**INTELLIGENT FILE SCORING SYSTEM FOR MALWARE DETECTION**

ABSTRACT

Currently, the most signi¯cant line of defense against mal-

ware is anti-virus products which focus on authenticating

valid software from a white list, blocking invalid software

from a black list, and running any unknown software (i.e.,

the gray list) in a controlled manner. The gray list, con-

taining unknown software programs which could be either

normal or malicious, is usually authenticated or rejected

manually by virus analysts. Unfortunately, along with the

development of the malware writing techniques, the num-

ber of ¯le samples in the gray list that need to be analyzed

by virus analysts on a daily basis is constantly increasing.

In this paper, we develop an intelligent ¯le scoring system

(IFSS for short) for malware detection from the gray list by

an ensemble of heterogeneous base-level classi¯ers derived

by di®erent learning methods, using di®erent feature rep-

resentations on dynamic training sets. To the best of our

knowledge, this is the ¯rst work of applying such ensem-

ble methods for malware detection. IFSS makes it practical

for virus analysts to identify malware samples from the huge

gray list and improves the detection ability of anti-virus soft-

ware. It has already been incorporated into the scanning

tool of Kingsoft's Anti-Virus software. The case studies on

large and real daily collection of the gray list illustrate that

the detection ability and e±ciency of our IFSS system out-

*AUTHORS*

*MRUNAL MUNJE,*

*PRACHI DAKHOLE,*

*JUILI KOTANGALE*

*¤*The author is also a±liated with Anti-virus laboratory,

KingSoft Corporation.

Permission to make digital or hard copies of all or part of this work for

personal or classroom use is granted without fee provided that copies are

not made or distributed for profit or commercial advantage and that copies

bear this notice and the full citation on the first page. To copy otherwise, to

republish, to post on servers or to redistribute to lists, requires prior specific

permission and/or a fee.

*KDD’09,* June 28–July 1, 2009, Paris, France.

Copyright 2009 ACM978-1-60558-495-9/09/06 ...$5.00.

Categories and Subject Descriptors

I.2.6 [Arti¯cial Intelligence]: Learning; D.4.6 [Operating

System]: Security and Protection - Invasive software

General Terms

Algorithms, Experimentation, Security

Keywords

malware detection, gray list, ensemble

1. INTRODUCTION

1.1 Malware Detection from the Gray List

Malware is a generic term [5] to denote all kinds of un-

wanted software(e.g., viruses, backdoors, spyware, trojans

and worms). Numerous attacks made by the malware have

posed a major security threat to computer users. Therefore,

malware detection is one of the computer security topics

that are of great interest. Currently, the most signi¯cant

line of defense against malware is anti-virus software prod-

ucts, such asNOD32, Kaspersky and Kingsoft's Antivirus.

These widely-used malware detection software tools mainly

use signature-based method to recognize threats. Signature

is a short string of bytes which is unique for each known mal-

ware so that future examples of it can be correctly classi¯ed

with a small error rate.

In order to capture as many malware samples as possible,

besides authenticating valid software from a white list and

blocking invalid software from a black list using signature-

based method, most of the existing anti-virus software prod-

ucts run any unknown software (i.e., the gray list) in a con-

trolled manner. The gray list, containing unknown software

programs which could be either normal or malicious, is usu-

ally authenticated or rejected manually by virus analysts.

Unfortunately, with the development of the malware writ-

ing techniques, the number of ¯le samples in the gray list

that need to be analyzed by virus analysts on a daily basis

is constantly increasing. For example, the gray list collected

by the Anti-virus Lab of a large software corporation usually

contains more than 100,000 ¯le samples per day. The gray

list is not only large in size, but also very complicated since

it contains the variants of known malware and previously

1385

unknown malware samples. In order to remain e®ective, it

is of paramount importance for the anti-virus companies to

be able to quickly analyze the gray list and detect malware

samples.

