Automated Cyber Threat Sensing and Responding: Integrating Threat Intelligence into Security-Policy-Controlled Systems

Peter Amthor  
Technische Universität Ilmenau  
Ilmenau, Germany  
peter.amthor@tu-ilmenau.de

Winfried E. Kühnhauser  
Technische Universität Ilmenau  
Ilmenau, Germany  
winfried.kuehnhauser@tu-ilmenau.de

Daniel Fischer  
Technische Universität Ilmenau  
Ilmenau, Germany  
daniel.fischer@tu-ilmenau.de

Dirk Stelzer  
Technische Universität Ilmenau  
Ilmenau, Germany  
dirk.stelzer@tu-ilmenau.de

ABSTRACT

Cyber security management requires fast and cost efficient responses to threat alerts. Automation of cyber threat sensing and responding is one way to achieve immediate reactions to imminent threats. There are already tools for an extensive automation of threat sensing, e.g. threat intelligence sharing platforms. Methods, techniques and tools for reacting to menacing states and events, e.g. security-policy-controlled systems, have also been explored and published for some time. What is still missing, however, is the integration of these two approaches. This paper describes first steps towards an integration of threat intelligence sharing platforms and security-policy-controlled systems. We present a conceptual design for threat reaction strategies, security architectures and mechanisms and information representation requirements. We use two exemplary threat scenarios to demonstrate our proposals.

CCS CONCEPTS

• Security and privacy → Formal security models: Intrusion/anomaly detection and malware mitigation; • Applied computing → Event-driven architectures;

KEYWORDS

Threat Intelligence Sharing Platforms, Security Policies, Security Automation, Cyber Threat Sensing and Responding, Conceptual Integration Design

1 INTRODUCTION

The current cyber security situation is characterized by an increasing number of threats and security incidents and a growing sophistication of cyber-attacks. At the same time, the amount of security-related information is continuously increasing. Many threats require an immediate response. However, security officers in most institutions, particularly in SMEs, are confronted with a lack of resources. Even in companies with enough sufficiently trained and experienced security specialists and adequate financial and IT resources available, a purely manual reaction to warnings is often not fast enough. This requires more automated reception, analysis, evaluation and, above all, reaction to threat alerts [12, 29, 37].

Cyber security management comprises protecting information, data and other IT related assets from attacks whose origins lie outside the institution to be protected [54]. This covers both sensing of and responding to threats to cyber security [8]. The phrase sensing and responding has been used in control theory and other disciplines, e.g. biology and management, to denote the pattern of perceiving relevant information and reacting accordingly [22]. We use these terms to describe the interaction between cyber threat intelligence and other elements of cyber security management, e.g. security incident management. ISO 27035 [26] lists key activities of incident management: prevent, protect, detect, analyze, respond and resolve. This list illustrates that the output of threat intelligence sharing must be integrated into other areas of cyber security management in order to be fully effective. This paper aims at a greater automation of sensing and responding to cyber threats. This should help to speed up reacting to impending risks, to use existing resources more efficiently and to improve accuracy and appropriateness of responses taken when security alerts have been received. The objective of the paper is to make a contribution to the automation of cyber security management. We do this using the example of integrating threat intelligence sharing platforms (TISPs) and security-policy-controlled systems (SPCSs).

In this paper, we describe first steps towards an integration of TISPs and SPCs. Following the guidelines of [39] on structuring design tasks, we describe a conceptual design for integrating the two approaches and demonstrate our concept using two exemplary threat scenarios.

In the next section, we give a brief overview of threat intelligence, threat intelligence sharing and corresponding platforms which are
tools to support cyber threat sensing. Section 3 discusses security policies and security-policy-controlled systems that support responding to cyber threats automatically. The subsequent section 4 deals with the question of how TISPs and SPCPs can be integrated. We distinguish two different types of integration, direct and indirect, and we then focus on a conceptual design for direct integration. We discuss threat reaction strategies, security architectures and mechanisms and information representation requirements. We use two exemplary threat scenarios to demonstrate our proposals. In the last two sections, we give an outlook on our future research and conclude with a short summary.

2 THREAT INTELLIGENCE SHARING PLATFORMS FOR BETTER THREAT SENSING

In this section, we provide a brief overview of research into threat intelligence sharing and threat intelligence sharing platforms. We explain the contribution of threat intelligence sharing platforms to faster and better threat sensing and discuss why such platforms need to be integrated as intensively as possible into systems that can respond to cyber threats.

