Electronic Preprint for Journal of Information Processing Vol.26

Regular Paper

Toward Collaborative Defense Across Organizations

Takayuki Sasaki[1,a)] Katsunari Yoshioka[1,2,b)] Tsutomu Matsumoto[1,2,c)]

Received: February 22, 2018, Accepted: September 7, 2018

Abstract: New attack methods, such as new malware and exploits are released every day. Attack information is essential to improve defense mechanisms. However, we can identify barriers against attack information sharing. One barrier is that most targeted organizations do not want to disclose the attack and incident information because they fear negative public relations caused by disclosing incident information. Another barrier is that attack and incident information include confidential information. To address this problem, we propose a confidentiality-preserving collaborative defense architecture that analyzes incident information without disclosing confidential information of the attacked organizations. To avoid disclosure of confidential information, the key features of the proposed architecture are (1) exchange of trained classifiers, e.g., neural networks, that represent abstract information rather than raw attack/incident information and (2) classifier aggregation via ensemble learning to build an accurate classifier using the information of the collaborative organizations. We implement and evaluate an initial prototype of the proposed architecture. The results indicate that the malware classification accuracy improved from 90.4% to 92.2% by aggregating five organization classifiers. We conclude that the proposed architecture is feasible and demonstrates practical performance. We expect that the proposed architecture will facilitate an effective and collaborative response to current attack-defense situations.

Keywords: malware, information sharing

1. Introduction

New attack methods are developed in short cycles to circumvent existing defense mechanisms, which makes malware detection difficult. For example, the availability of the source code for the Mirai bot (malware that targets IoT devices) has led to the creation of numerous Mirai variants. Even if malware source code is unavailable, malware authors often reuse binary code to reduce development burden. Given a large number of malware variants, maintaining antivirus signatures to cover all malware variants is difficult. Moreover, automatic malware creation tools, such as ZeuS, can generate variants by combining malicious functions. In addition, various automated exploit generation techniques [11], [14], [15] have been proposed to accelerate malware development.

To construct a strong defense mechanism, collecting attack information such as on targets and intrusion mechanisms is essential. With such information, we can develop effective countermeasures based on attack features to differentiate attack events from normal events. To construct effective defense mechanisms that cover all attack vectors, organizations should collect and share attack information. To facilitate information exchange, Structured Threat Information eXpression (STIX) [8], which allows organizations to exchange information using a unified format, has been specified. Moreover, an information exchange protocol named TAXII (Trusted Automated eXchange of Indicator Information) [9] has been proposed.

However organizations may not want to share attack information because confidential information could be disclosed. For example, a company that provides antivirus products can use malware executables, infected emails, and attacker activities to improve their products. However, customers may refuse to provide various attack information (e.g., malware executables, emails containing malware, and attacker activities in compromised systems), because such information frequently contains confidential information such as system architectures and configurations. Antivirus companies are also reluctant to share attack information with other security companies because such information is used to differentiate their products. However, widespread attacks can cause significant damage, which has become a critical social problem; thus, such entities should collaborate to address this issue.

Some organizational approaches to sharing attack information without disclosing confidential information have been developed. For example, the Information Sharing and Analysis Center (ISAC) collects information about cyber infrastructure attacks and provides a mechanism to share the information without revealing confidential information. Note that information shared among companies, such as antivirus security companies, is generally protected by nondisclosure agreements. In this paper, we address technical approaches and examine how security companies can protect customers without using customer malware information. Some technical approaches have been developed. For example, VirusTotal (a free online virus, malware, and URL scanning service) uses more than 40 virus engines to analyze user-submitted files and websites. VirusTotal also provides samples

[1] Graduate School of Environment and Information Sciences, Yokohama National University, Yokohama, Kanagawa 240–8501, Japan
[2] Institute of Advanced Sciences, Yokohama National University, Yokohama, Kanagawa 240–8501, Japan
[a)] sasaki-takayuki-kx@ynu.jp
[b)] yoshioka@ynu.jp
[c)] tsutomu@ynu.jp

of the submitted malware, which encourages information sharing. However, the submitted information is shared with VirusTotal members, many of whom are antivirus vendors. Thus, if the submitted information contains confidential information, e.g., Microsoft Word documents about company projects, disclosure of such information would represent a leak. To avoid such risks, users often only submit the hash values of suspicious files [*1]. Unfortunately, by themselves, hash values are insufficient to construct defense mechanisms. To promote attack information sharing, we propose a collaborative defense architecture that does not require confidential attack information. The proposed architecture solves the following research problems, i.e., how to exchange attack information without disclosing confidential information and how to aggregate information provided by multiple organizations to develop defense mechanisms based on that information.

To address these problems, the proposed architecture exchanges trained classifiers rather than raw attack information, and aggregates classifiers via ensemble learning. The proposed architecture has the following features.

(1) The proposed architecture employs a mechanism that abstracts information, which is then used to train a classifier. The trained classifier's parameters are then sent to an analysis organization to aggregate the information. Attack information is abstracted from raw information as classifier parameters; thus, the proposed architecture does not disclose confidential information of attacks against specific organizations.
(2) The analysis organization aggregates the classifier via ensemble learning to cover all attack vectors observed by attacked organizations.

In addition, we propose a metric to measure the degree of information disclosure caused by exchanging classifier parameters. The proposed metric is similar to differential privacy which is a concept used in privacy-preserving data mining. To demonstrate a practical application of the proposed architecture, we implemented and evaluated an initial prototype using the scikit-learn (a Python-based machine learning tool) to train and aggregate classifiers. The results show that classification accuracy improved from 90.4% to 92.2% by aggregating classifiers from five organizations. However, compared to a method without confidentiality preservation, the accuracy degrades by 3.3% due to our confidential-preserving mechanism. In addition we demonstrate that our initial implementation is practical and scalable.

Our primary contributions are summarized as follows.

- We propose an architecture that shares attack information without disclosing confidential information. In addition, we construct an integrated classifier that covers all attack vectors observed by the attacked organizations.
- We propose metrics to measure the degree of information disclosure caused by exchanging the classifier.
- We implement a prototype and demonstrate that the proposed architecture is feasible and practical.

The remainder of this paper is organized as follows. First, the problem statement is given in Section 2. In Sections 3 and 4, we respectively describe core concepts and the proposed architecture. In Section 5, we propose metrics to measure the degree of information disclosure caused by exchanging classifiers. We describe an initial implementation in Section 6 and evaluate its performance in Section 7. The limitations of the proposed architecture are discussed in Section 8, related work is reviewed in Section 9, and conclusions are given in Section 10.

[*1] VirusTotal offers virus check based on the hash values of executables.

2. Problem statement

Our goal is to build an analysis platform that can leverage security information without disclosing confidential information. Here, we first define information provider and analysis organization and we describe realistic situations that require sharing attack information. Next, we define a trust model between the organizations. We also identify the requirements of a system that would allow collaboration between the organizations.

2.1 Roles of organizations and situations

Information provider organizations possess information about the attacks such as malware executables and their activity logs, phishing emails, etc. **An analysis organization** examines the attack information and develops security measures such as detection and protection mechanisms. In the following, we describe two realistic situations.

Information sharing between security company and customers. Collecting malware information is an important task for security companies because antivirus signatures are generated by analyzing malware executables. In addition, malicious IP addresses for communicating with command and control (C&C) servers can be identified through dynamic analysis of malware communication methods. Thus, companies that provide antivirus products want to use customer information to improve the signatures of their antivirus detection engines. However, some organizations such as government and military organizations refuse to share attack information because confidential information could be revealed. Moreover, many security companies sell products globally and it is unknown whether an organization in one country can disclose malware information to a security company in another country. Moreover, communication logs are likely to contain data stolen from the organizations.

In this case, the analysis organization corresponds to antivirus vendor, and the information provider organizations correspond to its customers.

Information sharing among security companies. Security companies are reluctant to share information because relative to generating antivirus signatures, such information provides a competitive advantage and sharing information could impact profit margins. However, security companies must collaborate when an attack causes serious damage to many systems. For example, Japanese security companies Internet service providers, and the government have collaborated in the Cyber Clean Center project to identify malicious bots [1].

In this case, the analysis organization corresponds to a collaborative organization such as ISAC, and the information provider organizations correspond to security vendors.

