Analyzing Enterprise Architecture Models by Means of the Meta Attack Language

Adina Aldea[1] and Simon Hacks[2(✉)]

[1] University of Twente, Enschede, The Netherlands
a.i.aldea@utwente.nl
[2] University of Southern Denmark, Odense, Denmark
shacks@mmmi.sdu.dk

Abstract. The digital transformation exposes organizations to new threats endangering their business. A way to uncover these threats is threat modeling and attack simulations. However, modeling an entire organization by hand is time consuming and error prone. Therefore, we propose to reuse Enterprise Architecture (EA) models. In this work, we propose a mapping from ArchiMate, a common EA modeling language, to coreLang, a threat modeling language, and use the resulting models to perform attack simulations to foresee possible attack paths. Then, we play back the results of the attack simulations to the EA model and complete the round-trip. To demonstrate our approach, we developed a prototype performing the transformation from ArchiMate to coreLang and applied our approach to the well-known ArchiSurance example.

Keywords: ArchiMate · Attack simulations · Automated analysis · EA security

1 Introduction

Digital transformation has been a topic of interest for many organizations in the past decade, as it changes customer relationships, internal processes and value creation [1]. On the one hand, it offers new possibilities for business model innovation and the ability to disrupt markets to gain lasting competitive advantage [2]. On the other hand, it exposes organizations and customers to new threats. Common types of cyber-attacks include email bombing, information and data theft, Distributed Denial of Service (DDoS) attacks, Trojan viruses, and hacking the data or the system that accesses it [3]. The COVID-19 pandemic has accelerated the process of digital transformation as many organizations needed to support a remote workforce and to provide their products and services online [4]. In 2020 58% of customer interactions were digital compared to 36% in 2019 [5].

While some organizations benefit from the effects of the digital transformation [5], others struggle more than before [4]. Social engineering attacks are especially dangerous in this period of time due to more people working and studying

X. Franch et al. (Eds.): CAiSE 2022, LNCS 13295, pp. 423–439, 2022.
https://doi.org/10.1007/978-3-031-07472-1_25

remote and using videoconferencing software. The US Federal Trade Commission estimated that 12 million dollars were lost from COVID-19 related scams between January and April 2020, with similar reports from other countries [6].

Thus, the importance of cyber security is becoming more apparent. While research in the area of cyber security is progressing, there are not many studies that focus on the security aspects of Enterprise Architecture (EA). EA can be used by organizations to manage their digital transformation by designing, planning, and implementing organizational change from the point of view of their business, application, technology, and physical architecture [7]. One popular language for EA modeling is the ArchiMate specification [7]. We have chosen to use the ArchiMate language for our research due to its popularity [8], it is used by many organizations. However, while the EA models created with ArchiMate can help organizations with visualizing their IT landscape, they do not natively support any kind of security analysis.

Current research addressing this gap has several limitations, such as a manual security analysis [9], the information from the security analysis being used without using the models as input [10], or the use of other languages than ArchiMate as a source for the security analysis [11]. In this research, we aim to address these limitations and contribute to the state-of-the-art by proposing a way to automate the transformation and analysis of ArchiMate models from a security perspective. First, we propose a mapping between the concepts of the ArchiMate language and coreLang. Second, we perform an analysis of the possible security vulnerabilities with the help of the securiCAD attack simulations. Third, we enrich the ArchiMate models with the results of the analysis in a way that illustrates the most likely attacks on the EA of an organization.

Based on these limitations and our previous research [12], we assume the case of an organization that has a large EA model repository (often in the notation of ArchiMate [7,8]) at hand and desires to perform a security analysis to reduce the risk of unwanted compromise of the organization. Usually, one would scan the systems and perform an automated vulnerability analysis. However, this is just partly possible, as EA models usually lack detailed information that is necessary to identify concrete vulnerabilities (such as concrete versions). Due to the size of the repository, it is not feasible, to enrich the EA model itself with this additional security relevant information as proposed by other authors [9,13], because it would be too much effort and enterprise architects classically lack the needed security knowledge. From these observations, we deduct following requirements:

1. ***Should contribute towards automation of the cyber security analysis of ArchiMate models***. With the help of model transformation rules, we map the ArchiMate metamodel to the coreLang metamodel, to facilitate the automated transfer of models. Once this step is completed, we can perform the automated cyber security analysis with the help of the securiCAD attack simulations. Thus, the EA models can be reused for security analysis and every time the EA model is updated, the security analysis can be updated as well.

2. ***Should reuse existing ArchiMate models.*** Effort for security analysis can be substantially reduced by reusing information that is already available in an organization. One possible source for the underlying structure of an organization can be EA models [9], that are often maintained in ArchiMate [7,8]. Thus, we rely on this notation for our approach, while aiming for a generalization for other notations in future. However, most EA practitioners model their architecture at a high level of abstraction. This could pose certain challenges when trying to perform security simulations due to the difference in the levels of abstraction, which we address with the next requirement.
3. ***Should not add additional security elements to existing ArchiMate models.*** Considering the fact that EA models do not natively support cyber security analysis, it stands to reason that the information that is needed to perform these analyses is also not included in the models (separation of concerns). Thus, we consider that our approach to transfer the models into an environment that is designed with the purpose of performing these analyzes and adding the relevant information is suitable for achieving our goals. Moreover, we close the gap between the high abstraction of EA models and the needed detail level for security simulations by using different configurations of the elements to reflect uncertainty about missing information such as concrete versions of applications [12].
4. ***Should show a simple overview of the cyber security analysis results in the original ArchiMate models.*** Considering the complexity of data resulting from the attack simulations which can span through 1000 s s of scenarios, it stands to reason that not all of the results are useful to transfer back to the ArchiMate models. Especially, managers are usually interested in abstracted information [14]. Thus, the existing elements of the EA model will be enriched with properties reflecting an abstraction of a few scenarios (based on likelihood and impact).

