The Right Tool for the Job: a Case for Common Input Scenarios for Security Assessment

Xinshu Dong[1], Sumeet Jauhar[1], William G. Temple[1],
Binbin Chen[1], Zbigniew Kalbarczyk[2], William H. Sanders[2],
Nils Ole Tippenhauer[3] and David M. Nicol[2]

[1] Advanced Digital Sciences Center, Singapore
[2] University of Illinois at Urbana-Champaign, IL
[3] Singapore University of Technology and Design, Singapore

Abstract. Motivated by the practical importance of security assessment, researchers have developed numerous model-based methodologies. However, the diversity of different methodologies and tool designs makes it challenging to compare their respective strengths or integrate their results. To make it more conducive to incorporate them for practical assessment tasks, we believe it is critical to establish a common foundation of security assessment inputs to support different methodologies and tools. As the initial effort, this paper presents an open repository of Common Input Scenarios for Security Assessment (CISSA) for different model-based security assessment tools. By proposing a CISSA design framework and constructing six initial scenarios based on real-world incidents, we experimentally show how CISSA can provide new insights and concrete reference points to both security practitioners and tool developers. We have hosted CISSA on a publicly available website, and envision that community effort in building CISSA would significantly advance the scientific and practical values of model-based security assessment.

1 Introduction

Understanding the system security level against cyber threats is critical for today's IT or IT-enabled infrastructures, such as cloud storage and computing services, banking and payment systems, or cyber-physical systems such as smart grids. Industries today adopt various compliance standards (e.g., NERC CIP [17]) to exercise best practices in assessing their systems' level of security. Despite promoting security in general, such compliance practices often do not sufficiently capture the inherent relationships among different security-related aspects of the studied infrastructure. To provide deeper insight and more rigorous assessment into the overall security of the infrastructures, recent years have witnessed a surge of interest in model-based security assessment. In model-based security assessment, the various aspects of systems, threats, security measures, and more importantly, how these aspects interact with each other, are abstracted into models, often through rigorously defined formalisms. Such methodologies then use these underlying models to evaluate — often quantitatively via some forms of calculi — the security level of a system against specified cyber threats [29, 11, 24].

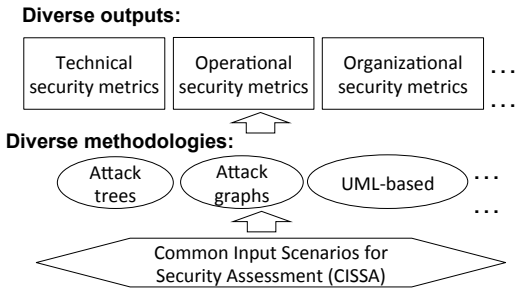

Fig. 1. Security assessment process driven by CISSA

Notable examples of model-based security assessment methodologies include the attack tree [25] and its enhancements (e.g., Boolean logic Driven Markov Process [23] and Attack-Defense Tree [10]), the attack graph [21, 15] and its embodiments (e.g., [26, 18]), Unified Modeling Language (UML)-based formalisms (e.g., Cyber Security Modeling Language [27]), and petri-net based formalisms (e.g., ADversary VIew Security Evaluation [14]). By applying techniques from this impressive range of methodologies, security practitioners could potentially assess the security of their systems in a systematic, rigorous, and holistic manner.

However, these methodologies manifest great diversity in selecting the information on system details and adversaries that are used as input, in representing and processing them (i.e., the formalisms, the calculi, the tool designs etc.), as well as in the aspects of security assessment (e.g., security metrics) they produce as output [29]. It is therefore difficult to understand their different strengths, compare their results, or integrate them in meaningful ways to present a multifaceted assessment of the systems. We have faced these challenges developing our own model-based security assessment methodology [5, 30], and we believe that industry practitioners seeking to adopt security modeling tools face similar dilemmas in vetting and selecting methodologies.

In this paper, we propose the creation of a set of *common input scenarios for security assessment* (CISSA). As illustrated in Figure 1, a rich and open repository of input scenarios would allow both the security practitioners and the research community to better compare different methodologies, which in turn will drive future research and development to address various real-world security assessment needs and challenges. Our work propose a specification framework for defining CISSA and provide six sample scenarios based on real-world attacks against various IT and IT-enabled systems. We use the samples to analyze the real-world security assessment needs and evaluate three representative model-based assessment tools accordingly. Ultimately, we hope that our initial efforts will encourage the community to build a rich repository of CISSA for many different systems. Such a repository can be referenced by both tool developers and security practitioners in discussing the needs and features of security assessment. In summary, we make the following contributions in this paper:

- We propose a first blueprint design (CISSA) for specifying common input scenarios for different model-based security assessment tools.
- To initiate the effort, we construct six CISSA cases based on real-world incidents, which are hosted on a public CISSA repository [1]. Such example cases show the

feasibility of building a diverse, realistic, structured, and precise set of CISSA based on the framework.
– We experimentally demonstrate the potential usefulness of CISSA by leveraging the six CISSA cases to investigate real-world needs of security assessment. We test three representative model-based security assessment tools and demonstrate that CISSA could provide new insights and concrete references to both security practitioners and tool developers.

In the rest of the paper, we propose our vision in Section 2, and then describe the elements of common input scenarios in Section 3. Section 4 reports our experiences in constructing six input scenarios, while a more detailed illustration of respresenting a real-world incident with the CISSA specification is presented in Appendix A. Section 5 uses the CISSA cases to investigate the security assessment needs and to test three representative security assessment tools. We discuss the roadmap for further developing CISSA in Section 6 and conclude in Section 7.

2 CISSA Vision

Devising effective means to assess the security of complex systems is one of the greatest challenges in security research [29, 24], but the reward for solving this problem is profound. The complexity and diversity inherent in today's enterprise IT systems and critical infrastructures make it important to leverage a model-based approach to answer various security assessment questions, e.g.: How to design more resilient systems? How to make better investment decisions for different defense mechanisms? To realize the potential of model-based security assessment methodologies, we believe it is necessary to establish widely-accepted open input scenarios. Such common input scenarios will allow researchers and industry stakeholders to better understand the data required for security modeling, as well as the trade-offs and blind spots inherent in picking a tool or formalism for their system.