2.2 Trust model

Here, we define a trust model between the information provider and analysis organization. We assume that information provider organizations do not want to disclose confidential information but they are motivated to contribute to the analysis organization. For example, improving a security product benefits both the company and its customers. Note that we make the following assumptions relative to this trust model. We assume that information provider organizations are honest and do not provide false information. We also assume that the analysis organization is honest and does not intentionally attempt to obtain an information provider's confidential information.

We further assume that the communication channels between the analysis and information providing organizations are encrypted using SSL/TLS or other secure protocols. We do not consider information leaks caused by a third party, e.g., an adversary who steals an organization's information. In addition, we do not consider the possibility that there may be malicious insiders in both organizations.

2.3 Requirements

The overall goal of this study is to provide a system that satisfies the following requirements.

Requirement 1. Information provider organizations should be able to share attack information with an analysis organization without disclosing confidential information. Confidential information could be revealed in the analysis of malware executable, communications between malware and C&C servers, and malware activity logs that may contain information about the target systems. For example, Stuxnet can automate an attack because the network of a SCADA system for uranium enrichment is isolated from the Internet. To attack such an isolated system, the automation function would contain an internal structure of the infrastructure. Note that different types of confidential information and these examples are discussed in Section 2.4.

Requirement 2. The analysis organization should be able to aggregate information provided by information provider organizations and develop a countermeasure based on the aggregated information. The countermeasure should cover all attack vectors observed by information provider organizations.

Requirement 3. To reduce attack analysis costs, the system should perform the above tasks automatically.

2.4 Example of confidential information used for defense mechanisms

Here, we discuss confidential information used for defense mechanisms.

2.4.1 Confidential information shared between security company and customers

Here, we consider malware targeting general victims and malware targeting a specific organization (e.g., a spear attack). To improve the likelihood of success, the malware would contain organization-specific information, such as the following.

- The malware would contain information of the victim organizations. For example, an author of Stuxnet studied target systems comprising computers and a SCADA system including PLC. To attack systems isolated from the Internet, Stuxnet has an attack automation function based on the actual system design. Obviously, sharing such malware discloses the targeted system's information.
- Malware activity logs could include system architecture information because malware typically scans the infected system's internal network to facilitate lateral movement. The activity log may also include file names and content, which could reveal internal organization information. System administrators and incident response team members could remove confidential information from malware activity logs, however this requires significant time and effort.
- Infected emails contain organizational information, such as divisions and teams names. Moreover, they include the recipient's name and affiliation. Recently, there have been sophisticated targeted attacks where the attacker exchanged several emails with the victim to build a relationship of trust. Once a relationship of trust is established, the attacker sends an email with malware. In this case, the body of the email contains information about the victim's organization.

In the non-spear type attacks, e.g., malware against general targets, the malware does not contain confidential information. However, typically, information provider organizations do not have security experts; thus, they cannot differentiate general malware attacks and attacks that target a specific organization. Thus, they cannot determine whether the malware information should be shared. Therefore, a confidentiality-preserving method is required for both types of attacks.

2.4.2 Confidential information shared among security companies

In this case, the malware itself is the confidential information. Antivirus vendors want to improve their products' detection ratio. Thus, malware information that directly affects the detection ratio is important. Moreover, most antivirus vendors invest in malware collection systems. For example, vendors offer online virus submission forms and their antivirus products include a function to upload suspicious programs [5], [6], [7]. Thus sharing attack/malware information with a competitor would not be advantageous and could reduce their return on investment.

3. Core concepts

In this section, we propose core concepts considered to satisfy the requirements. For information sharing (Requirement 1), inspired by privacy protection techniques, we propose a concept to abstract information to be shared. To aggregate the organization's attack information (Requirement 2), we propose a concept that leverages ensemble learning techniques.

3.1 Confidentiality-preserving information sharing

We can identify a similar problem in the privacy protection field, e.g., privacy-preserving data mining. In this case, an organization is motivated to utilize privacy data to improve services. However, such information must be protected and not revealed outside the organization. To ensure privacy protection, the information is made anonymous prior to sharing.

K-anonymity [34] ensures that k-persons have the same at-

tribute set; thus, it is not possible to identify a specific person in the attribute set. Prior to sharing privacy data, the database owner must anonymize the data to ensure that the k-anonymity degree is greater than a given threshold.

Differential privacy [17] has been proposed to control privacy disclosure when querying a database. The concept of differential privacy is that, if the data are relatively common, the privacy-degree of the data is low. For example, if an analyst sends a query to the two independent databases and the results are the same, it is assumed that the results do not contain privacy information, otherwise, the database is considered to have disclosed privacy information.

We can adopt a similar approach to share information between the information provider and the analysis organizations, wherein we send the information provider's malware information to the analysis organization via an information abstraction process.

Concept 1. Inspired by privacy-preserving techniques, we propose the following system model.

- The analysis organization sends an analysis template that corresponds to a query in differential privacy to information provider organizations.
- Information provider organizations use their attack information to train classifiers and return the classifiers to the analysis organization. Here, the analysis results are abstracted as trained classifiers, thus the results do not contain confidential information. This analysis step corresponds to anonymization (noise addition) in differential privacy and k-anonymity approaches.

3.2 Collaborative defense mechanism using ensemble learning

Ensemble learning is a technique to build a classifier from weak classifiers by aggregating their decisions. Bootstrap aggregation (bagging) employs majority voting of classifiers, each of which is trained using different dataset. Bagging reduces bias-variance caused by bias in the training datasets. In a discrete case (classification case), outputs are calculated based on majority voting by the classifiers. In a non-discrete case (regression case), bagging uses the average values of the classifiers as follows.

$$H(x) = \frac{1}{T}\sum h_t(x) \quad (1)$$

where T is the number of classifiers and $h_t(x)$ is the decision of classifier h_t.

The bagging dataset is generated by bootstrap sampling which selects data randomly from the entire dataset to create sub-datasets. Note that bootstrap sampling allows overlap between the sub datasets created from the entire dataset.

Concept 2. Each information provider organization has its own malware dataset that is part of a global malware dataset. Therefore, this is similar to bootstrap sampling. Thus, we employ the bagging technique to combine the output of the classifiers trained in consideration of concept 1.

4. Architecture

Based on the above concepts, we design the proposed architecture to enable collaborative analysis. Various types of attack

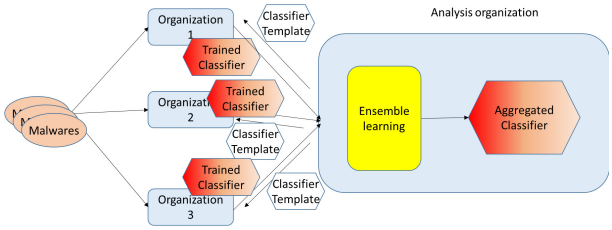

Fig. 1 Architecture for collaborative malware detection.

analysis have been proposed such as malware analysis, incident analysis (forensic), and adversary analysis (investigation of attackers). Note that each type of analysis requires different techniques. However, we focus on malware detection in this paper and we embody the concepts as an architecture by performing the following steps (**Fig. 1**).

- The analysis organization sends a non-trained classifier as a template to the information provider organizations (Section 4.1.1).
- Each information provider organization trains the classifier and returns the trained classifier to the analysis organization. Note that the information provider organization provides the trained classifier rather than raw attack information, which is essentially confidential information (Sections 4.1.2 and 4.1.3).
- The analysis organization builds a combined classifier from the classifiers trained by the information provider organizations (Section 4.1.4).

Here, we discuss two problems, i.e., malware detection to determine if a binary is malware or a normal application and malware family classification when a malicious executable is given.

Malware detection. This is a two-class classification problem to determine whether a given executable belongs to a set of normal/legitimate software or a set of malware. This problem is formalized as outlier detection that models the normal behavior of a system and considers anomalies as attacks.

Note that the proposed architecture does not fit this problem because what constitutes normal behavior differs between organizations. For example, each organization has its own CRM, web servers, etc., so normal behavior cannot be defined across different organizations. Therefore, it is expected that a classifier trained by an organization would mistakenly recognize the normal behaviors of other organizations as anomalies, thereby causing false positives. Thus, in this paper we focus on the malware family classification problem.

Malware family classification. This is a multi-class classification problem, where a classifier identifies a malware family from multiple malware families when a malware is given.