For this research, we adhere to the guidelines of the Design Science Research by Peffers et al. [15], which is also reflected in the structure of this paper. Section 2 presents the background on the ArchiMate and MAL languages and discusses the related work. In Sect. 3 we propose the mapping between the ArchiMate and the coreLang, while in Sect. 4 we demonstrate how this can be used with the help of a case study. Sections 5 and 6 focus on the discussion and conclusions.

2 Background

2.1 ArchiMate

ArchiMate is a language maintained by the Open Group that can be used to model EAs and their transformations in relation to the motivation behind them [16]. The language is structured according to a two-dimensional framework which contains six layers and four aspects.

At the intersection of the six layers and the three aspects (except for Motivation), certain views are defined. These views determine, which concepts and relations can be used to visualize a certain part of the architecture that is relevant to this part of the framework. Thus, for each of the views a metamodel is defined.

For each of these layers and aspects, certain concepts are defined. The Business layer contains concepts relating to the services offered to customers and the processes and actors that support it, while the Application layer includes concepts relating to the software applications and data used to support the business layer, the Technology layer focuses on the infrastructure of the organization with servers where data can be stored and networks that connect the devices, and the Physical layer helps describe how certain physical products can be produced.

The concepts in the Motivation aspect can be used by organizations to model the reasons for the changes in the architecture, while the Strategy layer can help define how these changes can be done at a higher abstraction level than the other four aforementioned layers. The Implementation and Migration layer, on the other hand, helps organizations with planning how these architectural transformations can be performed in order to achieve the best outcomes.

2.2 MAL

MAL is a meta-meta language, which combines probabilistic attack and defense graphs with object-oriented modeling. It is used to create Domain-Specific Languages (DSLs) that provide a meta language, which can be used to create models for attack simulations. Such a DSL defines the required information for the models and specifies the generic attack logic about the domain studied. For a detailed overview of MAL, we refer to the original paper [17].

To create a MAL-based language, one needs to identify all relevant assets and their associations. Each asset is comprised by multiple attack steps, which lead to "−>" 0 to n attack steps. Each attack step is either of the type OR "|" or AND "&". If one parent attack step of an OR attack step is compromised, an attacker can elaborate on this OR attack step. If all parent attack steps of an AND attack step are compromised, an attacker can elaborate on this AND attack step. Additionally, one can define the expected effort an attacker needs to spend on an attack step also called time-to-compromise (TTC). Further, an asset can contain defenses "#". Combining all possible attack paths lead to the attack/defense graph used for the attack simulation.

MAL provides the frame to create a DSL for threat modeling and attack simulations from scratch. At the same time, we recognized that many DSLs created with MAL share common concepts. Thus, we proposed `coreLang` [18] as means to reduce unnecessary redundant work. coreLang provides a basic set of assets that serve as starting point to model more advanced MAL DSLs or act as a basic language to model simple environments [11]. Figure 1 illustrates the overall structure of coreLang with respect to the concepts used in this work and presented next. For more details, we refer to the original publication [18].

The central concept that we use in this work is `Application`, which is characterized by its executability i.e., being software. Moreover, an `Application` is

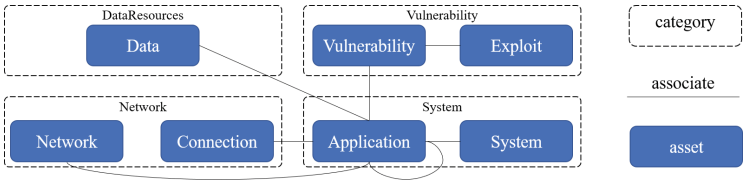

Fig. 1. Excerpt of `coreLang` containing used concepts (adapted from [18]).

able to execute another `Application` such as an operation system executing a program. For the execution itself, some hardware is needed that performs the actual calculations. This functionality is modeled as `System`.

Between two `Application` not just an execution relationship is possible, but they can also communicate with each other e.g., via interfaces. Such a communication is realized by a `Connection`, which is always intended and can take place within the same or different networks. On the other hand, `Application` can be contacted unintended e.g., by performing port scans. To reflect this, an `Application` can be exposed to a `Network`. If an attacker has access to such a `Network`, they can connect to all exposed `Application`.

Usually, an attacker is interested in the `Data` that is processed by the `Application`. Generally, `Data` can have two states. Firstly, it can be stored by an `Application`. Then, an attacker would be able to read, write, or delete the `Data`. Secondly, `Data` can be in transition between two `Application`. Then, the attacker can affect the `Data`, e.g., by a man-in-the-middle (MITM) attack.