Such common input scenarios need to include the inputs required by security assessment tools based on individual real-world incidents or synthesized attacks. For example, many tools require the network topology and system configurations, as well as attacker capabilities as inputs to their assessment. Other tools may require specific additional information as inputs, such as the details of security countermeasures, and the estimated response time when suspicious behaviors are detected. The common input scenarios should be as comprehensive as possible in covering the general classes of inputs required by major security assessment tools. We envision that:

– For security practitioners, a common set of realistic CISSA cases will help them to conduct meaningful comparison and integration among different methodologies. Since assessment methods are validated on common, openly available, and realistic inputs, CISSA can help reduce the barrier to adoption for industry.
– For researchers and tool developers, a realistic, rich, and heterogeneous set of CISSA will make it clear which aspects to model and the important questions to answer. This will help guide the further development of the methodologies to meet real needs.

– An open repository of CISSA cases will allow any interested parties to contribute new cases that will be available to all. Overall, it will benefit the entire community in tool development and security assessment with greater interactions, synergy, and standard practices, enabling bigger impact of model-based security assessment for real-world systems.

Of course, realizing this vision will require a concerted effort on the part of the security assessment community. By proposing a blueprint for designing CISSA cases, hosting an open CISSA repository with six input scenarios as initial examples, we demonstrate the potential feasibility and usefulness of CISSA, hence encouraging other security researchers and tool developers to share the input scenarios they use.

3 Elements of an Input Scenario

Conceptually, an input scenario collects together the representation of the important security-relevant information surrounding a system. For industry practitioners, the scenario representation should allow them to describe the system they are responsible for and to express their concerns and design choices in cybersecurity aspects. For academics, the scenario representation will provide a reasonable proxy for real systems, to enable research and development of security assessment tools and methodologies.

We make the following choices when designing an initial blueprint for the CISSA framework:

– *Methodology-independent*: We decouple the raw security-related inputs/facts from any specific assessment methodologies. This would ensure that the input scenarios could be generally applied to study different security methodologies.
– *Comprehensive*: We want CISSA to cover different types of information, as long as it is security-related, i.e., the inclusion of the information can potentially affect the security assessment results. For example, CISSA should not only include the details of the attack, but elaborate on the corresponding environment (e.g., certain network topology, or specific software configuration) where the attack can occur as well. Because CISSA is not bound to any specific methodology, it is easy to define and further extend the framework to cover additional aspects that arise from the different kinds of systems/scenarios.
– *Realistic*: To help bridge the gap between academia and industry security practitioners, we believe CISSA should be made as realistic as possible. One way to ensure a high level of realism is to require CISSA to be based on real-world systems and security incidents. We expect the development of a CISSA uses the best information one can gather from the field.
– *Precise*: The information included in CISSA should represent different aspects of information in a structured way, and strive for minimum ambiguity, so as to enable objective and fair comparison among the tools.

Note that there are implicit conflicts and trade-offs among the different goals we strive for, e.g., in order to make a case cover comprehensive information, one often

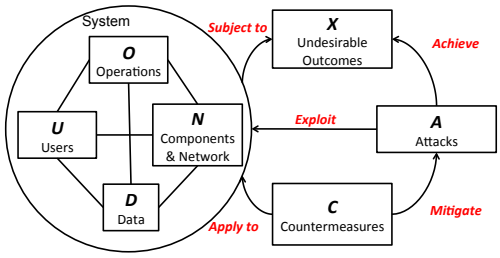

Fig. 2. Elements in an input scenario and their relationships

needs to include unstructured data; on the other hand, to ensure that the included information is grounded, realistic, and precise, one often needs to exclude hypothetical or unsubstantiated information, hence sacrificing the comprehensiveness.

Guided by these design considerations, we propose a schema for representing security-relevant information. While there may be other ways to represent such information, we believe that our schema (Figure 2) could serve as a valuable starting point for the CISSA concept. Existing resources such as security incident reports are largely unstructured, and it would require human comprehension and transformation before they can be used by security assessment tools. Databases such as NVD provide machine-readable format, but only include information on the specifics of the vulnerabilities, which is far from sufficient for most model-based security assessment tools. In the design of CISSA, the identified elements and their attributes provide a unified way to represent information on both the target system and the security incident.

As shown in Figure 2, a common input scenario for security assessment consists of seven core elements, which are interrelated: system *components and network*, *data*, *users*, and *operations*, as well as *undesirable outcomes*, *attacks*, and *countermeasures*. We represent these elements as a 7-tuple:

$$< N, D, U, O, X, A, C > .$$

We now elaborate on the characteristics of each element.

System Descriptions. The first four elements describe the system (in its general definition) to be assessed: N specifies the *components & network*, including both the specifications of the devices in the system (hardware / software, their configurations, and vulnerabilities), as well as the inter-connections and trust among them; D specifies the *data* that the system produces, stores, or consumes; U specifies the key *users* that interact with the system; and O specifies the *routine operations* that the system carries out. These four elements are tightly coupled: take the operation O as an example, it is defined in respect to N (e.g., which device is being controlled by a command, and which connection transmits the command), D (e.g., which sensor readings are used in an operation), and U (e.g., who takes charge of which step of the operation).

Undesirable Outcomes. Given the above system description, X specifies the *undesirable outcomes*, i.e., the security concerns, of the system. X is determined by the nature of the system and reflects its security requirements, including but not limited to the traditional Confidentiality, Integrity, and Availability triples. For example, for a rail-

CISSA Scenario	Unique Characteristics
Stuxnet	Multi-step, several zero-day exploits, broken "air-gap", command & control
Maroochy	Insider attack, poor auditing and access control policy, sabotage
Dragonfly	Targeted attack, watering hole, multi-step, trojanized software update, command & control
Target	Data breach, multi-step, integrated but insufficient security mechanisms, delayed incident response
SK Communications	Highly targeted attack, data breach, poor security policy, trojanized software update
Syrian Electronic Army	Targeted attack, multi-step, evading detection and defense

Table 1. Characteristics of Example CISSA Scenarios

way system, engine failure, train crash, and fare evasion are some of the undesirable outcomes that can be potentially caused by attacks.

Attack. The element A describes the attack aspect, including both the concrete attack steps taken by an attacker and the information about the attacker itself: $A :=< \alpha, \Sigma >$, where α denotes the concerned attacker, and Σ is a set of possible attack steps.

Countermeasures. There are different types of countermeasures to safeguard systems from potential attacks. These include system hardening mechanisms that raise the difficulty of launching successful attacks (e.g., timely patching), intrusion detection and prevention systems, as well as the resiliency mechanisms to limit the potential damage caused by invasions from attackers. We define countermeasures as a first-level element in CISSA to highlight their importance. In relation to the other elements in CISSA, a countermeasure is about what attacks it addresses and which parts of the system it hardens. When defining a concrete input scenario, we expect one to list the actual countermeasures that are in place.