Adversaries develop many malware variants for various reasons, e.g., to update functionality, fix bugs, or reuse code from different malware authors. Malware source code is sometimes disclosed. For example, the Mirai source code was made open-source [4], and attackers can use this code to create their own malware. Moreover, even if the source code is not disclosed, attackers can extract part of an executable and merge the code into their own malware. Malware family classification helps security engineers develop countermeasures because an effective protec-

List 1: Sample template

```
{
  "features":[
    "pe_imports",
    "file_read"
  ],
  "classifier":{
    "name": "SVM",
    "kernel": "rbf"
  }
}
```

tion method would be common for variants of a given malware family.

In the following, we focus on the malware family classification problem.

4.1 Details of the proposed architecture

Here, we detail the steps of the proposed architecture.

4.1.1 Classifier template distribution

First, the analysis organization distributes a template that specifies a method to extract features from malware. The template also specifies a classifier and its pre-defined parameters. The template comprises the following:

- A feature extraction template specifies static or dynamic analysis and its parameters to extract features.
- A classifier template specifies the classifier type. It also specifies classifier's parameters. For example, with a neural network, the classifier template specifies the number of layers, the number of units in the layers, and the learning rate.

List 1 shows an example template. It specifies imported libraries (pe_import) and files read by the malware (file_read) as features. The template also specifies a support vector machine (SVM) as the classifier and a classifier parameter indicating the use of a Gaussian kernel (rbf).

4.1.2 Feature extraction

Feature extraction is performed based on static and/or dynamic analysis. The feature extraction logic is predefined and the program used to extract features is distributed to the information provider organization from the analysis organization in advance.

Note that we employ existing feature extraction methods [10] and do not claim to make any contributions relative to feature extraction. To extract features by static analysis, we leverage the general features of an executable. For example, we can use statistics of API calls, disassembled order sets, call graphs, control flows, and executable metadata (file size, ELF/PE header information, etc.). For the dynamic analysis, we can also use general features such as API calls, file accesses, and network access. Ahmadi et al. have summarized various features [10], and we can use such features as follows.

There are two types of features generated by static analysis. One type is byte-code level features that are extracted directly from the malware. The other type is operation code (op-code) features which are extracted using a disassembler and have more semantic meaning compared to byte-code level features

Byte code level features.

- N-gram of binary that counts the number of specific patterns. In a case where $N = 1$ (i.e., a 1-byte monogram), the input is a vector with 2^8 dimensions, each of which shows the number of patterns in the binary.
- File metadata (file size, an address of first executable code, etc.).
- Image generated from byte code. Ahmadi et al. have proposed a technique to convert a binary executable to a gray-scale figure which is then used as the input for the image classification methods.

Op-code level features. Disassemble code (op-code) can also be used as a classifier features. A previous study [10] introduced the following features.

- Counts of op-codes (op-code distribution) such as the number of arithmetic operations and control operations (cmp&jmp, call, etc.). These features are particularly promising for malware family classification because malware in a family has similar op-code distribution due to code reuse.
- Counts of system calls (e.g., exec, fork, read, and write calls). The events of API calls such as libc and libssl can also be used as input features.
- Call graph and control flow which represent how a program calls subroutines, functions, and external code. Differ from op-code counts and system calls, call graphs and control flows reflect the overall structure of a program because it is expressed as a graph.
- Constant values and strings used in executables. Malware often contains unique features, e.g., an encryption key, the URLs of C&C servers, and command lists that the malware supports. Moreover, file names and registry keys are hard-coded as constant strings.

Dynamic analysis. We also use features collected by running the malware and monitoring the malware's behavior.

- Access events on local computer resources such as files and processes (IPC).
- Access events on network resources in HTTP, DNS, and IRC communications.

In the proposed architecture, there is no limitation relative to feature extraction, thus we should select the best features. For example, the winner of a malware classification challenge [3] used byte-code counts (2-, 3-, and 4-gram), segment counts, single byte frequencies, function names, etc.

4.1.3 Training classifier

After extracting features, each information provider organization trains a classifier according to the template. The organization sets up the classifier such as a neural network, SVM, or decision tree, and also sets learning parameters according to the template. Here we assume periodic exchange of the classifier at interval T, e.g., weekly and monthly. Each organization trains the classifier using newly captured malware in period T.

4.1.4 Classifier aggregation

Once the analysis organization receives trained classifiers from the information provider organizations, the analysis organization combines the classifiers using ensemble learning techniques. Some ensemble learning techniques calculate the best weights of the classifiers to minimize error in the combined result by

weighted average. However, to calculate weights, the analysis organization must possess training data. Therefore, the proposed architecture performs majority voting and does not employ the weighted average.

To perform majority voting, malware names must be shared among organizations. We discuss this issue in the Appendix.

5. Information disclosure metric

Here, we investigate information disclosure caused by exchanging trained classifiers. Specifically, we answer the following fundamental questions about the proposed architecture. *How do classifiers disclose information? How can we formalize and measure the disclosed information caused by exchanging the classifier? How can we mitigate or reduce information disclosure by exchanging classifier?*

To answer these questions, we categorize the possibilities of information disclosure into three cases: classifier output, i.e., hyperplane in the feature vector space (Section 5.1); classifier parameters (Section 5.2); and reconstruction of classifier inputs (Section 5.3). Then, we discuss the above questions relative to each case.

5.1 Classifier output

As discussed in Section 4, a similar situation can be found in differential privacy cases. Specifically, *a query* and *its result* in differential privacy respectively correspond to *a classifier template* and *a trained classifier* in the proposed architecture. Here, we define a metric based on differential privacy concept. Differential privacy is described using the following equation.

$$\ln(Pr(K(D1) = x) - \ln(Pr(K(D2) = x) \leq \epsilon \quad (2)$$

where $Pr()$ denotes a probability density function, and D1 and D2 denote databases whose one entry differs from another database. $K()$ represents database outputs modified by a privacy protection mechanism, and x denotes the output of the database. ϵ is a threshold that specifies the degree of privacy information disclosure. Inspired by differential privacy, we propose a metric to describe the degree of information disclosure by a classifier.

$$\ln(C1(x)) - \ln(C2(x)) \leq \epsilon \quad (3)$$

Here, x is an input vector of a classifier and $C(x)$ is an output of a classifier considered as a probability density. Specifically, $C1$ is a classifier trained by an organization and $C2$ represents converged classifiers. This equation means that if the trained classifier is similar to another classifier, it is assumed that the classifier does not contain much confidential information. Of course, the confidential protection degree (a low threshold is better) and usefulness of the classifier (a high threshold is better) form a trade-off relationship because a useful classifier will necessary contain unique information.

Note that there are two options if the trained classifier does not meet the threshold. One is giving up disclosing the classifier, and the other is adding noise, which is similar to differential privacy adding Laplace noise. However, this noise introduces false positives and false negatives, thus we give up disclosing the classifier over the threshold.

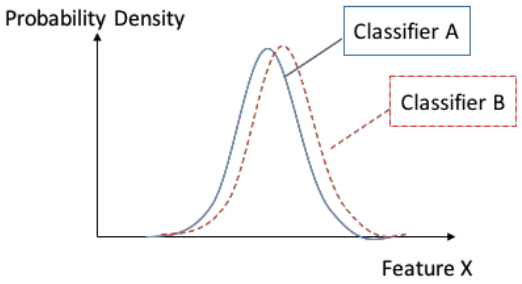

Fig. 2 Differential-privacy like metric.

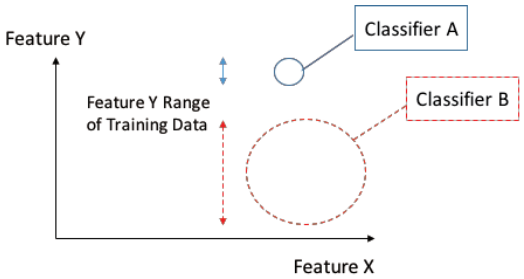

Fig. 3 L-divercity like metric.

There is another metric in the study of privacy, i.e., L-diversity. L-diversity expresses the number of attribute candidates when a person in a group is given. For example, if a person belongs to a group of North America citizens, the address for the person can be expected to be in Canada, the U.S.A, or Mexico. An area inside a hyperplane in the proposed architecture corresponds to a group in L-diversity (**Fig. 3**). A small hyperspace means that the classier was trained using malware with the specific feature values and a large hyperspace means that there are many possible feature values in the training data. Therefore we can define the volume of the hyperspace as an information disclosure degree. In the example of Fig. 3, classifier A discloses specific malware features compared to classifier B.

