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

Wenjun Xiong¹ · Emeline Legrand² · Oscar Åberg¹ · Robert Lagerström¹

Received: 14 July 2020 / Revised: 5 February 2021 / Accepted: 25 May 2021 / Published online: 18 June 2021
© The Author(s) 2021

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 framework and focuses on describing system assets, attack steps, defenses, and asset associations. The attack steps in the language represent adversary techniques as listed and described by MITRE. This entity-relationship model describes enterprise IT systems as a whole; 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 more effectively. Our proposed language is tested with a number of unit and integration tests. This is visualized in the paper with two real cyber attacks modeled and simulated.

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

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

To proactively deal with security issues of enterprise systems, threat modeling [58] is one approach that includes identifying the main assets within a system and threats to these assets. It is used to both assess the current state of a system and as a security-by-design tool for developing new systems. The approach can be coupled with attack simulations to provide probabilistic evaluations of security, e.g., time to compromise (TTC) [22,24]. On the basis of such

Communicated by Perry Alexander.

✉ 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 <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>.

objective evaluations, security controls can be chosen to counter anticipated threats.

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 [22]. The MITRE Enterprise ATT&CK Matrix⁵ 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), and defenses (e.g., Privileged Account Management, Execution Prevention, and Network Segmentation). Based on a system model and using available tools, *enterpriseLang* allows 1) analyzing weaknesses related to known attack techniques and 2) providing mitigation suggestions for these attacks. Therefore, stakeholders of an enterprise can assess threats to their enterprise IT environment and analyze what security settings that could be implemented to secure the system more effectively.

The rest of this paper is structured as follows. In Sect. 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 for the proposed language. Section 5 describes *enterpriseLang* in detail. In Sect. 6, we test the proposed language. Our work is discussed in Sect. 7 and finally concluded in Sect. 8.

2 Related work

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

2.1 Threat modeling

Threat modeling is a process to analyze potential attacks, threats, and risks [49]. It is often used as a structured approach to secure software in the design phase, by focusing on adversary goals when attacking a system [4].

According to a recent systematic literature review [58], most threat modeling methods can be categorized into manual modeling [1,59], automatic modeling [13,35], formal modeling [35,59], and graphical modeling [1,30,34], where formal modeling is based on mathematical models and graphical modeling can, for instance, be based on attack trees, attack and defense graphs, or tables. From the perspective

of system evaluation, through threat modeling, the system architecture is represented and analyzed, potential security threats are identified, and appropriate mitigation techniques are selected [10,13]. From the perspective of application development, threat modeling is often used to assist software engineers to identify and document potential security threats associated with a software product, providing development teams a systematic way of discovering strengths and weaknesses in their software applications [3]. Some focus on threat modeling as a process to analyze the security and vulnerabilities of an application or network services [9]. It provides a systematic way to identify threats that might compromise security; it is a well-accepted practice by the industry [33].

2.2 Attack simulations

Threats can be modeled using attack trees or attack graphs [28,41–43,53]. 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., MAL, also provides probabilistic simulation results [22]. Furthermore, several attack-graph-based tools have been developed. For example, MulVAL [20] derives logical attack graphs by associating vulnerabilities extracted from scans with the probability that an adversary could successfully conduct an attack. k-Zero Day Safety [52] extends MulVAL with the computation of zero-day attack graphs. In addition, NAVIGATOR [8] considers the identified vulnerabilities as directly exploitable, assuming that an adversary has access to the vulnerable system. Moreover, the topological vulnerability analysis (TVA) tool [37] models security conditions in networks and uses a database of exploits as transitions between the security conditions. Similarly, NetSecuritas [14] uses scanner output and known exploits to generate attack graphs and the corresponding security recommendations, e.g., the mitigation metric. These tools depend on available information about existing systems, from which attack graphs can be generated.

Some researchers have investigated the combined use of threat modeling and attack simulations in domains such as energy [51] and vehicular IT [27,57]. CySeMoL [45] is a cyber security modeling language for enterprise-level system architectures; P²CySeMoL [19], which is further development of CySeMoL, is an attack graph tool for estimating the cyber security of EAs. It couples attacks and defenses of objects in a system architecture, which a system owner can use to model and understand the system. P²CySeMoL differs from MulVAL, k-Zero Day Safety, and the TVA tool in that all the attack steps and defenses are related using Bayesian networks. In addition, pwnPr3d [24] was proposed as a probabilistic threat modeling approach for automatic attack graph generation; it provides both a high-level overview and tech-

⁵ <https://attack.mitre.org/>.

nical 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

The Zachman framework [60] is a representative EA base. EA has become an established discipline in business and software system management [40]. EA models can be used to increase the general understanding of enterprise systems and perform various types of analysis [29]. A key underlying assumption is that they should provide more aggregated knowledge than the information that was initially modeled, as in 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 [54]. Österlind et al. [38] 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.