4 Constructing initial CISSA cases

In this section we describe the six common input scenarios that we have defined using the proposed CISSA schema in Section 3. We challenge ourselves by selecting these scenarios from a diverse body of threats and application domains. We have put the specification files for all of our six initial scenarios online [1]. In Appendix A we illustrate the CISSA specification for one of the scenarios. We conclude this section by summarizing our experiences of constructing these initial CISSA cases.

4.1 A Diverse Set of CISSA Cases

We challenge ourselves to construct a diverse set of cases, with their characteristics summarized in Table 1. The following descriptions highlight for each of the six cases their unique aspects that our CISSA cases manage to incorporate.

Stuxnet. The Stuxnet attack [7, 12], widely publicized in 2010, is arguably the most well-known example of a cyber attack targeting systems comprising of both cyber and physical equipment. The Stuxnet attack targeted the centrifuge machinery in Iranian nuclear enrichment facilities. The attack first enters into the cyber systems via public Internet and escalates its privilege, just as any ordinary cyber attack would do. This step involves multiple zero-day vulnerabilities. It then attempts to bypass the "air-gap" that is supposed to physically segregate the control networks from the Internet, by infecting

USB storage devices. Once in the control network, the attacker intermittently changes the spinning speed of centrifuges, to cause lowered productivity and physical damage.

Maroochy. This scenario is based on the real-world incident that occurred in Maroochy Shire, Australia in 2000 [2]. In that attack, a disgruntled former employee stole a company laptop to send malicious radio commands to the sewage treatment system in Maroochy Shire. The attacker disguised his commands to appear as if coming from one of the pumping stations, causing pumps to malfunction; he also disabled alarms to conceal the attack. As result, sewage spilled at one of the pumping stations. The water treatment company noticed the "faults" and dispatched technicians to apply some countermeasures that were ineffective.

Dragonfly. The Dragonfly attack, publicized in 2014, was carried out by an advanced persistent threat actor [9]. The attack, which seems to target organizations from the energy and/or pharmaceutical sectors [28, 13] involves the installation of Remote Access Tools, ostensibly for the purpose of information theft. Dragonfly was carried out via three attack vectors: (i) email spear phishing, (ii) a watering-hole attack designed to compromise industrial control system vendors' systems, and (iii) trojanized software designed to spread from the compromised vendors to their customers' systems.

Target. The data breach at Target Corporation in 2013 resulted in the loss of credit card data from 40 million customers [8, 3]. This multi-stage attack began with the theft of an HVAC vendor's credentials for Target's vendor management web portal. From there, attackers were able to penetrate deep into Target's corporate network, steal personally identifiable information (PII) from a database, and deploy malware to pull customers' credit card information off of Target's point-of-sale machines. Stolen data were exfiltrated via FTP from inside the corporate network. We analyze this attack in greater detail as an example CISSA in Appendix A.

SK Communications. This scenario is based on a 2011 incident involving the theft of customer data from SK Communications: a South Korean internet service provider [6]. The attackers compromised a software vendor, and created a trojanized software update that installed a remote administration tool on more than 60 machines in the corporate network. The attackers leveraged this malware to exfiltrate personal information from over 35 million of SK Communications' customers.

Syrian Electronic Army. In 2013, a hacktivist group thought to have ties to the Syrian government initiated a number of phishing attacks against Western media organizations [16]. The attack follows two stages: (i) an initial phishing campaign from external email accounts to gain access to a user account in the target organization, and (ii) a second, internal, phishing campaign to obtain more desirable user accounts (e.g., with access to the company website's content management system). Once the attackers gain access to the company's website or social media account, they undertake web defacement or publication of material supporting their political agenda.

4.2 Our Experiences Constructing CISSA Cases

One lesson learned from the process of defining CISSA cases is the need for iteration. Since each of the real-world security incidents described above are unique, there was a

Tool	Category	Features
MulVAL [19]	Attack-graph-based	Integrating network & system vulnerability assessment
CySeMoL [27]	UML-based	Modeling various aspects of information system, with built-in knowledge base for quantitative evaluation
BDMP [23, 22]	Attack-tree-alike, with extra modeling power	Modeling dynamic behaviors by Markov-chain-enhanced attack trees

Table 2. Model-based security assessment tools used in experiments

need to progressively update the input specification. In fact, the *users* input U was added to the framework after identifying a gap in a previous ontology related to the modeling of human-centric attack vectors such as phishing. We believe future work and attention from the broader security community will further strengthen the CISSA framework and case descriptions.

A second critical issue is the level of detail to provide for each input class in the tuple. Real-world systems and cyber incidents are undoubtably complex, and model-based assessment tools often require different representations of the target system. We sought to address this by placing greater emphasis on *what should be specified* rather than *how it should be represented*. While there may be no "right" level of detail, we believe one of the primary benefits of CISSA is making clear the starting point for various model-based security assessment efforts.

Tool	Network	Data	User	Operations	Undesirable Outcomes	Attack	Countermeasure
MulVAL	OVAL/Nessus/Firewall scan reports	Nil	Built-in extensible library, e.g., hasAccount, etc.	Nil	Built-in extensible library, e.g., execCode, etc.	Nil	Nil
CySeMoL	Predefined extensible library, e.g., NetworkZone, NetworkInterface, etc.	Predefined extensible library, e.g., Dataflow, Datastore, etc.	Predefined extensible library, e.g., Person, Account, etc.	Technical, operational, organizational levels modeled in templates	Nil	Predefined attack steps	Predefined defense steps
BDMP	User-constructed attack tree	User-constructed attack tree	Nil	Nil	Root of attack tree	Tree leaves, e.g., AA, ISE, TSE	Enable/disable detection mode

Table 3. CISSA elements used as inputs to assessment tools

5 Putting CISSA into use

As summarized earlier in Table 1, our small yet diverse set of scenarios exhibit some unique characteristics, e.g., on the nature and design of the system, the tactics, techniques, and procedures of the attack, and the response from the victim. Based on these characteristics, we group the six scenarios into three categories corresponding to different security features that are typically assessed: *technical*, *operational*, and *organizational*. As we discuss below, CISSA provides a convenient and concrete context for security analysts to explore technical, operational, and organizational security aspects, and to understand which security assessment tools or methods may be most appropriate for each one.