Over the last few years, many research e®orts have been

conducted on developing intelligent malware detection sys-

tems [19, 30, 33, 26]. In these systems, the detection pro-

cess is generally divided into two steps: *feature extraction*

and *categorization*. In the ¯rst step, various features such as

Application Programming Interface (API) calls and program

strings are extracted to capture the characteristics of the ¯le

samples. In the second step, intelligent techniques such as

decision trees are used to automatically categorize the ¯le

samples into di®erent classes based on computational analy-

sis of the feature representations. These intelligent malware

detection systems are varied in their use of feature represen-

tations and categorization methods. For example, IMDS [33]

performs association classi¯cation on Windows API calls ex-

tracted from executable ¯les while Naive Bayes methods on

the extracted strings and byte sequences are applied in [26].

1.2 Contributions of The Paper

Di®erent feature representations and categorization meth-

ods have their own advantages and limitations in malware

detection. None of the single feature set can immune or resis-

tant to mimicry designed to confuse the anti-virus software

as di®erent feature representation typically capture di®er-

ent characteristics of ¯le samples. For example, API calls

typically re°ect the behavior of program code pieces while

program strings consist of reused code fragments, author sig-

natures, ¯les names and system resource information. On

the other hand, di®erent categorization methods have dif-

ferent strengths and may excel at di®erent situations. A

natural question arises: can we combine di®erent feature

representations and categorization methods to improve the

performance of malware detection?

Previous research has shown that ensemble methods, by

combining multiple input systems, are a popular way to

overcome instability and increase performance in many ma-

chine learning tasks, such as classi¯cation, clustering and

ranking [9, 11]. In this paper, we develop an intelligent ¯le

scoring system (IFSS for short) for malware detection from

the gray list by an ensemble of heterogeneous base-level clas-

si¯ers derived by di®erent learning methods, using di®erent

feature representations on dynamic training sets. To the

best of our knowledge, this is the ¯rst work of applying such

ensemble methods for malware detection.

Our IFSS system has the following major traits:

*² Diverse feature representations:* Two sets of extracted

features, API calls and interpretable string, are used in

our system. API calls re°ect the behavior of program

code pieces and the interpretable strings carry seman-

tic interpretations and re°ect an attacker's intent and

goal. For example, (1) the API call like\GetVersionExA"

in\KERNEL32.DLL" actually executes the function of

obtaining extended information about the version of

the recently running operating system; (2) the string of

\*<script language=`javascript'>window.open(`readme.eml')*"

always exists in the worms of \Nimda" and implicates

that they try to infect the scripts.

*² Dynamic training sets:* Note that malware techniques

are constantly evolving and new malware samples are

produced on a daily basis. To account for the temporal

trends of malware writing, our IFSS system makes use

of two di®erent datasets for training purpose: *DB T*1

which consists of ¯le samples from the historical data

collection and *DB T*2 which contains most recent ¯le

samples. The training sets are dynamically changing

to include new samples while retaining the character-

istics of historical data. In addition, training on di®er-

ent training sets also helps to increase the diversity of

individual classi¯ers.

*² Heterogeneous base classi¯ers:* Associative classi¯ers

and support vector machines have been chosen as our

base classi¯ers. Both classi¯ers have been successfully

used in malware detection [33, 19] and have distinct

properties.

*² Human-in-the-Loop:* Our system provides a user-friendly

mechanism for incorporating the expert knowledge and

expertise of virus analysts. It should be pointed that

in many cases, classifying a ¯le sample from the gray

list as malware will still be the prerogative of virus an-

alysts. Our IFSS system uses a simple voting scheme

to combine the prediction of individual classi¯ers and

produces a ¯le scoring list which is simple for virus an-

alysts to interpret and understand. Virus analysts can

then look at the top ranked ¯le samples and manually

authenticated and rejected those samples. New labeled

samples can then be used to update the training sets.

*² Simultaneous model construction and testing:* With

the dynamic training sets and human-in-the-Loop, our

IFSS system performs simultaneous model construc-

tion and testing in malware detection, an environment

and a task that is constantly evolving over the time.

In our IFSS system the following three steps are iter-

atively conducted on the daily basis: 1) classi¯ers are

¯rst constructed to classify/rank the ¯le samples in the

gray list; 2) virus analysts manually analyze these top

ranked samples; and 3) labeled samples are then used

to dynamically generate new training sets.

All these traits make our IFSS a practical solution for help-

ing virus analysts identify malware samples in the gray list

and improving the detection ability of anti-virus software.

