Cyber security threat modeling based on the MITRE Enterprise ATT&CK Matrix

Journal:	*Software and Systems Modeling*
Manuscript ID	Draft
Manuscript Type:	Regular Paper
Keyword:	Threat modeling, Domain-specific language, Attack simulations, Enterprise systems

Note: The following files were submitted by the author for peer review, but cannot be converted to PDF. You must view these files (e.g. movies) online.

SoSyM2020.zip

Noname manuscript No.
(will be inserted by the editor)

Cyber security threat modeling based on the MITRE Enterprise ATT&CK Matrix

Wenjun Xiong[1] · Emeline Legrand[2] · Oscar Åberg[1] · Robert Lagerström[1]

Received: date / Accepted: date

Abstract Enterprise systems are growing in complexity, and the adoption of cloud and mobile services has greatly increased the attack surface. To proactively address these security issues in enterprise systems, this paper proposes a threat modeling language for enterprise security based on the MITRE Enterprise ATT&CK Matrix. It is designed using the Meta Attack Language (MAL) framework and focuses on describing system assets, attack steps, defenses, and asset associations. The attack steps in the language representing adversary techniques are listed and described by MITRE. This entity-relationship model describes enterprise IT systems holistically; by using available tools, the proposed language enables attack simulations on its system model instances. These simulations can be used to investigate security settings and architectural changes that might be implemented to secure the system effectively. The effectiveness of our proposed language is tested and verified by modeling two real-world cyber attack scenarios.

Keywords Threat modeling · Domain-specific language · Attack simulations · Enterprise systems

Wenjun Xiong
wenjx@kth.se
Emeline Legrand
emeline.legrand@ensta-paristech.fr
Oscar Åberg
noav@kth.se
Robert Lagerström
robertl@kth.se

[1] KTH Royal Institute of Technology, Stockholm, Sweden

[2] ENSTA ParisTech, Paris, France

1 Introduction

Cyber security continues to be a key concern and fundamental aspect of enterprise IT systems. Recent years saw some of the largest, most sophisticated, and most severe cyber attacks, such as WannaCry,[1] the Equifax breach,[2] and the Facebook data leak,[3] which affected millions of consumers and thousands of businesses. In addition, cloud computing has become a major enterprise IT trend today and further increases the attack surface. For example, the instance metadata API featured in public cloud platforms can be used as a Trojan horse that can be queried by an adversary via the API to obtain access credentials to the public cloud environment by any process running on the instance.[4]

Threat modeling [43] is one approach to proactively improving the security of enterprise systems; it has attracted increasing attention and includes holistic identification of the main assets within a system and threats to these assets. It is used to both assess the current and future states of a system and as a security-by-design tool for developing new systems. A recent advance in threat modeling is to couple it with attack simulations to provide probabilistic evaluations of security, e.g., the Time-To-Compromise (TTC) [15,17]. On the basis of such objective evaluations, security controls can be chosen to counter anticipated threats.

[1] https://www.csoonline.com/article/3227906/what-is-wannacry-ransomware-how-does-it-infect-and-who-was-responsible.html

[2] https://www.wired.com/story/equifax-breach-no-excuse/

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

[4] https://redlock.io/blog/instance-metadata-api-a-modern-day-trojan-horse

In this paper, we propose a threat modeling language for enterprise systems called enterpriseLang. It is a domain-specific language (DSL) based on the Meta Attack Language (MAL) framework [15]. The MITRE Enterprise ATT&CK Matrix[5] serves as a knowledge base for our proposed threat modeling language, which describes adversary behaviors in order to measure the resilience of an enterprise system against various cyber attacks. The MITRE ATT&CK database contains useful information for a threat modeling language, such as assets (e.g., Computer, Service, OS, Firewall, Internal and External Network), attack steps (e.g., Spearphishing Attachment, User Execution, and Data Destruction) for these assets, and defenses (e.g., Privileged Account Management, Execution Prevention, and Network Segmentation) against these attack steps. By using available tools, the designed language allows attack simulations on instance models for a specific enterprise IT system and supports analysis of which security settings might be used to secure the system effectively.

The rest of this paper is structured as follows. In Section 2, we review the state of the art in threat modeling, attack simulations, and enterprise architecture (EA). Section 3 describes the background of this paper, including the MITRE Enterprise ATT&CK Matrix and the Meta Attack Language. Section 4 describes the design methodology of the proposed language. Section 5 describes enterpriseLang in detail. In Section 6, we test the proposed language. Our work is discussed in Section 7 and finally concluded in Section 8.

2 Related Work

In this section, we review the state-of-the-art research papers and software tools for securing IT systems.

2.1 Threat Modeling

Threat modeling is used to analyze potential attacks or threats and can be supported by threat libraries or attack taxonomies [36]. It provides a structured approach to secure software design in which an adversary's goal in attacking a system is understood in terms of the system assets [2]. The work of Shostack [32] and the Microsoft Threat Modeling tool[6] are commonly used in this area. They are used mainly to design secure software applications and do not consider the system holistically, e.g., by taking into account software, data, infrastructure, and processes. A well-known initiative for holistic threat modeling from a system-wide perspective is UMLsec [18], which is an extension of the Unified Modeling Language (UML) for developing security-critical systems. In addition, Cardenas et al. [6] provided a holistic view of the security requirements and threat models of sensor networks, with a focus on high-level security goals. They also developed a function to capture the overall effect of various attacks on high-level security requirements.

2.2 Attack Simulations

Threats can also be modeled using attack trees or attack graphs [20,28–30,39]. These methods aim to show all the paths through a system that end in a state where an adversary has successfully achieved his or her goal. Some work, e.g., the MAL, also provides probabilistic simulation results [15]. Furthermore, several attack-graph-based tools have been developed. For example, MulVAL [14] derives logical attack graphs by associating vulnerabilities extracted from scans with the probability that an adversary could successfully conduct an attack, and k-Zero Day Safety [38] extends MulVAL to the computation of zero-day attack graphs. In addition, NAVIGATOR [7] considers the identified vulnerabilities as directly exploitable, assuming that an adversary has access to the vulnerable system. Moreover, the topological vulnerability analysis (TVA) tool [25] models security conditions in networks and uses a database of exploits as transitions between the security conditions. Similarly, NetSecuritas [10] uses scanner output and known exploits to generate attack graphs and the corresponding security recommendations, e.g., the mitigation metric. These tools are commonly based on the available information about existing systems or infrastructures, from which attack graphs can be automatically generated.

Some researchers have investigated the combined use of threat modeling and attack simulations in domains such as energy [37] and vehicular infrastructures [19,42], and enterprise security modeling languages have been proposed. CySeMoL [34] is a cyber security modeling language for enterprise-level system architectures; P^{2}CySeMol [13], which is based on CySeMoL, is an attack graph tool for estimating the cyber security of EAs. It couples attacks on and defenses of objects in a system architecture that the system owner can easily model and understand. P^{2}CySeMol differs from MulVAL, k-Zero Day Safety, and the TVA tool in that all the attack and defense steps are related by Bayesian networks. In addition, pwnPr3d [17] was proposed as a probabilistic threat modeling approach for automatic attack graph generation; it offers a holistic

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

[6] https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling/

approach and provides both a high-level overview and technical details. The common idea is to automatically generate attack graphs for a given system specification that include a predictive security analysis of the system model.

2.3 Enterprise Architecture

EA has become an established discipline in business and software system management [27]. EA models can be used to increase the general understanding of enterprise systems and perform various types of analysis [21]. A key underlying assumption is that they should provide more aggregated knowledge than the information that was initially modeled, as in holistic threat modeling and attack simulations.

Metamodels are the core of EA and describe the fundamental artifacts of enterprise systems. These high-level models provide a clear view of the structure of and dependencies between relevant parts of an organization [40]. Lagerström et al. [21] described some factors that need to be considered when creating a metamodel for EA analysis. First, many factors affect the system properties. Second, these factors are related in a complex manner. The researcher or practitioner who sets out to model these interdependencies thus inevitably faces an unreasonably large number of modeling choices, all of which to some extent affect the ability of the final assessment framework to support decision-making.

2.4 Tools Based on the ATT&CK Framework

Most threat modeling and attack simulations work remains be done manually, which can be time-consuming and error-prone [12,24]. To extend the automation level, Applebaum et al. [1] designed an automated adversary emulation testbed based on the ATT&CK framework, which focuses on the tactical level to model adversaries operating within a Windows enterprise network. Similarly, CALDERA[7] was designed as an automated adversary emulation system based on the ATT&CK framework; it enables automated assessments of a network's susceptibility to adversary success by associating abilities with an adversary and running the adversary in an operation. However, none of the tools covers the full range of attacks (techniques) found and detailed by the MITRE ATT&CK Matrix.

[7] https://github.com/mitre/caldera

According to a technical report,[8] the ATT&CK Matrix has not been applied in published research yet. Using a combination of the above disciplines, we propose a threat modeling language that can assess the enterprise resilience against various cyber attacks. The information on assets, associations, adversary techniques, and mitigations is extracted from the ATT&CK Matrix framework. The proposed language enables users to model enterprise systems holistically and generate attack graphs for system models.

3 Background

3.1 MITRE ATT&CK Matrix for Enterprise

MITRE ATT&CK is a globally accessible knowledge base of adversary tactics and techniques based on real-world observations. This knowledge base can be used as a foundation for the development of specific threat models and other types of methodologies and tools. Our focus here is on its Enterprise Matrix.[9]

3.1.1 Adversary Tactics

The Enterprise Matrix contains 12 tactics representing an adversary's tactical objective for acting:

- `Initial Access`. This tactic represents the techniques used by adversaries to establish a foothold in an enterprise system. For instance, they may launch a spearphishing campaign, e.g., *Spearphishing Attachment*; steal user credentials using *Valid Accounts*; or use removable media, e.g., a compromised USB stick, to break into the system.
- `Execution`. After gaining initial access to a local or remote computer, adversaries may directly execute malicious code by techniques such as interaction via a *Command-Line Interface* or *Graphical User Interface*, or wait for *User Execution* to trigger exploitation.
- `Persistence`. The footholds gained by adversaries through `Initial Access` within an enterprise system may be eliminated when users change their passwords. To maintain access, adversaries may hijack legitimate code on the victim system to remain and move deeper into the system.
- `Privilege Escalation`. Adversaries often enter an enterprise system with unprivileged access, and they

[8] https://www.slideshare.net/attackcon2018/mitre-attckcon-20-flashback-with-attck-exploring-malware-history-with-attck-20032018-kris-oosthoek-delft-university

[9] https://attack.mitre.org/matrices/enterprise/

may acquire more resources within the victim system and elevate their permissions. Specifically, they may gain increased privileges by exploiting vulnerabilities in applications and servers within the enterprise system.