2.1 Threat Intelligence

A widely accepted definition of threat intelligence (TI) is still missing. However, several authors have proposed definitions [1, 24]. Shackleford [48] describes TI as a “set of data collected, evaluated, and applied regarding security threats, threat actors, exploits, malware, vulnerabilities, and compromise indicators.” Lee [31] denotes TI as “process and product resulting from the interpretation of raw data into information that meets a requirement as it relates to the adversaries that have the intent, opportunity and capability to do harm.” Friedman and Bouchard [20] refer to TI as “knowledge about adversaries and their motivations, intentions, and methods.” Tounsi and Rais [52] emphasize that it is essential to prevent attacks or shorten the time window between an actual compromise and its detection. McMillan [33] describes TI as “evidence-based knowledge, including context, mechanisms, indicators, implications and actionable advice, about an existing or emerging menace or hazard to assets that can be used to inform decisions regarding the subject’s response to that menace or hazard.” We define TI as evidence-based knowledge, information or data about existing or emerging threats that can be used to prevent or mitigate them.

For a detailed description of TI, Chismon and Ruks distinguish four subtypes [10, 52]:

- **Strategic TI** is high-level information consumed by senior decision-makers or at board level in an organization. It describes business (or more specifically: financial) impacts of threats, attack trends and areas that might impact high-level business decisions. Strategic TI is intended to enable strategists understand current risks and identify future risks. Strategic TI is almost always provided in the form of textual reports, briefings or conversations, mostly without technical details.
- **Operational TI** is more actionable information relating to specific attacks against the organization. It provides information on the nature of attacks and attackers’ identity and capabilities. It often indicates when the attacks might take place. Operational TI is used by higher-level security staff (security managers or heads of incident response).
- **Tactical TI** is information about how threat actors are conducting attacks including their tools and methodologies. It is often referred to as Tactics, Techniques and Procedures (TTPs). Tactical TI is consumed by defenders and incident responders to ensure that their sensing, alerting and defenses are prepared for current tactics of aggressors.
- **Technical TI** comprises technical details of an attacker’s assets. It is different from tactical TI as it focuses on specific Indicators of Compromise (IoC). Typical examples of such indicators are virus signatures, MD5 hashes of malware files or URLs of botnet command and control servers. Technical TI is regularly consumed by security operations center staff and/or automated systems, e.g., firewalls or mail filtering devices.

More detailed differentiations of TI can be found in description standards and exchange protocols, e.g., the Incident Object Description Exchange Format (IODEF) or the Structured Threat Information Exchange Format (STIX) [34]. STIX 2.0, for example, distinguishes twelve STIX Domain Objects (SDOs): attack pattern, campaign, course of action, identity, indicator, intrusion set, malware, observed data, report, threat actor, tool, and vulnerability [15]. Specific attributes are defined in STIX 2.0 for each SDO.

2.2 Threat Intelligence Sharing

Exchanging evidence-based knowledge about actual or potential threats across companies and authorities has been labelled threat intelligence sharing (TIS) [28, 36]. We define it as cross-organizational collaboration for collecting, aggregating, analyzing and disseminating threat-related information [1, 28, 36, 47]. The label ‘threat intelligence sharing’ is not used uniformly. Rather, there are a number of synonyms that refer to essentially the same content. The following term gives an overview of the labels that are often used together: [cyber] AND [threat | security] AND [intelligence | information | data | knowledge] AND [sharing | exchange | collaboration].

TIS leads to an improved level of information in all participating organizations and enables the bundling of security resources [28, 52]. For example, collaborative collection, aggregation and analysis of TI support sensing of coordinated attacks by providing for correlation of seemingly disparate information [48]. It facilitates emergency preparedness by supporting rapid dissemination of information concerning threats or response requirements [62]. Thus, organizations participating in TIS are able to identify and mitigate new and emerging threats more quickly, easily and cost-effective compared to a situation where they were on their own [10, 51].

In recent years, research into TIS has been significantly extended. Skopik et al. divide research on TIS into five groups [51]. The first group examines opportunities and challenges of TIS. Scholars analyze sharing scenarios, discuss advantages and disadvantages of TIS, describe requirements for efficient cooperation and coordination and determine benefits of TIS. The second group is dedicated to legal and regulatory aspects of TIS. Authors discuss legal conditions and examine whether and how legislation can create the basis for an effective exchange of threat intelligence. Papers in the third group explore standardization initiatives relating to TIS. Researchers discuss standardization efforts, compare standards and explore use
and dissemination of standards. The fourth group consists of papers exploring tools that support TIS. Scholars describe concepts, prototypes and products, analyze and compare selected tools and give recommendations for further development. The last group includes papers that examine integration aspects of TIS. Relevant papers explore how TIS can be integrated into organizational routines and processes.