However, these EA initiatives can lack semantics making it difficult for both humans and systems to understand the architecture description in an exact and common way [25]. Ontology-based approaches can be applied to solve this issue. An ontology includes definitions of concepts and an indication of how concepts are inter-related, which collectively impose a structure on the domain and constrain the possible interpretations of terms [47]. It can be used to assist in communication between human agents to achieve interoperability among computer systems and to improve the process and quality of engineering software systems [48]. An ontology-based EA can be employed to solve the communication problems between humans, between systems, or between human and system [25]. Moreover, it can be used to address the lack of domain knowledge and mismatched data granularity in automating threat modeling [50].

2.4 Tools based on the ATT&CK framework

Most threat modeling and attack simulations work remains to be done manually, which can be time-consuming and error-prone [18,36]. To extend the automation level, Applebaum et al. [2] 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, CALDERA⁶ 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.

According to a technical report,⁷ 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 as a whole 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.⁸

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.

⁶ <https://github.com/mitre/caldera>.

⁷ <https://www.slideshare.net/attackcon2018/mitre-attckcon-20-flashback-with-attck-exploring-malware-history-with-attck-20032018-kris-oosthoek-delft-university>.

⁸ <https://attack.mitre.org/matrices/enterprise/>.

- **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 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.⁹
- **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.

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 spearphishing e-mails with malicious attachments or links to trick users into sharing their credentials. For example, a cyber attack on the Ukraine power grid [44] was initiated by sending a spearphishing e-mail with a malicious file attached to a Microsoft Office Word document. When an employee opened the document and executed the file, the adversaries 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

⁹ <https://www.malwarepatrol.net/command-control-servers-c2s-fundamentals/>.

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)¹⁰ 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,¹¹ and search for vulnerabilities within the victim system. For example, when given an IP address, Nmap will report open ports and the services running on these ports. 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 e-mails sent from a trusted employee within the same enterprise [32]. 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 e-mail. If a user downloads the malicious attachment, the adversary can exploit the system upon *User Execution* [46].

There are multiple methods of defense against each technique.¹² 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 Fig. 1.

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 steps/defenses for each asset. A domain model can easily become too complex if the scope

¹⁰ <https://nvd.nist.gov/vuln/detail/CVE-2014-4076>.

¹¹ <https://nmap.org/>.

¹² <https://attack.mitre.org/mitigations/enterprise/>.

Fig. 1 MAL modeling hierarchy [22]

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 *vehicleLang* [27] for modeling cyber attacks on vehicle IT infrastructures, *powerLang* [15] for modeling attacks on power-related IT and OT infrastructures, *coreLang* [26] for modeling attacks on common IT infrastructures, and *awsLang*¹³ for assessing the cloud security of AWS environment 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 defenses (mitigations) (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 `&`.) Defenses (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 and defenses are given in camelCase, e.g., *userCredentials*, *userTraining*.

¹³ <https://foreseeti.com/securicad-vanguard-for-aws/>.

¹⁴ <https://github.com/mal-lang/mal-documentation/wiki>.

Table 1 MAL symbols

Symbol	Meaning	Description
<code>-></code>	Leads to	Successful compromise by this attack step allows adversaries to execute further attack steps, or this defense can defend against one or more further attack steps.
<code> </code>	OR	An OR attack in which step A can be reached if any of the attack steps that refer to A are reached.
<code>&</code>	AND	An AND attack in which step A can be reached only when all of the attack steps that refer to A are reached.
<code>#</code>	Defense	Unlike attack steps, defenses are Boolean. A defense represents the countermeasure to an attack step. A defense can be enabled or disabled by setting the defense value to TRUE or FALSE, respectively.
<code>E</code>	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.
<code>+></code>	Append	When parent and child assets have overlapping attack steps or defenses, the expressions defined for the child attack steps/defenses are appended to those of the parent attack steps/defenses.
<code>X.A</code>	Collect operator	Attack step A on asset X is referenced.
<code>TTC</code>	Time to compromise	TTC is a probability distribution that reflects the effort needed for an adversary to complete the related attack step. Each attack step can have a local TTC, and the MAL simulations calculate/aggregate the global TTC over models/scenarios.