The case studies on large and real data collections collected

by the Anti-virus Lab of Kingsoft corporation illustrate that:

(1) After being scanned by all the popular anti-virus soft-

ware products, such as NOD32 and Kaspersky, malware in

the gray list still can be e®ectively detected by our IFSS.

(2) The performance and e±ciency of our IFSS outperform

other classi¯cation methods in detecting malware from the

gray list. (3) Our IFSS reduces the number of ¯le samples

that need to be analyzed by virus analysts. Our case studies

show that the percentage of malware samples in the gray list

is about 0*:*5% while the percentage of malware samples in

the top 100 ranked ¯les samples of the ¯le scoring list gener-

ated by our IFSS system is 35%. Therefore IFSS can greatly

save human labor. As a result, our IFSS has already been

incorporated into the scanning tool of Kingsoft's Anti-Virus

software.

1.3 Organization of The Paper

The rest of this paper is organized as follows. Section 2

gives an overview of our IFSS system and Section 3 discusses

1386

the related work. Section 4 describes the feature representa-

tion and extraction; Section 5 introduces the two base classi-

¯ers; Section 6 presents the ensemble framework used in our

IFSS system for generating the ¯le scoring list. In Section 7,

we systematically evaluate the e®ects and e±ciency of our

IFSS system in comparison with other classi¯cation meth-

ods. In Section 8, based on the daily data collection obtained

from Kingsoft Anti-virus lab, we examine the detection abil-

ity and e±ciency of IFSS in comparison with other popular

anti-virus software such as NOD32 and Kaspersky. Finally,

Section 9 concludes.

2. SYSTEM ARCHITECTURE

In this paper, resting on the analysis of Windows API

(Application Program Interface) calls which can re°ect the

behavior of program code pieces and interpretable strings

which carry semantic interpretations and re°ect an attacker's

intent and goal, we develop the Intelligent File Scoring Sys-

tem(IFSS) to detect malware from the gray list. Figure 1

shows the malware detection procedure of IFSS:

Figure 1: Malware detection °ow of IFSS

*²* Training:

1. Feature Extractor: IFSS ¯rst uses the feature ex-

tractor to extract the API calls and interpretable

strings from the collected Windows Portable Ex-

ecutable (PE) ¯les of \black list" and \white list",

converts them to a group of 32-bit global IDs as

the features of the training data, and stores these

features in the signature database. A sample sig-

nature database is shown in Figure 2, in which

there are 8 ¯elds: record ID, PE ¯le name, ¯le

type (\0" represents benign ¯le while \1" is for

malicious ¯le), called APIs name, called API ID,

the total number of called API functions, string

ID, and the total number of interpretable strings.

The transaction data can also be easily converted

to relational data if necessary.

Figure 2: A sample signature database after data

transformation

2. Feature Selector: Feature selection is important

for many pattern classi¯cation systems. As not all

of the extracted features are contributing to mal-

ware detection, feature selector is used to identify

the most representative features.

3. Base Classi¯ers: Base classi¯ers are constructed

by applying associative classi¯er and SVM us-

ing di®erent feature representations on di®erent

training sets (denoted by *DB T*1 and *DB T*2).

Coupled with the two di®erent feature represen-

tations, we have four di®erent combinations of

training sets and feature representations: *DB T*1

with API calls, *DB T*1 with interpretable strings,

*DB T*2 with API calls, and *DB T*2 with inter-

pretable strings. Using the two di®erent classi¯-

cation methods, we thus obtain 8 di®erent base

classi¯ers.

*²* Malware Detection from the Gray List

The daily collection of ¯le samples is ¯rst scanned

by the existing popular anti-virus software products.

Valid software programs from a white list are authen-

ticated and invalid software programs from a black list

are blocked or rejected. The gray list, containing un-

known software programs which could be either normal

or malicious, is then fed into our IFSS system. After

feature extraction and selection, 8 di®erent classi¯ers

are applied to the gray list. A simple voting scheme

is used to combine base classi¯ers and generate a ¯le

scoring list. The ¯le score list ranks the input ¯le sam-

ples from the gray list. Virus analysts can then look

at the top ranked ¯le samples and manually authen-

ticated and rejected those samples. These manually

labeled ¯le samples can then be used to update the

training sets.