2.3 Threat Intelligence Sharing Platforms

Threat intelligence sharing platforms (TISPs) are tools that support organizations in sharing threat intelligence. We define these platforms as inter-organizational systems that provide TI to organizations. TISPs offer functions for collecting, aggregating and analyzing TI that can be disseminated to other organizations via the platforms [12]. These platforms promise to enable customizable, controlled multilateral TIS, to allow exchange of information between non-compatible data formats, to provide collaboration for generating, refining and vetting of threat-relevant information and to provide interfaces for both humans and machines [8]. In recent years, various TISPs have been put into operation that more or less meet these requirements.

Lately, many researchers have focused on studying basic requirements and challenges for the development of TISPs [8, 12, 45]. In contrast, there are only few studies that describe and compare existing platforms or analyze the use of platforms in organizations and their impact on organizational processes and decisions [16, 40, 44, 52, 55]. An even smaller number of scholars have investigated how TISPs can be integrated into organizational systems, processes and routines [4, 24].

2.4 The need for TISP integration

TISPs are powerful tools that allow organizations to more effectively obtain and share technical, tactical, operational and strategic TI [51]. With the help of the platforms, organizations gain access to significantly more and higher quality TI. TISPs also enable them to process TI more automated. TISPs significantly improve an organization’s ability to perceive cyber security threats [52]. Increased speed and accuracy of threat sensing are particularly noteworthy [48].

However, to unfold their full potential, TISPs need to be integrated as intensively as possible into an organization’s IT infrastructure [40, 52]. In other words, TISPs support sensing of cyber security risks, but they need to be complemented by systems in organizations that support adequate reactions. As TISPs provide automated TI promising faster, more targeted and more efficient threat sensing, it would be ideal if systems that receive TI could also respond quickly and efficiently, i.e. as fully automated as possible. Candidates for such systems are security tools, e.g. vulnerability management systems, incident management systems or security information and event management systems [20, 24]. Such systems are frequently used by larger organizations. They ensure that security staff and administrators are informed about unusual IT events in their systems, so that they can react instantly. However, in most cases they still have to react manually. Another significant improvement will be achieved if TI obtained from TISPs can be directly received and processed by security-critical hardware and software systems. A straight-through processing of TI without delays due to human intervention would be possible. This requires that these systems have mechanisms in place which enable them to adequately react to the received TI. We understand security policies as mechanisms that provide such features.

3 SECURITY-POLICY-CONTROLLED SYSTEMS FOR AUTOMATED THREAT RESPONSE

The goal of this section is to examine the role of security policies as a technology for responding to cyber threats. We discuss both the merits and costs of employing this technology. We conclude with an inherent problem of security-policy-controlled systems that we address by integrating these systems with a dynamic threat sensing approach using TISPs.

3.1 Security Policies

In section 2 we have discussed goals and merits of TISPs on the level of managing security-critical IT infrastructures. On the technical level, such infrastructures are composed of hardware and software. The latter enforces application-specific security goals (e.g. confidentiality of customer information, integrity of accounting information etc.). Examples for security-critical software systems that are part of an organizational IT infrastructure include database information systems, enterprise resource planning and workflow management systems, and also the underlying runtime software e.g. operating systems (OS) and middleware. To protect security-critical resources such systems increasingly rely on a security policy: an automatically enforced set of rules that control and restrict access to such resources. If any technical access decision in a software system is governed by a security policy, we speak of a security-policy-controlled system (SPCS).

Security policies can be found in a multitude of application domains:

- **Access control (AC)** policy rules control evaluating, granting and revoking access privileges [18, 46]. Being one of the most prominent policy classes in practice, AC policies can be found in OSs [32, 57], information systems for large organizations and enterprises [41], web services middleware [61], or social network systems and online community services [25].

- **Information flow (IF)** policy rules control tracing, classifying and confining potential flows of information [5, 58]. They are elements of workflow management systems [11, 56] and, increasingly, in OSs [23].

- **Non-interference (NI)** policy rules control isolating domains of system users and resources [21, 42]. They are used in cloud computing systems and also in OSs for mobile devices [43].