Such a common context enables comparison of different security assessment tools in their evaluations from various aspects. For instance, from the technical aspect, CISSA allows security analysts using assessment tools to answer questions such as *how do the network topology and configuration of the assessed system affect its security standing, for a particular scenario?* (**Experiment I**). With a given CISSA scenario with details

in the systems, security analysts can tweak the network topology and configuration, and quickly re-evaluate the security of the resulted systems under the same settings. From the operational aspect, CISSA can assist analysts in answering questions like *how much would a better incident response procedure change the system's resilience against the given attack?* (**Experiment II**). For example, in the Target case, it took weeks before the victim confirmed the breach. If the response were more prompt, would the impact be lowered? Such hypothetical questions can be assessed by adopting different incident response options for the original CISSA case, and comparing the evaluation results from various assessment tools. In addition, CISSA can be useful in providing the context for evaluating questions like *how effective could a better security awareness program be at thwarting an attack in a given environment?* (**Experiment III**). Such analysis will be especially relevant to studying how cases such as Maroochy attack can be better prevented. Our experiments as detailed next focus on the three aspects, demonstrating how CISSA can be useful in comparing and informing different security assessment methodologies and tools.

5.1 Using CISSA to Compare and Inform Methodologies

Ultimately, the usefulness of CISSA to security practitioners and security assessment researchers depends on: 1) whether using CISSA provides insight on comparing and selecting existing security assessment tools; 2) whether using CISSA reveals aspects where existing assessment tools are doing well and/or lagging behind. To provide some initial insights into the usefulness of CISSA in the above aspects, we focus on three representative model-based security assessment methodologies with good tool support, among many others (e.g., ADTree, ADVISE, NETSPA). As summarized in Table 2, the three tools we choose, i.e., Multi-host, Multi-stage Vulnerability Analysis Language (*MulVAL*) [19], Cyber Security Modeling Language (*CySeMoL*) [27], and Boolean logic Driven Markov Process (*BDMP*) [23, 22][4], use different formalisms to incorporate different security-related aspects, and subsequently (as we will show later), take in significantly different information and format as inputs. Table 3 describes how each tool takes in specific CISSA elements as input.

Feeding CISSA into the tools. When conducting our experiments, we apply a *best-effort* approach, by 1) applying CISSA scenarios to assessment tools as much and as directly as possible, and 2) exploring the available features of the tools as much as needed—as we will show, this often involves the use of some advanced features in the tools.

One practical benefit of having CISSA is to encourage different security assessment tools to unify their input formats. This would help security practitioners who need to work with different tools. Currently, our XML-based CISSA scenarios [1] cannot be directly used by the three tools in an automatic way. Instead, we manually convert the XML files into the appropriate input formats for each tool. From this exercise, we see that it should be feasible to automatically translate CISSA scenarios for use in some

[4] In this paper, we refer to the particular tool available at `http://researchers.edf.com/software/kb3-44337.html`.

tools. For example, MulVAL has built-in parsers to collect input from certain vulnerability scanners, and CySeMoL supports XML import/export[5] However, it is less clear how to automate the procedure to convert CISSA scenarios for use in tools like BDMP, which is designed for manual use. One potential direction is to construct a basic attack tree as a starting point for BDMP by chaining together different attack steps using their pre-conditions and post-conditions.

While we strive to be comprehensive in CISSA, all three tools need extra information (beyond CISSA inputs) to conduct their assessment. For example:

- BDMP requires the probability distributions for launching an attack step. Since we cannot readily obtain such information from real-world measurements and reports, it is not included in CISSA.
- MulVAL computes the risk associated with different vulnerabilities by obtaining the corresponding vulnerability descriptions from a database.
- CySeMoL relies on built-in logical structures and probability relationships among different security-related concepts, in order to cater for non-expert users. While CySeMoL includes a meta-class editor to change these default settings, CISSA does not provide enough information to allow us to do so.

In the future, we believe that CISSA will evolve to provide more complete coverage for different tools. However, we foresee that there will always be certain tool-specific information that is not suitable for inclusion in CISSA. We believe that each security assessment should explicitly state the input information that was considered, and that any delta with the CISSA information should be made explicit.

In the remainder of this section, we use the above inputs (both inside and outside of CISSA) to conduct several tool-based experiments attempting to address three typical questions in the technical, operational, and organizational aspects. The purpose of these experiments is to show how CISSA can provide a realistic context for analyzing the processes used by the tools to gather inputs and produce these outcomes. It is not, however, intended to be a commentary on the soundness of the tools nor the accuracy of the outcomes.

Experiment I (technical aspect). Here we challenge the tools to answer this question: how does the network topology and configuration of the assessed system affect its security standing for a particular scenario?

First, we use MulVAL to model the Stuxnet scenario, which is characterized by a sophisticated multi-step attack and the utilization of several vulnerability exploits as attack vectors. By design, MulVAL reports unpatched vulnerabilities in a network by leveraging various sources, such as OVAL, the Nessus scanner, NVD, etc. It integrates the discovery of such vulnerabilities into the assessment. To model the Stuxnet scenario, we find that we need to use some advanced features of MulVAL, e.g., creating new inference rules for generating attack graphs. For example, the default MulVAL engine cannot model USB-based penetration of the "air-gap". Hence, we add a few new rules, including:

```
interaction_rule(
 (execCode(Host, Perm) :-
```

[5] The PRM-based version in [27] provides XML import/export. However, we cannot find the format and semantics of the files, nor available tools for generating/processing them.

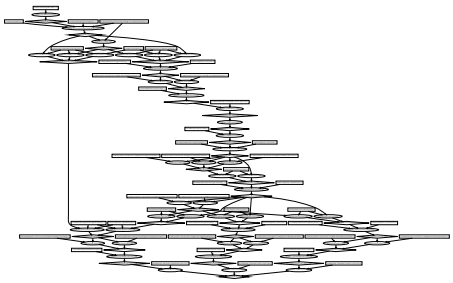

Fig. 3. Stuxnet: with Web Navigator server

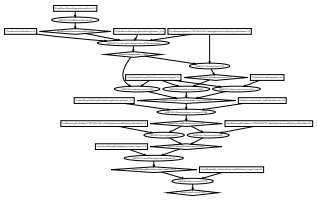

Fig. 4. Stuxnet: no Web Navigator server. Fig 3 and 4 are presented to illustrate the different number of attack paths for the two settings, and not meant for showing the details.

Tool	10 Days' Attack	30 Days' Attack	180 Days' Attack
CySeMoL	.38	.43	.45
BDMP	.05	.34	.60

Table 4. Results on attack success probability for Stuxnet scenario from CySeMoL and BDMP

```
vulExists(Host,_,Software,_,privEscalation),
localService(Host,Software,Perm),
usbMounted(Host,USB_Drive),
malwareLocated(USB_Drive)),
rule_desc('Exploit via Infected USB ',1.0)).
```

With these changes and the CISSA inputs, MulVAL generates a high-quality attack graph that closely resembles the attack graph used in an in-depth Stuxnet analysis report published by Tofino Security [4][6]. In the generated graph, attackers have more than 210 different possible attack paths to reach the process control network, by combinatorially exploiting seven different attack vectors over multiple network segments. MulVAL produces an attack success probability of 0.861 for this scenario.

