

# Towards Early Warning Systems – Challenges, Technologies and Architecture\*

Martin Apel<sup>1</sup>, Joachim Biskup<sup>1</sup>, Ulrich Flegel<sup>2</sup>, and Michael Meier<sup>1</sup>

<sup>1</sup> TU Dortmund, Computer Science VI, 44221 Dortmund, Germany  
{martin.apel, joachim.biskup, michael.meier}@cs.tu-dortmund.de

<sup>2</sup> SAP Research, Vincenz-Prießnitz-Str. 1, 76131 Karlsruhe, Germany  
ulrich.flegel@sap.com

**Abstract.** We present the architecture of an *automatic* early warning system (EWS) that aims at providing predictions and advice regarding security threats in information and communication technology without incorporation of cognitive abilities of humans and forms the basis for drawing a situation picture. Our EWS particularly targets the growing malware threat and shall achieve the required capabilities by combining malware collectors, malware analysis systems, malware behavior clustering, signature generation and distribution and malware/misuse detection system into an integrated process chain. The quality and timeliness of the results delivered by the EWS are influenced by the number and location of participating partners that share information on security incidents. In order to enable such a cooperation and an effective deployment of the EWS, interests and confidentiality requirements of the parties involved need to be carefully examined. We discuss technical details of the EWS components, evaluate alternatives and examine the interests of all parties involved in the anticipated deployment scenario.

**Keywords:** automated process chain, early warning, clustering, confidentiality, intrusion detection, signature learning.

## 1 Introduction

Along with the growing dependency of our society on information technology (IT) systems, concerns regarding IT security are becoming more urgent. While up to now primarily preventive measures and mechanisms have been focused, it becomes increasingly apparent that IT security cannot be achieved by prevention alone. Rather, preventive measures and reactive aspects need to complement one another. Precondition to reaction on security incidents is a dependable and timely detection of respective situations. A cooperative information exchange between different institutions is not only advantageous, but also mandatory in order to detect distributed and coordinated attacks. From a large-scale acquisition of pertinent information by an early warning system (EWS) arises the opportunity to draw up the situation picture that allows the detection of trends and upcoming threats, thereby allowing to take appropriate measures.

---

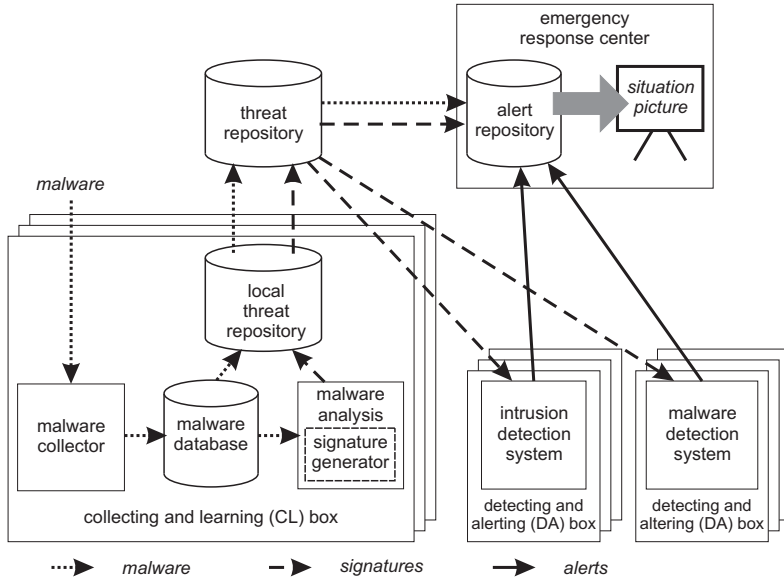
\* This work was accomplished in cooperation with the German Federal Office for Information Security (BSI) and the German Federal Ministry of Economics and Technology (BMWi).

The need for integrating data in order to construct such a situation picture is widely accepted (cf. e.g., [1,2,3,4,5]). However, typically there exist reservations concerning the distribution of information allowing outsiders insights into security incidents of individual institutions. These reservations are opposing the integration of information and so far prohibit the creation of a situation picture. A practical EWS needs to take the conflicting interests of the participating parties into consideration. A resolution of the conflicts can be achieved by using information reductions, e.g., pseudonymization.