While introducing coreLang, we mentioned its basic structure, which causes the abstract nature of the concepts. Thus, these assets cannot contain concrete attack steps and related TTC. Instead, just attack concepts, such as MITM on `Connection`, are contained. To enable the end-user to use coreLang nonetheless, the concepts of `Vulnerability` and `Exploit` have been introduced. They allow the end-user to relate a `Vulnerability` to an `Application`, which allows the attacker to compromise the `Application` via an `Exploit`. Moreover, they mirror the Common Vulnerability Scoring System (CVSS)[1] allowing to compute TTC.

3 Mapping

Before, we presented the basic concepts of ArchiMate, MAL, and coreLang. In our mapping, we rely on ArchiMate, as it is a wide-spread standard for EA modeling [7,8]. We opted for MAL, as it provides a framework for automated attack simulations, while being flexible towards different domains [17,19]. Finally, we opted for coreLang as it represents basic concepts in IT systems [18] and it has already been successfully used in similar contexts [12,20]. Next, we motivate our mapping from ArchiMate to coreLang. However, ArchiMate contains a broad

[1] https://www.first.org/cvss/specification-document.

range of different element types and, therefore, we reason which categories of element types we consider for being mapped.

First, we check which layers of ArchiMate are eligible to be mapped to coreLang. The *Strategy* layer models strategic directions and decisions of organizations [16]. As coreLang contains solely technical assets, there are no equivalents and, thus, we neglect this layer. A similar argumentation can be applied to the *Business* layer, as it technological-independently models the organization [16].

The *Application* layer describes the structure, behavior, and interactions of the organization's applications [16]. This relates to the executable assets [18] and, thus, we consider this layer for our mapping. The *Technology* layer models the technical infrastructure of organizations [16], which is classically referred to executables that are not exposed to the end-user. From a security perspective, these executables behave as on the Application layer. Moreover, the Technology layer includes also the *Physical* layer, which is used to model the physical world, i.e., hardware [16]. coreLang also foresees the modeling of hardware components [18], which lets us consider this layer.

The *Implementation & Migration* layer encodes the evolution of the EA [16]. However, MAL DSLs take a static view on the architecture [17], which makes this layer not of interest for our mapping. Finally, the *Motivation* layer depicts the motivation for the EA's evolution [16], which also has no equivalent in coreLang.

As we have determined the layers to be considered, we can shrink further down the number of element types that we need to evaluate for the mapping, as there are basically three different aspects in ArchiMate.

Active Structure element types are performing actions in the EA, i.e., applications serving the business. Consequently, these element types are of interest for security analysis. Such Active Structures perform *Behavior*, which are the dynamic aspects of organizations [16]. Behavior unites the activities of several Active Structures and are a logical grouping. Consequently, we do not consider those elements for our mapping. *Passive Structure* element types represent data objects [16]. Attackers are interested in the data of organizations and, therefore, they constitute the ultimate goal in our attack simulations.

Next, we take a close look at the remaining ArchiMate element types to be mapped to coreLang (cf. Table 1). Obviously, we map *Application Component* to `Application` and *Data Object* to `Data`. As *Application Collaboration* represents multiple Active Structure elements [16], it is a logical construct and we do not map it. Similarly, *Application Interface* is a point of contact related to a certain *Application Component* and is not related to an own execution.

Equally, we can argue for *Technology Collaboration* and *Technology Interface*. In contrast, *Node* and *Device* are characterized by hosting executables [16] and are in line with the conceptualization of `System`. Such executables in the Technology Layer are *System Software*, which execute *Application Component* [16] and are mapped to `Application`. The communication between several *Node* is modeled by *Path*. This communication can also take place along several networks [16], which leads to a mapping to `Connection`. If the *Node* share the same network, this connection is codified by a *Communication Network* [16], which is in line with `Network`. Finally, *Artifact* is "a piece of data" [16], which matches `Data`.

Table 1. Mapping from ArchiMate to `coreLang`

Layer	Element type	Application	Data	System	Network	Connection	PhysicalZone
Application	Application component	✖					
	Application collaboration						
	Application interface						
	Data object		✖				
Technology	Node			✖			
	Device			✖			
	System software	✖					
	Technology collaboration						
	Technology interface						
	Path					✖	
	Communication network				✖		
	Artifact		✖				
Physical	Equipment			✖			
	Facility						✖
	Distribution network						
	Material						

Equipment are physical machines that process materials [16]. These machines are controlled by software, which they host. Consequently, we map *Equipment* to `System`. A *Facility* is a grouping of *Equipment* deployed at one physical location [16]. The best representation for this is `PhysicalZone`. A *Distribution Network* transports *Material* or energy [16]. As coreLang is not equipped with assets representing physical concepts, there is no representation neither for both.

4 Demonstration

To realize the automated attack simulations for EA models, we follow a five-stepped process. (1) The EA model is translated to its graph representation. (2) The graph is modified to match coreLang specificities. (3) The graph is transformed into its representation, which can be interpreted by securiCAD. (4) Additional vulnerability information is added to the securiCAD model. (5) The simulation results are imported and visualized in the original EA model. Hitherto, the steps (1) to (3) are fully automatized, while (4) and (5) need still some manual work.

4.1 Illustrative Example

We choose to demonstrate the application of our proposed mapping and simulation with the help of the ArchiSurance case study, which is used in several previous papers [7,21]. We expand upon the previous details available in this case and enrich it with information that is necessary for our demonstration. Below, we provide a short description of the case.