To assess how network topology affects the system's security, we remove the vulnerable Web Navigator server in the topology to implement a "true air-gap", i.e., physical segregation between enterprise and control network, instead of firewall-isolated one. With this change, there are fewer than five attack paths still available to attackers, and the attack success probability produced by MulVAL decreases to 0.539 accordingly. We present the generated attack graphs for comparison in Figures 3 and 4, respectively. We also experiment with two alternative configuration setups: one with all USB drives prohibited, and the other with remote employee access disabled. It turns out that security enhancement gained from these is less obvious than a true air-gap. MulVAL produces 0.825 and 0.667 attack success probabilities, respectively, for the two alternatives. This experiment shows how a tool like MulVAL can use information in CISSA to provide a concrete answer for the question posed earlier.

We also test BDMP and CySeMoL with the same Stuxnet scenario. For BDMP, unlike MulVAL which can automatically generate the attack graph, we have to manually construct the model. This is more time consuming, especially when we vary the network topologies and configuration setup of the system, such as removing the Web

[6] The main difference is that the hypothesis in [4] about Contractor Remote Access attack vector is not included in our CISSA case.

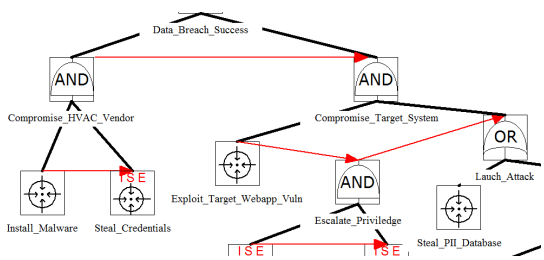

Fig. 5. Target data breach modeling with BDMP (partial)

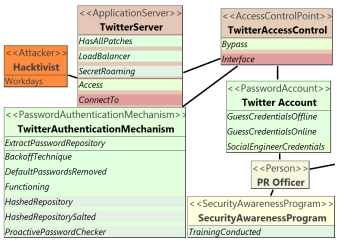

Fig. 6. SEA modeling with CySeMoL (partial)

Navigator server to answer some "what if" questions. With CySeMoL, we also need to manually construct the network topologies for the assessed system. However, once it is constructed, it is fairly straightforward to alter it and re-run the assessment.

Differed inputs, differed outputs: A key observation is that although these tools use similar metrics to represent the assessment result, i.e., the attack success probability, they produce vastly different values for the same system assessed. Part of the reason could be the presence of additional, tool-specific inputs, as discussed previously.

We now compare the attack success probabilities produced by the tools when we model the same Stuxnet scenario. Since both CySeMoL (the latest version of v2.3)[7] and BDMP require the specification of attack duration in order to compute the metric, while MulVAL does not have the notion of time, we can only compare the results produced by CySeMoL and BDMP meaningfully. Table 4 shows the attack success probabilities produced by CySeMoL and BDMP for the Stuxnet scenario under varying attack duration. As shown in the table, the results differ between tools for the same attack duration, and the BDMP tool exhibits much higher sensitivity to the attack duration. For security practitioners to better interpret differences like these, it is necessary to clearly specify the extra inputs used to conduct a security assessment.

Experiment II (operational aspect). Here we challenge the tools to answer this question: how much would a better incident response procedure change the system's resilience against the given attack?

We have varying degrees of success in conducting operational-level security assessment using the three tools. MulVAL does not have a built-in notion of time, nor about reactive defense procedures. Although we were able to introduce some customized rules for MulVAL in our last experiment, because the notions of time and reactive countermeasures are so fundamental, it is unclear how to enhance MulVAL to model these aspects. For CySeMoL, we can *indirectly* model the impact of reactive countermeasures by varying the attack duration parameter. We observe that the final risk value computed for the Target case is highly influenced by this parameter, e.g., the attack success probability can be reduced from 58% to 14% if the countermeasure can reduce the available time for the attacker from 1 week to 1 day.

In comparison, we find that some advanced features offered by BDMP [22] can be very useful in answering this question. In particular, BDMP allows a user to express the detection effectiveness for each individual attack step directly at four different stages:

[7] Earlier PRM-based versions of CySeMoL assume constant attack duration.

initial, ongoing, final, and a-posteriori. For the initial (when an attack step gets started) and final (when an attack step completes) steps, the user can specify a probability for the attack to be detected. For the ongoing (when an attack step is being carried out) and a-posteriori (when an attack step has been completed so the attacker can proceed to the next step if any), the user can specify the mean-time-to-detection assuming the time follows an exponential distribution.

We test these features by using BDMP to model the Target scenario (partially shown in Figure 5). To do so, we need to provide BDMP with the required detection parameters. As argued earlier, getting ground-truth values for these parameters is difficult in the real world, and thus they are not included in CISSA. In our experiment, we compare the outputs of BDMP with an average ongoing / a-posteriori mean-time-to-detection of 1 day and 1 week respectively. The results show that the expected time for the attacker to stay in a state with access to the credit card information, without being detected, increases by around 7 times under the two settings. This modeling capability can be very useful for security practitioners to understand and promote the importance of rapid incident response, if the required estimation for time-to-detection can be obtained with reasonable accuracy.

While BDMP provides direct and detailed modeling for conducting these studies for the Target scenario, we find that as of now the tool can only deal with a one-time attack. This is insufficient to model repeated malicious attempts, such as in the Maroochy case. One possible enhancement to BDMP would be to integrate more complicated Markov models where a defeated attacker can adapt and re-launch the attack.

Experiment III (organizational aspect). Here we challenge the tools to answer this question: how effective could a better security awareness program be at thwarting the attack in the given environment?

We find that organizational aspects are not well modeled by the studied tools: only CySeMoL provides some simple modeling of a security awareness program. We thus created a model (partially shown in Figure 6) in CySeMoL for the Syrian Electronic Army (SEA) scenario, exploring how the security awareness program would affect the security assessment. Specifically, for a model without the awareness program, CySeMoL produces a 28% probability of success for the attacker to access the victim's Twitter account. After we enable the security awareness program for the targeted organization, the risk reduces to 20%. On the other hand, a technical security mechanism like authentication protection can reduce the risk from the original 28% to 17%. CySeMoL allows the combined application of both organizational and technical security controls into the same model. For example, with both security controls applied, the risk is reduced to 7%.