- `Defense Evasion`. To avoid detection and bypass security controls, adversaries often clear or cover their traces to continue their malicious activities.
- `Credential Access`. To achieve malicious objectives and maintain access to the victim system, adversaries may capture more usernames and passwords through the *Bash History* or *Keychain* of a compromised computer.
- `Discovery`. After gaining access to an enterprise system, adversaries may attempt to explore and gather more information about the system to support their objectives. These attempts include the discovery of possible vulnerabilities to exploit, data stored in the system, and network resources through *Network Service Scanning*.
- `Lateral Movement`. After compromising one asset within the enterprise network, adversaries may shift from the compromised user account to other user accounts within an office area through techniques such as *Internal Spearphishing*, which enable them to exploit the trusted internal accounts to increase the probability of tricking other users.
- `Collection`. Adversaries may collect data that help them achieve their malicious objectives. Data can be collected from a compromised computer or its peripheral devices (e.g., webcams or USB memory) using the *Data from Removable Media* technique. The next step can be data exfiltration.
- `Command and Control`. This tactic enables adversaries to control their operations within an enterprise system remotely. When adversaries have control over the enterprise, their compromised computers may then become botnets within the enterprise that can be controlled by the adversaries.[10]
- `Exfiltration`. After data are collected, adversaries may package it using techniques such as *Data Compression* to minimize the data size transferred over the network, making the exfiltration less conspicuous to bypass detection.
- `Impact`. Adversaries can breach the confidentiality, degrade the integrity, and limit the availability of assets within an enterprise system after achieving their objectives. For instance, *Disk Structure Wipe* and *Disk Content Wipe* can be used to make computers unable to boot and reboot.

[10] https://www.malwarepatrol.net/command-control-servers-c2s-fundamentals/

The number of adversary techniques included in the above tactics ranges from 9 (for the `Exfiltration` tactic) to 69 (for the `Defense Evasion` tactic).

3.1.2 Adversary Techniques

The above tactics are treated as tags for adversary techniques, depending on the malicious intent. For instance, the *Valid Accounts* technique is categorized as four tactics: `Initial Access`, `Defense Evasion`, `Privilege Escalation`, and `Persistence`. If adversaries aim to gain `Initial Access` to a system, they may steal the credentials of a specific user or service account using *Valid Accounts*, whereas if they wish to bypass security controls (i.e., `Defense Evasion`), they may use the compromised *Valid Accounts* within the enterprise network to make them harder to detect.

A total of 266 techniques are listed in the Enterprise ATT&CK Matrix. Twelve of these techniques from the above list are chosen as examples to illustrate how adversaries use them to achieve their malicious tactical goals.

Spearphishing Attachment. Adversaries commonly conduct spearphishing campaigns, e.g., by sending spear-phishing emails with malicious attachments or links to trick users into sharing their user credentials. For example, a cyber attack on the Ukraine power grid [31] was initiated by sending a spearphishing email with a malicious file attached to a Microsoft Office Word document. When an employee opened the document and executed the file, the attackers then penetrated the office network. A possible mitigation is *User Training*, where enterprises can decrease the risk by conducting security awareness training; consequently, employees would be more aware of these social engineering attacks and know how to behave if tricked.

User Execution. Adversaries may not be the only ones involved in a successful attack; sometimes users may involuntarily help by performing what they believe are normal activities. *User Execution* can be performed in two ways: executing the malicious code directly or using a browser-based or application exploit that triggers users to execute the malicious code. For instance, after conducting a spearphishing campaign, adversaries will rely on users to download malicious attachments or click malicious links to gain execution.

Create Account. When adversaries have obtained admin accounts from an enterprise system, they might not use them directly for malicious activities because these accounts are more frequently monitored and could thus trigger security alarms. To avoid losing access, adversaries may create local accounts to ensure their continued presence.

Exploitation for Privilege Escalation. Some vulnerabilities in operating systems (e.g., CVE-2014-4076[11]) and applications can be exploited by adversaries to gain higher system permissions such as admin privileges. To counter this technique and make it difficult for them to advance their operations, enterprise servers and software can be updated regularly to patch these vulnerabilities.

Disabling Security Tools. Adversaries try to avoid detection of their tools and activities; for instance, they may try to disable security software or event logging processes, delete registry keys so that tools do not start at run time, or use other methods of interfering with security scanning or event reporting.

Keychain. Keychain is a built-in tool in macOS that stores user passwords and accounts. An adversary who knows the credential access for the login to Keychain can access all the other credentials stored in it. To make it harder for adversaries to access user credentials, additional credentials need to be used.

Network Service Scanning. Adversaries may attempt to obtain a list of network services running within an enterprise system by using network and vulnerability scanners, e.g., the Nmap scanner,[12] and search for vulnerabilities within the victim system. For example, when given an address, Nmap will report open ports and the services they support, as well as the users that provide the services. This technique enables adversaries to learn more about the victim system.

Internal Spearphishing. Even when employees are trained, they may not be fully capable of detecting spearphishing attacks, especially those in emails sent from a trusted employee within the same enterprise [23]. Internal spearphishing is used when the account credentials of an employee have already been compromised during `Credential Access`, and the compromise is not easily discovered by a detection system.

Data from Removable Media. Adversaries are often interested in sensitive information. Sensitive data can be collected from removable media, e.g., USB memory, that are connected to a compromised computer. This technique can be used before data exfiltration, for instance.

Commonly Used Port. Adversaries may conduct C2 communications over commonly used ports, e.g., ports 80 (HTTP) and 443 (HTTPS), so that their communications are mixed with normal network activities and bypass network detection systems. *Network Intrusion Prevention* can be used to thwart such attempts at the network level.

Data Compressed. After sensitive data are collected, an adversary may compress the data to make them portable before sending them over the network. The data are compressed according to a program or algorithm, and transmission can be prevented by using *Network Intrusion Prevention* to block certain file types such as ZIP files.

Disk Content Wipe. Adversaries may try to maximize their impact on the target enterprise system by limiting the availability of system and network resources. They may wipe specific disk structures or files or arbitrary portions of disk content. *Data Backup* can be used to recover the data.

Adversaries often combine techniques from many different tactics to achieve broader goals. For example, adversaries may expand their damage to the victim system by using techniques from other tactics, such as *Data Destruction*, to limit the availability of data stored on a computer. These techniques are applied during an attack from an entry point such as a hardware/software component to successfully compromise a target enterprise system using a multistage approach. For instance, to perform a spearphishing attack, an adversary may first use the *Spearphishing Attachment* (belonging to the `Initial Access` tactic) to attach a file to the spearphishing email. If a user downloads the malicious attachment, the adversary can exploit the system upon *User Execution* [35].

There are multiple methods of defense against each technique.[13] For instance, *Spearphishing Attachment* and *User Execution* can both be mitigated by *User Training*, where employees can be trained to identify certain social engineering techniques. Thus, they will be more suspicious of spearphishing campaigns. Note that not all techniques can be mitigated.

The MITRE Enterprise ATT&CK Matrix contributes to our proposed language by providing adequate information about adversary techniques, that is, the platforms, required permissions, mitigations, and possible combinations of the techniques, to create threat models of enterprise systems.

3.2 Meta Attack Language

The proposed enterpriseLang is based on the MAL. The MAL is a threat modeling language framework that combines probabilistic attack and defense graphs with object-oriented modeling, which in turn can be used to create DSLs and automate the security analysis of instance models within each domain. The MAL modeling hierarchy is shown in Figure 1.

[11] https://nvd.nist.gov/vuln/detail/CVE-2014-4076
[12] https://nmap.org/
[13] https://attack.mitre.org/mitigations/enterprise/

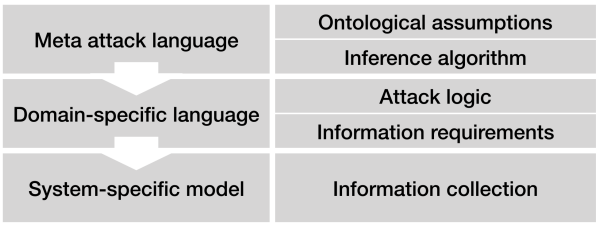


Fig. 1 MAL modeling hierarchy [15]

The construction of a domain-specific threat modeling language is based on an understanding of the system (domain) that is being modeled and its scope. For enterprise systems, we collect information about the system assets, asset associations, and possible attack/defense steps for each asset. A domain model can easily become too complex if the scope is too broad or too detailed. When the domain is understood well and the scope is set, the next step is to create the DSL. DSLs such as vehicular IT infrastructures [19] and Amazon Web Services [33] have been created. For the enterprise domain, enterpriseLang can be used to generate attack graphs for enterprise systems automatically and suggest appropriate mitigations. It follows Java-like coding standards, so it efficiently describes domain assets (e.g., `OS`), specific instances (e.g., `Linux 19.3`), attack steps (e.g., *bashHistory*), and defense (mitigation) steps (e.g., *operatingSystemConfiguration*).

3.2.1 MAL Symbols

The most common MAL symbols used in enterpriseLang are shown in Table 1 and are excerpted from the MAL Syntax.[14] Attack steps are connected to each other, and each of them is of the type OR (represented by |) or AND (represented by &). It is important to thoroughly analyze each attack step and find potential defenses as well as the possible subsequent attack steps. One successfully compromised attack step can lead to a second step (represented by "–>"). A combination of attack steps is sometimes required to reach a second attack step (thus, the resulting attack step is of type &). Defense steps (represented by #) have Boolean values to indicate their status, where "enabled" or "disabled" is represented by setting the defense value to TRUE or FALSE, respectively. If the value is FALSE, the associated attack step against which it defends can be reached.

3.2.2 MAL Coding Standards

MAL follows Java-like coding standards, where:

- Asset names start with an uppercase letter, e.g., asset `User`, `OS`, `Linux`.
- Attack steps are given in camelCase, e.g., *userCredentials*, *bashHistory*.
- Role names are in lowercase, e.g., `user`.

In the following basic MAL example, *bashHistory* on `Linux` can be the starting point for an adversary to initiate an attack, which is defended by *operatingSystemConfiguration*. If the defense is disabled (i.e., its value is FALSE), the *userAccount.userCredentials* attack step will be reached, which enables an attack on the *userCredentials* of the connected `UserAccount` asset. By contrast, if the defense is enabled, the *userCredentials* step will not be reached. Furthermore, *userCredentials* itself can also be an entry point for an attack, for example, if the adversary has already obtained a `UserAccount`, which may lead to other attack steps (not shown in this example).

```
asset UserAccount {
    | userCredentials
}

asset Linux {
    & bashHistory
        ->  userAccount.userCredentials

    # operatingSystemConfiguration
        ->  bashHistory
}

associations {
UserAccount [userAccount] 1 <--Access--> * [linux] Linux
}
```

As in UML class diagrams,[15] association ends are bound to types with multiplicities. Similarly, in this MAL example, `UserAccount` and `Linux` assets are associated by association `Access`. Moreover, one `UserAccount` can `Access` multiple (*) `Linux` operating systems.

The relevant disciplines and background sources that contribute to our designed enterpriseLang are shown in Figure 2.