Though the term EWS has been used in the literature, there is no common, accepted definition of what early warning is (and no differentiation to, e.g., intrusion detection systems (IDS)). A vague definition of an early warning information system (EWIS) is given by [6], who defines EWIS by sketching its purpose: ‘EWIS assists experts and policy makers in assessing desired options for...’ several particular security measures. By outlining one particular realization technique, [6] defines ‘[an EWS] for IT security surveillance is based on a specific procedure to detect as early as possible any departure from usual or normal observed frequency of phenomena.’ A more general operational definition of early warning is used by [1]: ‘In case of perceptible indicators, and no or (still) a low number of victims, information must be distributed, to help others – not yet victims, including response organizations, in order to avoid a major crisis.’ We adopt the more declarative definition given by [7]: ‘EWS aim at detecting unclassified but potentially harmful system behavior based on preliminary indications and are complementary to intrusion detection systems. Both kinds of systems try to detect, identify and react before possible damage occurs and contribute to an integrated and aggregated situation report (big picture). A particular emphasis of EWS is to establish hypotheses and predictions as well as to generate advices in still not completely understood situations. Thus the term early has two meanings, a) to start early in time aiming to avoid/minimize damage, and b) to process uncertain and incomplete information.’

This paper sketches the architecture of such an early warning system that is particularly targeting the malware threat and is currently under development. The fundamental requirements of an early warning system are a) automatic detection of known as well as unknown automated security violations, and b) automated indication of security incidents in the form of alerts, which can be combined into a situation picture. These capabilities shall be achieved by combining the following technologies to an integrated automated process chain: 1) capturing active malware using honeypot technology based malware collectors, 2) analysis of malware and generation of patterns for detecting malware (signatures) using machine learning techniques, 3) central consolidation and storage of the generated signatures, 4) distribution and deployment of signatures to signature-based detection systems, 5) central alerting to an emergency response center.

Our contribution is threefold. Firstly, we describe the architecture of our EWS in section 2, followed by a discussion of the anticipated deployment scenario in section 3. Secondly, we discuss the technical and organizational challenges, which need to be considered when implementing the EWS, and sketch technical details of the components of our EWS in section 4. Thirdly, deployment issues, in particular regarding participating parties, their interests and the resulting privacy and confidentiality requirements for an EWS, are examined in section 4.5. Finally, we discuss related work and summarize in section 5.



**Fig. 1.** Architecture of the early warning system

## 2 Architecture

We propose the following architecture for an EWS as shown in fig. 1. It comprises four basic components. The *collecting and learning (CL) box* bundles the functional components to collect malware, to analyze malware to extract features used for learning and to generate appropriate detection patterns. Automated signature generation is a primary focus of the described project. Because of the honeypot component placed upstream our approach can exploit the assumptions that the collected and analyzed files are malware indeed. The *threat repository* is used to centrally manage information on malware and detection criteria delivered by CL boxes. It supplies the information needed for detecting malware to the *detecting and alerting (DA) boxes*, which contain the functional components to detect respective security violations and to generate alerts. Alerts generated by the DA boxes as well as information on malware supplied by the threat repository form the basis for constructing a *situation picture* and they are centrally managed in the *alert repository*. Please note that, even if not the case in the particular design discussed here, multiple threat and alert repositories are possible and cooperative exchange of information between different EWS at the repository level can be foreseen. The overall procedure allows to match signatures for detecting misuse of observed threats in a timely and fully automated manner. It is designed to simultaneously achieve the following: retaining the advantage of misuse detection to provide specific alerts with a low false positive rate and compensating for the original weakness of misuse detection to detect only a priori known security violations.

### 3 Deployment Scenario – Parties and Interests

Both the protection level achieved for the participating systems and the quality of the situation picture depend on the number and placement of deployed CL boxes as well as DA boxes. The more CL boxes are deployed at suitable locations in the network the higher will be the likelihood that new malware is collected during an early stage of its distribution and signatures for detection systems are supplied by the threat repository early enough to observe, detect and restrict further distribution of the malware. The more DA boxes are suitably placed in the network the more comprehensive the information base for constructing the situation picture will be. The installation of a larger number of CL boxes as well as DA boxes in different network domains is anticipated as sketched in Fig. 2. A primary prerequisite is that the domain owners agree to cooperatively exchange information on security incidents. Without additional protective measures, there exists, e.g., for the owner of the threat repository the possibility to gain knowledge about the occurrence of security incidents in the domains of CL boxes. Analogously, the owner of the emergency response center gains insight into security incidents detected in the domains of DA boxes. Since this may conflict with the interests of the domain owners, an agreement about exchanging the required information is hard to achieve. To resolve this problem and to rebut respective reservations, we investigate requirements on data and privacy protection for required information flows and consider complying information reductions, e.g., pseudonymization (cf. ‘data and confidentially protection’ in fig. 2). In particular, such information reductions effectively confine owners of central components to acquire sensitive information, without affecting the functionality of the system or significantly reducing the quality of the situation picture. In the following we refine the deployment scenario (cf. fig. 2) by discussing the suitable parties for operating the system components and elaborating on their interests.