The above metrics work when the feature space is not large. In **Figs. 2** and 3, there are one or two features; thus we can evaluate the differences of the two classifiers for the first metric and the volume of the hyperplane for the second metric by making a grid search over the entire feature space. However, for a large feature space, we cannot perform a grid search due to the large search space. Note that metrics for a large feature space will be the focus of future space.

5.2 Classifier parameters

In some types of classifiers, training data can be inferred from the classifier's parameters. For example, a decision tree, which comprises automatically generated if-then rules, can reveal training data because the if-then rules represent human-understandable logic that separate the hyperspace using a threshold of the features. Specifically, we assume the following decision logic.

> If an executable contains the "sales division" string, then if the potential malware contains "John Smith", it is considered malware.

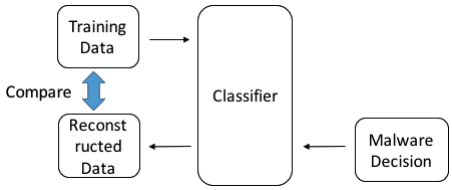

Fig. 4 Feature reconstruct.

This decision tree would disclose that there is an employee named John Smith in a sales division.

A Bayesian filter would also disclose the training data. A Bayesian filter is based on a probabilistic distribution of words in a document and it calculates the total score from each word score. Therefore, a Bayesian filter would contain specific words that are confidential. Specifically, a likelihood is calculated using $\prod P(w_i|S)$ that means the probability of word w_i when document S is given. This parameter discloses feature w_i.

Note that we do not have a clear mitigation method for this problem. We recommend complex classifiers such as multi layer neural network rather than classifiers that disclose decision logic (i.e., decision tree and a Bayesian filter).

5.3 Reconstruct training data from trained classifier

DCGAN [31] has shown that the human face can be regenerated/reconstructed from compressed feature vectors, which means that the classifier (an autoencoder in DCGAN), would be able to regenerate original training data from the likelihood of a malware. In addition to the deep neural network, a Bayesian network would infer input from its output in a probabilistic manner.

To measure this type of information disclosure, we define a metric using the similarity between the training data (original attack information) and the reconstructed input from the classifier (**Fig. 4**). Similarity can be defined by cosine similarity, edit distance, and Jaccard distance, and if the similarity is less than threshold ϵ, the trained classifier is discarded and not disclosed.

6. Implementation

Here, we discuss prototype implementation.

6.1 Feature extraction and classifier training

For the classifier training, we used scikit-learn 1.8 and its SVM and neural network packages. We wrote a Python script that reads a template file and loads specified features from a dataset. Then, the script imports the classifier package specified by the template and trains the classifier using these features.

6.2 Aggregating trained classifiers

For simple implementation, we employ a majority voting function to ensemble classifiers. We input the same test data into the classifiers and selected the label predicted by the greatest number of classifiers.

7. Evaluation

To evaluate the proposed architecture, we answer the following research questions.

- Is classification accuracy improved by classifier aggregation? To answer this question, we compared accuracy by changing the number of classifiers.
- How much does detection performance degrade due to the confidentiality-preserving approach? To answer this question we compared the detection ratios of the proposed method and a directly trained classifier that does not preserve malware information.
- Is the proposed method scalable? To answer this question, we measured the classification processing time by varying the number of organizations and the sample size.

7.1 Classification accuracy

To evaluate classification accuracy, we used a malware set [2] [*2] that contains 292 APT1 samples, 434 Locker Samples, and 2,014 Zeus samples. We evaluated classification accuracy by varying the number of information provider organizations. Here, we assumed five information provider organizations, where each organization had 1/5 of the total number of malware samples in the training samples. The samples were shuffled randomly and divided into five groups for five organizations. Then, we evaluated the performance of one classifier corresponding to one organization, an aggregated three classifiers corresponding to a collaboration of three organizations, and an aggregated five classifiers corresponding to a collaboration of five organizations. We also evaluated a classifier trained using the full dataset without confidentiality preservation, which corresponds to a situation where the analysis organization collects raw malware information and trains the classifier using those data.

For feature extraction, we used the pe_imoprt feature, which identifies the dynamic link libraries imported by a malware. We selected well-used libraries in the dataset and converted the feature into a vector using one-hot encoding. Here, we selected libraries used more than 20 times and the number of libraries (i.e., the length of the feature vector) was 5,439.

With the above dataset, we evaluated the accuracy using the SVM and neural network provided by scikit-learn library. For the SVM, we used the svm.SVC class. Note that an SVM with an RBF kernel only has two parameters, i.e., penalty parameter C and kernel coefficient γ; therefore, we performed a grid search to find the best parameters. For the penalty parameter C, we searched 1, 10, 100, and 1,000, and for the kernel coefficient γ, we searched 0.1, 0.01, 0.001, and 0.0001. We selected penalty parameter $C = 100$ and kernel coefficient $\gamma = 0.01$, which produce the best accuracy. For the neural network, we used the MLPClassifier class. A neural network involves many parameters, such as the number of layers, the neuron size of each layer, and the penalty; thus, a grid search is impractical. Therefore, we used the default parameter values of the MLPClassifier class. To help the convergence, we set the maximum number of iterations to 100,000.

Tables 1 and **2** respectively show the confusion matrix of the SVM and neural network. Here, we measured 10 times and calculated the average. For each measurement we reshuffled the samples by changing the random seed and created five

[*2] This dataset also contains Locker and other malware, but we use top 3 families in the dataset for simple evaluation.

Table 1 Confusion matrix (SVM).

(a) Confusion matrix of full data set (Accuracy=0.955)

	Apt1	Crypto	Zeus
Apt1	137.1	3.5	6
Crypto	2.3	182.3	29.2
Zeus	2.0	16.5	939.1

(b) Confusion matrix of 1 organization classifier (Accuracy=0.904)

	Apt1	Crypto	Zeus
Apt1	123.4	3.4	19.8
Crypto	4.0	127.6	82.2
Zeus	2.8	14.4	940.4

(c) Confusion matrix of aggregated 3 organization classifiers (Accuracy=0.916)

	Apt1	Crypto	Zeus
Apt1	127.3	2.5	16.8
Crypto	5.0	131.9	76.9
Zeus	0.6	8.5	948.5

(d) Confusion matrix of aggregated 5 organization classifiers (Accuracy=0.922)

	Apt1	Crypto	Zeus
Apt1	127.2	3.3	16.1
Crypto	3.4	142.7	67.7
Zeus	0.6	11.9	945.1

Table 2 Confusion matrix (Neural network).

(a) Confusion matrix of full data set (Accuracy=0.958)

	Apt1	Crypto	Zeus
Apt1	138.2	2.4	6.0
Crypto	1.5	179.0	33.3
Zeus	1.7	10.6	945.3

(b) Confusion matrix of 1 organization classifier (Accuracy=0.915)

	Apt1	Crypto	Zeus
Apt1	127.4	2.9	16.3
Crypto	5.1	136.0	72.7
Zeus	2.6	11.9	943.1

(c) Confusion matrix of aggregated 3 organization classifiers (Accuracy=0.927)

	Apt1	Crypto	Zeus
Apt1	129.7	2.3	14.6
Crypto	6.0	144.4	63.4
Zeus	0.7	8.8	948.1

(d) Confusion matrix of aggregated 5 organization classifiers (Accuracy=0.929)

	Apt1	Crypto	Zeus
Apt1	128.9	2.7	15.0
Crypto	4.2	147.5	62.1
Zeus	0.9	9.1	947.6

groups. In Tables 1 and 2, each row shows the classifier decisions and each column shows the ground truth provided by the dataset. Therefore, diagonal elements represent the number of correct classifications and non-diagonal elements represent incorrect classifications. We also show accuracy which is calculated as $\frac{\text{TrueClassifications}}{\text{TrueClassificatons+FalseClassifications}}$.