1387

3. RELATEDWORK

3.1 Data Mining Methods for Malware

Detection

Signature-based methods are widely used in malware de-

tection to recognize threats [12]. A signature is a short

string of bytes which is unique for each known malware.

However, this classic signature-based method always fails to

detect variants of known malware or previously unknown

malware. The problem lies in the signature extraction and

generation process, and in fact these signatures can be eas-

ily bypassed [27]. In order to overcome the disadvantages of

the widely-used signature-based malware detection method,

data mining and machine learning approaches are proposed

for malware detection [19, 26, 30, 7]. The performance of

such methods used for malware detection critically depend

on the set of features and the classi¯er [10].

Neural Networks as well as immune system are used by

IBM for computer virus recognition [28]. Naive Bayes, Sup-

port Vector Machine(SVM) and Decision Tree classi¯ers are

used to detect new malicious executables based on small

data collection in the previous studies [19, 26, 30]. Recently,

associative classi¯cation [22], with its ability to utilize re-

lationships among attributes, has been also applied in [33].

Note that the class distribution in the gray list of our col-

lection is quite imbalanced with the malware samples as the

minority class. Many accuracy driven classi¯ers may fail on

such a large and imbalanced gray list. For example, neu-

ral networks and naive Bayes consistently biased towards

the majority class at any given size and prone to treat the

minority (malware) class as noise [18]. Decision trees algo-

rithms (C4.5) are also not performing well in the presence of

imbalance: they might lose big parts of the minority (mal-

ware) class at the pruning phase or lead to the trees of large

size and over-¯tting of the minority class [18]. Hence in our

work, we choose association classi¯er and SVM as our base

classi¯ers.

3.2 Ensemble Classification

Previous research has shown that ensemble methods, by

combining multiple input systems, are a popular way to

overcome instability and increase performance in many ma-

chine learning tasks, such as classi¯cation, clustering and

ranking. For example, an ensemble of classi¯ers is a set of

classi¯ers whose individual predictions are combined in some

way (typically by voting) to classify new examples. Gener-

ally there are two types of classi¯er ensemble: 1) Homoge-

neous ensemble: the base classi¯ers are constructed using

a single learning algorithm, such as decision trees or neural

networks [9]. Typically base classi¯ers are generated by ma-

nipulating the training set (as done in boosting or bagging),

manipulating the input features, manipulating the output

targets or injecting randomness in the learning algorithm [8].

The individual classi¯ers are then typically combined by vot-

ing or weighted voting. 2) Heterogeneous ensemble: the base

classi¯ers are constructed by applying di®erent learning al-

gorithms (with heterogeneous model representations) to a

single dataset [24]. More complicated methods such as stack-

ing are used for combining classi¯ers [32]. In our IFSS sys-

tem, the base classi¯ers are constructed by di®erent learning

methods (association classi¯cation or SVM), using di®erent

feature representations (API calls or Interpretable strings)

on di®erent training sets (*DB T*1 and *DB T*2). We expect

that our construction of base classi¯ers would increase their

diversity and improve the classi¯cation performance. Our

work is the ¯rst e®ort on applying such ensemble classi¯er

methods for malware detection.

4. FEATURE EXTRACTION AND

SELECTION

Our IFSS system is performed directly on Windows PE

code. PE is designed as a common ¯le format for all °a-

vor of Windows operating system, and PE malware are in

the majority of the malware rising in recent years. If a PE

¯le is previously compressed by a third party binary com-

press tool such as UPX and ASPack Shell or embedded a

homemade packer, it needs to be decompressed ¯rst. We

use the dissembler W32Dasm developed by KingSoft Anti-

Virus Laboratory to dissemble the PE code and output the

assembly instructions as the input for feature extraction.

4.1 Feature Extraction

API Calls: The Windows API execution calls for each

benign/malicious executable is generated by a PE parser.

Through the API query database, the API execution calls

generated by the PE parser can be converted to a group of

32-bit global IDs which represents the static execution calls

of the corresponding API functions. For example, the API

\KERNEL32.DLL, OpenProcess" executes the function that

returns a handle to an existing process object and it can be

encoded as 0x00500E16. Then we use the API calls as the

signatures of the PE ¯les and store them in the signature

database.