Despite rules differ semantically among these classes they all aim at guarantees towards security properties of SPCSs. To this end, security policies enable the use of formal methods to prove correctness with respect to the respective application domain.

In the following, we first take a look at the process of engineering security policies to illustrate how correctness guarantees can be achieved. This requires highly specialized expertise and tends to be cumbersome. This motivates the need for a more dynamic and
A modern approach for this is based on encapsulating strategies for model analysis. This makes it dramatically increases the cost of detecting and correcting errors. One of the reasons is that model analysis is often pestered by computational complexity. This approach often are rather forbidding. One of the reasons is that model analysis is often pestered by computational complexity. This dramatically increases the cost of detecting and correcting errors. Because of these re-engineering costs, rules in a security policy implementations \[5, 49, 57\]. While model-based security engineering achieves considerable improvements in the effectiveness and correctness of a system's security properties, the costs of this approach often are rather forbidding. One of the reasons is that model analysis is often pestered by computational complexity. This dramatically increases the cost of detecting and correcting errors. Because of these re-engineering costs, rules in a security policy tend to remain static throughout a system's lifetime. This makes it difficult to effectively and efficiently adapt a SPCS to threats and incidents that were unexpected or unknown in the original policy engineering step. Current standard approaches for enforcing security policies \[17, 19\] have not been designed to adapt to changing operational risks.

In the absence of adaptive security policies, the dynamics of threats in modern communicating IT systems increasingly require an immediate response. Human action is too slow for re-engineering a policy implementation once an unexpected threat arises. In a resource constrained environment, e.g. in SMEs, this problem is exacerbated by a general lack of security expertise required for such re-engineering.

The basic problem here is comparable to the problem of changing requirements of application software: for application software and security policies, a new requirement (e.g. a new class of threats) leads to a revision of costly engineering steps, most prominently the formal analysis in case of security policies. In software engineering, such problems are solved by an adaptive software design which leads to a product that is dynamically and, to some extent, automatically reconfigured (instead of re-engineered) based on changing requirements within a certain range. Such reconfiguration is done during software operation rather than software development. Similarly, security engineering has devised an approach called context-aware security policies [13, 50, 59]. Such policies are based on context-aware security models that allow for a formal definition and verification of a range of security properties depending on the required degree of adaptiveness [7, 49, 60].

### 3.3 Context-aware Security Policies

Context-aware security policies enable a system to react to changes of physical or logical features, represented by a local variable. The value of such variables may be either computed (e.g. time, date, resource usage) or perceived by sensors (e.g. temperature, geographic location, NFC device proximity). Such security policies can be found in any of the classes presented in section 3.1, especially in AC policies \[7, 25, 49\]. In many of their application domains, e.g. mobile devices \[43, 50\], the notion of context is based on the immediate values of measured or computed features local to a device. However, to dynamically adapt policy decisions to changing threats as a more abstract concept, AC systems also demand a more abstract notion of context. A first step towards this goal is taken by risk-based security policies.

Such policies use context information to derive an abstract quantification of "threat level", called risk. It is basically computed from a multitude of contextual features as inputs. The numerical output of the risk computation is used by a decision algorithm, e.g. based on fuzzy logic \[38\]. The algorithm is an element of the security policy rules for determining an access decision.

### 3.4 The Problem of Automated Threat Response

Risk-aware security policies may serve as a basis for responding to unexpected or unknown threats and incidents \[9, 14, 38\]. However, the paradigm reveals some troubling issues in practice:

- Context information gathered by the system itself is local by nature. A significant share of threats in information systems originate from outside the system, e.g. exploits of interface vulnerabilities of the software run by a server. To counter an attack,
a risk-based policy relies on collecting information *after* an attack has occurred. This limits the possible success of such policies to merely mitigating the effects of rather insignificant threats. In practice, this explicitly excludes root kits or similar classes of automated malware with system-scale impact.

- Context is based on primitive values, while a security threat is an abstract concept. This means that physical or logical features of the local system cannot be used to indicate important classes of threats. In particular, this includes complex incidents e.g. a firewall penetration or system intrusion by manipulating the behavior of privileged applications. In practice, collecting and analyzing such information requires specialized functionality, which is not necessarily part of a SPCS.