For the current setting, we expect that the *alert repository* is operated by a government agency, such as the German Federal Office for Information Security (BSI), which

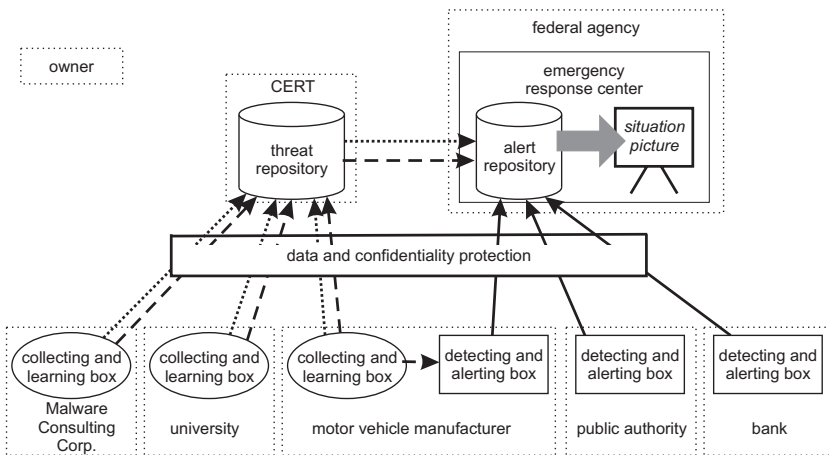


Fig. 2. Domains of an example deployment scenario

is officially commissioned to provide a situation picture. The agency might also outsource the operation to a CERT (computer emergency response team) or a private sector enterprise. Alternative models would be a private company providing the situation picture as a payable service, or a closed consortium of the EWS member organizations.

The *threat repository* may also be operated by a government agency, a CERT, a private sector company or a consortium of EWS member organizations. If the operating organizations of the alert and the threat repository differ, suitable combinations of operating organizations are subject to further investigation. Organizations that already operate an IDS are natural candidates for member organizations of an EWS. The primary driving force behind installing a *DA box* is the organization's interest in the information provided by the situation picture and its value for safeguarding its assets. All sectors that rely on IT services are promising candidates for operating DA boxes, i.e., government agencies, academia and private sector organizations.

For the *CL boxes*, the same consideration as before applies. In particular, a service provider may be specialized on providing a broad and representative collection of malware samples. While it is perfectly possible that an organization specializes either on collecting malware and generating signatures, or on detecting incidents and providing alerts, a combined operation of DA and of CL boxes is valuable. Operating both kinds of boxes allows the organization to generate new signatures to cater to its own need for detecting locally occurring malware and thwarting its recurring occurrence.

## 4 Details

### 4.1 Malware Collector

For collecting malware, server-honeypots like Nepenthes [8] and Amun [9], which provide vulnerable services in order to attract spreading malware, are used in our EWS. These systems are presently able to collect malware that exploits known vulnerabilities. Therefore they emulate vulnerable services. Spreading malware exploits vulnerable services to enter the victim system and downloads the malware binary from a *malware distribution system* (MDS) afterwards. Payload targeting an emulated service vulnerability is captured and analyzed by the malware collector, the URL of the malware binary on the MDS is extracted and the malware binary is downloaded eventually.

### 4.2 Malware Analysis System – Sandbox

Malware analysis systems are used for inspecting and extracting features of malware that are appropriate for characterizing and distinguishing malware and benign programs. Static and dynamic analysis can be distinguished. *Static analysis* focuses on static features that can be directly extracted from a malware sample. Sequences of data and instructions of a sample are typical examples. Morphing (aka obfuscation) techniques and tools demonstrate that programs of similar functionality do not need to share similar instruction and data sequences, leading to polymorphic and metamorphic variants of malware [10]. Consequently, higher-level structural features like the control-flow graphs (CFG) of programs, which can be extracted by disassemblers, are

studied (e.g. [11]). These techniques assume that similar programs need to share a similar structure. Unfortunately, albeit not yet commonly used by today’s malware authors, techniques exist to change the program structure without changing its functionality.

*Dynamic analysis* avoids these drawbacks by focusing on the dynamic behavior of a program, which can be extracted by executing a program and closely observing its activities, e.g., at the system call level. As polymorphic variants of the same malware are behaving almost identical, polymorphic samples are easily detected based on their common behavior [12]. Even some metamorphic techniques can be thwarted that way. A difficulty with dynamic techniques is to trigger the malicious execution path of the program under observation. Nevertheless, we postulate that dynamic analysis is the most promising alternative for malware analysis and detection. The actions of a malware are described by its interaction with the operating system (OS), specifically the system calls it uses. The *trace of a malware* represents the list of all system calls that are performed by the malware. As a malware can start multiple threads or even processes, the trace contains multiple blocks, each one describing one thread. Inside of the blocks the system calls are ordered chronologically. In the context of the project described here, CWSandbox [13] is used as dynamic malware analysis system.