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 [11]. Thus, the application of DSR is appropriate here as it guides the development of enterpriseLang. In this work, enterpriseLang is designed

[14] https://github.com/mal-lang/mal-documentation/wiki

[15] http://www.agilemodeling.com/artifacts/classDiagram.htm

Table 1 MAL symbols

Symbol	Meaning	Description
–>	Leads to	Successful compromise by this attack step allows the adversaries to then execute further attack steps, or this defense step can defend against one or more further attack steps.
|	OR	An OR attack in which step A can be reached if any of the attack steps that refer to A are reached.
&	AND	An AND attack in which step A can be reached only when all of the attack steps that refer to A are reached.
#	Defense	Unlike attack steps, defenses are Boolean. A defense step represents the countermeasure to an attack step. A defense step can be enabled or disabled by setting the defense step value to TRUE or FALSE, respectively.
E	Existence	This operation is used when the existence of a connected/associated asset is checked. It acts as a defense, but the Boolean value is automatically assigned according to the existence of the asset.
+>	Add operator	This operation is applied to the child asset. One attack step (e.g., access) under the child asset results in that attack step as well as the corresponding attack step on the parent asset.
X.A	Collect operator	Attack step A on asset X is referenced.
TTC	Time-To-Compromise	TTC is a probability distribution that reflects an adversary's ability to compromise an asset. Attack steps can have a TTC, and the MAL simulations calculate/aggregate the TTC over models/scenarios.

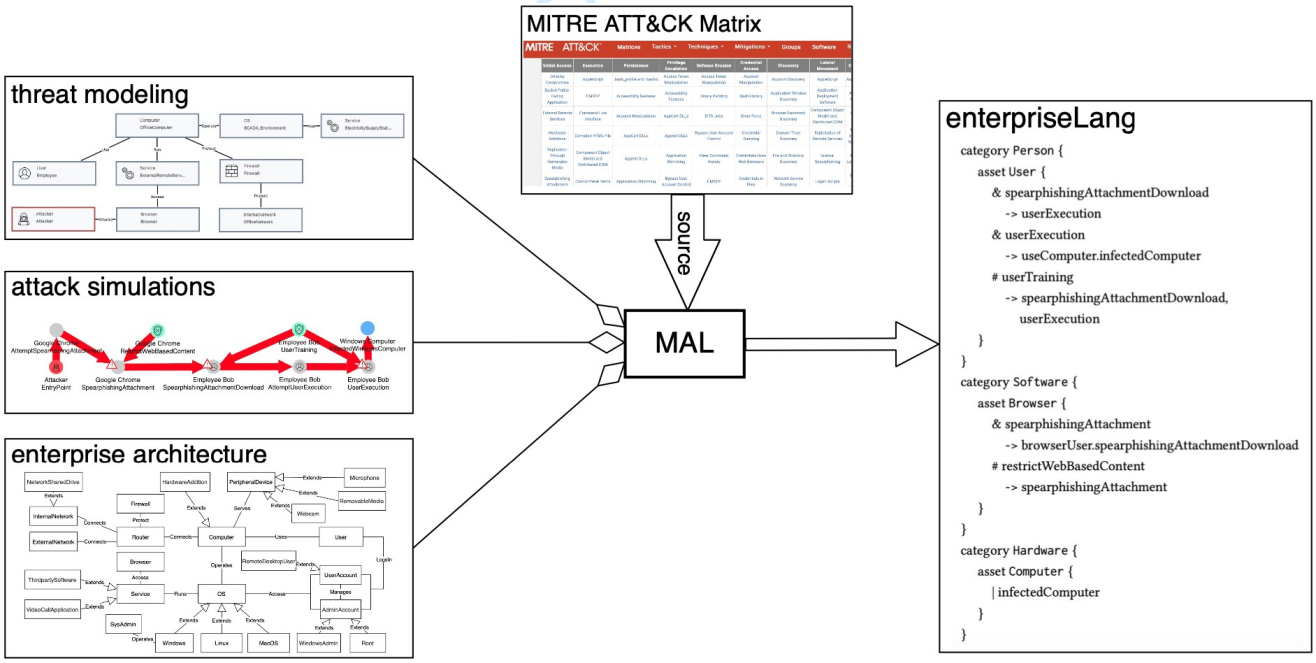


Fig. 2 Contributions of various resources and disciplines to enterpriseLang

according to the DSR guidelines of Peffers et al. [26], which include six steps:

– `Step 1: Identify Problem & Motivate`:
Because cyber security is a key concern for enterprise IT systems, it is necessary to increase the security level of enterprise systems so that they are more resistant to cyber attacks. This goal can be achieved by modeling threats to essential IT assets and the associated attacks and mitigations. This work aims to develop a threat modeling language for assessing the cyber security of enterprise IT systems. By using available tools, the proposed language enables the simulation of attacks on its system model instances and supports analysis of the security settings that might be implemented to secure the system effectively.
– `Step 2: Define Objectives`:
To develop a holistic threat model of the enterprise

system, it is important to obtain reasonable coverage of attacks on enterprise systems and understand how these attacks can be associated. The full range of attacks/defenses (techniques/mitigations) detailed by the MITRE ATT&CK Matrix is covered in our proposed enterpriseLang, and the associations between attacks/defenses are described using MAL symbols. In addition, to determine which security settings can be applied for a specific enterprise, attacks can be simulated using the system model instantiated in enterpriseLang, and enterpriseLang supports analysis of which security settings may be useful.

– `Step 3: Design & Development`:
The MITRE ATT&CK Matrix is used as a knowledge base, and MAL is used as the underlying modeling framework for enterpriseLang. First, the DSL, enterpriseLang, is constructed according to the construction process described in Section 5.1; it can be compiled to generate a generic attack graph. In addition, a metamodel containing essential enterprise IT assets and associations is modeled during the construction process. Furthermore, to threat model a specific enterprise system, a tool called securiCAD [9] can be used to simulate attacks on the system model. Therefore, enterpriseLang can affect to-be models of a specific enterprise system by providing simulation results for different security settings.
– `Step 4 & 5: Demonstration & Evaluation`:
To demonstrate enterpriseLang, we created a set of threat models for two real-world cyber attacks. The two cases are demonstrated using an attack graph excerpted from the generic attack graph of enterpriseLang, which shows the attack and defense steps for the relevant system model assets, as well as how they are associated.
Each case is evaluated to determine (1) whether the techniques used are present in enterpriseLang and behave as expected and (2) whether enterpriseLang can provide security assessments and suggest security settings that might be implemented on the basis of a threat model and attack simulation of the system.
– `Step 6: Communication`. The research is communicated by the publication of the paper itself and the peer review process of the journal. In addition, enterpriseLang is an open-source project, and the entire code base of enterpriseLang, including instructions on how to use it, is publicly available from the GitHub repository.[16]

[16] https://github.com/mal-lang/mitreattacklang

5 Enterprise Threat Modeling Language

enterpriseLang is designed as an adversary-technique-based threat modeling language that can assess the security of enterprise systems against various attacks. It is a DSL based on the MAL framework (see Section 3.2). In this section, the construction process of enterpriseLang is described, and its underlying metamodel is created during this process.

5.1 Construction Process

The construction process of enterpriseLang has three steps: (1) extracting information for each adversary technique from the ATT&CK Matrix (e.g., *Access Token Manipulation*), (2) converting the extracted information into MAL files (e.g., *accessTokenManipulation.mal*), and (3) combining the created files into one language (i.e., *enterpriselang.mal*).

5.1.1 Extracting Technique Information from the ATT&CK Matrix

The goal of the first step is to extract sufficient information about each adversary technique needed for our threat modeling language. The extracted information includes the following:

– `Technique`. A total of 266 enterprise techniques are extracted from the ATT&CK Matrix.
– `Description`. According to the description of each technique, the adversary information is extracted, i.e., how an adversary can exploit this technique and how this technique can be combined with other techniques to achieve broader goals. For example, an adversary performs a *Spearphishing Attachment* attack and relies upon *User Execution* to realize execution.
– `Platform`. This information represents the platform on which an adversary can implement a technique. The platform could be a Computer, Service, or OS. For example, *Keychain* is a feature of macOS that records user passwords and credentials for many services and features; thus, the platform for using *Keychain* is macOS.
– `Permissions Required`. This information indicates the minimum permission level required for an adversary to use a technique. For instance, the permission required to perform *Process Discovery* is Administrator, and thus an adversary with a `UserAccount` could not use this technique. In addition, some adversary techniques, such as *Spearphishing Attachment*, do not require any permissions.

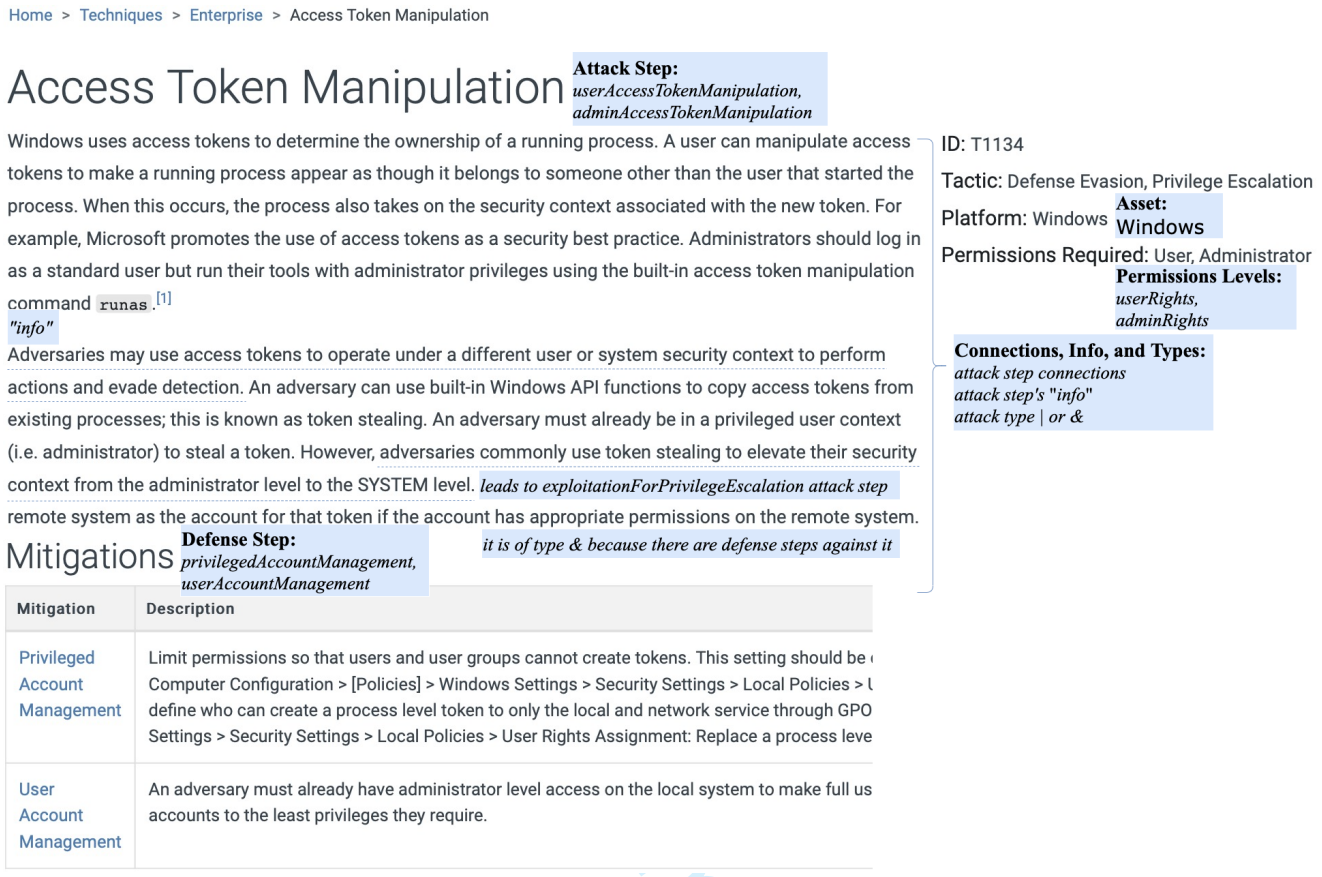


Fig. 3 Example of extraction and conversion of information from the Access Token Manipulation technique. Screenshot from the MITRE ATT&CK Matrix