The results demonstrate that classification accuracy is improved when information from multiple information provider organizations is used. Thus, our collaborative mechanism with ensemble learning improves the accuracy. For example, as shown in Table 1, the classifiers accuracy was 0.904, 0.916, and 0.922 when using the data of one, three, and five organizations, respectively. However, the classification accuracy of the classifier trained using the full dataset without confidentiality preservation was 0.955, which is better than that of the aggregated classifiers. This difference in accuracy is a drawback caused by our confidentiality-preserving techniques. Nonetheless, we conclude that the proposed architecture works and provides better classification capability to the analysis organization.

In the above evaluation, we assumed three and five organizations and three malware families. Relative to the number of organizations, larger numbers are better for accuracy. However, the evaluation results demonstrate that the proposed approach is effective even with three organizations. Relative to the number of malware families, it is expected that classification accuracy deteriorates as the number of malware families becomes larger. This is a general problem with machine learning-based classification and will serve as a focus for future work.

7.2 Scalability

To evaluate the scalability of the proposed architecture, we measured the processing time of classification at the analysis organization.

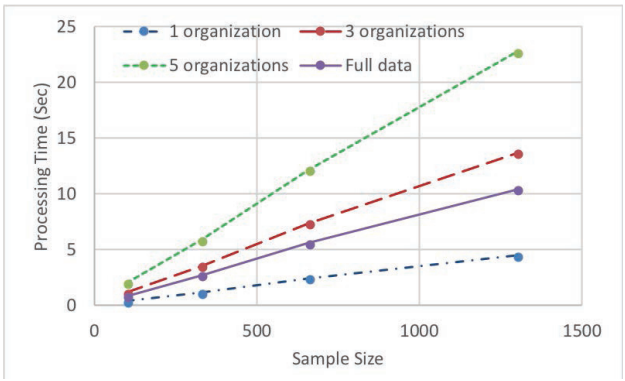

Fig. 5 Processing time.

In this evaluation, we examined cases with one, three, and five organizations, as well as a case without confidentiality preservation. In these four cases, we varied the sample size from 100 to 1,300 and measured the processing time. Here, we used a VirtualBox VM with 4 GB memory and two assigned CPUs (i5-6360U, 2.00 GHz). We also used Ubuntu 16.04 and the SVM classifier provided by scikit-learn 1.8. We measured processing 10 ten times and calculated the average.

Figure 5 shows the results. The processing times of all cases are proportional to the number of samples to be classified. The processing time is also proportional to the number of organizations. The reason for this in the case of one organization is that only a classifier is used, however in the multiple-organization case, multiple classifiers must be used. Using the full data without confidentiality preservation, there is only a single classifier, however the processing time is longer than that of the single orga-

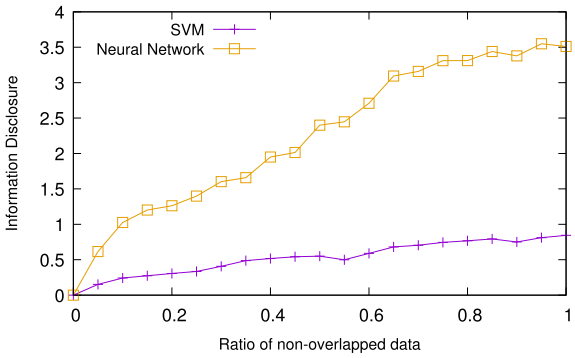

Fig. 6 Information disclosure degree.

nization case, which also uses a single classifier. Here, we expect that the hyperplane of the classifier becomes complex. In other words, the number of support vectors becomes large compared to the single organization case due to the difference in dataset size.

The above results demonstrate that the processing time is proportional to sample size and the number of organizations. To reduce processing time, we can adopt a multi-processing strategy because the classifications of each sample are independent of each other. Therefore, we conclude that the proposed architecture is scalable and works even if sample size or number of organizations is large.

In the above evaluation, we assumed the sample sizes of each organization were equal, however in real-world situations, there are differences in such sample sizes. We expect that the impact of such deviations would be small because processing only requires a few dozen seconds in the above evaluation. Even if the sample size were to increase by two orders of magnitude, the classification processing of all malware would finish in one hour.

7.3 Information leakage metric

Next, we evaluated the information disclosure caused by classifier exchange. **Figure 6** shows the information disclosure degrees calculated based on differential privacy-like metric (Eq. (3); Section 5.1). We calculated the degree of information disclosure using the following equation.

$$\frac{\sum_{x \in X} \sum_{i \in I} \mid \ln(C1_i(x)) - \ln(C2_i(x)) \mid}{\mid X \mid \mid I \mid} \tag{4}$$

Here, X is the test dataset and I represents the classes (APT1, Crypto, and Zeus). We evaluated the exchanges of the SVM and neural network by changing the training dataset. In Fig. 6, the x-axis of shows the overlap of the training dataset (0 means two classifiers were trained using the same data; 1 means the classifiers were trained using a completely different dataset). The results show that a dataset with unique data results in a greater degree of information disclosure.

Here, the meaning of the disclosure degree value remains a problem. For example, what does "0.5" disclosure mean? Developing a method to determine a proper information disclosure threshold will be the focus of future work.

8. Discussion

8.1 Applicability to other attack information

In the previous section, we focused on the malware classification using malware features, however attack information is not limited to such malware features. Therefore, here, we discuss the applicability of the proposed architecture to other types of attack information. The proposed architecture shares classifiers; thus, theoretically, it can be applied to a defense mechanism using the classifiers. We identify the following use cases that the proposed architecture can be applied to.

Phishing email detection. Classifiers are also used to determine phishing emails [18], [38]. In this case, the attack information is the content of the body of an email, such as links and words. We can aggregate classifiers that learn phishing emails received by each organization.

Malicious HTTP request detection. Methods to distinguish normal and application layer attacks have been proposed previously [25], [29]. These methods use SVM classifiers to learn the features of HTTP Get requests, such as keywords in the requests, which correspond to attack information. By aggregating classifiers, organizations can efficiently differentiate between malicious and legitimate requests.

Insider threat detection. Attacks can come from internal sources, such as malicious employees. Mayhew et al. proposed a method to detect insider threats using machine learning [26]. They use email exchanges and HTTP requests as features, which correspond to attack information, and these features can contain an organization's confidential information. The proposed architecture contributes to sharing such information without disclosing raw information.

8.2 Exchanging classifier vs exchanging input features of classifiers

Features extracted from attack data are abstracted and do not contain the original data, however, features sometimes can disclose part of the original data. For example, if the feature is a string in the malware binary, it may contain target information about hosts/servers. Moreover, some classifiers such as Bayesian network must disclose the original data, e.g., the words in an email that carries a malware. Therefore, we exchange classifiers rather than input features.

8.3 Packed and evasive malware

There are some techniques to block malware analysis: malware packing and evasion techniques. Such techniques disturb achievement of Requirement 3 (automatic analysis) defined in Section 2.3.

Malware packing is an obfuscation technique that encodes a malware executable to avoid detection by antivirus scan engines. Specifically, packed malware is encoded, and at the beginning of malware execution, the unpack routine decodes the malware executable and invokes the malware body. There are many types of packers such as UPX, NsPack, and ASpack, which make automated unpacking difficult. However, we can use existing automated unpacking techniques [33] to address this issue. Even if automated unpacking is unavailable, we can employ a dynamic analysis that does not require unpacking. In this case, the information provider organization only sends a classifier trained using dynamic analysis features.

Evasive malware is also a problem relative to automated feature extraction. To evade malware analysis, recent malware includes a sandbox detection function and the malware terminates if it detects a sandbox [24], [36], [39]. For example, the malware checks various hardware features such as NIC devices which are often emulated. To handle this type of malware, we can use bare-metal analysis [22] which does not require a sandbox. Another solution is nEther [30]. To achieve high transparency, nEther provides out-of-the-guest malware analysis using hardware virtualization.

8.4 Anonymity of information provider organization

The proposed architecture does not disclose attack information, however it reveals who joins a collaboration. Some organizations do not want to disclose the source of the attack information in addition to the attack information itself. To satisfy this requirement, the proposed architecture could employ anonymization mechanisms to remove linkability between the organization and attack data to ensure pseudonymity. We can use existing techniques [23] to achieve this. However this topic is beyond the scope of this paper.