- 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 con-

nected *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] * <--Accesses-->
  1 [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 *Accesses*. Moreover, one *Linux* can *Access* multiple (*) *UserAccounts*.

An illustration of how the relevant disciplines and background sources contribute to our designed *enterpriseLang* is shown in Fig. 2, where the MITRE ATT&CK Matrix serves as inputs for constructing the threat modeling language *enterpriseLang*, and *enterpriseLang* serves as an input to analyze the behavior of adversaries within the system model. By performing attack simulations on an enterprise system model using available tools, stakeholders can assess known threats to their enterprise, mitigations that can be implemented, shortest attack paths that can be taken by adversaries in the modeled system, and the shortest time required (i.e., the global TTC) for adversaries to reach various available attack steps and compromising individual attack steps (i.e., the local TTC) from the entry point, where a larger TTC value represents a more secure system.

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 [17]. Thus, the application of DSR is appropriate here as it guides the development of *enterpriseLang*. In this work, *enterpriseLang* is designed according to the DSR guidelines of Peffers et al. [39], 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 more effectively.
- Step 2: Define Objectives: To assess and enhance the security of enterprise systems, security-related assets of enterprise systems need to be understood, and 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 Sect. 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* [12] 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 evaluate *enterpriseLang*, 79 test cases are developed to check if the attack simulations executed by *enterpriseLang* behave as expected, and attacks and potential defenses are modeled accurately. Then, two enterprise system models of known real-world cyber attacks are created 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 to be implemented for the system models.

¹⁵ <http://www.agilemodeling.com/artifacts/classDiagram.htm>.

Fig. 2 Contributions of various resources to enterpriseLang, and how enterpriseLang can be practically usable for enterprise systems

To demonstrate enterpriseLang, two enterprise system models of known real-world cyber attacks are demonstrated using an attack graph excerpted from the generic attack graph of enterpriseLang, which shows the attack steps and defenses for the relevant system model assets, as well as how they are associated.

– 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

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 Sect. 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

In this step, we manually extract the information needed for constructing enterpriseLang from the ATT&CK Matrix. We consider each adversary technique as an attack step that can be performed by adversaries to compromise system assets. From the technique description, we learn how this technique (attack step) can be potentially used by adversaries with other techniques (attack steps) to form an attack path, and its corresponding attack type (OR or AND), where OR (|) signifies that adversaries can start working on this attack step as soon as one of its parent attack steps is compromised, and AND (&) requires all its parent attack steps to be compromised to reach this step.

Overall, 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 use a technique. The platform can 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

¹⁶ <https://github.com/mal-lang/enterpriseLang>.

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.

- 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 an adversary 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 Fig. 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. Mitigations of a technique can be converted to Defenses against an Attack. A Defense can defend against (->) multiple Attack Steps, and one Attack Step can be defended by multiple Defenses.
- 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 steps/defenses. (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 perform an Attack or an enterprise can implement a Defense. An Attack Step/Defense 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 Windows, 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 Account {
  asset UserAccount {
    | userRights
    -> windows.userAccessTokenManipulation
    # userAccountManagement
    -> windows.userAccessTokenManipulation
  }
  asset AdminAccount {
    | adminRights
    # privilegedAccountManagement
  }
  asset WindowsAdmin extends AdminAccount {
    | adminRights
    +> windows.adminAccessTokenManipulation
    # privilegedAccountManagement
    +> windows.adminAccessTokenManipulation
  }
}