Interpretable Strings: The interpretable strings are ex-

tracted using a feature parser. The feature parser reads the

PE ¯le. If there is a sequence of consecutive bytes belonging

to the same Character Set, such as ASCII, GB2312, Big5

and Unicode, then the parser exacts them as our features.

Figure 3 shows a sample interpretable strings extracted by

our feature parser. These strings are extracted from a mal-

ware named *Backdoor ¡ Redgirl:exe*. From Figure 3, we

can see the behaviors of the malware and the attacker's in-

tent explicitly.

Since these two sets of features are representation of PE

¯le samples at di®erent semantic levels, we use them for

building base classi¯er respectively.

4.2 Feature Selection

API Calls: As not all of the API calls are contributing

to malware detection, we rank each API call using Max-

Relevance algorithm [25] and select a set of API calls with

the highest relevance to the target class, i.e. the ¯le type, for

later classi¯cation. Given *ai* which represents the API with

ID *i*, and the ¯le type *f* (\0" represents benign executables

and \1" is for malicious executables), their mutual informa-

tion is de¯ned in terms of their frequencies of appearances

*p*(*ai*), *p*(*f*), and *p*(*ai; f*) as follows.

*I*(*ai; f*) =

Z Z

*p*(*ai; f*)*log*

*p*(*ai; f*)

*p*(*ai*)*p*(*f*)

*d*(*ai*)*d*(*f*)

With this algorithm, we select the top *m* APIs in the de-

scent order of *I*(*ai; f*), i.e. the best *m* individual features

correlated to the ¯le types of the PE ¯les.

1388

Figure 3: Interpretable strings sample extracted by

feature parser

Interpretable Strings: For Interpretable strings, we

¯rst use the corpus of natural language to ¯lter the can-

didate interpretable strings. If the string consists most of

the unusual characters which are not in the corpus, like

\!0&0h0m0o0t0y0", it will be pruned by our feature parser.

We then also apply Max-Relevance algorithm [25] to select a

set of the most representative strings for later classi¯cation.

5. BASE CLASSIFIERS

In this section, we brie°y describe the base classi¯ers used

in our IFSS. We use association classi¯er and SVM as our

base classi¯ers for the following reasons: 1) The gray list

is large and quite imbalanced and many accuracy driven

classi¯ers including neural networks, naive Bayes and de-

cision trees may fail on such a large and imbalanced gray

list [18]. On the other hand, association classi¯er and SVM

seems work well on imbalanced datasets. 2) Both associa-

tive classi¯cation and SVM have been successfully applied

in malware detection [33, 19]. In particular, association clas-

si¯cation can discover interesting relationships among input

features that are explicitly related to malware/benign ¯le

class and SVM can identify good classi¯cation boundaries

for malware detection.

5.1 Association Classification

*5.1.1 Introduction*

For malware detection in this paper, the ¯rst goal is to ¯nd

out how a set of input features (e.g., API calls) supports the

speci¯c class objectives: *class*1 = *Malicious*, and *class*2 =

*Benign*.

*²* (Support and con¯dence) Given a dataset DB, let

*I* = *fI*1*; :::; Img* be an itemset and *I ! Class*(*os; oc*)

be an association rule whose consequent is a class ob-

jective. The support and con¯dence of the rule are

de¯ned as:

*os* = *supp*(*I;Class*) =

*count*(*I [ fClassg*)

*jDBj*

*£* 100%

*oc* = *conf*(*I;Class*) =

*count*(*I [ fClassg*)

*count*(*I;DB*)

*£* 100%

where the function *count*(*I[fClassg*) returns the num-

ber of records in the dataset *DB* where *I [ fClassg*

holds.

*²* (Frequent itemset) Given *mos* as a user-speci¯ed

minimum support. *I* is a frequent itemset/pattern in

DB if *os ¸ mos*.

*²* (Classi¯cation association rule) Given *moc* as a

user-speci¯ed con¯dence. Let *I* = *fI*1*; :::; Img* be a

frequent itemset. *I ! Class*(*os; oc*) is a classi¯cation

association rule if *oc ¸ moc* where *os* and *oc* are the

support and con¯dence of the rule.

Apriori [1] and FP-Growth [13] algorithms can be ex-

tended to associative classi¯cation [21, 22]. In general, FP-

Growth algorithm is much faster than Apriori for mining

frequent item sets. In our work, we use FP-Growth algo-

rithm to conduct the classi¯cation association rule mining.