– `Mitigation`. In the ATT&CK Matrix, each technique has multiple mitigations. A mitigation method prevents a technique from working or having the desired outcome. For example, the methods of mitigating *Access Token Manipulation* include *Privileged Account Management* and *User Account Management*, where the former limits permissions so that users and user groups cannot create tokens, and the latter can be applied to limit users and accounts to the least privileges they require so that they cannot make full use of this technique.

5.1.2 Converting Adversary Techniques into MAL Files

After the above items are extracted for each adversary technique, they are converted by applying MAL symbols and coding standards to the following items. We take *Access Token Manipulation* as an example to show the process, which is illustrated in Figure 3.

– `Attack Step`. Each technique can be converted to one (or more) attack step(s). For example, *Access Token Manipulation* is converted to two attack steps, *userAccessTokenManipulation* and *adminAccessTokenManipulation*, as specified by its `Permissions Required`, which are User and Administrator.
– `Defense Step`. `Mitigations` of a technique can be converted to `Defenses` against an `Attack`. A `Defense Step` can be related to (–>) multiple `Attack Steps`, and one `Attack Step` can be related to multiple `Defense Steps`.
– `Connections, Info, and Types`. According to the `Description` of each technique, (1) the attack step connections, including "how an adversary can use this technique" and "how an adversary can take advantage of this technique," can be converted using the "–>" symbol to represent the consequence of this attack. An adversary is likely to try multiple paths after completing one attack step; thus, one attack step can lead to ("–>") multiple further attack steps. (2) The "info" for an attack step provides information for end-users about the associated attack/defense steps. (3) The attack type of each attack step can be specified as type | or &; it is of type | when an adversary can start working on this

attack step as soon as one of its parent attack steps is completed, and it is of type & when all of its parent attack steps have to be completed to reach this step, or there is at least one `Defense` against this `Attack`.

- `Asset`. Assets reflect where adversaries can make an `Attack` or an enterprise can implement a `Defense`. An `Attack/Defense Step` can be placed under multiple `Assets`. For example, *Logon Scripts* are related to both macOS and Windows; thus, when this information is converted to a MAL file, *logonScripts* are assigned to both the `macOS` and `Windows` assets.
- `Permissions Levels`. The `Permissions Levels` include *userRights* (for the `UserAccount` asset) and *adminRights* (for the `AdminAccount` asset), where `WindowsAdmin` is specified for the Windows operating system, and `Root` is specified for Linux and macOS. An adversary holding a `UserAccount` cannot use a technique that requires Administrator permission. By default, an adversary who holds *adminRights* automatically has *userRights*. Moreover, an adversary can level up through `Privilege Escalation` tactic to gain *adminRights* from *userRights*.

According to the above two steps, *Access Token Manipulation* can be converted into a MAL file *accessTokenManipulation.mal* as follows:

```
category Software {
asset UserAccount {
    | userRights
        ->  windows.userAccessTokenManipulation

    # userAccountManagement
        ->  windows.userAccessTokenManipulation
}

asset AdminAccount {
    | adminRights
}

asset WindowsAdmin extends AdminAccount {
    | adminRights
        +>  windows.adminAccessTokenManipulation

    # privilegedAccountManagement
        ->  windows.adminAccessTokenManipulation
}

asset Windows {
    & userAccessTokenManipulation
    info: "Adversaries may use access tokens to operate
    under a different user or system security context
    to perform actions and evade detection."
        ->  service.exploitationForPrivilegeEscalation

    & adminAccessTokenManipulation
        ->  service.exploitationForPrivilegeEscalation
}

asset Service {
    | exploitationForPrivilegeEscalation
}
}

associations {
UserAccount [userAccount] * <--Access--> * [windows] Windows
AdminAccount [adminAccount] * <--Access--> * [windows] Windows
Windows [windows] * <--Runs--> * [service] Service
}
```

The assets are categorized according to their functions, types, or fields of use. In this example, all the assets belong to the `Software` category, where `WindowsAdmin` extends `AdminAccount` to specify that the platform on which this technique can be used is the `Windows` operating system.

Considering the attack/defense steps, the asset `UserAccount` contains one attack step and one defense step. First, *userRights* leads to (–>) *windows.userAccessTokenManipulation*, which means that an adversary who holds *userRights* can perform *userAccessTokenManipulation* in the `Windows` OS. Similarly, an adversary who holds *adminRights* can perform *adminAccessTokenManipulation*, which may lead to further attacks owing to its higher permission level.

The asset `Windows` contains two attack steps: *userAccessTokenManipulation* and *adminAccessTokenManipulation*. They are of type &, as several steps need to be completed before they can be implemented. When the value of *userAccountManagement* defense is set to TRUE, the corresponding *userAccessTokenManipulation* attack step cannot be reached; when the value is set to FALSE, the *userAccessTokenManipulation* attack step can be reached, and the attack step *exploitationForPrivilegeEscalation* becomes accessible.

In addition, each asset association is created to connect two assets, and assets play roles in associations. In this example, multiple (*) `UserAccounts` can `Access` multiple (*) `Windows`.

After this file is compiled, an attack graph can be generated, as shown in Figure 4, where the circles represent the attack steps of type OR (|), the squares represent the attack steps of type AND (&), and the upside-down triangles represent the defense steps.

5.1.3 Integrating Techniques into enterpriseLang

In the construction process, 266 adversary techniques are converted to MAL files. As we aim to cover the full range of techniques found and detailed by the MITRE ATT&CK Matrix, and adversary techniques are usually not used in isolation, it is thus necessary to integrate these files into a single language, enterpriseLang, for holistic threat modeling of enterprise systems.

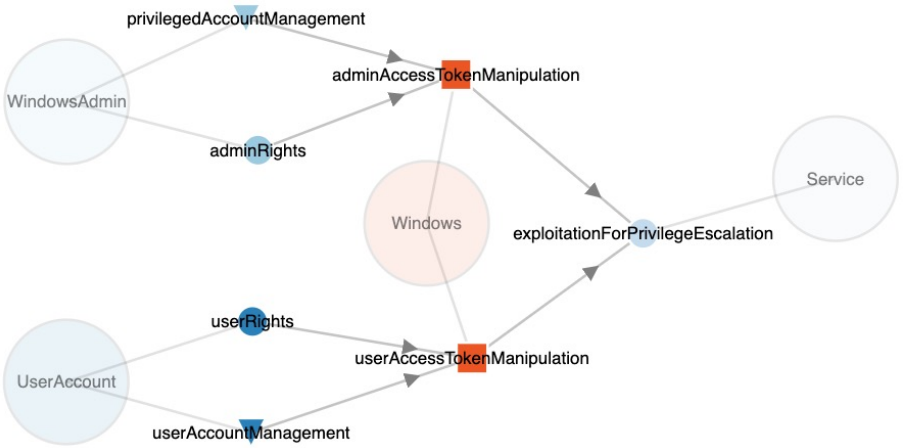


Fig. 4 Attack graph representation of the modeled Access Token Manipulation

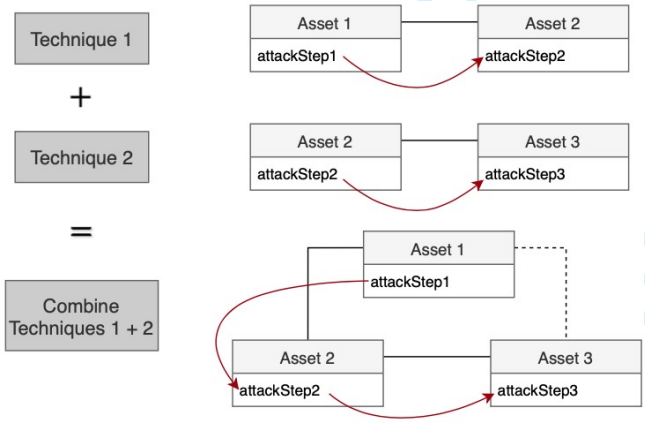


Fig. 5 Illustration of need to integrate combined adversary techniques into enterpriseLang

As shown in Figure 5, `Asset 1` and `Asset 2` are directly associated according to the description of *Technique 1*. Similarly, for *Technique 2*, we know that `Asset 2` and `Asset 3` are directly associated. To model a more complicated scenario in which an adversary combines these two techniques, `Asset 1` and `Asset 3` are indirectly associated, and the attack steps for these two assets are indirectly linked to one another.

The designed enterpriseLang can then be converted by a MAL compiler,[17] which generates Java code from enterpriseLang. Several files are created in the specified output folder. One is an HTML file, which can be opened in a web browser to visualize the overall attack graph of enterpriseLang.