On the one hand, the state of the art in context-aware AC provides us with a paradigm to enable automated threat response in SPCSs. On the other hand, both the type and the sources of context information currently used to determine risk as an abstract decision criterion does not meet the requirements that arise from today’s most important and practically relevant threats. From a policy engineering perspective, it is yet an unsolved problem how to react to dynamic information about threats from arbitrary and heterogeneous sources. Solving this problem could lead to a new class of SPCSs, whose security properties are neither tied to a costly re-engineering process nor to the expertise required to perform it.

4 INTEGRATION OF TISPS AND SPCSS

In the previous two sections, we have explained that TISPs support sensing of threats and SPCSs responding to threats. This enables a certain level of security automation. By integrating TISPs and SPCSs, however, the automation can be increased significantly. In this section, we explore the integration of the two approaches. First, we describe pros and cons of a direct and indirect TISP integration. Subsequently, we present a feasible concept for direct integration between TISPs and SPCSs. We then discuss the advantages of such integration using the example of two threat scenarios.

4.1 Direct and Indirect Integration

*Indirect integration* means that TISPs are integrated into an organization’s security tools, e.g. intrusion detection systems, vulnerability management systems, incident management systems or security information and event management systems. With the help of these tools, organizations can automatically access and process TI provided by TISPs [20, 24]. This helps to inform security staff and administrators more quickly and accurately about cyber threats. However, the initiation and execution of threat responses is usually not yet fully automated. If a security-critical system must be better protected from acute danger, human interventions are still required. TI obtained from TISPs are not forwarded to these systems directly. Security staff and administrators usually must perform this integration task.

We refer to *direct integration* when TI obtained from TISPs is received and processed directly by security-critical systems. A straight-through processing of TI occurs, i.e. both cyber threat sensing and responding are performed without any human intervention. Security staff and administrators can focus on monitoring. They must intervene only selectively if required. Full automation enables a significantly faster reaction to cyber threats with less personnel. A more targeted response can also be expected, as the characteristics of individual security-critical systems can be taken into account better. However, most security-critical systems have to be adapted to include mechanisms that can adequately handle TI provided by TISPs. This regularly requires extensive intervention in individual systems to support automated threat response. SPCSs provide an excellent basis for direct integration. No comprehensive studies and solutions for this integration option have been published so far.

4.2 Conceptual Design for Direct Integration

We now discuss basic technical concepts for designing SPCSs that directly integrate TISPs. We focus on architectural properties and interfaces based on the assumption that implementation techniques already in use for SPCSs will not be affected in any way. As we have already introduced both risk-based, context-aware AC policies as a promising starting point, we now use the term “AC system” to denote any required functionality that implements the enforcement of such policies in a local software system.

Any local AC system that integrates TISPs should be part of a distributed security information system to manage interactions between local security mechanisms and TI providers. To achieve this, we propose three key design questions to be addressed:

(1) Which strategies should be used by the AC system for responding to threats provided by TISPs?

(2) Which communication interfaces for TI and which mechanisms for threat reaction are needed in the AC system, and where are they placed in a security architecture?

(3) Which standard for TI representation and exchange should be used?

It should be highlighted that the answers to questions 2 and 3, as going beyond the merely technical aspects, first require an answer to question 1: An abstract framework for threat reaction strategies is needed to identify basic functional requirements for any communication models, data structures, interfaces and mechanisms used to communicate and process TI. Because of this dependency, we first identify the required high-level policy semantics derived from context-aware security policies, which already focus on operational flexibility (see section 3.3).

*Question 1: Threat Reaction Strategies.* In the following we treat any context-aware AC policy as a black box of rules $P$ that authorize any access operation $op \in OP$ (e.g. get for an HTTP transfer or execute for a binary file execution) related to a vector of entities $e = (e_i)_{i=1}^n \in E^n$ (e.g. user processes, HTML documents or files) and a vector of context values $v = (v_j)_{j=1}^m \in V^m$ (e.g. time or geographic location). We then do not need any information about the concrete policy class, other than being context-aware, since any more fine-grained semantics e.g. roles, attributes or hierarchies are subject to our black box. Instead, we define its interface by a semantically neutral access function $f_P : E^n \times V^m \times OP \rightarrow \{true, false\}$. An access request to transfer a website document could be modeled as $f_P((142.144.122.42, http://www.example.com/foobar), (2019-08-26T09:00:00+01:00, 51.297273N 1.063485E), get)$ (1)

$\langle http://www.example.com/foobar, 2019-08-26T09:00:00+01:00, 51.297273N 1.063485E, get \rangle$
which represents information about the calling entity (identified by its IP address), the target entity (identified by its URI), context values for “time” and “geographic location”, and the operation get. The response to this request (true or false) is determined by the SPCS by checking these information against $P$.