Although it is interesting to see the modeling of organizational security considerations together with other aspects in the same framework, we believe more organizational security details can be included: for example, the existence (or absence) of a policy for isolating different accounts (as in the SEA case), the effectiveness of the auditing procedures for employees who leave a company (as in the Maroochy case), or the trust policy for third-party vendors (as in the SK Communications case).

6 CISSA: Benefits, Limitations, and Roadmap

Our experiments using example scenarios demonstrate how CISSA provides a concrete context for analysis and identification of strengths and weaknesses of security assessment tools. The pros and cons we discussed and the quantitative values generated by the tools are provided only for the purposes of illustrating potential benefits of using CISSA. Although the results could be further refined, the process of using the different tools to answer specific security questions was illuminating, and we believe that CISSA can play an important role in the continued development of this research area.

6.1 Using CISSA to Advance the State-of-the-Art

From our evaluations we see three areas for improving model-based security assessment methodologies.

Clarifying inputs. As we mention in Experiment I, differing inputs and input formats between different assessment tools has been a bottleneck, complicating meaningful comparison of tools or corroboration of results. Our sample CISSA scenarios have provided some initial but meaningful analysis on what different assessment tools seem to agree on, and where they differ in terms of inputs. When security practitioners need to conduct an assessment, they can have a better understanding about which tool or tools would be most suitable by looking at how they model similar scenarios.

Integrating tools. Our experiments above demonstrate that different tools are designed to focus on specific aspects of security assessment. Enhancing each one individually to incorporate richer inputs is possible, albeit at the risk of making them more complicated and hence reducing their usability. A better alternative is to continue using different specialized tools to answer different questions, but to find certain ways to integrate them together, e.g., the way MulVAL integrates OVAL/Nessus scanning results. On this front, common inputs are just the first step towards interoperability between tools. Common interfaces, consistent outputs, and methodologies or guidelines in partitioning assessment tasks also need to be established after inputs are agreed upon by different tools.

Modeling security beyond the technical level. All three tools we experiment with provide detailed modeling in the lower technical aspects. However, our success decreases as we move up to the operational and organizational level. Cross-reference between different levels is also preliminary in all three tools. By designing CISSA to contain information at all three aspects (and in a more balanced manner), CISSA can help promote a more holistic treatment of these different levels.

6.2 Present Limitations

The main limitation of our current CISSA definition is that it does not guarantee on the completeness of information. However, the idea is to populate the CISSA repository with such information as commonly required by popular security assessment tools. CISSA can be readily extended to include additional information about systems being assessed or the details about the threats as necessary. However, CISSA at least makes

input to security assessment explicit, which can significantly avoid possible misunderstandings and misassumptions on how systems are assessed.

Another potential limitation is that the conversion from CISSA cases to tool-specific inputs is non-trivial. As a one-time effort, once a particular CISSA case has been converted, it can be readily reused for similar scenarios by minor edits. We plan to develop tools to help automate such processes. Our hope is the usefulness of CISSA demonstrated in practice will motivate gradual and eventual converge in the format of inputs to different security assessment tools.

A third limitation could lie in the process of security analysts selecting most relevant CISSA cases for the systems under assessment, when the CISSA repository grows much richer. In fact, we would encourage a worst-case analysis by considering all attack scenarios. Ruling out security vulnerabilities is by itself very challenging.

6.3 The Future of CISSA

Pinpointing areas where model-based security assessment can improve is one of the key roles we believe CISSA can play. At the same time, in better shaping security assessment methodologies we expect CISSA itself to evolve.

Our initial experiences in constructing sample CISSA scenarios indicate that CISSA is more of an evolving effort than a static collection. The seven-element CISSA framework we propose is also to be iteratively refined with additional effort on constructing additional sample CISSA scenarios, e.g., in terms of whether to include / exclude some information, how to define the delta accurately, and how to enable automatic translation. We understand this has to be a community-wide effort, and we plan to incorporate CISSA modeling and CISSA-enabled comparison with other tools in the development of our own security assessment software [30]. Furthermore, we plan to organize workshops to provide the forum for developers of security assessment tools to discuss how to make CISSA more useful. We hope other security researchers will share our vision, so we can work together to continue building an ever-richer CISSA repository.

We believe this type of sustained community effort will be the first step in making model-based security assessment more scientific and valuable. Establishing a commonly accepted input repository for security assessment will form a basis for benchmarking security assessment tools. We have seen benchmarks in many other areas, such as the JavaScript engine, GPU rendering, routing protocols, etc. Unfortunately, we are not yet close to having common benchmarks to gauge model-based security assessment tools, e.g., for evaluating their breadth in taking all relevant information into account, their sensitivity to network topological changes, their run-time performance to assess a particular scenario, etc. Security researchers first need to largely agree on the inputs, before they can work out how the outputs from security assessment tools can be interpreted as benchmarking results.

7 Conclusion

This paper presents an open repository of common input scenarios for security assessment (CISSA). By constructing six sample input scenarios and experimenting with three

assessment tools, we show the potential usefulness of CISSA for security practitioners and for the future development of security assessment tools. In particular, well specified common inputs could facilitate the comparison and integration of different assessment tools. We hope our work will promote a concerted community effort to build a richer repository of CISSA for supporting model-based security assessment studies.

Acknowledgements

This study is supported by the research grant for the Human-Centered Cyber-physical Systems Programme at the Advanced Digital Sciences Center from Singapore's Agency for Science, Technology and Research (A*STAR).