### 4.3 Automatic Signature Creation

The creation of behavioral signatures consists of two steps (fig. 3). The first step is to group similar behavior reports together. To do so, a function is needed, that computes the (dis)similarity of two traces. Which function is used has a great impact on the resulting groups, as it determines the used features and their weighting. In the second step a signature is created for each of the resulting groups. During this step an additional set of traces is used, which contains the behavior of benign executables and is called the *good pool*. The usage of the good pool helps keeping the *false-positive* rate low.

**Clustering.** The grouping of malware traces can be done using a cluster algorithm (which groups elements from a set  $S$  regarding certain features into subsets  $C_i$  (clusters) such that  $S = \bigcup_i C_i$ ). Most algorithms will create disjoint clusters, i.e.,  $C_i \cap C_j = \emptyset$  for  $i \neq j$ . Unfortunately, there exist malware samples that show behavior common to multiple families, and thus their traces could be put into different groups. There are even samples that contain more than one individual malware. Overlapping cluster algorithms that

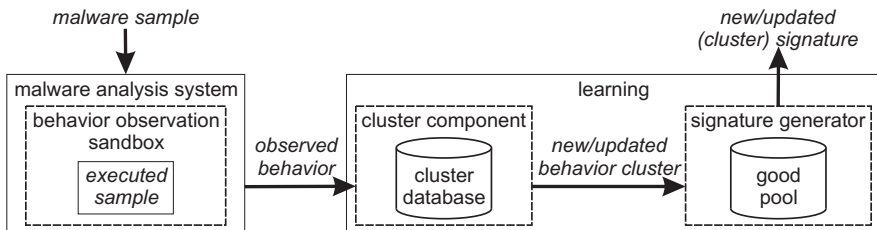


Fig. 3. Signature generation process

allow the presence of one element in multiple  $C_i$  are therefore of special interest to us. A cluster algorithm suited for our needs should fulfill the following criteria:

1. The only input parameter the algorithm needs are pairwise distances. This means that no special metric is required and makes the used metric interchangeable.
2. It should be able to handle noisy data (be robust). This provides the algorithm with a better ability to handle outliers and enables stronger generalization.
3. The number of clusters to be generated should not be part of the input, as an appropriate choice depends on the data set and varies between data sets.

Cluster algorithms fulfilling the above criteria are for example hierarchical clustering algorithms (single-, complete-link, WPGMA, UPGMA) [14] and fuzzy clustering [15] which also provides overlapping clusters, which would certainly improve the quality of the clustering, though it is not necessarily needed. The quality of the clustering depends on the *distance metric* used. We have to compute distances between traces of every pair of malware samples, and to do so repeatedly and thus highly efficiently. The distance between two traces relies on the properties that the metric will take into account. These properties can be quite different between different metrics and it has to be carefully examined which one to choose.

A well known approach is to treat the traces as long strings and use the *edit distance*, which computes the *cost* of making the two strings equal and relates these costs to the length of the strings. The edit distance has been used in malware analysis by Lee and Mody [16]. Unfortunately, the runtime complexity of the edit distance is  $O(n^2)$  in the length of the strings. Another possibility is the use of the *normalized compression distance (NCD)* [17], which approximates the Kolmogorov complexity using compression algorithms. The runtime of *NCD* is tied to the runtime complexity of the used compression algorithm. It has been employed successfully for malware analysis [18] and in many other fields. The used compression algorithm has to be chosen carefully as shown by Cebrián et. al. [19]. A fundamental problem with *NCD* is that it only measures the structural complexity of a string and ignores any semantic context. It is debatable whether the structural differences of the traces of different malware samples reflect their behavioral distance indeed. A quite different idea is based on an *embedding* function which embeds sequential data into a high-dimensional vector space using a formal language  $L$ . Each dimension of a vector corresponds to a specific word of  $L$  and holds the number of occurrences in the data. The dimensionality of each vector depends on the number of words in  $L$  which is usually very high. As the resulting vectors are very sparse, special data structures can be used to store them [20]. To compute the (dis)similarity between elements of the vector space the  $L_m$  (Minkowski) metrics can be chosen, which is defined as  $L_m(x, y) = (\sum_{i=1}^n |x_i - y_i|^m)^{\frac{1}{m}}$ . Another possibility is the usage of similarity coefficients.

Our experiments [21] showed that the usage of the Manhattan distance  $L_1$  (along with tries) outperformed the NCD (using ppmd and lzma compression) as well as the edit distance. The edit distance yielded better results than the NCD variants. Thus, for further development of our EWS, we chose the vectorization using the Manhattan distance.