5.2 Overview of the Language

A metamodel of enterpriseLang showing the essential enterprise IT assets and their associations is created during the construction of enterpriseLang; the model was inspired by the work of Ek and Petersson [8] and is shown in Figure 6. The following asset categories are captured:

- The `Person` category includes one main `Asset`, `User`. `Users` are employees in the office area who use `Computers` and can also be social engineering victims.
- The `Software` category includes six main `Assets`: `OS`, `UserAccount`, `AdminAccount`, `SysAdmin`, `Service`, and `Browser`.
  `AdminAccount` has two inherited assets, `WindowsAdmin` and `Root`, depending on the operating system, and `ReomoteDesktopUser` is inherited from `UserAccount`, which represents the rights that are currently held by adversaries or could be compromised and obtained by them. When `OSs` boot up, they can run programs or applications called `Services` that perform functions. One commonly accessed `Service` is `Browser`.
  In addition, `OS` has three inherited assets, `Windows`, `Linux`, and `macOS`, where `SysAdmin` is a security principal for operating the `Windows` system. Some attack steps can be reached for all three `OSs`, e.g., *connectionProxy* and *bruteForce*, whereas some can be reached only for `Linux` and `macOS`, e.g., *sudo*.
  Furthermore, `ThirdpartySoftware` and `VideoCallApplication` are inherited from `Service`. For example, *Spearphishing via Service* can be reached for third-party services rather than typical trusted services.
- The `Network` category contains four main `Assets`: `InternalNetwork`, `ExternalNetwork`, `Router`, and `Firewall`. A `Firewall` represents a local firewall on the `Computer` that provides a layer of defense to permit incoming and outgoing network

[17] https://github.com/mal-lang/malcompiler

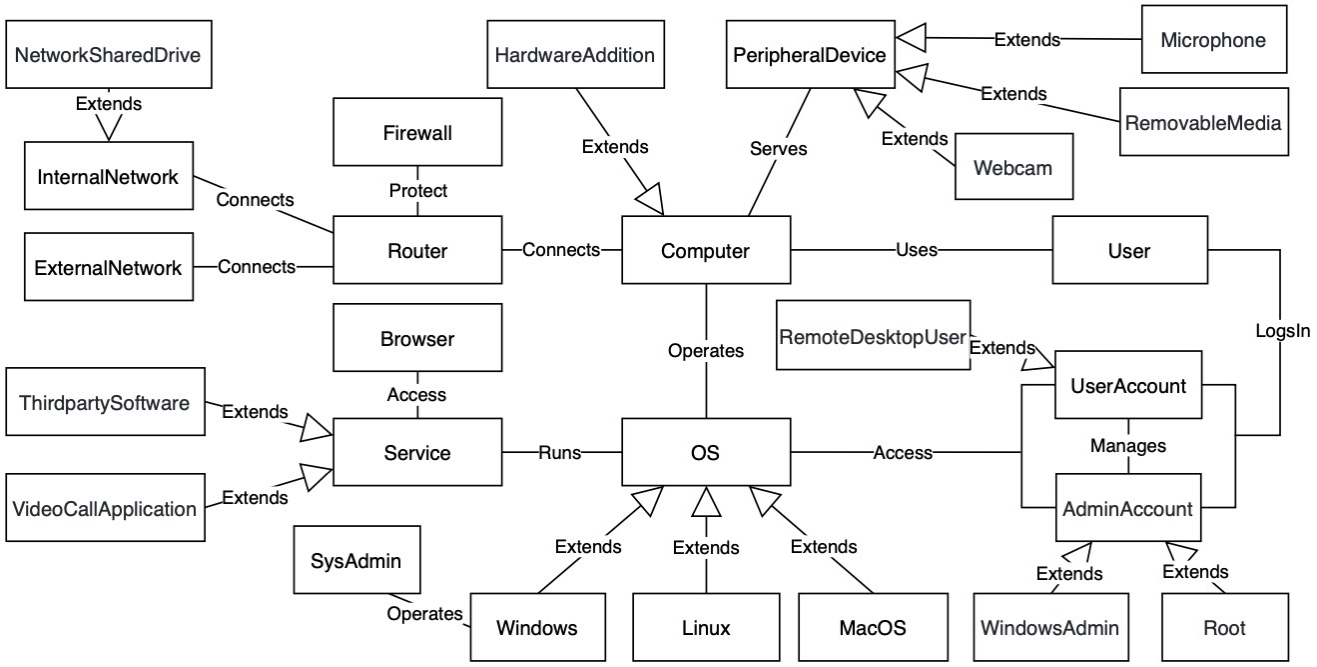


Fig. 6 The enterpriseLang metamodel containing enterprise assets and associations

traffic; its rules can also be discovered by adversaries to shape their subsequent behavior. In addition, `NetworkSharedDrive` is inherited from `InternalNetwork` and contains, e.g., host-shared directories and network file servers.

– The `Hardware` category contains two main `Assets`: `Computer` and `PeripheralDevice`. `HardwareAddition` extends `Computer` to include resources such as encryption breaking or adding wireless access to an existing network. In addition, `PeripheralDevice` has three inherited assets: `Microphone`, `RemovableMedia` (e.g., USB), and `Webcam`. For example, adversaries may search for `PeripheralDevice`s and search for sensitive data stored on them.

A total of 26 enterprise IT `Assets` (13 main `Assets` and 13 inherited `Assets`) are extracted from the MITRE ATT&CK Matrix and included in enterpriseLang. Although it is not shown in this metamodel, each `Asset` is associated with a pair of attack and defense steps. For example, concerning the number of attack steps that can be reached for a certain asset, 222 attack steps are associated with `Windows`, 134 are associated with `Linux`, and 160 are associated with `macOS`.

After enterpriseLang is designed, its security scope is established. It contains 41 defenses that can be implemented in general enterprise systems, and 8 of them can potentially defend against more than 20 attack steps, as shown in Table 2. Because it is difficult to achieve perfect security, security controls need to be prioritized for a specific enterprise; this can be realized through, for instance, attack simulations.

After a system model is created for a specific enterprise, securiCAD can be used to simulate attacks on the system model and provide a measure of the system security in terms of resilience against various attacks. For example, to assess the cyber security of enterprise A, a system model can be created by specifying the contained assets, asset associations, and implemented security mechanisms. After attacks on the system model are simulated, the security measures in the modeled as-is system are provided. Consequently, enterprise A can prioritize its security settings (e.g., defenses) by changing them in to-be models and observing the changes in the simulation results, e.g., how attack paths can be disrupted, as shown in Figure 7.

6 Testing

In this section, we test the proposed enterpriseLang by modeling two known attack scenarios: the Ukraine cyber attack and the Cayman National Bank cyber heist. The evaluation of both cases considers two issues: (1) whether the techniques used in each case are present in enterpriseLang and behave as expected, and (2) whether enterpriseLang can identify which mitigations are missing and could be implemented.

Table 2 Defenses that defend against more than 20 attack steps

Defense name	Description	Number of attack steps
Privileged Account Management	To protect privileged accounts, e.g., `AdminAccount`s, from abuse and misuse, enterprises should limit their use, modification, and permissions.	37
User Account Management	This defense is associated mainly with the `AdminAccount` asset to ensure proper User permissions.	37
Execution Prevention	Enterprises may use application whitelisting tools to block scripts and unapproved software that can be misused by adversaries.	32
Restrict File and Directory Permissions	Enterprises can apply the least privilege principle to limit access to files and directories.	29
Network Intrusion Prevention	Enterprises can use an intrusion prevention system to examine network traffic flows, e.g., to scan and remove malicious email attachments.	28
Disable or Remove Feature or Program	To prevent misuse of vulnerable software, some features should be disabled.	24
Audit	The security of enterprise systems should be systematically evaluated; this evaluation should include checking file system permissions for opportunities for abuse.	23
Network Segmentation	An architectural approach that divides a network into subnetworks, each of which is a network segment, to improve the network performance and security.	22

6.1 The Ukraine Cyber Attack

The Ukraine power grid attack of 2015 was identified as a coordinated attack and resulted in hours of blackouts for approximately 225,000 people in various parts of Ukraine. According to a comprehensive report,[18] the `Attacker`s first conducted a spearphishing campaign that aimed to enter the office area of the network operators. When an `Employee` downloaded and executed the malicious attachment through `UserAccount`, the `Attacker`s were able to compromise the `OfficeComputer`s and obtain credentials through `ExternalRemoteService`s to gain access to and control of the central `SCADAEnvironment`. They continued by obtaining remote access to the human-machine interface system, shutting down the electricity supply system, and disabling the protective relays.

Let us analyze this case in terms of the attack steps. First, the `Attacker`s sent a *spearphishingAttachment* by email as an initial attack vector. They relied on *userExecution* to attack the *infectedComputer* within the office area. The `Attacker`s then used *externalRemoteServices* and harvested *validAccounts*, which were used to interact directly with the client application through the *graphicalUserInterface* in the SCADA environment to open breakers. Then, the `Attacker`s used malicious *systemFirmware* and scheduled disconnects of the compromised power supply systems, which finally caused *systemShutdownOrReboot*. They also performed *fileDeletion* of files stored on the infected computers to make it difficult to restore the system. In addition, they conducted an *endpointDenialOfService* attack against the center of the substation, which caused a protective *serviceStop*.

For the first evaluation, we check whether the adversary techniques used in this case and the attack step connections are present in enterpriseLang. Figure 8 shows the attack graph of the Ukraine cyber attack; all of the attack steps are present and behave as expected. The arrows indicate the potential target attack step after reaching each step, and together they constitute a complete attack path. There are three main results for this attack, which are indicated by red lines: *fileDeletion*, *systemShutdownOrReboot*, and *serviceStop*. In addition to showing the actions of the `Attacker`s, the attack graph also shows how to potentially mitigate this attack (the defense steps are indicated by upside-down triangles). First, unknown or unused attachments can be blocked (i.e., by using *restrictWebBasedContent*). In addition, *userTraining* can decrease the likelihood of successful *spearphishingAttachmentDownload* and *userExecution*, and *limitAccessToResourceOverNetwork* can further prevent `Attacker`s from using remote access tools. Finally, *passwordPolicies* can make user accounts within the environment harder to obtain, and *restrictRegistryPermissions* can prevent `Attacker`s from disabling or interfering with critical services.

[18] https://www.antiy.net/p/comprehensive-analysis-report-on-ukraine-power-system-attacks/

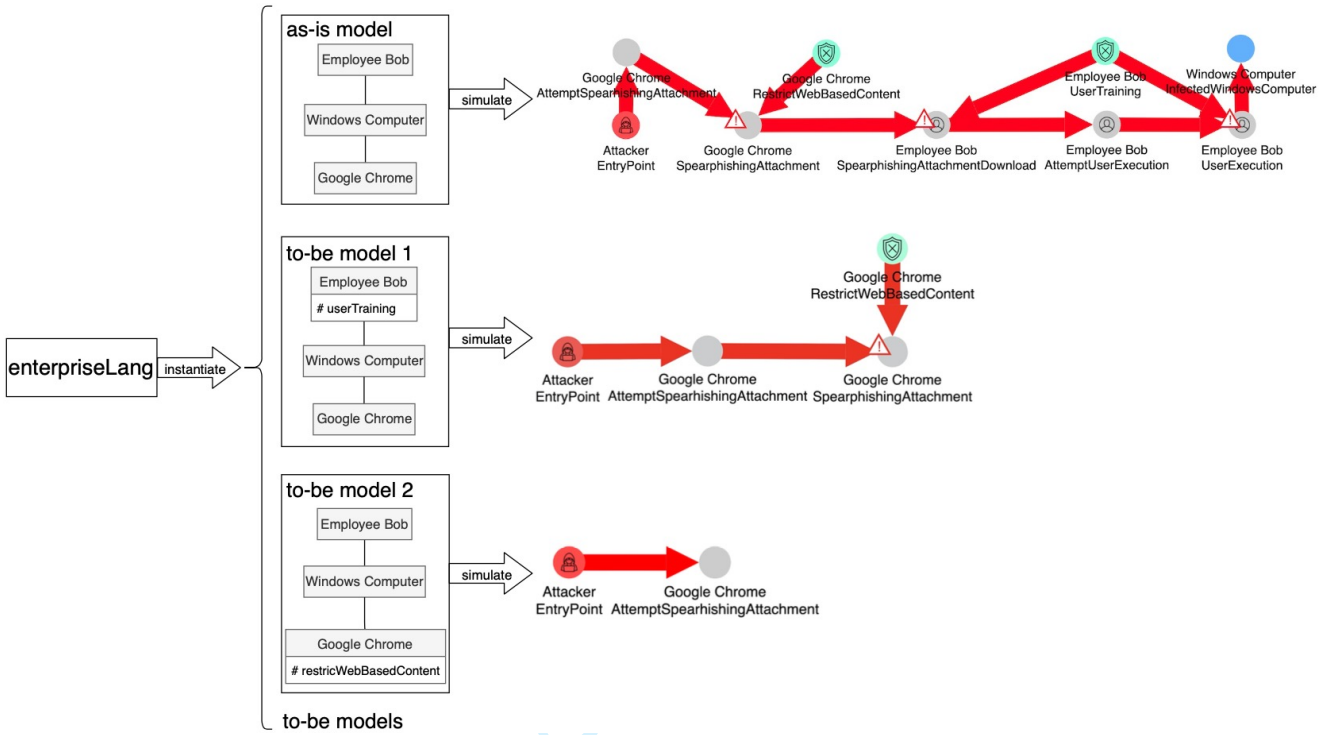


Fig. 7 Instantiation and use of the proposed language for to-be scenario decision-making