category Software {
  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] * <--Accesses-->
  1 [windows] Windows
  AdminAccount [adminAccount] * <--Accesses-->
  1 [windows] Windows
  Windows [windows] 1 <--Runs-->
  * [service] Service
}
```

Access Token Manipulation

Attack Step:
userAccessTokenManipulation,
adminAccessTokenManipulation

Windows uses access tokens to determine the ownership of a running process. A user can manipulate access tokens to make a running process appear as though it belongs to someone other than the user that started the process. When this occurs, the process also takes on the security context associated with the new token. For example, Microsoft promotes the use of access tokens as a security best practice. Administrators should log in as a standard user but run their tools with administrator privileges using the built-in access token manipulation command `runas`.^[1]

"info"
 Adversaries may use access tokens to operate under a different user or system security context to perform actions and evade detection. An adversary can use built-in Windows API functions to copy access tokens from existing processes; this is known as token stealing. An adversary must already be in a privileged user context (i.e. administrator) to steal a token. However, adversaries commonly use token stealing to elevate their security context from the administrator level to the SYSTEM level. *lead to exploitationForPrivilegeEscalation attack step* remote system as the account for that token if the account has appropriate permissions on the remote system.

Mitigations

Defense:
privilegedAccountManagement,
userAccountManagement

Mitigation	Description
Privileged Account Management	Limit permissions so that users and user groups cannot create tokens. This setting should be Computer Configuration > [Policies] > Windows Settings > Security Settings > Local Policies > Define who can create a process level token to only the local and network service through GPO Settings > Security Settings > Local Policies > User Rights Assignment: Replace a process level
User Account Management	An adversary must already have administrator level access on the local system to make full use accounts to the least privileges they require.

ID: T1134

Tactic: Defense Evasion, Privilege Escalation

Platform: Windows **Asset:**
Windows

Permissions Required: User, Administrator

Permissions Levels:
userRights,
adminRights

Connections, Info, and Types:

attack step connections
attack step's "info"
attack type | or &

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

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

Considering the attack steps/defenses, the asset UserAccount contains one attack step and one defense. First, *userRights* leads to ($\rightarrow$) *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, one Windows can Access multiple (*) UserAccounts and AdminAccounts, and one Windows can Run multiple (*) Services.

After this file is compiled, an attack graph is generated, as shown in Fig. 4, where the circles represent the attack steps of type OR ($\vee$), the squares represent the attack steps of type AND ($\&$), and the upside-down triangles represent the defenses.

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 threat modeling of enterprise systems.

As shown in Fig. 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 sce-

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

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

nario in which an adversary combines these two techniques, Asset 1 and Asset 3 are indirectly associated, and the attack steps and defenses 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, which is inspired by the work of Ek and Petersson [11] and is shown in Fig. 6. The following asset categories are captured:

- The Person category includes one asset—User—and reflects the human aspect of cyber security.
- The Account category includes two assets: User Account 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 AdminAccount has two inherited assets, WindowsAdmin (for Windows) and Root (for Linux and macOS), depending on the OS running on a certain Computer.
- The Software category includes three assets: OS, Service, and Browser. When OSs boot up, they can run programs or applications called Services 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 OSs. Further, Service has two inherited assets: CloudService and Thirdparty Software.
- The Network category contains four assets: Internal Network, 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, Removable Media (e.g., USB), and Webcam.

A total of 22 enterprise IT Assets (12 main Assets and 10 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 steps and defenses. For example, concerning the number of attack steps that can be reached for a

¹⁷ <https://github.com/mal-lang/malcompiler>.

Fig. 6 The enterpriseLang metamodel containing enterprise assets and associations

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.

To implement enterpriseLang to assess the cyber security of an enterprise system, first, we load enterpriseLang in a simulation tool called securiCAD. Then, we create a system model by specifying the system assets and their associations and specify the adversaries' entry point that represents the attack step can be performed by adversaries to enter the modeled system. When we perform attack simulations on the system model, the various attacks that the system is vulnerable to can be discovered and possible mitigation strategies can be tested. The shortest path that can be taken by adversaries from the entry point to various other points in the modeled system can be explored together with potential mitigations throughout the path.