**Behavioral Signature Creation.** The creation of the signatures is performed for a group of malware traces found to be similar using the clustering approach. The goal

is to determine sequences of system calls that are shared among these traces, but are absent in *normal* programs taken from the *good pool*. One possibility is to find all substrings  $s_1, \dots, s_l$  shared between all traces in the group and build a signature that matches a sequence if all of the shared strings are contained. The shared substrings can be found using *generalized suffix trees*, which can be built in time  $O(n_1 + n_2 + \dots + n_m)$ , where  $n_i$  is the length of the  $i$ -th trace, using the algorithm of Ukkonen [22]. Another approach relies on the embedding into a vector space and the use of a support vector machine (SVM) (which finds an optimal hyperplane between two sets of vectors). Choosing the sets  $S_m$  and  $S_b$  as the sets of vectors belonging to the malicious traces and the benign traces, respectively, the hyperplane found by a SVM can be taken as a signature.

**Optimization.** The clustering as well as the signature creation can be time consuming and thus, for the early warning context, demands to deploy optimization techniques like the following one. If a new sample arrives its behavior is tested against the existing signatures first. If any signature matches no further action is taken. If no signature matches, the distances of the new sample (resp. its behavior) to the existing groups of malware traces are evaluated. The new sample is added into the group that exhibits the smallest distance to the sample's trace and the signature for that group is recreated. As this procedure might worsen the quality of the clusters (creating a new singleton cluster or splitting an old cluster may be better), a complete re-clustering is performed periodically for all traces.

**Validation.** There are essentially two problems that can occur. The signature can be too specific or too general. To ensure that the signature is not too specific, a technique called  $X$ -fold-Cross validation is used. The group of malware traces for which a signature is to be created is divided into  $X$  parts.  $X - 1$  of these parts are used to create the signature while the remaining part is used for testing. If the signature is good and generalizes to a suitable extent, the traces not considered for signature creation should be detected as well. To ensure that the signature is not too general, the signature is tested against the *good pool*.  $X$ -fold-Cross validation can be used here as well.

#### 4.4 Malware/Misuse Detection Systems

Misuse detection systems are used to detect security incidents based on observations of security relevant events and signatures. Examples of available solutions that support this functionality include intrusion detection systems, virus scanners and firewalls. It is beneficial to integrate as many of the available detection products into the EWS as possible. Due to space restrictions we abstain from discussing common technical requirements of detection systems here and focus on requirements that are specific to the deployment of the detection systems within the EWS.

When integrated with EWS, detection systems need to be able to receive new signatures from a threat repository, to deploy them on the fly, and to forward generated alerts to an alert repository. Further, the signatures supplied by the EWS need to be compatible with the employed detection system. In particular the feature domain used for generating signatures in the EWS and the features observed and analyzed by the detection systems need to be compatible. That is, the behavioral features of malware that are extracted using CWSandbox and used for signature generation need to be observed/monitored by

the detection system in order to support effective matching of the supplied signatures. For integrating existing detection systems possibilities to transform signatures between different domains need to be realized.

Besides the integration of existing detection systems in DA boxes, a detection system called *jSAM* (*Java Signature Analysis Module*) is in under development that provides full support of the behavioral features used in EWS supplied signatures. Further highlights of *jSAM* are the expressiveness of the used signature language *EDL* (*Event Description Language*) [23], which is also used as signature transfer language inside the EWS, as well as the optimized matching strategies.

#### 4.5 Privacy and Confidentiality Requirements

We first consider entities and their potential relation to personal data. Malware samples may contain (hidden) clues about their authors and endpoint addresses of *malware target systems* (MTS), which are targeted victims of attacks executed by malware (e.g., a denial of service attack) and *malware drop zones* (MDZ), where data copied from *victim systems* (VS) is uploaded. While MDZ may be operated by malware authors, this is usually not the case. The copied data is rather delivered to VS that are controlled by malware authors. VS often are poorly managed home computers operated by natural persons. Hence, we consider MTS as well as MDZ endpoint data as personal data. Malware spreads from so-called *malware host systems* (MHS), and the CL Box can observe the MHS endpoint of a malware trying to spread to the CL Box. From the observed exploit payload the CL Box can extract the *malware distribution system* (MDS) endpoint. In the vast majority of cases the MHS and MDS are VS. Hence, we consider MHS and MDS endpoint data as personal data. We do not protect the interests of malware authors and will therefore ignore them in the following.