ArchiSurance is an insurance merged from three independent insurance organizations, because they could not maintain their competitive advantage without significant investments in IT. By merging, they are able to reduce costs, invest in new technology, maintain customer satisfaction, and explore opportunities in emerging markets. After the merger, ArchiSurance went through several transformations of their landscape in order to unify in the customer facing processes, reduce redundancies, and merge customer data from the different data centers.

Due to these transformations, the organization is concerned that their landscape might be facing new threats. The need for insight into possible security vulnerabilities is emphasized due to a recent increase in cyber-attacks, especially in the insurance industry. Several insurances were subjected to ransomware attacks which resulted in a loss of trust from customers and potential lawsuits.

4.2 Processing

To perform the attack simulations for ArchiSurance, we followed a five-stepped process implemented in our prototype[2]. First, the XML of the ArchiMate model is translated to its graph-representation. Therefore, each element becomes a node and each relationship an edge. Moreover, we preserve the name, id, and ArchiMate element/relationship type as attributes to the node/edge.

Second, we modify the graph to match assumptions made in coreLang. On the one hand, we add additional nodes for certain relationships as the respective relationships are not foreseen in coreLang. This is the case for *Application Component serving* another *Application Component*. In coreLang, `Application` are just linked directly to each other if one is executing the other. A communication as indicated by "*serving*" is realized via a `Connection`. Therefore, we introduce a `Connection` for such relationships and link it to the related `Application`.

Additionally, it is not presumed that *Device* or *Node* (mapped to `System`) are directly exposed to a *Communication Network* (mapped to `Network`). Instead, the operation system (i.e., *System Software*) running on the *Device* or *Node* is communicating via the *Communication Network*. Consequently, we add for each of those communications a *System Software* and related edges.

The last modification is related to the access of ArchiMate elements to *Data Object/Artifact*, which is from the former to the latter. In contrast, coreLang defines that `Data` is accessed by `Application`. To overcome this, we simply inverse the direction of the respective edges in the graph-representation.

Third, the graph is translated into a securiCAD format [19], the tool we use for attack simulations. This translation is guided by the mapping presented in

[2] https://github.com/simonhacks/EA-Resilience.

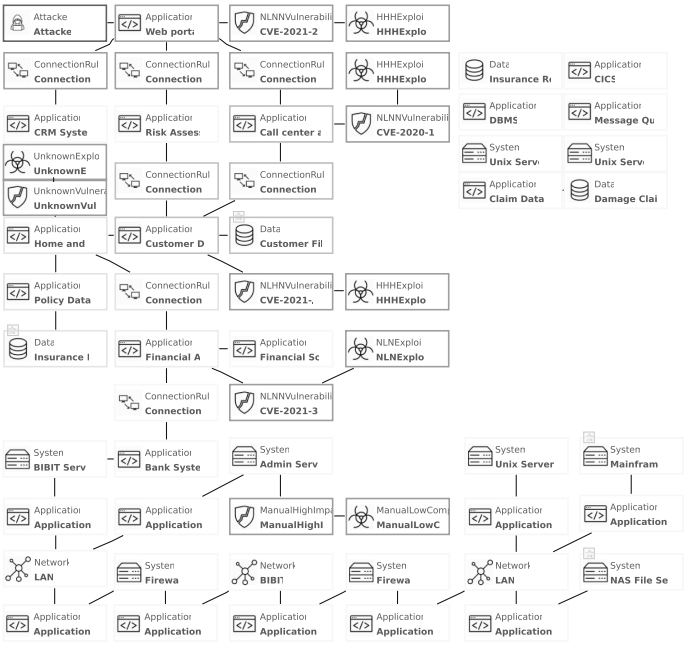

Fig. 2. securiCAD model with external attacker and possible vulnerabilities and exploits

Sect. 3 and if an ArchiMate element type is not in this mapping, the node and related edges will be ignored. Moreover, it takes a MAL DSL as input (e.g., coreLang [18]) to ensure that the produced result conforms to the DSL.

Fourth, we enrich the securiCAD model (cf. Fig. 2) with security relevant information on the attack surface, possible vulnerabilities, and exploits as coreLang does not provide such information by itself [18]:

- We assume that ArchiSurance uses WordPress to host their *Web portal* and WooCommerce as plugin to sell their insurances to the customers. The used WooCommerce version allows unauthenticated users to upload files even with executable content so that the installation can be overtaken[3].
- As the *Call center application* of ArchiSurance, they use the bitrix24 software. The used version allows for a server-side request forgery (SSRF)[4], which can be exploited to get deeper into ArchiSurance's network.
- *Customer Data Access* is provided by an application, which runs on an oracle cloud instance. Due to CVE-2021-2320[5], an attacker can take over the storage gateway and influence other applications hosted on that infrastructure.

[3] https://nvd.nist.gov/vuln/detail/CVE-2021-24212.
[4] https://nvd.nist.gov/vuln/detail/CVE-2020-13484.
[5] https://nvd.nist.gov/vuln/detail/CVE-2021-2320.

- *Home and Away Policy Administration* is an own implementation and, hence, it is not possible to gather vulnerabilities from a central repository. Therefore, we use the `UnknownVulnerability` and `UnknownExploit` with a low probability to be present to express that there might be a security issue in this application.
- *Financial Application* is hosted in an SAP environment and implemented in NetWeaver. A security scan could discover a weakness[6] that allows an attacker to inject commands that endanger the integrity of the system.
- Due to security restrictions, the *Admin Server* is not directly connected to the internet. To provide updates to the machine, it offers a USB interface that uses USB Pratirodh to allow just encrypted devices. However, an attacker can change usernames and passwords to take over the respective `System`[7].