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 Fig. 7.

5.3 Computational performance

The system model in the above example is rather small when comparing to real enterprise systems. The system models created for real enterprise IT systems can be large and comprised of thousands or millions of attack steps. Therefore, it is important to consider computational performance.

According to the MAL framework [22] that enterpriseLang is based on, it is assumed that rational adversaries would select the shortest path to reach an attack step. Therefore, the global TTC to execute an attack step A_{child} (i.e., $T_{glob}(A_{child})$) is an estimate of the shortest time required by adversaries to reach any of its parent attack steps $A_{parent1}, \dots, A_{parentn}$ plus the local time increment of this attack step (i.e., $T_{local}(A_{child})$). In addition, the local TTC value of each attack step is sampled from its assigned probability distributions [56].

For an attack step of type OR,

$$T_{glob}(A_{child}) = \min(T_{glob}(A_{parent1}), \dots, T_{glob}(A_{parentn})) + T_{local}(A_{child})$$

For an attack step of type AND,

$$T_{glob}(A_{child}) = \max(T_{glob}(A_{parent1}), \dots, T_{glob}(A_{parentn})) + T_{local}(A_{child})$$

The above algorithms are modified versions of the single-source shortest path (SSSP) algorithm [16], and the benefit of the modification is the ability to approximate AND attack steps with maintained computational efficiency. Also, the SSSP algorithm is deterministic. To perform probabilistic computations, the deterministic algorithm is enveloped in a Monte Carlo simulation. Thus, a large set of graphs is gener-

Table 2 Defenses that defend against more than 20 attack steps

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

ated with local TTC values for each attack step sampled from their probability distributions. Then, the SSSP algorithm is used to compute the global TTC for each attack step in each attack graph. The resulting set of global TTC values for each attack step then approximates the actual distribution [22]. On an Apple MacBook, the above algorithms could compute 1000 samples of graphs with half a million nodes in under three minutes. Therefore, by using relatively unimpressive hardware, large IT systems can be computed.

5.4 Evaluating enterpriseLang

According to Hevner et al. [17], five methods can be used to evaluate the output of DSR, including observations, analysis,

experiments, tests, and descriptions. Because the development of enterpriseLang is similar to the development of source code, we select testing as the enterpriseLang evaluation method.

Specifically, two types of testing are applied. First, 44 unit tests are implemented to ensure that each technique in enterpriseLang functions as expected. To verify the generated results, cross-checking is applied by another DSL developer working on a realization of the MAL for a related domain. Second, 35 integration tests are implemented to ensure that the combination of different techniques and mitigations function as expected, which are based on real-world cyber attacks and security alerts.

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

Overall, 79 test cases have been developed to verify enterpriseLang. These tests confirm that attack simulations executed by enterpriseLang behave as expected, and attacks and potential defenses are modeled accurately.

6 Testing

In this section, we use enterpriseLang to model 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 are present in enterpriseLang and behave as expected and (2) whether enterpriseLang can provide security assessments and suggest security settings to be implemented for the system models.

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 Attackers 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 Attackers were able to compromise the OfficeComputers and obtain credentials through ExternalRemoteServices 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.

To analyze this case in terms of the attack steps, first, the Attackers sent a *spearphishingAttachment* by e-mail as an initial attack vector. They relied on *userExecution* to attack the *infectedComputer* within the office area. The Attackers 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 Attackers 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 Attackers, the attack graph also shows how to potentially mitigate this attack. (The defenses are indicated by upside-down triangles.) First, unknown or unused attachments can be

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

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

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

blocked (i.e., by using *restrictWebBasedContent*). In addition, *userTraining* can decrease the likelihood of successful *spearphishingAttachmentDownload* and *userExecution*, and *limitAccessToResourceOverNetwork* can further prevent Attackers from using remote access tools. Finally, *passwordPolicies* can make user accounts within the environment harder to obtain, and *restrictRegistryPermissions* can prevent Attackers from disabling or interfering with critical services.