Next, we inspect the data flow from malware VS via a CL box to the central threat repository (cf. fig. 4). The CL Box initially receives the exploit of a given malware and can identify the originating MHS. In a second step, the CL Box extracts the download endpoint of the (possibly different) MDS (cf. section 4.1) where the complete malware sample is situated and finally downloads it. As follows, the CL Box receives the following data concerning the attacking MHS: sending endpoint (IP, port) and payload (exploit); and from the malware sample serving MDS: download endpoint (IP, port, protocol-specific path) and payload (malware sample). In addition the following data can be determined: receiving endpoint (CL Box IP, port), name of the vulnerable service, timestamp, malware sample id (e.g., md5 hash value). For further processing, the CL Box persists the following data to the local threat repository, which then will be sent to the global threat repository (as defined above): sending endpoint, receiving endpoint, download endpoint, name of vulnerable service, malware sample, timestamp, malware sample id. In addition malware analysis systems may extract additional potentially confidential data from malware samples (e.g., endpoint data of MTS, MDS or MDZ) and provide it to the threat repositories. CL boxes also provide automatically generated signatures to the threat repository, where a signature detects an observable behavioral pattern of the collected malware sample. The signatures are distributed to the DA boxes for malware detection. DA boxes may use a network interface (*observing endpoint*) for observing network behavior. When a DA box successfully matches a signature to observed behavior



**Table 1.** Summary of feature specific interests of the involved entities and interest support by the proposed balance of conflicting interests – c: confidentiality of own/domain-local feature; l: linkability of remote feature; d: disclosure of remote feature; blank: party has no interest; I: party's interest is not supported; S: party's interest is fully supported; P: party's interest is supported merely against selected other parties

feature	controlled by attacker	victim			CL box		DA box		threat repository		alert repository	
		c	l	d	c	l	d	l	d	l	d	
timestamp	Yes				P	S	I				S	S
alert signature name	Yes				P	S	I		S		S	S
sending endpoint of MHS	Yes	I	I	I	P	S	I		S		S	S
receiving endpoint of MTS	Yes	I	I	I	P	S	I		S		S	S
download endpoint of MDS	Yes	P	I	I	I	S	S		S		S	S
upload endpoint of MDZ	Yes	P	I	I	I	S	S		S		S	S
vulnerability module name	Yes				I	S	S		S		S	S
receiving endpoint of CL box					S	I	I		S		S	
observing endpoint of DA box					S	I	I		S		S	
malware exploit payload	Yes				I	S	S		S		S	
malware sample payload	Yes	P	I	I	I	S	S		S		S	

CL boxes, as well as DA boxes are assumed to be operated by organizations, not by natural persons. We also assume that the boxes are provided as stand-alone systems that have no users, except for initial administrative purposes. As a result we do not need to consider personal data regarding these boxes. It remains to consider the remaining data that is sent out to the repositories. This data can be considered sensitive when observed by customers or competitors, such that the organization is interested in keeping them confidential. At the same time a given organization may be interested in the malware incident data of other organizations to gain a competitive edge.

The threat repository and the alert repository merely collect data from the CL boxes and DA boxes, respectively. The data in the threat repository may be refined and enriched manually, but it will still not refer to the operating organization. Hence, we do not need to consider personal data of the operating organizations here. The role of the global threat repository is providing information to DA boxes for detecting and possibly directly blocking malware. For the purpose of transitory blacklisting sites involved in the outbreak of a given malware, the threat repository needs to disclose the endpoints of MDS, as well as MDZ. Proactively patching or shutting down services would be enabled by disclosing the receiving port numbers of the malware collector. Repository data refinement and analysis for enrichment is enabled by disclosure of the payloads (exploits and malware sample). Duplicate elimination is necessary for efficiency reasons. The alert repository needs to be able to link all data items to create a situation picture. Some data needs to be disclosed for detailed reporting and advisories for time-critical and transitory blacklisting.

Table 1 summarizes the interests of all parties, which, obviously, may conflict. For example, VS want to keep their endpoints confidential and CL boxes also want to keep their existence confidential. However, e.g., DA boxes of other domains are generally

interested in disclosing these features. It is therefore necessary to define a suitable balance between the conflicting interests. In some cases it suffices to support a given interest in linkability or disclosure only for the repository owners, in other cases it is necessary to support the given interest also for DA boxes. The proposed balance supports the confidentiality requirements of VS only partially: box owners and repository owners can in most cases link and disclose the VS endpoint. The confidentiality requirements of CL and DA box owners can only be supported for the receiving CL endpoint, the observing DA endpoint and the timestamp feature; for the other features it is necessary to let other DA boxes link and disclose them to allow for timely response. An array of techniques to balance interests was discussed in [25].

## 5 Related Work and Summary

A few EWS has been proposed in the literature. They all have the sharing and central collection of information in common but they can be differentiated regarding the kind of information that is processed and correlated. Another differentiating feature of EWS is the way they establish hypotheses and predictions and generate advice. Existing systems support information aggregation and visualization as well as statistic analysis but predictions or advice generation is left to the human user. In the following we discuss and compare related approaches regarding these differentiating features.