Finally, the attack simulations can be performed, and the results be exported. The export contains information on which attack steps have been compromised by the attacker in which time and with which likelihood. To include this information into the original EA model, we determine for each asset one attack step for which the asset can be seen as fully compromised, i.e., it is annotated with consequences for confidentiality, integrity, or availability. Then, the TTC of that attack step is considered to be reported in the EA model.

4.3 Analysis

While mapping the ArchiMate elements to the coreLang is straightforward, the same cannot be said about incorporating the results of the simulation back into the ArchiMate model. This is mostly due to the complexity of the simulation results which provide a multidimensional perspective on each element that is analyzed. Thus, for each element, several attack steps can be possible, with varying degrees of probability determined based on 1000 simulations.

For example, in Fig. 3 we see that the ArchiSurance organization uses a Policy Data Management application to support their Handle Claims business process. Based on the results of the simulation, this element of the architecture can be exposed to 24 different attack steps, of which 11 attack steps have a probability of success that is higher than 0. Thus, to map this information back to the ArchiMate model, we need to be able to relate all the different attack steps and their probability values to the Policy Data Management application component. However, this mapping is not possible to do automatically, as the language does not currently support this type of data mapping.

Following [22], we propose to use the language extension mechanism and introduce the concept of Metric as a specialization of Driver. With the help of the Metric concept, we can relate the different attack steps and their probability values to each element of the architecture. For this purpose, we create two metrics, namely External Attack (attack steps) and Probability External Attack

[6] https://nvd.nist.gov/vuln/detail/CVE-2021-33663.
[7] https://nvd.nist.gov/vuln/detail/CVE-2017-6911.

(probability). The first metric can store all of the possible attack steps while the second metric can store the numeric values associated with the probabilities.

To make the results of the analysis visible in the ArchiMate model, we use a color and label overlay. The color is used to show the probability while the label shows the attack step. The reason for this is that the most important information, in this case, the probability of an attack succeeding, should be the most visible. Since it is not possible to visualize in one view all the different attack steps and their probabilities, we have chosen to show in Fig. 3 only the most severe attacks that have a probability of success higher than 0. The other attack steps and their probabilities can be visualized in a similar manner.

5 Discussion

Our work contains several points that need discussion. First, one can criticize our decision on the used languages. To our best knowledge, there is no alternative to MAL that would allow automated attack simulation, while already providing an increasing ecosystem of threat modeling languages for different domains [11].

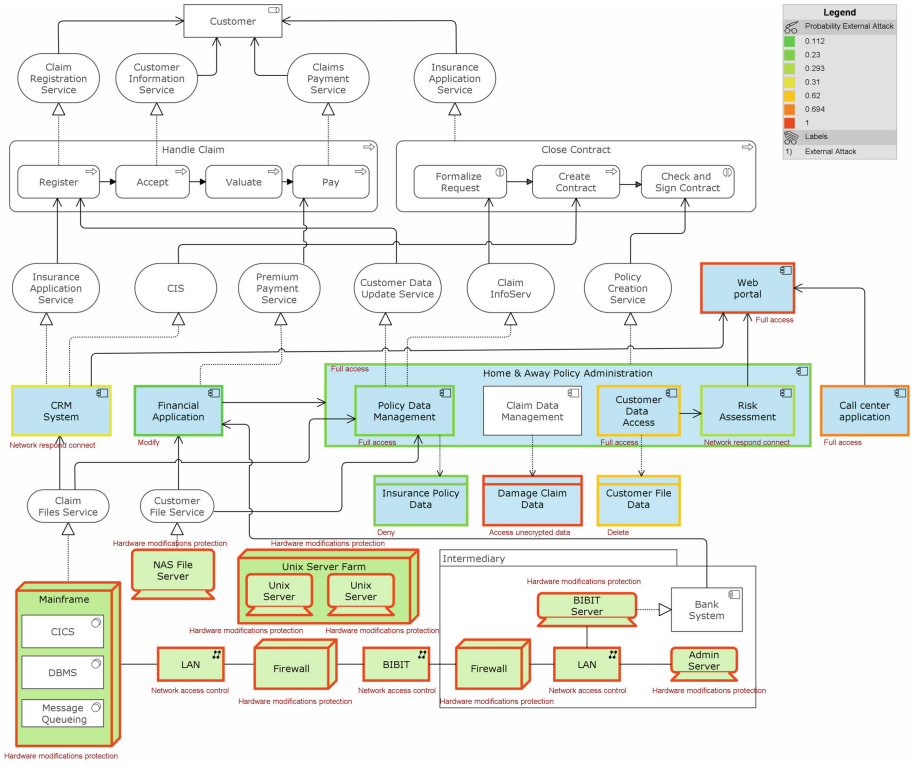

Fig. 3. ArchiSurance security analysis

Adapting our approach to other (not MAL) threat modeling languages, might be challenging as the model transformation would be to be implemented from scratch, while the rest of our approach could remain same.

coreLang is abstract and does not provide security relevant information by itself [18]. However, it is the MAL DSL at hand, which is closest to our needs [11] and has already been used in similar contexts [12,20]. Moreover, other languages are tailored to more concrete domains [23–25] making it challenging to find a mapping to a general-purpose language such as ArchiMate. Considering generalization, there is a growing ecosystem of languages that build on coreLang [11]. Consequently, our approach can be adapted to these languages by further refining the mapping. For languages that are not build on coreLang, a completely new mapping has to be sketched, while the rest of our approach would remain.