*5.1.2 Post-processing for Associative Classifier*

*Construction*

Since there is a huge number of rules generated from the

training set and it is infeasible to build the classi¯er used all

of rules, post-processing of associative classi¯cation is also

very important for improving the accuracy and e±ciency of

the classi¯er. Rule pruning and rule re-ordering are used for

post-processing associative classi¯er.

*Rule Pruning.* Several common rule pruning approaches

have been developed for associative classi¯ers to reduce the

generated rules [3, 4, 21, 22, 23, 29]: (1) *Â*2 (chi-square)

testing [21] to measure the signi¯cance of the rule itself, (2)

database coverage [22] to just keep the rules covering at least

one training data object not considered by a higher ranked

rule, and (3) pessimistic error estimation [22] to test the

estimated error of a new rule. These rule pruning techniques

mainly focus on individual rules. We have used the above

three pruning techniques in our application.

*Rule Re-ordering.* Rule re-ordering plays an important

role in the classi¯cation process since most of the associa-

tive classi¯cation algorithms utilize rule ranking procedures

as the basis for selecting the classi¯er [22, 21, 34]. In par-

ticular, CBA [22] and CMAR [21] use database coverage

pruning approach to build the classi¯ers, where the pruning

evaluates rules according to the rule re-ordering list. Hence,

the highest-order rules are tested in advance and then in-

serted into the classi¯er for predicting test data objects. For

rule re-ordering, there are ¯ve popular mechanisms [31]: (1)

Con¯dence Support size of Antecedent (CSA), (2) size of

Antecedent Con¯dence Support (ACS), (3) Weighted Rel-

ative Accuracy (WRA), (4) Laplace Accuracy, and (5) *Â*2

(chi-square) measure. CSA and ACS are belong to the pure

\support-con¯dence"framework and have been used by CBA

and CMAR for rule ranking. WRA, Laplace Accuracy and

*Â*2 measure are used by some associative classi¯cation al-

gorithms, such as CPAR [34], to weigh the signi¯cance of

each generated rule. In our work, we adopt hybrid rule re-

ordering mechanism by combining CSA and *Â*2 to re-order

the rules. We ¯rst rank the rules whose con¯dences are

100% by CSA and then re-order the remaining rules by *Â*2

measure. Because those rules whose con¯dences are 100%

can make the classi¯er accurate, while the remaining rules

should be considered by the combination of their supports

and con¯dences together.

1389

We use \Best First Rule" [31] method to predict the new

¯le samples. We select the ¯rst best rule that satis¯es the

new ¯le sample according to the rule list based on our hybrid

CSA/*Â*2 rule re-ordering method to predict whether the new

case is malware or not.

5.2 Support Vector Machine

Support Vector Machine (SVM) is a promising method

for data classi¯cation and regression and it has also been

successfully used in malware detection [16, 2, 15]. The key

to the success of SVM is the kernel function which maps

the data from the original space into a high dimensional

feature space. By constructing a linear boundary in the

feature space, the SVMproduces nonlinear boundaries in the

original space. The output of a linear SVM is *u* = *w£x¡b*,

where *w* is the normal weight vector to the hyperplane and

*x* is the input vector. Maximizing the margin can be seen

as an optimization problem:

minimize

1

2

*kwk*2*;* subject to *yi*(*w ¢ x* + *b*) *¸* 1*; 8i;*

where *x* is the training example and *yi* is the correct output

for the *ith* training example. Intuitively the classi¯er with

the largest margin will give low expected risk, and hence

better generalization.

6. ENSEMBLE CLASSIFIER

Base classi¯ers are constructed by applying associative

classi¯er and SVM using di®erent feature representations

on di®erent training sets (denoted by *DB T*1 and *DB T*2).