8.5 Confidentiality vs Usefulness

In privacy protection, the privacy and usefulness of data form a trade-off relationship. A strong differential privacy protection threshold makes the database outputs the same. We can find the same trade-off in our confidentiality-preserving mechanism. For example, when all information provider organizations require strong confidentiality thresholds, all trained classifiers become the same.

Obviously, the best value for this threshold depends on the relationship between the information providing and analysis organizations. For example, between a security vendor and its customers, the threshold can be loosely based on a loose agreement. On the other hand, between security companies in a competitive relationship, the threshold should be strict.

In the future, to determine the best confidential-preserving threshold value, we will investigate actual use cases involving the proposed architecture in order to propose best practices for various situations.

8.6 Confidentiality of trained classifier

There is an argument about whether the trained classifier itself is in fact confidential. Our answer is “no” because the proposed architecture measures the degree of information disclosure and shares the classifier only if the degree of disclosure is below a certain threshold.

8.7 Application to VirusTotal model

Here, we discuss application of the proposed architecture to a VirusTotal information sharing model.

VirusTotal provides two services; One is a virus scan service, where users upload suspicious files to VirusTotal, which then scans the files using and more than 40 virus scan engines. Note that this service is free and accessible by anyone. The other service is information sharing for antivirus vendors and security companies. The VirusTotal “About” page states “Files and URLs sent to VirusTotal will be shared with antivirus vendors and security companies so as to help them in improving their services and products.”

Currently, VirusTotal users do not seem overly concerned about confidential information in the uploaded files or do not know that the uploaded files are shared even though information sharing is identified on the VirusTotal submission page. Survey studies about VirusTotal [13], [27] have reported many documents are used as decoys, which may contain confidential information about the target organizations. It was reported that the decoy documents contain data related to the business of the target organizations that can be used to attract the document recipient.

We do not expect the VirusTotal model to continue being used in the future due to document confidentiality concerns. In fact, some organizations do not allow employees to use VirusTotal, therefore, we must investigate other information sharing models.

In the VirusTotal information sharing model, the proposed architecture can be deployed between VirusTotal and its customers. In this context, VirusTotal corresponds to the information provider organization and the customers correspond to the analysis organizations. In this model, users upload confidential information to VirusTotal; however VirusTotal does not disclose the confidential information to their customers because it provides the classifiers to its customers. The customers can aggregate their classifier and the classifier given by VirusTotal to achieve better classification.

Here, a limitation is that VirusTotal customers receive a classifier; thus, they cannot conduct detailed analysis using malware binaries, which reduces customer motivation to use VirusTotal. Note that the VirusTotal case is a single example. The proposed architecture can also be applied to the models described in Section 2.1

8.8 Attack against the proposed architecture

Here, we discuss attacks against the proposed architecture.

8.8.1 DoS against the analysis organization

An attacker may perform DoS attack against an interface to exchange classifiers. Here, the interface is only used by information provider organizations; thus, IP filtering should be performed to prevent DoS attacks against the interface.

8.8.2 Spoofing and falsification of classifier template

An attacker pretending to be a legitimate organization submits fake classifiers to disturb legitimate analysis. Moreover, if the attacker can falsify the communication path between the information providing and analysis organizations, they can modify the submitted classifier. Furthermore, the attacker can modify a classifier template such that the analysis organization cannot aggregate classifiers. To mitigate the above risks, the communication protocol must support both authentication and encryption (e.g., TLS and SSH).

8.8.3 Crafted malware

Information provider organizations train classifiers based on the template specifying features. If an attacker knows this template, they can generate crafted malware with the same feature. In this case, the classifier cannot distinguish such malware because

the extracted features are the same.

To mitigate these risks, the classifier template should not be disclosed since this would allow the attacker to know the features used to train the classifier.

8.9 Malicious organization

In Section 2.2, we assumed that the information provider and analysis organizations were benign. In contrast, in this situation, here we assume that one of these entities is malicious.

If an information provider organization is malicious, the organization would send a fake classifier to the analysis organization. As a result, the detection performance of an aggregated classifier would be reduced due to the fake classifier. Here, a solution is to increase the number of information provider organizations to reduce the effect of the fake classier by the dominant normal classifiers.

If an analysis organization is malicious, it can attempt to provide crafted classifier templates to the information provider organization. For example, a malicious analysis organization sends many templates, each of which discloses information below the disclosure threshold; however by aggregating the disclosed information, the malicious analysis organization would obtain information beyond that constrained by the threshold. This attack can be detected by the information provider organization by monitoring the classifier templates.

9. Related work

9.1 Malware detection and classification using machine learning

Many studies have examined detecting and classifying malware. To cluster malware, FIRMA [32] executes unlabeled malware and captures traffic such as HTTP, IRC, and SMTP, from/to the malware. Then FIRMA performs clustering on the traffic. In the HTTP case, FIRMA uses the URL path, URL parameters, and HTTP header as features for clustering. Then, FIRMA generates a signature by extracting high coverage and low false-positive tokens from the clusters. DeepSign [16] leverages deep learning for malware classification. Specifically, a deep belief network, which is a deep unsupervised network, generates general behavior of malware. DeepSign compresses 20,000 input features to 30 features using an eight-layer network. Then, it outputs a set of the 30 features as a malware signature. Hassen and Chan proposed a graph-based malware detection method [21] that extracts a function call graph from an executable and converts the graph to an input vector by representing the existence of links between functions. Ahmadi et al. reported features for malware classification [10]. They leveraged hex-dump-based and op-code features. In addition, they analyzed the importance of features based on "mean decrease impurity" and demonstrated that section information and data define (constant values) have greater influence. They also reported that most of the winners of the malware classification challenge used the XGBoost technique to aggregate classifiers trained using different features. Our key idea is information sharing by classifier exchange and aggregation; therefore we can introduce the above techniques to improve the accuracy of the proposed architecture.

Relative to evasive malware (Section 8.3), evasion techniques and countermeasures have been proposed. Vidas et al. presented evasion techniques to detect Android sandboxes [36]. They classified the techniques into four categories: (1) behavior differences (e.g., differences of API and network behaviors), (2) performance differences (e.g., CPU and graphic performance), (3) hardware (e.g., CPU type and peripherals), and (4) differences in software components (e.g., vendor specific software). To detect evasive malware that detects a sandbox and changes its behavior, BareCloud [22] runs malware in four different environments, i.e., bare-metal environments: Anubis, which is a QEMU-based analysis framework, Ether, which uses the Xen hypervisor, and the Cuckoo sandbox, which uses VirtualBox. BareCloud calculates the similarity between analysis and bare-metal environments by comparing system and network activities. Malware with low similarity is assumed to be evasive malware. The proposed architecture and these previous studies form a complementary relationship, and the proposed architecture can introduce these ideas to classify evasive malware.

9.2 Cybersecurity information sharing

Several studies have explored information sharing of cybersecurity information. Bhatia et al. analyzed privacy risk in cybersecurity data sharing [12]. They estimated *willingness to share* by administering a survey to security professionals. Their analysis results show that privacy-sensitive data and potentially confidential data have low willingness to share. Specifically, password, username, key-logging data, e-mails, chat history, video or image files, browser history, web sites visited, contact information, and keyword searches are listed as low willingness data. Moreover, temporary files, application session data, memory data, registry information, packet data, and sensor data are ranked after the above privacy-sensitive data, and these data appear to potentially contain confidential information of the organizations.

Fisk et al. also discussed privacy principles [19] relative to the least disclosure, qualitative evaluation, and forward progress concepts. For least disclosure, they proposed moderated queries that can limit trust or query issuers by restricting the query type. In addition, they proposed a rate limit for queries to control information disclosure. Murdoch et al. discussed anonymity and trust in cybersecurity collaborations [28], and they proposed a reputation system to balance the trade-off between anonymity (anonymous information provider) and trust in the information.

Serrano et al. discussed the design of a cybersecurity data sharing system [35] that focused on four problems: policy and legal issues, ontological issues, information sharing across communities, and uncertainty management. Wagner et al. proposed the Malware Information Sharing Platform (MISP) [37], in which they defined a data exchange model comprising events, attributes, and tags. The MISP also supports sharing levels such as organization only and community only. In addition, Microsoft has issued a report about a framework for cybersecurity information sharing [20]. They investigated the basic elements of information sharing such as the actors, type of information, model and exchange method of the information. Moreover, they recommend a design with privacy protection to respect privacy and civil lib-

erties. Structured Threat Information Expression (STIX) [8] is a language for exchanging threat information. STIX comprises eight information fields: campaigns, threat actors, TTPs (Tactics, Techniques, and Procedures), indicators, observables, incidents, courses of action, and exploit targets. Each field has its own structure and is linked to each other. For example, a campaign can be linked to actors and incidents. In addition, TAXII [9] is a framework to exchange threat information, e.g., information written in STIX.