ArchiMate is a wide-spread notation for EA models in organizations and research [7,8]. Therefore, our approach is applicable to a broad range of existing EA models. However, recent efforts have been taken, to understand EA models as Knowledge Graphs [26,27]. Considering Knowledge Graph representation of EA models as input to our approach, would allow us to include all EA model notations, which have a graph representation, and thus mean a large step towards generalization from this end. However, this would also include the need to a generalized mapping from EA model notations towards our desired MAL DSL. Unfortunately, such a general representation for EA models does not exist. Hence, this step towards generalization remains for future work.

The second point of improvement is also related to the decision for coreLang, as the absence of vulnerability information forces us to add this by hand. On the one hand, we added just exemplary vulnerabilities to the model to avoid overcrowding the visualization. In a real-world context, one would include all discovered vulnerabilities. For the demonstration purposes, adding a subset of possible vulnerabilities is sufficient as we can perform the related attack simulations and those would just increase in complexity. On the other hand, one could employ automated vulnerability scans to gather the respective vulnerabilities. However, this would demand an infrastructure to scan, which is not at our disposal. Similarly, the EA model contains no concrete information on the applications, which forces us to make assumptions on what exact applications and which versions are deployed in the EA. Having a real-world example at hand, which is planned for future work, we will not experience this issue.

Third, the mapping from ArchiMate to coreLang can be differently realized. The most influential decision was to neglect transitive relations between mapped elements related to the behavior aspect. For instance, if an *Application Component* is executed by a *System Software* but it is modeled by the consumption of a *Technology Service*, this information will be lost. Unfortunately, it is not that simple to solve this issue, because a deeper inspection is needed to determine if a transitive relation should be still contained in the security analysis, which we plan to address in a future iteration of our research.

Fourth, another issue is related to the fact that coreLang does not allow `System` to communicate directly over `Network` with each other. Therefore, we introduced additional `Application` symbolizing the operation system, which

take care for the communication. As we tried to keep it simple, we introduced for each connection of a `System` to a `Network` such an `Application`. If a `System` has several connections to `Network`, it results in multiple `Application` added to the model. In the next iteration, we aim for a more realistic solving of this issue, as one `System` solely can have one operation system.

Fifth, the EA model contains elements named "Firewall", which are translated to `System`, as we just consider the element type and do not perform a deeper inspection of the elements. However, such a deep inspection is pointless in our case as the firewall is an exception in coreLang and the rest of the assets is on an abstract level. In case that there will be a MAL DSL available, which is more specific than coreLang, then relying the mapping not just on the element types but also on further information available (e.g., element names) might be suitable.

Sixth, the transformation of the simulation results back to the ArchiMate model had to be done manually. Since the ArchiMate language does not support the complexity of the information resulting from the simulation, certain simplifications had to be made. For example, we could not visualize all the attack steps that would affect certain elements of the architecture but had to choose one attack step at a time. To do this, we chose the types of attack steps that had a probability higher than 0 and have the highest impact on the architectural element. Similarly, we did not consider all the values, but rather focus only on the probability value that was calculated. However, this simplified information can still provide enterprise architects with a good overview of which types of attack steps can have the most impact and their probability of success.

Furthermore, the purpose of ArchiMate is to support the high-abstraction modeling of EA elements. Thus, enriching ArchiMate models with the type of data required for a detailed security analysis as presented in this paper (e.g., version or instance number) would not be aligned with the purpose of the language and falls into the realm of IT service management. While previous work has proposed the extension of the ArchiMate language to include Security related concepts [28], this approach is not sufficient to support the simulation-based analysis that we propose in our paper. In future work, the feasibility of using a language extension mechanism to support our proposed approach could be investigated.

Lastly, while we have demonstrated how our approach can be used to conduct a detailed security analysis of ArchiMate models, additional qualitative analysis to validate the results is needed. Thus, we plan interviews with EA and Security experts to validate the usefulness and usability of our approach. Additionally, to better understand the business impact of these potential security threats, the analysis should be extended to encompass the link between the technical and business elements, similar to Ebbers et al. [29].

6 Related Work

There are different efforts to enrich ArchiMate with security information. Grandry et al. [13] mapped the concepts of an information system security risk

management to ArchiMate. The main limitation of this work in relation to our goals and requirements is that the cyber security analysis is done manually and based only on a conceptual representation. This relies on the ability of the person analyzing the architecture to analyze all of the relevant architectural risks, as opposed to a simulation-based analysis which is able to run through 1000 s of scenarios.

Band et al. [28] extended this work and demonstrated the connection between ArchiMate and other risk and security concepts. Their conclusion was that most of the common risk and security concepts can be realized in ArchiMate. Aldea et al. [30] identified relevant metrics to assess the overall resilience of a given EA model as a means for future research. These works are characterized by incorporating security or risk relevant information directly into ArchiMate. In contrast, for our approach it is not needed to make changes to the models itself as the security information is codified in MAL. Moreover, we perform simulations to see the architecture's influence on the security.