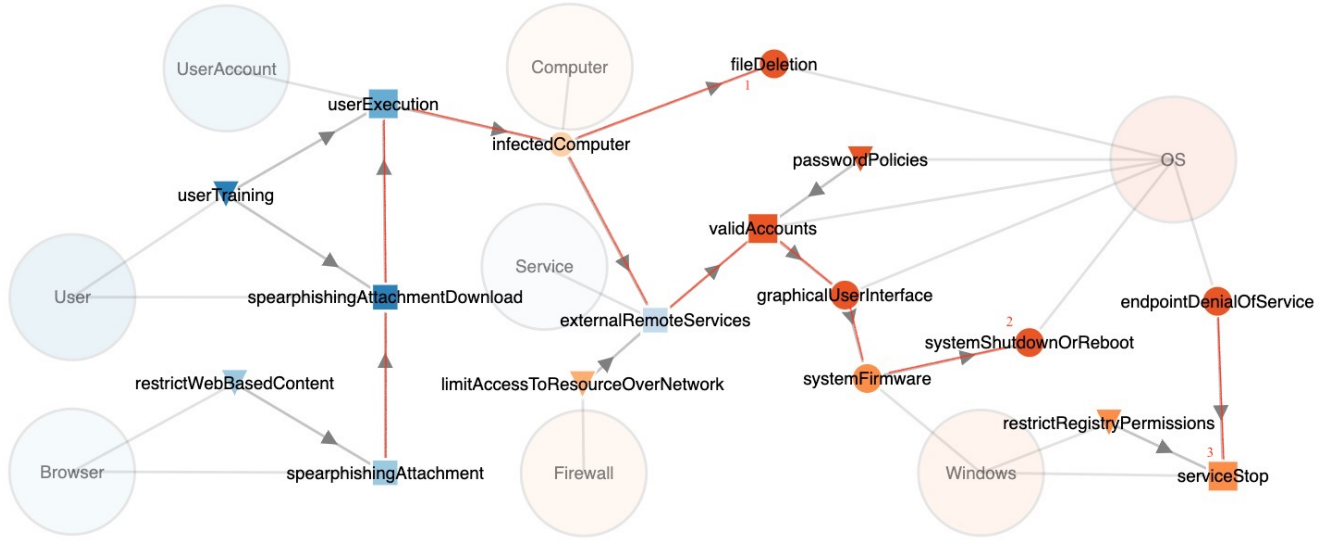
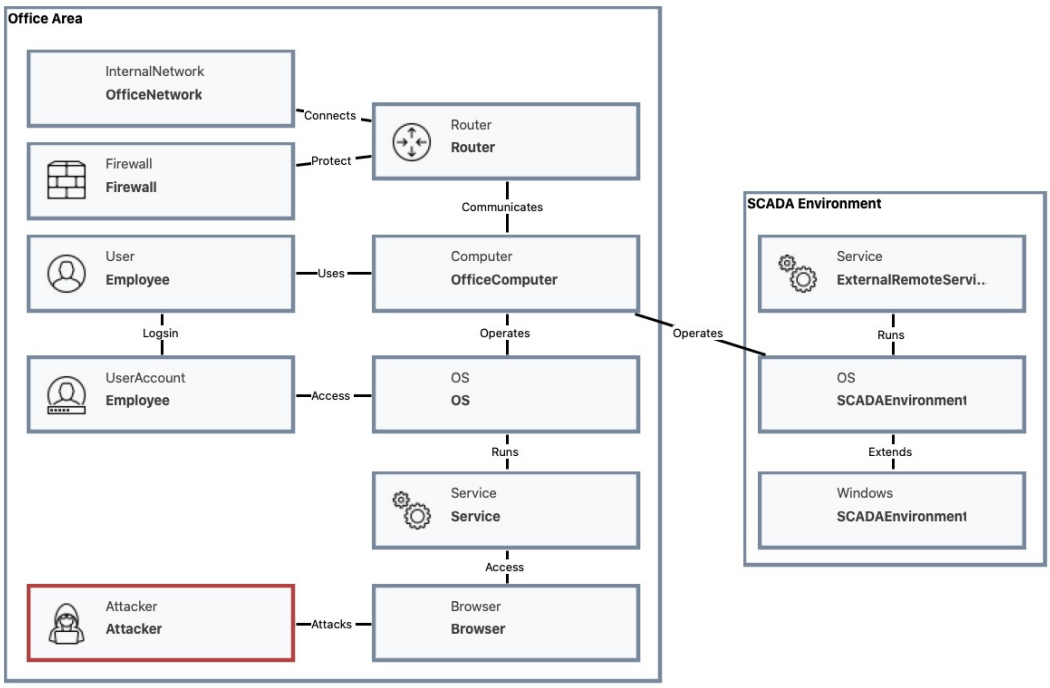


Fig. 8 Attack graph representation of the Ukraine cyber attack. Excerpt from the generic attack graph of enterpriseLang

In the second evaluation, we check whether enterpriseLang can indicate the security of the current system model and support better decision-making for to-be system models. First, we specify the assets and asset associations required to build a system model of this case, and we specify the entry point of the attack as *spearphishingAttachment* under `Browser` to make the threat model complete, as shown in Figure 9(a). We then simulate attacks on the system model using securiCAD. Figure 9(b) shows one of the critical attack paths that results in *systemShutdownOrReboot* from the simulation results. Possible defense steps to interrupt this attack, which could be implemented to increase the system security level, are indicated by green circles. In addition, the width of the lines between the attack and defense steps indicates the probability of the attack path. Here the lines are of equal width owing to the lack of probabilistic relations between the attack and defense steps.



(a) Model of Ukraine system

(b) Attack path to shut down the electricity supply system

Fig. 9 Threat modeling and attack simulations for the Ukraine cyber attack

In addition, enterpriseLang is intended to help enterprises make better decisions for their to-be scenarios. For example, when we enable the `Firewall` on *LimitAccessToResourceOverNetwork*, which prevents adversaries from using `ExternalRemoteServices` to access the SCADA environment, this attack can be blocked at the *InfectedComputer*, as shown in Figure 10. Furthermore, the attack path could probably be interrupted earlier at the *SpearphishingAttachmentDownload* step by *UserTraining*, where employees can be trained not to download malicious attachments from emails. Therefore, by comparing the two hypothetical scenarios of the system model, *UserTraining* could be prioritized as a security control to improve the system security level and thus make it harder for adversaries to achieve their final goals, i.e., *SystemShutdownOrReboot*.

6.2 Cayman National Bank Cyber Heist

The Cayman National Bank cyber heist of 2016 netted hundreds of thousands of pounds. According to a report,[19] the `Attackers` first obtained access to the `OfficeComputer` by scanning the Internet for all the vulnerable VPN `Services` for which there were exploits; they then gained a foothold in the bank's network. In addition, another group of `Attackers` first gained access to the `OfficeComputer` of the same workstation by sending an email with a malicious attachment from a spoofed email account to a bank `Employee`. They waited for the `Employee` to click the attachment, and finally the `OfficeComputer` was infected. After the bank discovered unauthorized SWIFT (Society for Worldwide Interbank Financial Telecommunication) transactions, an investigation was started. Furthermore, the `Attackers` obtained new passwords to follow the investigation by reading the emails of the persons involved. The `Attackers` remained active on the bank's networks for a few months and started the first transaction for a hundred thousand pounds.

[19] https://www.csoonline.com/article/3454443/how-a-bank-got-hacked-a-study-in-how-not-to-secure-your-networks.html