To allow for taking into account dynamic threats, another function related to risk-based AC is introduced: the risk function $r_M$. Again, the risk evaluation metric that implements this function (e.g. based on fuzzy logic [38] or, in the simplest case, on a risk evaluation metric $M$, both stored in a policy information point (PIP), against any access to a resource. This results in an access decision based on the hierarchical decisions of both tiers 1 and 2. The actual enforcement of this decision is realized through multiple policy enforcement points (PEP). PEPs are typically implemented as a part of the management software for different types of resources, e.g. in the file system management in case of access to secondary memory controlled by an OS. They are called in any operation on the accessed resource (e.g. read, append, execute, …for files) and in turn call a well-defined PDP interface that implements $f^+$. Finally, an event processing point (EPP) provides information about the physical and logical context of the AC systems as outlined in section 3.3. The EPP is implemented in a reactive manner based on asynchronous triggers instead of function calls. These may originate from local hard- and software components e.g. GPS, clock or temperature sensors, typically implemented by an interrupt mechanism of the underlying OS. Any access decision of the PDP may also trigger an event which is needed for logging and auditing.

As a general processing point for context-related events, the EPP is predestined to implement an interface for TI. Since TISPs represent separate infrastructures, each AC system is then connected to such platforms via an event-based communication model (e.g. a publisher-subscriber system, see [4]) that helps implementing TI events as external triggers, retrieved over some network infrastructure (see Fig. 1). We argue that by integrating such events on the basis of context-aware AC systems, any modification necessary to the system’s architecture is minimal, since only interfaces and communication standards are affected. Even more important, by modeling and implementing only the interfaces to the policy logic $P$, any previous verifications that ensure their correctness are never invalidated by integrating TISP support. In fact, we expect that the major portion of practical effort on the AC system side lies in an implementation of $M$ that has to interpret TI to determine the risk of an access.

In this architecture, any resource access is processed as follows: Whenever a client application, e.g. a user process running on a security-policy-controlled OS, calls an operation of the system’s application programming interface, e.g. read, a PEP is called as part of this operation’s implementation. The PEP mediates this access to a resource (e.g. a file to be read) by requesting a boolean decision from the PDP. As an element of this request, all information about the accessing entity (e.g. a process ID), the accessed entities (e.g. a file’s inode number), and the type of operation (e.g. read) is passed to the PDP. The PDP then returns the value of $f^+$ by evaluating both authorization tiers: for tier 1, $r_M$, TI-related context information is fetched. This is accomplished by an access event trigger, implemented e.g. using interrupts. It returns any context information gathered by the EPP. This information is asynchronously received, either from external sources (including a TISP) via a publisher-subscriber infrastructure, or from internal sources, e.g. any traditional context information such as hardware-induced events (local
triggers). Based on the full vector of \( e, v \) and \( op \), the implementation of \( M \) in the PIP is then queried for a risk-based access decision. For tier 2, \( fp \), the implementation of \( P \) in the PIP is subsequently queried in case \( R_M \) was evaluated as \( \text{true} \). The conjunction of both booleans is then returned as a boolean decision to either allow or deny the requested access. Finally, the PEP enforces this decision by either calling the appropriate resource management interface (e.g. a file system implementation) or returning a predefined error value to the client.

**Question 3: Information Representation.** For this design study we have selected representative technological standards in TISP and SPCS implementations. In particular, we propose MISP (Malware Information Sharing Platform) [55] as a TISP solution in a mature stage of usage within the TISP community. Its technical merits are

- flexible support for a multitude of data sources and data formats,
- Java-based deployment independent from OS and runtime environment, and
- open source and well-documented implementation.

On the side of SPCS implementation, we refrain from suggesting specific OSs or application protection systems due to the massive degree of diversity involved in security-critical application domains. We rather highlight the architectural principles that should be adhered to for incorporating the necessary interfaces for TI: As we have discussed for question 2, the AC systems must implement a reference monitor architecture that is functionally modular. These modules need to match the state of the art in AC enforcement as represented by the Policy Machine specification [19] and its successor, the NGAC (Next-Generation Access Control) standard [17]. Besides the security-related functionality, an asynchronous communication interface with a MISP server is required for external triggers. This can be provided by standard socket implementations (e.g. POSIX sockets).