Unfortunately, these frameworks do not consider information exchange without disclosing confidential information. We believe that the proposed architecture improves the above frameworks by providing confidentiality-preserving capabilities.

10. Conclusion

Confidential information is a barrier to information sharing across organizations. To address this problem, we propose a confidentiality-preserving collaborative defense architecture that analyzes the incident information without disclosing confidential information. The key features of the proposed architecture are the exchange of trained classifiers, e.g., neural networks, rather than raw attack/incident information and classifier aggregation using ensemble learning techniques. We implement an initial prototype and demonstrated that classification accuracy is improved 1.8% by aggregating the classifiers of five organizations. Although there are remaining issues, e.g., information disclosure metrics, we believe that the proposed architecture will improve collaborative malware analysis across organizations.

Acknowledgments A part of this work was funded by the WarpDrive: Web-based Attack Response with Practical and Deployable Research Initiative project, supported by the National Institute of Information and Communications Technology (NICT).

References

[1] Cyber Clean Center, The Cyber Clean Center project (online), available from ⟨https://www.telecom-isac.jp/ccc/en_index.html⟩ (accessed 2017-08-10).
[2] Marco Ramilli, Malware Training Sets: A machine learning dataset for everyone (online), available from ⟨http://marcoramilli.blogspot.it/2016/12/malware-training-sets-machine-learning.html⟩ (accessed 2018-02-10).
[3] Kaggle, Microsoft Malware Winners' Interview: 1st place, "NO to overfitting!" (online), available from ⟨http://blog.kaggle.com/2015/05/26/microsoft-malware-winners-interview-1st-place-no-to-overfitting/⟩ (accessed 2017-05-24).
[4] Mirai BotNet (online), available from ⟨https://github.com/jgamblin/Mirai-Source-Code⟩ (accessed 2017-05-25).
[5] McAfee, Submit a Virus or Malware Sample (online), available from ⟨https://www.mcafee.com/ca/threat-center/resources/how-to-submit-sample.aspx⟩ (accessed 2017-05-24).
[6] Symantec, Submit Virus Samples (online), available from ⟨https://www.symantec.com/security-center/submit-virus-samples⟩ (accessed 2017-05-24).
[7] TrendMicro, Submitting suspicious or undetected virus for file analysis to Technical Support using Threat Query Assessment (online), available from ⟨https://success.trendmicro.com/solution/1031392-submitting-suspicious-or-undetected-virus-for-file-analysis-to-technical-support-using-threat-query⟩ (accessed 2017-05-24).
[8] STIX Version 2.0. Part 1: STIX Core Concepts (2017).
[9] TAXII Version 2.0, Working Draft 01 (2017).
[10] Ahmadi, M., Ulyanov, D., Semenov, S., Trofimov, M. and Giacinto, G.: Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification, *Proc. 6th ACM Conference on Data and Application Security and Privacy*, *CODASPY '16* (2016).
[11] Avgerinos, T., Cha, S.K., Rebert, A., Schwartz, E.J., Woo, M. and Brumley, D.: Automatic Exploit Generation, *Comm. ACM*, Vol.57, No.2, pp.74–84 (2014).
[12] Bhatia, J., Breaux, T.D., Friedberg, L., Hibshi, H. and Smullen, D.: Privacy Risk in Cybersecurity Data Sharing, *Proc. 2016 ACM on Workshop on Information Sharing and Collaborative Security*, *WISCS '16* (2016).
[13] Blond, S.L., Gilbert, C., Upadhyay, U., Gomez-Rodriguez, M. and Choffnes, D.R.: A Broad View of the Ecosystem of Socially Engineered Exploit Documents, *24th Annual Network and Distributed System Security Symposium, NDSS* (2017).
[14] Brumley, D., Poosankam, P., Song, D. and Zheng, J.: Automatic Patch-Based Exploit Generation is Possible: Techniques and Implications, *Proc. 2008 IEEE Symposium on Security and Privacy* (2008).
[15] Cha, S.K., Avgerinos, T., Rebert, A. and Brumley, D.: Unleashing Mayhem on Binary Code, *2012 IEEE Symposium on Security and Privacy* (2012).
[16] David, O.E. and Netanyahu, N.S.: DeepSign: Deep learning for automatic malware signature generation and classification, *2015 International Joint Conference on Neural Networks* (*IJCNN*), pp.1–8 (2015).
[17] Dwork, C.: Differential privacy: A survey of results, *International Conference on Theory and Applications of Models of Computation*, Springer, pp.1–19 (2008).
[18] Fette, I., Sadeh, N. and Tomasic, A.: Learning to Detect Phishing Emails, *Proc. 16th International Conference on World Wide Web*, *WWW '07*, pp.649–656 (2007).
[19] Fisk, G., Ardi, C., Pickett, N., Heidemann, J., Fisk, M. and Papadopoulos, C.: Privacy Principles for Sharing Cyber Security Data, *2015 IEEE Security and Privacy Workshops* (2015).
[20] Goodwin, C., Nicholas, J.P., Bryant, J., Ciglic, K., Kleiner, A., Kutterer, C., Massagli, A., Mckay, A., Mckitrick, P., Neutze, J., et al.: A framework for cybersecurity information sharing and risk reduction, Technical Report, Microsoft Corporation (2015).
[21] Hassen, M. and Chan, P.K.: Scalable Function Call Graph-based Malware Classification, *Proc. 7th ACM on Conference on Data and Application Security and Privacy*, *CODASPY '17* (2017).
[22] Kirat, D., Vigna, G. and Kruegel, C.: BareCloud: Bare-metal Analysis-based Evasive Malware Detection, *23rd USENIX Security Symposium* (*USENIX Security 14*), pp.287–301 (2014).
[23] Kobsa, A. and Schreck, J.: Privacy Through Pseudonymity in User-adaptive Systems, *ACM Trans. Internet Technol.*, Vol.3, No.2, pp.149–183 (2003).
[24] Lindorfer, M., Kolbitsch, C. and Milani Comparetti, P.: Detecting Environment-sensitive Malware, *Proc. 14th International Conference on Recent Advances in Intrusion Detection*, *RAID'11* (2011).
[25] Makiou, A., Begriche, Y. and Serhrouchni, A.: Improving Web Application Firewalls to detect advanced SQL injection attacks, *2014 10th International Conference on Information Assurance and Security* (2014).
[26] Mayhew, M., Atighetchi, M., Adler, A. and Greenstadt, R.: Use of machine learning in big data analytics for insider threat detection, *MILCOM 2015 - 2015 IEEE Military Communications Conference* (2015).
[27] Morishima, S.: Analyzing Targeted Email Attacks with Decoy Document Collection System, *SCIS* (2017).
[28] Murdoch, S. and Leaver, N.: Anonymity vs. Trust in Cyber-Security Collaboration, *Proc. 2nd ACM Workshop on Information Sharing and Collaborative Security*, *WISCS '15* (2015).
[29] Ni, T., Gu, X., Wang, H. and Li, Y.: Real-time Detection of Application-layer DDoS Attack Using Time Series Analysis, *J. Control Sci. Eng.*, Vol.2013, p.4:4 (online), DOI: 10.1155/2013/821315 (2013).
[30] Pék, G., Bencsáth, B. and Buttyán, L.: nEther: In-guest Detection of Out-of-the-guest Malware Analyzers, *Proc. 9th European Workshop on System Security*, *EUROSEC '11* (2011).
[31] Radford, A., Metz, L. and Chintala, S.: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, *CoRR*, Vol.abs/1511.06434 (2015) (online), available from ⟨http://arxiv.org/abs/1511.06434⟩.
[32] Rafique, M.Z. and Caballero, J.: FIRMA: Malware Clustering and Network Signature Generation with Mixed Network Behaviors, *Proc. 16th International Symposium on Research in Attacks, Intrusions, and Defenses - Volume 8145*, *RAID 2013* (2013).
[33] Royal, P., Halpin, M., Dagon, D., Edmonds, R. and Lee, W.: PolyUnpack: Automating the Hidden-Code Extraction of Unpack-Executing Malware, *2006 22nd Annual Computer Security Applications Conference* (*ACSAC'06*), pp.289–300 (2006).
[34] Samarati, P. and Sweeney, L.: Protecting privacy when disclosing information: K-anonymity and its enforcement through generalization and suppression, Technical Report, SRI International (1998).
[35] Serrano, O., Dandurand, L. and Brown, S.: On the Design of a Cyber