Manzur et al. [31] took a step further and enhanced ArchiMate to xArchiMate, which can be used to perform simulations, experimentations, and analyze EAs by an extension to the ArchiMate meta-model. A similar work was done by Grov et al. [32] focusing more on risk aspects and developing a tool to visualize the effects in EA. This is similar to our approach, while we leave the ArchiMate meta-model untouched and transform the model instead to an instance of a MAL DSL.

Generally, EA models are a popular input for security assessments. Mathew et al. [9] elaborated on the reuse of EA related information in a security context. Therefore, they identified similarities between the management parts of TOGAF and BSI Grundschutz and proposed a mapping from ArchiMate to the German BSI Grundschutz' meta-model. Similarly, Xiong et al. [33] use EA repositories to predict effects of failing components on the entire architecture. Ebbers et al. [29] use EA models to aggregate vulnerabilities from assets to a level of relevance for the management. Lastly, Pavleska et al. [34] took EA models in a health context and transferred them by hand to securiCAD to perform security analysis. Similarly, Jiang et al. [35] did with models from the power domain. All works are like ours as EA related information is reused for security analysis. However, these were performed by hand, while we aim to automate it.

There are also works reusing security models for EA. Holm et al. [10] proposed a mapping of the NeXpose Scanner to ArchiMate. Later, they use these models as foundation for attack simulations in securiCAD [36]. König et al. [37] mapped the Substation Configuration Language (SCL) to ArchiMate to better ease the stakeholders' understanding of the Substation Automation (SA) system and its architecture. Comparing these works to ours, we recognize that the previous work focuses on models representing reality and including them in EA models, while we perform analysis on these models and solely play back the results.

Finally, there are works focusing on the reuse of existing models for attack simulations in MAL. Firstly, Hacks et al. [38] propose a method to automatically create a MAL DSL based on an EA model. Secondly, Hacks et al. [12] developed

a mapping from the Business Process Modeling Notation (BPMN) to coreLang [18], automatically transform these models to a graph representation, and perform attack simulations in securiCAD. We extend the first work by providing a tool that also translates ArchiMate models into instances of MAL DSLs and the second work by enabling such transformation for ArchiMate models.

7 Conclusion

In this work, we have presented an automated transformation from EA models to threat models that are then used to perform attack simulations. Therefore, we proposed a mapping from ArchiMate to coreLang. With the help of this mapping, existing ArchiMate models which do not contain any cyber security information can still be analyzed. This is in line with the first three requirements mentioned in the 1 section of the paper.

Finally, we incorporate the simulation results back to the EA model to visualize the generated insights for the EA practitioners. With the help of this approach, the original ArchiMate models maintain their value as the single source of truth regarding the EA and are enriched with the most relevant cyber security analysis results to help EA practitioners in their decision-making process. To demonstrate our approach, we developed a first prototype and used the well-known ArchiMate example of ArchiSurance. This is in line with the fourth requirement mentioned in Sect. 1.

Still, future work remains. As such, we plan to implement our approach in different organizations to analyze real-world EAs and further investigate the link between technical and business assets. Additionally, we will investigate further a suitable way to cope with the simulation results in the EA model. Moreover, we are looking forward to a MAL DSL that is tailored for office IT environments to avoid the need for adding vulnerabilities by hand.

References

1. Zaoui, F., Souissi, N.: Roadmap for digital transformation: a literature review. Proc. Comput. Sci. **175**, 621–628 (2020)
2. Verhoef, P.C., et al.: Digital transformation: a multidisciplinary reflection and research agenda. J. Bus. Res. **122**, 889–901 (2021)
3. Chowdhury, A.: Recent cyber security attacks and their mitigation approaches – an overview. In: Batten, L., Li, G. (eds.) ATIS 2016. CCIS, vol. 651, pp. 54–65. Springer, Singapore (2016). https://doi.org/10.1007/978-981-10-2741-3_5
4. Lallie, H.S., et al.: Cyber security in the age of COVID-19: a timeline and analysis of cyber-crime and cyber-attacks during the pandemic. Comp. Sec. **105**, 102248 (2021)
5. LaBerge, L., O'Toole, C., Schneider, J., Smaje, K.: How COVID-19 has pushed companies over the technology tipping point - and transformed business forever (2020)
6. Hakak, S., Khan, W.Z., Imran, M., Choo, K.K.R., Shoaib, M.: Have you been a victim of COVID-19-related cyber incidents? Survey, taxonomy, and mitigation strategies. IEEE Access **8**, 124134–124144 (2020)