Coupled with the two di®erent feature representations, we

have four di®erent settings for training base classi¯ers: *DB T*1

with API calls, *DB T*1 with interpretable strings, *DB T*2

with API calls, and *DB T*2 with interpretable strings. Us-

ing the two di®erent classi¯cation methods, we thus obtain

8 di®erent base classi¯ers. A simple voting scheme is used

to combine base classi¯ers. For an input ¯le, each base clas-

si¯er casts a vote for its prediction: i.e., 1 if the input ¯le is

predicted to be malicious and 0 otherwise. Therefore after

classi¯er voters, each ¯le sample can obtain a score ranging

from 8 to 0. If two ¯le sample have the same score, they

will be ranked by their matching association classi¯cation

rules' *Â*2 (chi-square) values [21] in descending order. IFSS

system then generates a ¯le scoring list which is a ranked

list of all input ¯le samples from the gray list. The ¯le score

list is simple for virus analysts to interpret and understand.

Virus analysts can then look at the top ranked ¯le samples

and manually authenticated and rejected those samples. In

practice, each virus analyst can analyze 20 new ¯le samples

per day and they pick the top 100 ¯le samples from the ¯le

scoring list for manual inspection. These manually labeled

¯le samples can then be used as new training data to im-

prove the system.

7. EXPERIMENTAL RESULTS AND

ANALYSIS

In this section, we conduct two sets of experimental stud-

ies using our data collection obtained from the Anti-virus

Lab of Kingsoft to compare our IFSS with other classi¯ers:

(1) The ¯rst set of experiments is to investigate the e®ects

of feature selection. (2) In the second set of experiments,

we compare our IFSS with di®erent ensemble methods ob-

tained using di®erent combinations of feature representa-

tions, training sets and base classi¯ers. All the experimen-

tal studies are conducted under the environment of Windows

XP operating system plus Intel P4 1.83 GHz CPU and 2 GB

of RAM.

7.1 Evaluation of Feature Selection

Identifying the most representative features is critical to

improve the performance and e±ciency of the classi¯ers [17,

20]. As not all of the features contributing to malware de-

tection, we rank each API call and interpretable string using

Max-Relevance algorithm [25] and select top *k* API calls and

interpretable strings as the features for later classi¯cation.

We obtain a whole week's data collection(from Jan. 1st,

2009 to Jan. 7th, 2009) from Kingsoft Anti-virus lab to tes-

tify the validation of the feature selection method in this

set of experiments. We use six days' data collection con-

taining 530,448 PE ¯le samples for training ( half of them

are recognized as benign executables and the other half are

malicious executables mainly consisting of backdoors, tro-

jans and worms) and one day's samples including 89,626

¯les for testing. There are 7,909 API calls and 32,123 inter-

pretable strings extracted from these ¯le samples. We use

*precision*[6] and *recall* [6] of the malware class to evaluate the

performance of the classi¯cation results, which can be de-

¯ned as follows: *precision*= *TP*

*TP*+*FP* , *recall*= *TP*

*TP*+*FN* , where

*TP* is the number of malicious ¯les correctly classi¯ed, *FP* is

the number of benign ¯les incorrectly classi¯ed as malicious

and *FN* is the number of malicious ¯les incorrectly classi-

¯ed as benign. Figure 4 and Figure 5 show that the testing

performance of Associative Classi¯er(AC) changes slightly

after the number of API calls reaches 100 and the number

of interpretable strings reaches 500. So, we select top 100

API calls and top 500 interpretable strings respectively as

the features for later classi¯cation.

Figure 4: AC performance with di®erent number of

API Calls

Figure 5: AC performance with di®erent number of

Strings

1390

Feature-Classi¯er-Train TP TP+FP Recall Precision

C1:API-AC-DB T1 13 1244 21.31% 1.05%

C2:API-AC-DB T2 13 896 21.31% 1.45%

C3:STR-AC-DB T1 30 4173 49.18% 0.72%

C4:STR-AC-DB T2 9 444 14.75% 2.03%

C5:API-SVM-DB T1 13 747 21.31% 1.74%

C6:API-SVM-DB T2 13 568 21.31% 2.29%

C7:STR-SVM-DB T1 29 3266 47.54% 0.89%

C8:STR-SVM-DB T2 30 803 49.18% 3.74%

Table 1: Detection results of di®erent base classi¯ers

on di®erent training sets using di®erent feature rep-

resentations. The test data is from the gray list of

Jan. 8th, 2009.