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 Fig. 9a. We then simulate attacks on the system model using securiCAD. Figure 9b shows one of the critical attack paths

that results in *systemShutdownOrReboot* from the simulation results. Possible defenses to interrupt this attack, which can be implemented to increase the security level of the system, are indicated by green circles. In addition, the width of the lines between the attack steps and defenses indicates the probability of the attack path. Here, the lines are of equal width owing to the lack of probability distributions that can be assigned to attack steps and defenses to describe the efforts required for attackers to exploit certain attack steps.

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 Fig. 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 e-mails. 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 e-mail with a malicious attachment from a spoofed e-mail 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 e-mails 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.

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 Fig. 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 Web sites 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 implemented 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 Fig. 12a. After attacks on the current system model are simulated, the shortest path to reach *transmittedDataManipulation* is generated and is shown in Fig. 12b.

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 Fig. 13.

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

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

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 could provide security assessments and support analysis of which security measures should be implemented in the system models by changing security settings (e.g., enabling or disabling a defense) for the current system model. According to the simulation results, different security settings can be compared and a to-be model can be selected 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 more effectively. The effectiveness of our proposed language is verified by application to known attack scenarios.

Although some capabilities of the proposed enterpriseLang are 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) database²⁰ contains a list of publicly known cyber security vulnerabilities, each of which is associated with a Common Vulnerability Scoring System score²¹ indicating its severity [23]. Other databases such as the Common Weakness Enumeration (CWE) database²² list various types of software and hardware weaknesses, and the Common Attack Pattern Enumeration and Classification (CAPEC) database²³ provides a comprehensive dictionary of known patterns of attack

employed by adversaries to exploit known weaknesses in cyber-enabled capabilities.

Furthermore, enterpriseLang 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. To produce more realistic simulation results, probability distributions need to be assigned to attack steps and defenses to describe the efforts required for adversaries to exploit certain attack steps. For example, a user clicking a *Spearphishing Link* follows a Bernoulli distribution with parameter 0.71 [6]. In addition, the defenses in enterpriseLang currently have only Boolean values (TRUE/FALSE) to indicate their status. If the value of a defense is TRUE, it can defend against all the corresponding attack steps. According to the results of comparisons in previous research [5,7], *User Training* in place follows a Bernoulli distribution with parameter 0.22. Vulnerability databases such as CAPEC and CWE can also provide information about the likelihood of attacks [55] and thus can help produce 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, probability distributions can be assigned to attack steps and defenses of enterpriseLang. These probability distributions are available from various information sources [31,56], including vulnerability scores and databases, previous research, vulnerability scanners, expert knowledge, hacker knowledge, logs and alerts, and system state information. Therefore, by using the probability distributions assigned to attack steps and defenses, instead of using binary relations, more accurate security assessments could be obtained, e.g., the probability of successful attack paths and risk matrices, as shown in Fig. 14.

There are several potential directions for further improvement of enterpriseLang. First, the information extraction process for creating enterpriseLang is done manually from the ATT&CK Matrix. For future research directions, we could automate the process by combining natural language processing (NLP) and information retrieval (IR) to extract the information needed for threat modeling from information sources [21]. Second, we concentrate on modeling known techniques in enterpriseLang. However, there are also attack steps (e.g., zero-day threats) that are unknown yet. To overcome this issue, we update enterpriseLang regularly to

²⁰ <https://cve.mitre.org/>.

²¹ <https://www.first.org/cvss/v2/guide>.

²² <http://cwe.mitre.org/>.

²³ <https://capec.mitre.org/>.

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

include new cyber threats. Also, by using enterpriseLang to perform attack simulations on an enterprise system, we could identify the greatest weaknesses in the system and patch corresponding vulnerabilities, which may help to protect the system against zero-day attacks indirectly. Finally, we have implemented test cases to ensure that enterpriseLang behaves as expected. Our future work also includes creating more test cases to achieve better test coverage and testing enterpriseLang in real-world settings.