DSShield's Internet Storm Center [2] operates on firewall logs of several organizations and incorporates human interpretation and action in order to generate predictions and advice. MyNetWatchman [26] also processes firewall logs of multiple organizations but supports automatic generation of email notifications. eCSIRT.net system [3] comprises of a sensor network of intrusion detection systems. It collects and correlates alerts of these system which are visualized for human inspection. The Internet Motion Sensor (IMS) [27] statistically analyses dark net traffic that need to be interpreted by humans. SURFids [28] stores malware binaries collected by a network of Nepenthes sensors in a central database and supports generation of several statistics. The Deepsight system uses about 19000 sensors that provide IDS alerts, firewall logs and honeynet data. Human analysis and data mining is incorporated in order to provide statistics. Zou et al [29] propose the Malware Warning Center that realizes worm detection based on an epidemic propagation model. It is focused on worms that uniformly scan the internet and aims at early detection of worm epidemics. The Internet Malware Analysis System (InMAS) [4] collects malware using honeypots, honeyclients and spamtraps and analyses collected files using CWSandbox. Predefined statistics on collected information can be visualized using a statistic backend. The Internet Analysis System (IAS) [30] collects and visualizes statistics on network packet data in order to support detection of anomalies by humans. eDare (Early Detection, Alert and Response system) [31] and the Agent-based EWS [5] are two further proposals to support early warning. While Agent-EWS basically propose to centrally collect information of different sensors the eDare systems propose to use supervised machine learning techniques for detecting unknown malware. ADWICE [32], which is part of the Safeguard project, uses unsupervised learning to generate a model of normal traffic and performs outlier detection to find anomalies. The Carmentis project [1], advanced by the German CERT association, is

another initiative towards cooperative sharing, central storage and visualization of different kinds of sensor data, which is planned to be extended by correlation techniques in order to automatically generate advice and predictions.

In comparison to these approaches, our system operates on different kinds of information, which are potentially new malware samples (collected, e.g., by Nepenthes), malware behavior (extracted, e.g., by CWSandbox), automatically generated and distributed signatures (generated using machine learning techniques) as well as detection alerts (generated by detection systems using the signatures). Due to the automatic derivation of new signatures and central reporting of occurrences of new malware threats, it forms a basis to generating predictions and advice without incorporation of cognitive abilities of humans and is therefore a large step towards an *Automatic EWS* that automatically contribute to a situation picture.

Accordingly, in this paper we have described the architecture of our automatic EWS and discussed details regarding alternative implementations of its key components. While malware collection and analysis is mainly realized using existing approaches, clustering of malware behavior and generating behavior signatures in the context of our EWS are focus of our research. Enforcing a balance between conflicting confidentiality and availability requirements is another key research challenge of this ongoing project. Based on a discussion of a deployment scenario, interests and requirements of all involved parties are examined. A balance between conflicting interests is proposed and mechanisms appropriate to enforce this balance are mentioned. Besides completing ongoing efforts to implement the EWS components, future work in this project includes the deployment and evaluation of the EWS.

## References

1. Grobauer, B., Mehlau, J., Sander, J.: Carmentis: A co-operative approach towards situation awareness and early warning for the internet. In: Proc. of IMF 2006. LNI, vol. 97, pp. 55–66. GI (2006)
2. DShield: DShield website (2008), <http://www.dshield.org>
3. Network, T.E.C.: The European CSIRT Network Website (2008), <http://www.ecsirt.net/>
4. Engelberth, M., Freiling, F., Göbel, J., Gorecki, C., Holz, T., Trinius, P., Willems, C.: Frühe Warnung durch Beobachten und Verfolgen von bösartiger Software im Deutschen Internet: Das Internet-Malware-Analyse-System (IAS) (in German). In: Sichere Wege in der vernetzten Welt – Tagungsband zum 11. Deutscher IT-Sicherheitskongress (in German), pp. 353–367. SecuMedia Verlag (2009)
5. Bsufka, K., Kroll-Peters, O., Albayrak, S.: Intelligent network-based early warning systems. In: López, J. (ed.) CRITIS 2006. LNCS, vol. 4347, pp. 103–111. Springer, Heidelberg (2006)
6. Gattiker, U.: The Information Security Dictionary. Kluwer, Dordrecht (2004)
7. Biskup, J., Hämmerli, B.M., Meier, M., Schmerl, S., Tölle, J., Vogel, M.: 08102 working group – early warning systems. In: Perspectives Workshop: Network Attack Detection and Defense. Dagstuhl Seminar Proceedings, vol. 08102 (2008)
8. Baecher, P., Koetter, M., Holz, T., Dornseif, M., Freiling, F.: The Nepenthes platform: An efficient approach to collect malware. In: Zamboni, D., Krügel, C. (eds.) RAID 2006. LNCS, vol. 4219, pp. 165–184. Springer, Heidelberg (2006)
9. Amun: Python HoneyPot, <http://amunhoney.sourceforge.net/>
10. Aycock, J.: Computer Viruses and Malware. Springer, Heidelberg (2006)