7. Aldea, A., Iacob, M.E., van Hillegersberg, J., Quartel, D., Franken, H.: Modelling value with ArchiMate. In: Persson, A., Stirna, J. (eds.) CAiSE 2015. LNBIP, vol. 215, pp. 375–388. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19243-7_35
8. Barbosa, A., Santana, A., Hacks, S., Stein, N.V.: A taxonomy for enterprise architecture analysis research. In: ICEIS, vol. 2, SciTePress, pp. 493–504 (2019)
9. Mathew, D., Hacks, S., Lichter, H.: Developing a semantic mapping between TOGAF and BSI-IT-Grundschutz. MKWI **5**, 1971–1982 (2018)
10. Holm, H., Buschle, M., Lagerström, R., Ekstedt, M.: Automatic data collection for enterprise architecture models. Softw. Syst. Model. **13**(2), 825–841 (2012). https://doi.org/10.1007/s10270-012-0252-1
11. Hacks, S., Katsikeas, S.: Towards an ecosystem of domain specific languages for threat modeling. In: La Rosa, M., Sadiq, S., Teniente, E. (eds.) CAiSE 2021. LNCS, vol. 12751, pp. 3–18. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-79382-1_1
12. Hacks, S., Lagerström, R., Ritter, D.: Towards automated attack simulations of BPMN-based processes. In: EDOC, pp. 182–191 (2021)
13. Grandry, E., Feltus, C., Dubois, E.: Conceptual integration of enterprise architecture management and security risk management. In: EDOCW, pp. 114–123, September 2013
14. Hacks, S., Brosius, M., Aier, S.: A case study of stakeholder concerns on EAM. In: EDOCW, pp. 50–56. IEEE (2017)
15. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. JIMS **24**(3), 45–77 (2007)
16. The Open Group: ArchiMate 3.1 Specification (2019)
17. Johnson, P., Lagerström, R., Ekstedt, M.: A meta language for threat modeling and attack simulations. In: ARES, p. 38. ACM (2018)
18. Katsikeas, S., et al.: An attack simulation language for the IT domain. In: Eades III, H., Gadyatskaya, O. (eds.) GraMSec 2020. LNCS, vol. 12419, pp. 67–86. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-62230-5_4
19. Ekstedt, M., Johnson, P., Lagerström, R., Gorton, D., Nydrén, J., Shahzad, K.: securiCAD by foreseeti: a CAD tool for enterprise cyber security management. In: EDOCW, pp. 152–155. IEEE (2015)
20. Hacks, S., Butun, I., Lagerström, R., Buhaiu, A., Georgiadou, A., Michalitsi Psarrou, A.: Integrating security behavior into attack simulations. In: ARES, pp. 1–13 (2021)
21. Aldea, A., Iacob, M.E., Quartel, D., Franken, H.: Strategic planning and enterprise achitecture. In: ES 2013, pp. 1–8. IEEE (2013)
22. Aldea, A., Iacob, M.E., Daneva, M., Masyhur, L.H.: Multi-criteria and model-based analysis for project selection: an integration of capability-based planning, project portfolio management and enterprise architecture. In: EDOCW, pp. 128–135 (2019)
23. 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). https://doi.org/10.1186/s42162-020-00134-4
24. Ling, E.R., Ekstedt, M.: Generating threat models and attack graphs based on the IEC 61850 system configuration description language. In: SAT-CPS, pp. 98–103. ACM (2021)
25. 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 ICISSP (2019)

26. Smajevic, M., Bork, D.: From conceptual models to knowledge graphs: a generic model transformation platform. In: ER. Springer. LNCS (2021)
27. Smajevic, M., Hacks, S., Bork, D.: Using knowledge graphs to detect enterprise architecture smells. In: PoEM, Springer International Publishing, pp. 48–63 (2021)
28. Band, I., Engelsman, W., Feltus, C., Paredes, S.G., Diligens, D.: Modeling enterprise risk management and security with the archimate ®. Language, The Open Group (2015)
29. Ebbers, F., Hacks, S., Thakurta, R.: The business impact of IIOT vulnerabilities. In: PACIS 2021 Proceedings, vol. 225 (2021)
30. Aldea, A., Vaicekauskaitė, E., Daneva, M., Piest, J.P.S.: Assessing resilience in enterprise architecture: a systematic review. In: EDOC, pp. 1–10 (2020)
31. Manzur, L., Ulloa, J.M., Sánchez, M., Villalobos, J.: XArchiMate: enterprise architecture simulation, experimentation and analysis. Simulation **91**(3), 276–301 (2015)
32. Grov, G., Mancini, F., Mestl, E.M.S.: Challenges for risk and security modelling in enterprise architecture. In: Gordijn, J., Guédria, W., Proper, H.A. (eds.) PoEM 2019. LNBIP, vol. 369, pp. 215–225. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-35151-9_14
33. Xiong, W., Carlsson, P., Lagerström, R.: Re-using enterprise architecture repositories for agile threat modeling. In: EDOCW, pp. 118–127 (2019)
34. Pavleska, T., Aranha, H., Masi, M., Grandry, E., Sellitto, G.P.: Cybersecurity evaluation of enterprise architectures: the E-SENS case. In: PoEM, pp. 226–241 (2019)
35. Jiang, Y., Jeusfeld, M., Atif, Y., Ding, J., Brax, C., Nero, E.: A language and repository for cyber security of smart grids. In: EDOC, pp. 164–170 (2018)
36. Holm, H., Shahzad, K., Buschle, M., Ekstedt, M.: P^2CySeMoL: predictive, probabilistic cyber security modeling language. TDSC **12**(6), 626–639 (2015)
37. König, J., Zhu, K., Nordström, L., Ekstedt, M., Lagerstrom, R.: Mapping the substation configuration language of IEC 61850 to ArchiMate. In: EDOCW, pp. 60–68 (2010)
38. Hacks, S., Hacks, A., Katsikeas, S., Klaer, B., Lagerström, R.: Creating meta attack language instances using ArchiMate: applied to electric power and energy system cases. In: EDOC, pp. 88–97 (2019)