**4.3 Threat Scenarios**

In the following sections, we discuss benefits of our direct integration concept (see section 4.2) based on two threat scenarios. First, a malware-based attack on discretionary access control mechanisms of end user systems and second, a distributed denial-of-service attack on a backbone system of security-critical IT infrastructure. We demonstrate how both attacks can neither be detected by an SPCS nor effectively mitigated by traditional, indirect TISP integration.

**4.3.1 Malware Attack.** The broad range of malware imposes one of the most persistent threats on software systems, especially in case of professional root kits [30]. Typically embedded in useful software functionality, attackers rely on root kits to gain complete control of a target system by compromising basic OS functionality, e.g. configuration, administration, and logging tools. Even with SPCSs attempting to mitigate the severity of a successful attack, they remain possible by exploiting implementation errors on any software layer. Even worse, the effectiveness of SPCSs is limited if users control or override policy decisions (discretionary access control, DAC) for usability reasons.

Examples for DAC mechanisms prioritized over security policy rules can be found in most OS, e.g. the Windows User Account Control (UAC) [23]: In principal, the execution of untrusted programs (e.g. downloaded as an email attachment) is heavily restricted by a Bell-LaPadula security policy [5] which prevents the program from manipulating system service binaries (among others). This prevents an effective root kit attack. However, since the knowledge of the security policy is static and independent from runtime context, it is impossible to determine if a program tries to execute legitimate functionality (e.g. creating an installation directory on a system-critical file system) or performs additional, malicious activities (e.g. replacing the task manager program by a modified binary that hides the execution of the root kit process). Instead, a runtime prompt for a user decision to override the policy restrictions relies on complementary technical and non-technical security measures, e.g. antivirus tools and user security awareness. In contrast to enforcing a verified security policy, these measures are imprecise (e.g. malware recognition heuristics), slow (e.g. daily virus definition updates or monthly security training) and they require expert skills.\(^1\)

A solution to this problem that avoids most of the above-mentioned practical drawbacks is to enhance security policy enforcement with TI as described in the previous section. For any local authorization request of a user process \( i \) to execute a program binary \( x \), basically two cases arise:

1. No relevant TI for \( x \) has been gathered by the EPP, e.g. no recent MISP event attributed with a matching MD5 hash value \( h \).\(^2\) It follows that our risk evaluation metrics \( M \) considers the execution of \( x \) not risky (\( r_M(i, x, h, \text{execute}) = \text{true} \)) which leads to another decision by \( P \) based on the same values. This tier-2-decision may check, as an example, a policy rule "the creating user of \( x \) ... " [27].

\(^1\)Cf. Jesper M. Johansson, former Microsoft Principal Security Engineer: “By extension, yes, there may be less malware, but that will depend on whether users keep UAC enabled [...] and that users stop viewing prompts as fast-clicking exercises [...]” [27].

\(^2\)This is just an example criterion, in practice, a more complex evaluation of different TI e.g. from the MISP categories "artifact dropped, network delivery or payload installation" could be used by \( M \), possibly in addition to quantifiers (e.g. "threat level") used in specific platforms.
must be the same that runs \( i^* \) or even an explicit, interactive user approval like Windows UAC (as already mentioned, any existing security policy semantics may still be applied). Once both queries are evaluated, the PDP returns an access decision for \( f^* \).

(2) There is relevant TI for \( x \) gathered by the EPP, e.g. an MD5 hash value matching for \( h \) that is blacklisted as (potential) root kit malware. Based on this information, \( r_{MD}(i, x, h, \text{execute}) \) evaluates to false which automatically leads to \( f^*(i, x, h, \text{execute}) = false \) which in turn denies the execution of \( x \) without considering any security policy rules (including possible user approval requests).

In the second case, we have automatically identified and prevented a possible root kit attack with minimal changes to the system architecture and no reliance on user competence. Since this scenario basically uses TI to anticipate and rigorously enforce security-compliant reactions to malware information, it also shows how the new protection concept actually complements traditional antivirus software (rather than replacing it): such software is still in charge of implementing the last line of defense for the user, just as it was before.

4.3.2 Denial-of-Service-Attack. A second scenario relates to software systems that provide basic functionality for any IT service, e.g. a web server backbone system. Whenever such systems are accessible from outside the technical realm of their organization, they face distributed denial-of-service attacks (DDoS) as a major threat. One simple example for such systems is a web server hosting an online shopping front end (other relevant use cases include database servers or proxies or media servers, e.g. for video-on-demand services).