References

1. Public Repository for CISSA. http://www.illinois.adsc.com.sg/cissa.
2. Marshall Abrams and Joe Weiss. Malicious control system cyber security attack case study - Maroochy water services, Australia, 2008.
3. Aorato Labs. The untold story of the target attack step by step. http://www.aorato.com/blog/untold-story-target-attack-step-step/, Aug 2014.
4. Eric Byres, Andrew Ginter, and Joel Langill. How stuxnet spreads – a study of infection paths in best practice systems. www.tofinosecurity.com/how-stuxnet-spreads.
5. Binbin Chen, Zbigniew Kalbarczyk, David M. Nicol, William H. Sanders, Rui Tan, William G. Temple, Nils Ole Tippenhauer, An Hoa Vu, and David K.Y. Yau. Go with the flow: Toward workflow-oriented security assessment. In *NSPW*, 2013.
6. Command Five Pty Ltd. Sk hack by an advanced persistent threat. http://www.commandfive.com/papers/C5_APT_SKHack.pdf, Sep 2011.
7. Nicolas Falliere, Liam O Murchu, and Eric Chien. Symantec security response: W32.stuxnet dossier. www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf.
8. iSightPartners. Kaptoxa point of sale compromise. http://www.securitycurrent.com/resources/files/KAPTOXA-Point-of-Sale-Compromise.pdf, Jan 2014.
9. Kaspersky Lab Global Research and Analysis Team. Energetic bear – crouching yeti. http://securelist.com/files/2014/07/EB-YetiJuly2014-Public.pdf, Jul 2014.
10. Barbara Kordy, Sjouke Mauw, Saša Radomirović, and Patrick Schweitzer. Foundations of attack-defense trees. In *FAST*, pages 80–95, 2011.
11. Barbara Kordy, Ludovic Pietre-Cambacedes, and Patrick Schweitzer. DAG-based attack and defense modeling: Don't miss the forest for the attack trees. *CoRR*, abs/1303.7397, 2013.
12. Siwar Kriaa, Marc Bouissou, and Ludovic Pietre-Cambacedes. Modeling the stuxnet attack with BDMP: Towards more formal risk assessments. In *Proc. of the International Conference on Risk and Security of Internet and Systems (CRiSIS)*, pages 1–8, Oct 2012.
13. Joel Langill. Defending against the dragonfly cyber security attacks. http://www.belden.com/docs/upload/Belden-White-Paper-Dragonfly-Cyber-Security-Attacks.pdf, 2014.
14. E. LeMay, M. Ford, K. Keefe, William H. Sanders, and C. Muehrke. Model-based security metrics using ADversary VIew Security Evaluation (ADVISE). In *QEST*, 2011.
15. R. P. Lippmann and K. W. Ingols. An annotated review of past papers on attack graphs, 2005.

16. Mandiant, a FireEye Company. Beyond the breach. https://dl.mandiant.com/EE/library/WP_M-Trends2014_140409.pdf, 2014.
17. North American Electric Reliability Corporation. Critical infrastructure protection standards. http://www.nerc.com/pa/Stand/Pages/CIPStandards.aspx.
18. Xinming Ou and Wayne F. Boyer. A scalable approach to attack graph generation. In *CCS*, 2006.
19. Xinming Ou, Sudhakar Govindavajhala, and Andrew W. Appel. Mulval: A logic-based network security analyzer. In *USENIX Security*, 2005.
20. PCI Security Standards Council. PCI SCC data security standards overview. https://www.pcisecuritystandards.org/security_standards/.
21. C. Phillips and L. Swiler. A graph-based system for network-vulnerability analysis. In *NSPW*, 1998.
22. L. Pietre-Cambacedes and M. Bouissou. Attack and defense modeling with bdmp. In *Proc. of the Mathematical Methods, Models and Architectures for Computer Network Security*, 2010.
23. L. Pietre-Cambacedes and M. Bouissou. Beyond attack trees: dynamic security modeling with boolean logic driven markov processes (BDMP). In *EDCC*, 2010.
24. William Sanders. Quantitative security metrics: Unattainable holy grail or a vital breakthrough within our reach? *IEEE-SPM*, April 2014.
25. B. Schneier. Attack trees: Modeling security threats. *Dr. Dobb's Journal*, 1999.
26. O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. Wing. Automated generation and analysis of attack graphs. In *IEEE S&P*, 2002.
27. T. Sommestad, M. Ekstedt, and H. Holm. The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures. *Systems Journal, IEEE*, 7(3):363–373, Sept 2013.
28. Symantec Security Response. Dragonfly: Cyberespionage attacks against energy suppliers. http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/Dragonfly_Threat_Against_Western_Energy_Suppliers.pdf, Jul 2014.
29. V. Verendel. Quantified security is a weak hypothesis. In *NSPW*, 2009.
30. An Hoa Vu, Nils Ole Tippenhauer, Binbin Chen, David M. Nicol, and Zbigniew Kalbarczyk. CyberSAGE: A tool for automatic security assessment of cyber-physical systems. In *QEST*, 2014.

A Appendix: CISSA Example

In this section we provide an example CISSA by describing the seven elements (N, D, U, O, X, A, C) for the Target Corporation data breach. The input files themselves are available online in XML format [1].

A.1 Brief Recap of the Target Incident

The data breach at Target Corporation in 2013 resulted in the loss of credit card data from 40 million customers [8, 3]. This multi-stage attack began with the theft of an HVAC vendor's credentials for Target's vendor management web portal. From there, attackers were able to penetrate deep into Target's corporate network, steal personally identifiable information (PII) from a database, and deploy malware to pull customers' credit card information off of Target's point-of-sale machines. Stolen data were exfiltrated via FTP from inside the corporate network.

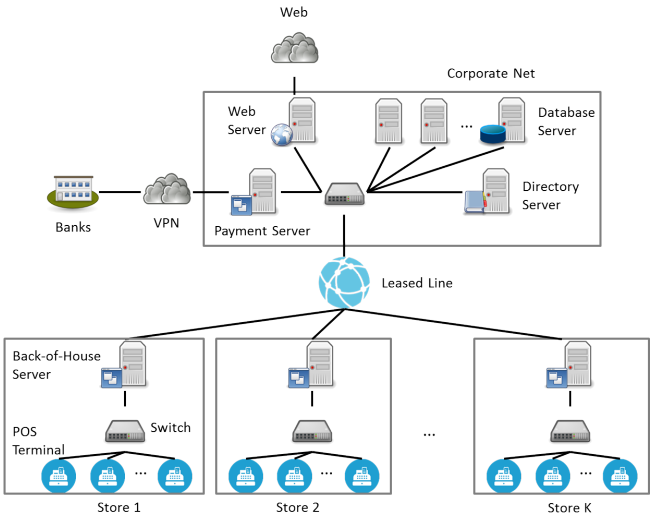

Fig. 7. Components & network for Target input scenario

A.2 CISSA Definition for the Target Scenario

To illustrate how CISSA specification can represent the details of read-world incidents for security assessment, we briefly present how we define the Target scenario under CISSA.

Components & network N. A large retailer such as Target typically has 100s or 1000s of locations, each with numerous point-of-sale (POS) stations that accept and process customer payments. Figure 7 depicts the network connections (E_N) and devices (V_N) considered in this scenario. The model includes K store locations, each with T_K POS machines which are connected via a switch to a back-of-house (BoH) server. This BoH server connects to a central payment server at the corporate network, which interfaces with external financial institutions to verify transactions. The corporate network also includes a directory server, a web server, and a database server. We abstract other corporate services from this model. In the XML files online [1] we specify additional information.