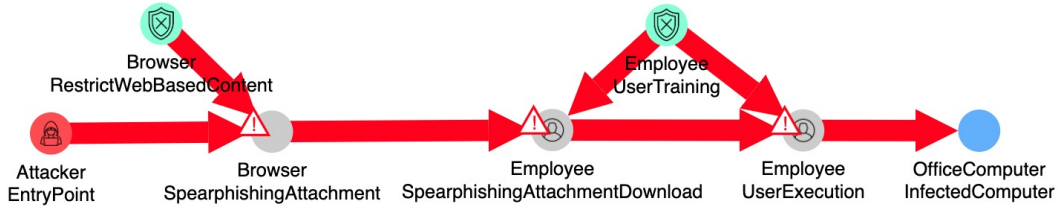


Fig. 10 Changing firewall settings to mitigate the attack

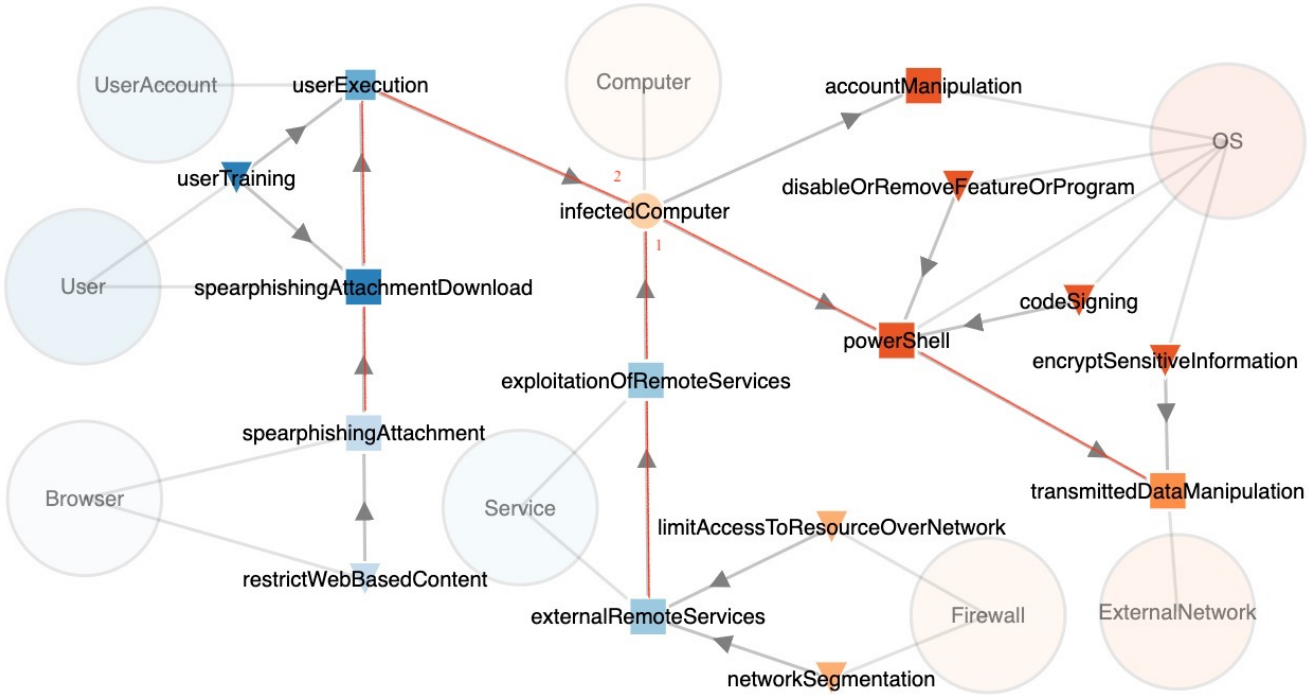
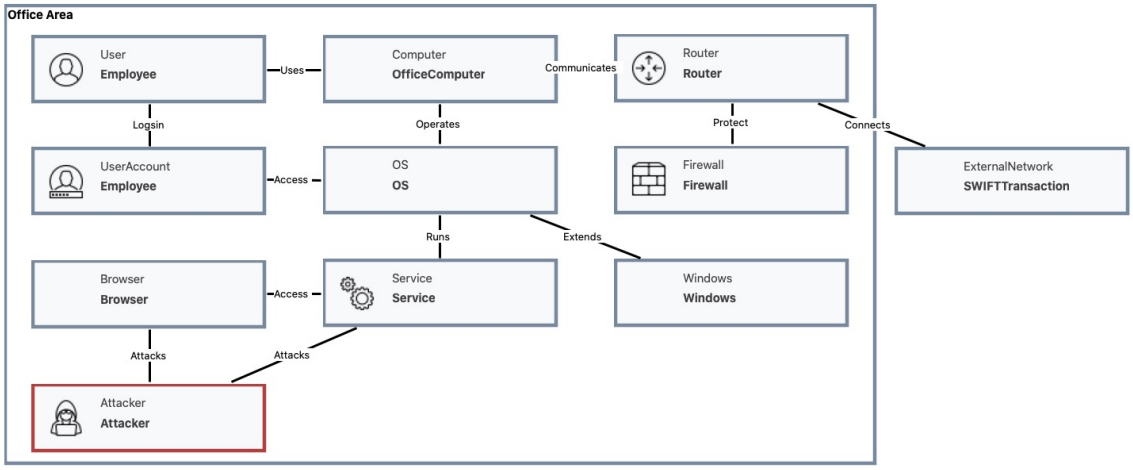


Fig. 11 Attack graph representation of the Cayman National Bank cyber heist. Excerpt from the generic attack graph by enterpriseLang

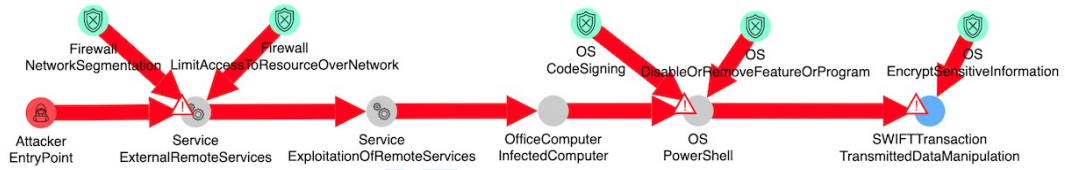
We analyze this case in terms of the attack steps. First, the `Attackers` gained access to the `OfficeComputer` in two ways. One group performed an attack on *externalRemoteServices*, where a Sonicwall SSL/VPN exploit was found, and they performed the *exploitationOfRemoteServices* to attack the *infectedComputer* and enter the office area. Another group used the *spearphishingAttachment* combined with *userExecution* to access the office area. Next, *accountManipulation* enabled the `Attackers` to follow the investigation and remain present on the network, and the use of *powerShell* made it possible for them to conduct *transmittedDataManipulation*.

Again, we check whether the adversary techniques used in this case and the connections between attack steps are present in enterpriseLang. As shown in Figure 11, there are two ways to compromise the `Computer` and finally perform *transmittedDataManipulation*, which are indicated by red lines. In addition, the `Attackers` performed *accountManipulation* to remain in the office area. Overall, the techniques used in this case are present in enterpriseLang and behave as expected.

In terms of mitigations of this attack, first, *restrictWebBasedContent* can be implemented to block certain websites that may be used for spearphishing. If they are not blocked and the malicious attachment is downloaded, *userTraining* can be used to defend against *spearphishingAttachmentDownload* and *userExecution*, making it harder for adversaries to access and attack the *infectedComputer*. Another way to attack the *infectedComputer* is by using *externalRemoteServices*, which can be mitigated by *limitAccessToResourceOverNetwork* and *networkSegmentation* by a `Firewall`. In addition, through the *infectedComputer*, `Attackers` could launch a *powerShell*, which can be defended by the use of *codeSigning* to execute only signed scripts and *disableOrRemoveFeatureOrProgram* to limit use to legitimate purposes and limit access to administrative functions. Finally, *encryptSensitiveInformation* can be im-



(a) Model of Cayman National Bank system



(b) Attack path of transmitted data manipulation

Fig. 12 Threat modeling and attack simulations for the Cayman National Bank cyber heist

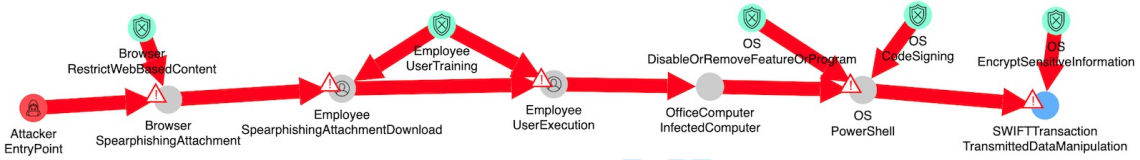


Fig. 13 Changing the attacker's entry point could lead to another attack path that achieves the same goal

plemented to reduce the impact of tailored modifications on data in transit.

For the second evaluation, we first specify the assets and asset associations to model the current system. We also specify that the entry points can be both `Browser` and `Service` to complete the threat model, as shown in Figure 12(a). After attacks on the current system model are simulated, the shortest path to realize *transmittedDataManipulation* is generated and is shown in Figure 12(b).

In addition, to see how enterpriseLang can support better decision-making, we enable both *limitAccessToResourceOverNetwork* and *networkSegmentation* in the `Firewall` settings to prevent `Attackers` from using *externalRemoteServices* and interrupt the attack path. However, these actions may not be sufficient to prevent `Attackers` from reaching *transmittedDataManipulation* because simply blocking the initial attack vector is only a first step. Access can still be obtained through a different entry point, as shown in Figure 13.

Overall, the effectiveness of the proposed language is verified by application to these two known cyber attack scenarios. First, the techniques used in both cases are present in enterpriseLang and behaved as expected. In addition, enterpriseLang provided security assessments and supported analysis of which security measures should be implemented in the system models by changing security settings (e.g., enabling or disabling each defense) for the current model. According to the simulation results, we can compare different security settings and select a to-be model that has suitable attack resilience.

7 Discussion

In this work, a DSL called enterpriseLang is designed according to the DSR guidelines. It can be used to assess the cyber security of enterprise systems and support analysis of security settings and potential changes that can be implemented to secure an enterprise system effectively. The effectiveness of our proposed language

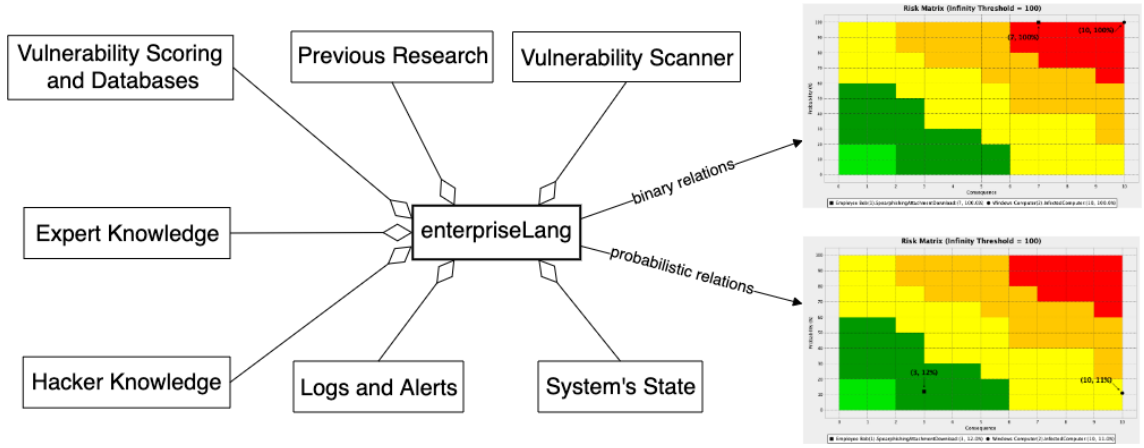


Fig. 14 Other sources that can be added to enrich enterpriseLang

is verified by application to two known cyber attack scenarios.

Although some capabilities of the proposed enterpriseLang were tested, there are still challenges. More known attacks could be used to further validate the language. In addition, larger enterprise systems could be modeled to test its usability. The MITRE Enterprise ATT&CK Matrix is used as a knowledge base for the proposed language. However, it may not cover all adversary techniques. Other information sources that record vulnerabilities are available. For instance, the Common Vulnerabilities and Exposures (CVE) database[20] contains a list of publicly known cyber security vulnerabilities, each of which is associated with a Common Vulnerability Scoring System score[21] indicating its severity [16]. Other databases such as the Common Weakness Enumeration (CWE) database[22] list various types of software and hardware weaknesses, and the Common Attack Pattern Enumeration and Classification (CAPEC) database[23] provides a comprehensive dictionary of known patterns of attack employed by adversaries to exploit known weaknesses in cyber-enabled capabilities.