As in the previous scenario, these are SPCSs in a professional setting which means there are AC mechanisms in place to prevent attacks on client information confidentiality, accounting information integrity etc. What cannot be prevented, however, is any attack on service availability: Once a botnet-based attack starts sending massive amounts of parallel requests to the server (e.g. HTTP get operations on product pages), it is impossible for the security policy to distinguish between an attack or bursts of legitimate client requests. This is especially problematic in scenarios where such bursts are part of typical usage patterns (e.g. the aforementioned video-on-demand server) combined with a sophisticated timing of the attack which complicates detection on a heuristic basis. As in the previous scenario, timing heuristics are an example for static, runtime-independent knowledge of the security policy.

We propose a solution based on direct integration of TISPs and SPCSs. We discuss two practical cases, both involving a client \( i \) (identified by its IP address) that attempts to access a web resource \( x \) through an HTTP get at some local time \( t \):

(1) No relevant TI for \( i \) has been gathered by the EPP, i.e. the client IP address does not appear in any recent MISP event. Assume no other risk indicators for this access, such that \( r_{MISP}(i, x, t, \text{get}) = true \). The access is then authorized by \( P \) as usual, e.g. by checking if \( i \) represents a logged-in user or if both \( x \) is restricted content and \( t \) after 22:00:00 local time. Just as in the previous scenario, the PDP combines both decisions in a decision \( f^* \).

(2) The IP address \( i \) matches with a TISP-reported blacklist of potential originators of DDoS attacks gathered by the EPP. Depending on risk evaluation logic, an internal risk value in \( M \) is increased.

The time stamp \( t \) is then compared with any last request time stamp from any blacklisted IP address. This further increases the risk value beyond some threshold \( M \). As a result, the risk evaluation leads to \( r_{MISP}(i, x, t, \text{get}) = false \) which prevents any further processing of the current and any following access requests originating from \( i \). For implementing the latter in a practical setting, the PDP may update a cache of previously made deny decisions (blacklisting) in the PIP.

Even more than the previous example, this threat scenario demonstrates the fact that the implementation of \( M \) based on interpreted TI is not a trivial task. As already mentioned, we expect this risk evaluation logic to require more principal research, adding to the theoretical foundations of model-based security engineering.

5 FUTURE WORK

We have identified three major steps to further substantiate the ideas presented in this paper.

First, a fundamental requirement for automated threat responses is a machine-readable description of TI events compatible with the context semantics used by the PDP. To this end, we are currently working on the formalization of TI event descriptions by ontologies that allow for (a) a tool-supported specification of TI events, (b) their integration in TISPs and (c) their automated interpretation by SP rules. Despite existing description standards (mentioned in section 2.1) cannot yet be interpreted fully automatically, they may serve as a starting point here. This facilitates a straightforward adaption of existing TISP implementations already using these standards.

Second, automated responses to TI imposes novel requirements on SP design: SPs have to (a) provide interfaces to TI ontologies, (b) employ a risk-based selection of threat reaction strategies and (c) enforce these strategies in compliance with the reference monitor principles. This requires new models and paradigms for specification, verification and implementation of responsive security policies.

Third, we will implement a prototype for experimentally assessing the feasibility of our approach, based on the software architecture outlined in Fig. 1. This allows a better understanding of required computing resources and the potential improvements in threat response timing, efficiency and quality.

6 CONCLUSION

This paper discusses technological foundations for automated responses to security crisis situations in IT systems. We argue that fast and fully automated responses to cyber threats can be achieved by integrating two recent technologies, threat intelligence sharing platforms for sensing and disseminating threat intelligence and responsive security policies that automatically react to threat alerts.

We consider the work in this paper a first step towards a fast, sophisticated and cost-efficient response to security incidents. To demonstrate the potential of the approach and to argue its feasibility we discussed two threat scenarios.

The focus of our approach is on technical threat intelligence. Future work will focus on ontologies for threat intelligence formalization and new design paradigms for responsive security policies.

3 From the technical side, this has to be complemented with other, not SPCS-related mechanisms such as firewall autoconfiguration to maximize the effect of DDoS countermeasures. We consider such system-internal coordination of automated defense mechanisms beyond the scope of this work.
Automated Cyber Threat Sensing and Responding

both used to support a correct and seamless implementation of threat responses.

It is to be expected that the relevance of strategic, operational and tactical threat information will increase. This trend must be given greater attention in the future. More research is needed to ensure fast and fully automated responses to such types of threat intelligence.

REFERENCES