Data D. Data plays a central role in the Target CISSA. The attackers' goal was the theft of customer data, and this was made possible by the acquisition of additional system-specific data. Table 5 describes the data items that are modeled in this scenario. We provide a unique identifier for each data item ID_D, a description of the data's properties L_D, and a list of devices that interact with the data Map_D (network links are omitted for brevity).

Users U. In this scenario, several different user types are affected by the incident. In particular, credentials from a *vendor* and, later, a *Domain Administrator*, enabled the adversaries to steal sensitive information relating to the company's *customers*. Table 6 summarizes the relevant user information.

Operations O. Arguably the most important system operation in this scenario is the handling of consumer credit card information during POS transactions. Figure 8 depicts

Identifier (ID_D)	Description (L_D)	Mapping (Map_D)
$D1$	Credit card number	POS terminal, BoH server, Payment server, Bank
$D2$	Customer PII (e.g., name, address)	POS terminal, BoH server, Payment server, Bank, Database Server
$D3$	Admin access token	Servers in corporate network
$D4$	Active Directory listing	Directory server

Table 5. Data items in the Target input scenario

Identifier (ID_U)	Description (L_U)	Access (Map_U)
$U1$	Contractor with vendor web portal account	Web server
$U2$	Domain Administrator for corporate network	All devices/links in N
$U3$	Customer in a store	POS terminal

Table 6. Users in the Target input scenario

this process. Here the vertices V_O denote the major operations from the POS terminal, a store's BoH Server, and the Bank responsible for clearing the transaction, while the edges E_O imply sequential order. The mapping function Map_O in this case assigns specific devices to the roles described above (e.g., POS Terminal 5 in Store #300). Additional system operations in this scenario could include the POS Terminal or BoH Server's software update process, or the processes for collecting and storing personally identifiable information (PII) in the company's database.

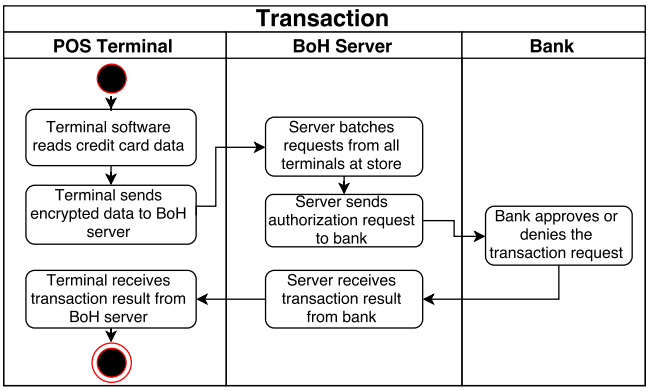

Fig. 8. Transaction operations for Target input scenario

Undesirable outcomes X. In this scenario we model the final undesirable outcomes of the attack as *loss of credit card data* and *loss of personally identifiable information*, as specified in Table 7. Due to space limit, we do not elaborate on intermediate undesirable outcomes x_1 to x_7 for each attack step, while providing brief summary in Table 8.

Attack A. The attacker input α is modeled with:

- **Goal:** theft of credit card data $X1$.
- **Access:** external attacker, with access to "Web" in N.
- **Skills:** use of existing tools and malware (no zero-days).

Identifier (ID_X)	**Description** (L_X)	**Mapping** (Map_X)	**Implications** (Imp_X)
$X1$	Loss of credit card data	POS machine	Fraud, legal liability
$X2$	Loss of PII	Database server	Legal liability, loss of goodwill

Table 7. Undesirable outcomes in the Target scenario

Attack Step (L_σ)	**Pre-Condition** (Pre_σ)	**Post-Condition** ($Post_\sigma$)
1. Steal credentials	<(Vendor's network access), (Server vulnerability exploiting techniques)>	<(Credentials of Target's systems), (), x_1 (Credential leak)>
2. Expl. web server	<(Credentials of Target's systems), (Server vulnerability exploiting techniques)>	<(Privilege to execute OS commands), (), x_2 (Privilege leak)>
3. Steal token	<(Access to Target's servers), (Know-how of collecting NT hashes from memory)>	<(Corporate network admin privilege), (), x_3 (Privilege escalation)>
4. Create account	<(Admin privilege to add new user to Domain), ()>	<(Access to corporate network), (), x_4 (Malicious admin account)>
5. Steal PII	<(Access to corporate network), (Skill to use database server)>	<(Access to customer records), (), x_5 (Unauthorized access)>
6. Install malware	<(Access to POS machines' writable folders), (Malware infection capabilities)>,	<(Access to data on POS), (), x_6 (Malware infection)>
7. Aggregate data	<(Access to FTP servers in corporate network, access to sensitive data), (Basic file transfer techniques)>	<(), (), x_7 (Sensitive data aggregated>
8. Exfiltrate data	<(Access to outward-facing internet connection, access to sensitive data), (Skills to stealthily exfiltrate files)>	<(), (), $X1 \cup X2$ (Data leak)>

Table 8. Attack steps for the Target input scenario

The attack on Target's corporate network and POS system is thought to consist of 11 steps [3]. We model a simplified 8-step attack, i.e., σ_1: Steal credentials of vendor, σ_2: Exploit vulnerability on Target web portal, σ_3: Steal Domain Admin access token, σ_4: Create new Domain Admin account, σ_5: Steal PII from database, σ_6: Install malware on POS machines, σ_7: Aggregate stolen data in network, and σ_8: Exfiltrate data via FTP.

The above text constitutes the attack step description input (L_ω). In Table 8 we specify the pre-conditions (Pre_σ) and post-conditions ($Post_\sigma$) for these attack steps.

Identifier (ID_C)	**Description** (L_C)	**Mapping** (Map_C)
$C1$	Multi-factor authentication	Users $U1$, $U2$
$C2$	Application whitelisting	POS terminal
$C3$	Real-time monitoring	Directory server, Web server

Table 9. Countermeasures in the Target input scenario

Countermeasures C. The credit card industry maintains a set of standards for data protection [20]. In addition to those guidelines—which were followed in this scenario—other countermeasures can potentially detect or prevent similar attacks.

- **Multi-factor authentication** for the outward-facing vendor portal, and for the Domain Administrators.
- **Application whitelisting** for the point-of-sale machines and the servers involved in transaction verification.
- **Real-time monitoring** of user lists and network queries to detect the addition of new user accounts (particularly admin accounts) and potentially identify lateral movement of an attacker within the network.