11. Dullien, T., Rolles, R.: Graph-based comparison of executable objects. In: Proc. of SSTIC 2005 (2005)
12. Rieck, K., Holz, T., Willems, C., Düssel, P., Laskov, P.: Learning and classification of malware behavior. In: Zamboni, D. (ed.) DIMVA 2008. LNCS, vol. 5137, pp. 108–125. Springer, Heidelberg (2008)
13. Willems, C., Holz, T., Freiling, F.: Toward automated dynamic malware analysis using CWSandbox. *IEEE Security & Privacy* 5(2), 32–39 (2007)
14. Lance, G.N., Williams, W.T.: A general theory of classificatory sorting strategies: II. Clustering systems. *The Computer Journal* 10(3), 271–277 (1967)
15. Frigui, H., Krishnapuram, R.: A robust clustering algorithm based on competitive agglomeration and soft rejection of outliers. In: Proc. of Computer Vision and Pattern Recognition, vol. 550. IEEE, Los Alamitos (1996)
16. Lee, T., Mody, J.: Behavioral classification. In: Proc. of EICAR 2006 (2006)
17. Cilibrasi, R., Vitanyi, P.: Clustering by compression. *IEEE Trans. on Information Theory* 51, 1523–1545 (2005)
18. Bailey, M., Oberheide, J., Andersen, J., Mao, Z., Jahanian, F., Nazario, J.: Automated classification and analysis of internet malware. In: Kruegel, C., Lippmann, R., Clark, A. (eds.) RAID 2007. LNCS, vol. 4637, pp. 178–197. Springer, Heidelberg (2007)
19. Cebrian, M., Alfonso, M., Ortega, A.: Common pitfalls using the normalized compression distance. *Comm. in Information and Systems* 5(4), 367–384 (2005)
20. Rieck, K., Laskov, P.: Linear-time computation of similarity measures for sequential data. *Journal of Machine Learning Research* 9, 23–48 (2008)
21. Apel, M., Bockermann, C., Meier, M.: Measuring similarity of malware behavior. In: Proc. of 34th LCN 2009. IEEE Computer Society Press, Los Alamitos (2009)
22. Ukkonen, E.: On-line construction of suffix trees. *Algorithmica* 14(3), 249–260 (1995)
23. Meier, M., Schmerl, S., Koenig, H.: Improving the Efficiency of Misuse Detection. In: Julisch, K., Krügel, C. (eds.) DIMVA 2005. LNCS, vol. 3548, pp. 188–205. Springer, Heidelberg (2005)
24. Flegel, U., Biskup, J.: Requirements of information reductions for cooperating intrusion detection agents. In: Müller, G. (ed.) ETRICS 2006. LNCS, vol. 3995, pp. 466–480. Springer, Heidelberg (2006)
25. Flegel, U.: *Privacy-Respecting Intrusion Detection*. Springer, Heidelberg (2007)
26. MyNetWatchman: MyNetWatchman website (2008), <http://www.mynetwatchman.com>
27. Bailey, M., Cooke, E., Jahanian, F., Nazario, J., Watson, D.: The internet motion sensor - a distributed blackhole monitoring system. In: Proc. of NDSS 2005, The Internet Society, pp. 167–179 (2005)
28. SURFids: SURFids Development Homepage (2008), <http://ids.surfnet.nl>
29. Zou, C., Gao, L., Gong, W., Towsley, D.: Monitoring and early warning for internet worms. In: Proc. of ACM CCS 2003, pp. 190–199 (2003)
30. Waibel, F.: Das Internet-Analyse-System (IAS) als Komponente einer IT-Sicherheitsarchitektur (in German). In: *Sichere Wege in der vernetzten Welt – Tagungsband zum 11. Deutscher IT-Sicherheitskongress* (in German), pp. 281–296. SecuMedia Verlag (2009)
31. Elovici, Y., Shabtai, A., Moskovitch, R., Tahan, G., Glezer, C.: Applying machine learning techniques for detection of malicious code in network traffic. In: Hertzberg, J., Beetz, M., Englert, R. (eds.) KI 2007. LNCS (LNAI), vol. 4667, pp. 44–50. Springer, Heidelberg (2007)
32. Burbeck, K., Nadjm-Therani, S.: Adwice – anomaly detection with real-time incremental clustering. In: Park, C.-s., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 407–424. Springer, Heidelberg (2005)