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 is developed according to the DSR guidelines. It is used for assessing the cyber security of an enterprise system as a whole against various cyber attacks. The MITRE Enterprise ATT&CK Matrix serves as a knowledge base for the attack steps and defenses. 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 more 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 steps/defenses in order to provide more realistic simulation results [56].

Acknowledgements This project has received funding from the European Union's H2020 research and innovation program under the Grant Agreement No. 832907, the Swedish Governmental Agency for Innovation Systems (Vinnova), and the Swedish Energy Agency.

Funding Open access funding provided by Royal Institute of Technology.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Al-Fedaghi, S., Moein, S.: Modeling attacks. *Int. J. Safety Secur. Eng.* **4**(2), 97–115 (2014)
2. 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)
3. Baquero, A.O., Kornecki, A.J., Zalewski, J.: Threat modeling for aviation computer security. *CrossTalk* November/December **1–12**, (2015)
4. Bedi, P., Gandotra, V., Singhal, A., Narang, H., Sharma, S.: Threat-oriented security framework in risk management using multiagent system. *Software Practice Exp.* **43**(9), 1013–1038 (2013)
5. Burns, A.J., Johnson, M.E., Caputo, D.D.: Spear phishing in a barrel: insights from a targeted phishing campaign. *J. Organiz. Comput. Electron. Commerce* **29**(1), 24–39 (2019)
6. Butavicius, M., Parsons, K., Pattinson, M., McCormac, A.: Breaching the human firewall: Social engineering in phishing and spear-phishing emails (2016). [arXiv:1606.00887](https://arxiv.org/abs/1606.00887)
7. 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)

8. Chu, M., Ingols, K., Lippmann, R., Webster, S., Boyer, S.: Visualizing attack graphs, reachability, and trust relationships with navigator. In: Proceedings of the 7th International Symposium on Visualization for Cyber Security, pp. 22–33. ACM (2010)
9. Dahbul, R.N., Lim, C., Purnama, J.: Enhancing honeypot deception capability through network service fingerprinting. *J. Phys. Conference Ser.* **801**, 1–6 (2017)
10. Dhillon, D.: Developer-driven threat modeling: lessons learned in the trenches. *IEEE Security Privacy* **9**(4), 41–47 (2011)
11. Ek, D., Petersson, J.: Abstraction of MITRE ATT&CK. Bachelor's thesis, KTH Royal Institute of Technology, Stockholm, Sweden (2020)
12. Ekstedt, M., Johnson, P., Lagerström, R., Gorton, D., Nydrén, J., Shahzad, K.: Securi cad by foreseeit: A cad tool for enterprise cyber security management. In: 2015 IEEE 19th International Enterprise Distributed Object Computing Workshop (EDOCW), pp. 152–155. IEEE (2015)
13. Frydman, M., Ruiz, G., Heymann, E., César, E., Miller, B.P.: Automating risk analysis of software design models. *Sci. World J.* **2014**, 1–12 (2014)
14. 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: Proceedings of the 2015 International Conference on Distributed Computing and Networking, p. 30. ACM (2015)
15. Hacks, S., Katsikeas, S., Ling, E., Lagerström, R., Ekstedt, M.: powerlang: a probabilistic attack simulation language for the power domain. *Energy Inf.* **3**(1), 1–17 (2020)
16. Harish, P., Narayanan, P.J.: Accelerating large graph algorithms on the GPU using CUDA. In: High Performance Computing—HiPC 2007, Lecture Notes in Computer Science, pp. 197–208. Springer (2007)
17. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. *MIS Quarterly* **28**(1), 75–105 (2004)
18. Holm, H., Buschle, M., Lagerström, R., Ekstedt, M.: Automatic data collection for enterprise architecture models. *Software Syst. Model.* **13**(2), 825–841 (2014)
19. Holm, H., Shahzad, K., Buschle, M., Ekstedt, M.: P²CySeMoL: predictive, probabilistic cyber security modeling language. *IEEE Trans. Depend. Secure Comput.* **12**(6), 626–639 (2015). <https://doi.org/10.1109/TDSC.2014.2382574>
20. Homer, J., Zhang, S., Ou, X., Schmidt, D., Du, Y., Rajagopalan, S.R., Singhal, A.: Aggregating vulnerability metrics in enterprise networks using attack graphs. *J. Comput. Secur.* **21**(4), 561–597 (2013)
21. Husari, G., Al-Shaer, E., Ahmed, M., Chu, B., Niu, X.: Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources. In: Proceedings of the 33rd Annual Computer Security Applications Conference, ACSAC 2017, p. 103–115. Association for Computing Machinery, New York, NY, USA (2017)
22. 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)
23. Johnson, P., Lagerström, R., Ekstedt, M., Franke, U.: Can the common vulnerability scoring system be trusted? a Bayesian analysis. *IEEE Trans. Depend. Secure Comput.* **15**(6), 1002–1015 (2018)
24. 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)
25. Kang, D., Lee, J., Choi, S., Kim, K.: An ontology-based enterprise architecture. *Expert Syst. Appl.* **37**(2), 1456–1464 (2010)
26. Katsikeas, S., Hacks, S., Johnson, P., Ekstedt, M., Lagerström, R., Jacobsson, J., Wällstedt, M., Eliasson, P.: An attack simulation language for the IT domain. In: Graphical Models for Security, pp. 67–86. Springer International Publishing (2020)
27. 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)
28. 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)
