, Keele University, Tech. Rep., 2007.

[41] A. Booth, A. Sutton, and D. Papaioannou, *Systematic Approaches to a Successful Literature Review*. Sage, 2016.

[42] J. Loxdal, M. Andersson, S. Hacks, and R. Lagerström, "Why phishing works on smartphones: A preliminary study," in *Proceedings of the 54th Hawaii International Conference on System Sciences*, 2021. [Online]. Available: https://doi.org/10.24251/HICSS.2021.863

[43] M. A. McQueen, W. F. Boyer, M. A. Flynn, and G. A. Beitel, "Time-to-compromise model for cyber risk reduction estimation," in *Quality of Protection*, D. Gollmann, F. Massacci, and A. Yautsiukhin, Eds. Boston, MA: Springer US, 2006, pp. 49–64. [Online]. Available: https://doi.org/10.1007/978-0-387-36584-8_5

[44] K. Mandia, "FireEye Shares Details of Recent Cyber Attack, Actions to Protect Community," Available: https://www.fireeye.com/blog/products-and-services/2020/12/fireeye-shares-details-of-recent-cyber-attack-actions-to-protect-community.html, 2020.

[45] E. Johansson and P. Johnson, "Assessment of enterprise information security - estimating the credibility of the results," in *Proceedings of the Symposium on Requirements Engineering for Information Security (SREIS 05)*, 2005.

[46] P. Technologies, "Cybersecurity threatscape: Q1 2019," Available: https://www.ptsecurity.com/ww-en/analytics/cybersecurity-threatscape-2019-q1, 2019.

[47] J. Walter, "Trickbot update: Brief analysis of a recent trickbot payload," Available: https://labs.sentinelone.com/trickbot-update-brief-analysis-of-a-recent-trickbot-payload/, 2019.

[48] R. Mendrez, "Tale of the two payloads – trickbot and nitol," Available: https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/tale-of-the-two-payloads-trickbot-and-nitol/, 2017.

[49] Red Canary, "2020 Threat Detection Report," Available: https://redcanary.com/threat-detection-report/techniques/process-injection/, 2020.

[50] E. Ling, R. Lagerström, and M. Ekstedt, "A systematic literature review of information sources for threat modeling in the power systems domain," in *Critical Information Infrastructures Security*. Springer International Publishing, 2020, pp. 47–58. [Online]. Available: https://doi.org/10.1007/978-3-030-58295-1_4

[51] G. Karsai, H. Krahn, C. Pinkernell, B. Rumpe, M. Schindler, and S. Völkel, "Design guidelines for domain specific languages," in *9th OOPSLA Workshop on Domain-Specific Modeling (DSM' 09)*, 2009, pp. 1–7.

[52] G. Czech, M. Moser, and J. Pichler, "Best practices for domain-specific modeling. a systematic mapping study," in *Proc. of the 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2018, pp. 137–145. [Online]. Available: https://doi.org/10.1109/SEAA.2018.00031

[53] G. Kahraman and S. Bilgen, "A framework for qualitative assessment of domain-specific languages," *Software & Systems Modeling*, vol. 14, pp. 1505–1526, 2015. [Online]. Available: https://doi.org/10.1007/s10270-013-0387-8

[54] N. Hersén, S. Hacks, and K. Fögen, "Towards measuring test coverage of attack simulations," in *Proc. of the 25th International Conference, EMMSAD 2021, Held at CAiSE 2021 (to be published)*, 2021, pp. 1–15.