Security Data Sharing System, *Proc. 2014 ACM Workshop on Information Sharing & Collaborative Security*, *WISCS '14* (2014).
[36] Vidas, T. and Christin, N.: Evading Android Runtime Analysis via Sandbox Detection, *Proc. 9th ACM Symposium on Information, Computer and Communications Security*, *ASIA CCS '14* (2014).
[37] Wagner, C., Dulaunoy, A., Wagener, G. and Iklody, A.: MISP: The Design and Implementation of a Collaborative Threat Intelligence Sharing Platform, *Proc. 2016 ACM on Workshop on Information Sharing and Collaborative Security*, *WISCS '16* (2016).
[38] Yasin, A. and Abuhasan, A.: An intelligent classification model for phishing email detection, *CoRR*, Vol.abs/1608.02196 (2016) (online), available from ⟨http://arxiv.org/abs/1608.02196⟩.
[39] Yokoyama, A., Ishii, K., Tanabe, R., Papa, Y., Yoshioka, K., Matsumoto, T., Kasama, T., Inoue, D., Brengel, M., Backes, M. and Rossow, C.: *Sandprint: Fingerprinting malware sandboxes to provide intelligence for sandbox evasion*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol.9854 LNCS, pp.165–187, Springer Verlag (2016).

Appendix

A.1 Malware family classification (Malware naming issue)

Here, we extend the proposed architecture to classify malware when malware names are not shared among information provider organizations. A key problem is how the analysis organization performs clustering/grouping of the trained classifier because majority voting is performed based on the malware name information. For example, assuming one information provider organization names a malware A and another organization names the same malware α, the analysis organization cannot sum the voting because the analysis organization does not know malware A and malware α are the same. To associate malware names across information provider organizations, we identify two possible approaches.

Approach 1. Each information provider organization analyses the malware family and sends the trained classifier with its malware family tag in a unified manner to the organizations. The analysis organization groups the trained classifiers based on the tags. Then, for each grouped classifier, the analysis organization aggregates the classifiers into a single classifier.

Approach 2. In the previous approach, each organization must determine a family for each malware. If the information provider organizations are security companies, that would be possible (Section 2.4.2). However, information provider organizations are non-security companies (Section 2.4.1); thus, malware family identification is impractical due to a lack of security knowledge. To address this case, the analysis organization could perform clustering of the trained classifiers. Then the analysis organization can combine the classifiers into clusters, each of which corresponds to a malware family.

A.1.1 Clustering of trained classifier

Here, we further investigate the second approach, which clusters classifiers. Here, classifier clustering can be performed based on classifier parameters such as the weight values of the neural network and SVM.

A.1.1.1 Hyperplane based clustering

A classifier separates a feature space by a hyperplane, and a hyperspace enclosed by the hyperplane represents a malware family.

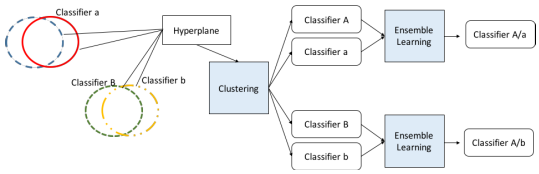

Fig. A·1 Clustering classifier based on hyperplane.

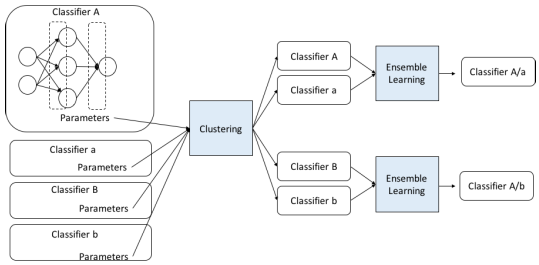

Fig. A·2 Clustering classifier based on trained parameters.

Table A·1 Similarities between malware classes.

	Apt1	Crypto	Zeus
Apt1	0.47	1.08	2.19
Crypto	1.05	0.53	1.32
Zeus	1.99	1.54	0.36

Thus, it is assumed that classifiers trained using the same malware family have similar hyperspaces. Therefore we can cluster the classifier using the similarity of the hyperspace. **Figure A·1** shows clustering using similarities among hyperspaces. The similarity of two classifiers is calculated using Jaccard distance as follows.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{A.1}$$

Here, A and B represent the decisions of classifiers A and B, respectively. $|A \cap B|$ denotes the number of equal decisions from the sample and $|A \cup B|$ denotes the number of samples. Classifiers with a similarity score greater than a threshold are assumed to handle the same malware family.

If the classifier output is probability, we can use the proposed information disclosure metric (Eq. (4)) as the similarity between classifiers. **Table A·1** shows a preliminary evaluation of the similarity between classifiers of two organizations. Here, we use the malware samples used in Section 7 as test data.

As can be seen, the similarity scores between two classifiers trained using the same malware family (diagonal elements) have low scores. On the other hand, scores between two classifiers trained using a different malware family (non-diagonal elements) are high compared to the diagonal elements. Therefore, the analysis organization can infer associations among classifiers using the scores even if the malware names are not shared.

A.1.1.2 Parameter based clustering

Another classifier clustering approach is parameter based clustering. Here, it is also assumed that classifiers trained using the same malware family have similar internal parameters, e.g., weight values of the SVM and neural networks.

To perform clustering, parameters are extracted from each classifier. Then, similarity is calculated over the parameter values. For simplicity, we use Euclidean distance and perform clustering based on this distance. We define this distance as follows.

$$D(A, B) = \sqrt{\sum_i a_i^2 + b_i^2} \quad \text{(A.2)}$$

Here, a_i and b_i denote the i-th parameter of classifiers A and B, respectively. Note that each information provider organization uses the same type of classifier specified by the classifier template; thus, the structures of the classifiers are the same and comparable.

Takayuki Sasaki received his M.S. degree in Physics from the University of Tokyo in 2006. He worked as a visiting researcher at ETH Zürich from 2015 to 2016. He is currently a principal researcher at NEC Security Research Laboratories. He is also a Ph.D. student at Yokohama National University. His main research interests include security architecture for cloud computing, SDN, and IoT.

Katsunari Yoshioka received his B.E., M.E. and Ph.D. degrees in Computer Engineering from Yokohama National University in 2000, 2002, and 2005, respectively. From 2005 to 2007, he was a Researcher at the National Institute of Information and Communications Technology, Japan. Currently, he is an Associate Professor at the Graduate School of Environment and Information Sciences, Yokohama National University. His research interest covers wide range of information security, including malware analysis, network monitoring, intrusion detection, etc. He was awarded 2007 Prizes for Science and Technology by The Commendation for Science and Technology by the Minister of Education, Culture, Science and Technology.

Tsutomu Matsumoto is a professor of Faculty of Environment and Information Sciences, Yokohama National University and directing the Research Unit for Information and Physical Security at the Institute of Advanced Sciences. He received Doctor of Engineering from the University of Tokyo in 1986. Starting from Cryptography in the early 80's, he has opened up the field of security measuring for logical and physical security mechanisms. Currently he is interested in research and education of Embedded Security Systems such as IoT Devices, Network Appliances, Mobile Terminals, In-vehicle Networks, Biometrics, Artifactmetrics, and Instrumentation Security. He is serving as the chair of the IEICE Technical Committee on Hardware Security, the Japanese National Body for ISO/TC68 (Financial Services), and the Cryptography Research and Evaluation Committees (CRYPTREC) and as an associate member of the Science Council of Japan (SCJ). He was a director of the International Association for Cryptologic Research (IACR) and the chair of the IEICE Technical Committee on Information Security. He received the IEICE Achievement Award, the DoCoMo Mobile Science Award, the Culture of Information Security Award, the MEXT Prize for Science and Technology, and the Fuji Sankei Business Eye Award.