29. Lagerström, R., Johnson, P.: Using architectural models to predict the maintainability of enterprise systems. In: 2008 12th European Conference on Software Maintenance and Reengineering, pp. 248–252. IEEE (2008)
30. Lavrova, D.S., Pechenkin, A.I.: Adaptive reflexivity threat protection. *Automatic Control Comput. Sci.* **49**(8), 727–734 (2015)
31. 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)
32. Maleki, N., Ghorbani, A.A.: Generating phishing emails using graph database. In: Information Security Practice and Experience, pp. 434–449. Springer (2019)
33. Marback, A., Do, H., He, K., Kondamarri, S., Xu, D.: A threat model-based approach to security testing. *J. Software Practice Exp.* **43**(2), 241–258 (2013)
34. Meszaros, J., Buchalcevova, A.: Introducing OSSF: a framework for online service cybersecurity risk management. *Comput. Security* **65**, 300–313 (2017)
35. Musman, S., Turner, A.J.: A game oriented approach to minimizing cybersecurity risk. *Int. J. Safety Security Eng.* **83**(2), 212–222 (2018)
36. Närman, P., Johnson, P., Lagerström, R., Franke, U., Ekstedt, M.: Data collection prioritization for system quality analysis. *Electron. Notes Theor. Comput. Sci.* **233**, 29–42 (2009)
37. 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)
38. Österlind, M., Johnson, P., Karnati, K., Lagerström, R., Vålja, M.: Enterprise architecture evaluation using utility theory. In: 2013 17th IEEE International Enterprise Distributed Object Computing Conference Workshops, pp. 347–351. IEEE (2013)
39. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. *J. Manage. Inf. Syst.* **24**(3), 45–77 (2007)
40. Ross, J.W., Robertson, D.C., Weill, P.: Enterprise Architecture As Strategy: Creating a Foundation for Business Execution. Harvard Business School (2006)
41. Saini, V., Duan, Q., Paruchuri, V.: Threat modeling using attack trees. *J. Comput. Sci. Colleges* **23**(4), 124–131 (2008)
42. 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)
43. Schneier, B.: Attack trees - modeling security threats (1999)
44. 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)
45. Somestad, T., Ekstedt, M., Holm, H.: The cyber security modeling language: a tool for assessing the vulnerability of enterprise system architectures. *IEEE Syst. J.* **7**(3), 363–373 (2013)
46. Sood, A.K., Enbody, R.: Chapter 3 - infecting the target. In: Targeted Cyber Attacks, pp. 23–35. Syngress, Boston (2014)
47. Uschold, M.: Knowledge level modelling: concepts and terminology. *Knowl. Eng. Rev.* **13**(1), 5–29 (1998)

48. Uschold, M., Jasper, R.: A framework for understanding and classifying ontology applications. In: Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5), pp. 1–12 (1999)
49. Uzunov, A.V., Fernandez, E.B.: An extensible pattern-based library and taxonomy of security threats for distributed systems. *Comput. Stand. Interfaces* **36**(4), 734–747 (2014)
50. Vålja, M., Heiding, F., Franke, U., Lagerström, R.: Automating threat modeling using an ontology framework. *Cybersecurity* **3**(1), 1–20 (2020)
51. Vernotte, A., Vålja, M., Korman, M., Björkman, G., Ekstedt, M., Lagerström, R.: Load balancing of renewable energy: a cyber security analysis. *Energy Inf.* **1**(1), 5 (2018)
52. 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)
53. Weiss, J.D.: A system security engineering process. In: 14th National Computer Security Conference, pp. 572–581 (2006)
54. 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), p. 30. IEEE (2006)
55. 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)
56. Xiong, W., Hacks, S., Lagerström, R.: A method for assigning probability distributions in attack simulation languages. *Complex Syst. Inf. Model. Quarterly* **26**, 55–77 (2021). <https://doi.org/10.7250/csimq.2021-26.04>
57. 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)
58. Xiong, W., Lagerström, R.: Threat modeling—a systematic literature review. *Comput. Security* **84**, 53–69 (2019)
59. Xu, D., Nygard, K.: Threat-driven modeling and verification of secure software using aspect-oriented petri nets. *IEEE Trans. Softw. Eng.* **32**(4), 265–278 (2006)
60. Zachman, J.A.: A framework for information systems architecture. *IBM Syst. J.* **26**(3), 276–292 (1987)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Emeline Legrand graduated in 2020 from the French engineering school ENSTA Paris with a Master's degree in Computer Science and Cybersecurity. She has since joined Wavestone and is now working as a cybersecurity consultant in France.

Oscar Åberg is a student at KTH Royal Institute Of Technology, Sweden in the combined master's and bachelor's programme in Computer Science. His bachelor thesis is about validating the Meta Attack Language using the MITRE ATT&CK matrix. Since then he has participated in the development of enterpriseLang. He is currently working at Ericsson AB.

Robert Lagerström is an associate professor at the School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Sweden. His research is focused on enterprise security, threat modeling, attack simulations, vehicle security, and viable cities. Robert is a member of the Young Academy of Sweden and one of the founders of Foreseeti AB

Wenjun Xiong is a Ph.D. student in Electrical Engineering at the School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Sweden. She obtained her M.Sc. degree in Telecommunications Engineering from Wuhan University, China, in 2017. Her research interests include threat modeling, enterprise security, and attack simulations.