Furthermore, enterpriseLang currently assumes that all attack steps reachable by adversaries can be performed instantly. However, successful real-world attacks usually involve a certain cost, probability, and effort. An attack usually takes time, e.g., the TTC, and an adversary completes this attack step successfully with a certain probability. We assume that adversaries will take the shortest path, i.e., the least time-consuming route to the end node. For example, the probability that a user will click a *Spearphishing Link* has been found to be subject to a Bernoulli distribution with a probability of 71% [4]. In addition, the defense steps in enterpriseLang currently have only Boolean values (TRUE/FALSE) to indicate their status. If the value of a defense step is TRUE, it can defend against all the corresponding attack steps. According to the results of comparisons in previous research [3,5], *User Training* is subject to a Bernoulli distribution with a probability of 22%. Vulnerability databases such as CAPEC and CWE can also provide the likelihood of attacks [41], and thus more accurate simulation results.

Because the MAL, on which the proposed enterpriseLang is based, offers the option of using probability distributions to provide more accurate attack simulation results, the correct probability distribution types and probability values for the included attack steps could be added to enrich enterpriseLang. These distribution types and values are available from information sources, according to a systematic literature review [22]; they include vulnerability scores and databases, previous research, vulnerability scanners, expert knowledge, hacker knowledge, logs and alerts, and system state information. Therefore, by using the probabilistic relations provided for each attack/defense step instead of binary relations and implementing Bayesian network-based attack graphs, more accurate security measurements could be obtained, e.g., the probability of successful attack paths and risk matrices, as shown in Figure 14.

8 Conclusion

Assessing the cyber security of enterprise systems is becoming more important as the number of security issues and cyber attacks increases. In this paper, we propose a MAL-based DSL called enterpriseLang that was developed according to the DSR guidelines. It is used for

[20] https://cve.mitre.org/
[21] https://www.first.org/cvss/v2/guide
[22] http://cwe.mitre.org/
[23] https://capec.mitre.org/

holistic assessment of the cyber security of enterprise systems against various cyber attacks. The MITRE Enterprise ATT&CK Matrix serves as a knowledge base for the attack and defense steps. By using available tools, enterpriseLang enables attack simulations on its system model instances, and the simulation results can support analysis of the security settings and architectural changes that might be implemented to secure the system effectively. The proposed enterpriseLang is evaluated by examining two real-world cyber attacks.

The proposed DSL will be improved in our future work. First, although the MITRE ATT&CK Matrix provides reasonable coverage of attacks on enterprise systems, other information sources (e.g., the CVE and CWE databases) can be used to enrich enterpriseLang. Further, enterpriseLang is intended to be able not only to model enterprise systems but also to provide probabilistic security measures. Thus, another direction for future work is to assign probability distributions to the attack/defense steps in order to provide quantitative simulation results.

References

1. Applebaum, A., Miller, D., Strom, B., Foster, H., Thomas, C.: Analysis of automated adversary emulation techniques. In: Proceedings of the Summer Simulation Multi-Conference, pp. 1–12 (2017)
2. Bedi, P., Gandotra, V., Singhal, A., Narang, H., Sharma, S.: Threat-oriented security framework in risk management using multiagent system. Software: Practice and Experience **43**(9), 1013–1038 (2013)
3. Burns, A.J., Johnson, M.E., Caputo, D.D.: Spear phishing in a barrel: Insights from a targeted phishing campaign. Journal of Organizational Computing and Electronic Commerce **29**(1), 24–39 (2019)
4. Butavicius, M., Parsons, K., Pattinson, M., McCormac, A.: Breaching the human firewall: Social engineering in phishing and spear-phishing emails (2016). URL https://arxiv.org/abs/1606.00887
5. Caputo, D.D., Pfleeger, S.L., Freeman, J.D., Johnson, M.E.: Going spear phishing: Exploring embedded training and awareness. IEEE Security & Privacy **12**(1), 28–38 (2014)
6. Cardenas, A.A., Roosta, T., Sastry, S.: Rethinking security properties, threat models, and the design space in sensor networks: A case study in scada systems. Ad Hoc Networks **7**(8), 1434–1447 (2009)
7. Chu, M., Ingols, K., Lippmann, R., Webster, S., Boyer, S.: Visualizing attack graphs, reachability, and trust relationships with navigator. In: Proc. of the 7th Int. Symp. on Visualization for Cyber Security, pp. 22–33. ACM (2010)
8. Ek, D., Petersson, J.: Abstraction of mitre att&ck. Bachelor's thesis, KTH Royal Institute of Technology, Stockholm, Sweden (2020)
9. Ekstedt, M., Johnson, P., Lagerstrom, R., Gorton, D., Nydrén, J., Shahzad, K.: Securi cad by foreseeti: A cad tool for enterprise cyber security management. In: 2015 IEEE 19th International Enterprise Distributed Object Computing Workshop (EDOCW), pp. 152–155. IEEE (2015)
10. Ghosh, N., Chokshi, I., Sarkar, M., Ghosh, S.K., Kaushik, A.K., Das, S.K.: NetSecuritas: An integrated attack graph-based security assessment tool for enterprise networks. In: Proc. of the 2015 Int. Conf. on Distributed Computing and Networking, p. 30. ACM (2015)
11. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS Quarterly **28**(1), 75–105 (2004)
12. Holm, H., Buschle, M., Lagerström, R., Ekstedt, M.: Automatic data collection for enterprise architecture models. Software & Systems Modeling **13**(2), 825–841 (2014)
13. Holm, H., Shahzad, K., Buschle, M., Ekstedt, M.: P^2cysemol: Predictive, probabilistic cyber security modeling language. IEEE Transactions On Dependable And Secure Computing **12**(6), 626–639 (2015)
14. Homer, J., Zhang, S., Ou, X., Schmidt, D., Du, Y., Rajagopalan, S.R., Singhal, A.: Aggregating vulnerability metrics in enterprise networks using attack graphs. Journal of Computer Security **21**(4), 561–597 (2013)
15. Johnson, P., Lagerström, R., Ekstedt, M.: A meta language for threat modeling and attack simulations. In: Proceedings of the 13th International Conference on Availability, Reliability and Security, p. 38. ACM (2018)
16. Johnson, P., Lagerström, R., Ekstedt, M., Franke, U.: Can the common vulnerability scoring system be trusted? a Bayesian analysis. IEEE Transactions on Dependable and Secure Computing **15**(6), 1002–1015 (2018)
17. Johnson, P., Vernotte, A., Ekstedt, M., Lagerström, R.: pwnpr3d: an attack-graph-driven probabilistic threat-modeling approach. In: Availability, Reliability and Security (ARES), 2016 11th International Conference on, pp. 278–283. IEEE (2016)
18. Jürjens, J.: Umlsec: Extending uml for secure systems development. pp. 412–425. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
19. Katsikeas, S., Johnson, P., Hacks, S., Lagerström, R.: 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)
20. Kordy, B., Mauw, S., Radomirović, S., Schweitzer, P.: Foundations of attack–defense trees. In: International Workshop on Formal Aspects in Security and Trust, pp. 80–95. Springer (2010)
21. Lagerström, R., Franke, U., Johnson, P., Ullberg, J.: A method for creating enterprise architecture metamodels - applied to systems modifiability analysis. International Journal of Computer Science and Applications **6**, 89–120 (2009)
22. Ling, E., Lagerström, R., Ekstedt, M.: A systematic literature review of information sources for threat modeling in the power systems domain. In: 15th International Conference on Critical Information Infrastructures Security (2020)
23. Maleki, N., Ghorbani, A.A.: Generating phishing emails using graph database. In: Information Security Practice and Experience, pp. 434–449. Springer (2019)
24. Närman, P., Johnson, P., Lagerström, R., Franke, U., Ekstedt, M.: Data collection prioritization for system quality analysis. Electronic Notes in Theoretical Computer Science **233**, 29–42 (2009)
25. Noel, S., Elder, M., Jajodia, S., Kalapa, P., O'Hare, S., Prole, K.: Advances in topological vulnerability analysis. In: Conference For Homeland Security, 2009. CATCH

'09. Cybersecurity Applications Technology, pp. 124–129 (2009)
26. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. Journal of Management Information Systems **24**(3), 45–77 (2007)
27. Ross, J.W., Robertson, D.C., Weill, P.: Enterprise Architecture As Strategy: Creating a Foundation for Business Execution. Harvard Business School (2006)
28. Saini, V., Duan, Q., Paruchuri, V.: Threat modeling using attack trees. Journal of Computing Sciences in Colleges **23**(4), 124–131 (2008)
29. Salter, C., Saydjari, O.S.S., Schneier, B., Wallner, J.: Toward a secure system engineering methodology. In: Proceedings of the 1998 workshop on New security paradigms, pp. 2–10. ACM (1998)
30. Schneier, B.: Attack trees - modeling security threats (1999)
31. Shehod, A.: Ukraine power grid cyberattack and us susceptibility: Cybersecurity implications of smart grid advancements in the us. Ph.D. thesis, MIT 22.811 Sustainable Energy (2016)
32. Shostack, A.: Threat modeling: Designing for security. John Wiley & Sons (2014)
33. Singh Virdi, A.: Awslang: Probabilistic threat modelling of the amazon web services environment. Master's thesis, KTH Royal Institute of Technology, Stockholm, Sweden (2018)
34. Sommestad, T., Ekstedt, M., Holm, H.: The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures. IEEE Systems Journal **7**(3), 363–373 (2013)
35. Sood, A.K., Enbody, R.: Chapter 3 - infecting the target. In: Targeted Cyber Attacks, pp. 23–35. Syngress, Boston (2014)
36. Uzunov, A.V., Fernandez, E.B.: An extensible pattern-based library and taxonomy of security threats for distributed systems. Computer Standards & Interfaces **36**(4), 734 – 747 (2014)
37. Vernotte, A., Välja, M., Korman, M., Björkman, G., Ekstedt, M., Lagerström, R.: Load balancing of renewable energy: a cyber security analysis. Energy Informatics **1**(1), 5 (2018)
38. Wang, L., Jajodia, S., Singhal, A., Cheng, P., Noel, S.: k-zero day safety: A network security metric for measuring the risk of unknown vulnerabilities **11**(1), 30–44 (2014)
39. Weiss, J.D.: A system security engineering process. In: 14th National Computer Security Conference, pp. 572–581 (2006)
40. Winter, R., Fischer, R.: Essential layers, artifacts, and dependencies of enterprise architecture. In: 2006 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06), pp. 30–30. IEEE (2006)
41. Xiong, W., Gülsever, M., Kaya, K.M., Lagerström, R.: A study of security vulnerabilities and software weaknesses in vehicles. In: The 24th Nordic Conference on Secure IT Systems (NordSec 2019), pp. 1–15. Springer (2019)
42. Xiong, W., Krantz, F., Lagerström, R.: Threat modeling and attack simulations of connected vehicles: Proof of concept. In: Information Systems Security and Privacy (ICISSP 2019), pp. 272–287. Springer (2020)
43. Xiong, W., Lagerström, R.: Threat modeling - a systematic literature review. Computers & Security **84**, 53–69 (2019)