7.2 Comparisons of Different Classification

Methods

In this set of experiments, we compare our IFSS with

di®erent ensemble methods obtained using di®erent com-

binations of feature representations, training sets and base

classi¯ers. In particular, we use: (1) API calls and inter-

pretable strings as diverse features, (2) *DB T*1 consisting of

491,733 PE ¯le samples obtained from the history data set

of Kingsoft Anti-virus lab, and *DB T*2 containing 530,448

PE ¯les which is the data collection of the week from Jan.

1st, 2009 to Jan. 7th, 2009, (3) associative classi¯er de-

scribed in Section 5.1 and linear SVM [14] implemented in

LibLinear package as heterogenous base classi¯ers, to con-

struct di®erent classi¯ers. To evaluate the performance of

di®erent classi¯ers and ensembles, we use the gray list of

Jan. 8th, 2009 obtained at Kingsoft Anti-virus lab. We

randomly sample 10% from the gray list for testing. The

test dataset contains 12,365 ¯les, 61 of which are malware

samples.

Table 1 shows the detection results of di®erent base clas-

si¯ers on di®erent training sets using di®erent feature rep-

resentations. From Table 1, we observe that the precision

of each classi¯er is too low and the number of the ¯le sam-

ples misclassi¯ed as malware is too large. Obviously, the

single classi¯cation result is infeasible for real applications.

Ensemble classi¯ers are quite popular in many data min-

ing applications due to their potential for e±cient parallel

implementations and high accuracy.

Table 2 and Figure 6 show the detections results of di®er-

ent ensembles constructed by di®erent combinations of the

feature representations, training sets and base classi¯ers. In

particular, E1-E4 are the ensemble methods constructed by

a single classi¯er with a single feature representation on dif-

ferent training sets; E5-E6 are the ensemble methods con-

structed by a single classi¯er with diverse feature represen-

tations on di®erent training sets. These methods are typi-

cal approaches for constructing ensembles. For comparison

purpose, we also include the results of human expert. F1

measure, de¯ned as *F*1 = 2*£Recall£Precision*

*Recall*+*Precision* , is also used

to evaluate the classi¯cation performance of di®erent algo-

rithms. From the comparison, we observe that our IFSS

outperforms other ensembles as well as human experts.

In addition, the detection by our IFSS can be done very

e±ciently using a couple of minutes (it uses 21.5 minutes to

detect these 12,365 ¯le samples, including feature extraction

time). A virus analyst has to spend 5 days to analyze the

100 ¯le samples in the gray list, since he/she can analyze 20

Ensemble TP Recall Precision F1

E1:C1+C2 2 3.28% 2% 0.0248

E2:C3+C4 3 4.92% 3% 0.0373

E3:C5+C6 3 4.92% 3% 0.0373

E4:C7+C8 4 6.56% 4% 0.0497

E5:C1+C2+C3+C4 12 19.67% 12% 0.1491

E6:C5+C6+C7+C8 9 14.75% 9% 0.1118

IFSS:C1-C8 35 57.38% 35% 0.4348

Human Expert 2 3.28% 2% 0.0248

Table 2: Detection results of di®erent ensembles.

Remarks: We select the top 100 ¯les from the rank-

ing list generated by each ensemble according to the

simplest voting and ranking mechanism described in

Section 6 and evaluate the performances of di®er-

ent ensembles. For comparison purpose, our virus

analysts also select 100 ¯les from the gray list to

analyze.

new ¯le samples per day. Our case studies shows that the

percentage of malware samples in the gray list is about 0*:*5%

while the percentage of malware samples in the top 100 ¯les

samples of the ¯le scoring list generated by our IFSS system

is 35%. Because of its high e±ciency and e®ectiveness, our

IFSS system makes it practical for human experts to inspect

the top rank ¯les.

Figure 6: F1 measures of di®erent ensembles based

on part of the gray list of Jan. 8th, 2009.

8. REAL APPLICATION OF IFSS

It should be pointed out that our IFSS system for malware

detection has already been incorporated into the product of

Kingsoft's Anti-virus software. Figure 7 shows the interface

of the IFSS system. We call the new scanning tool of King-

soft's Anti-Virus software which incorporates IFSS system

as KS-IFSS. The old scanning tool of Kingsoft's Anti-Virus

software is referred as KS. The main purpose of IFSS is to

help virus analysts ¯nd out malware samples in the gray list

on which all other popular scanners fail and to improve the

malware detection ability of anti-virus software. Therefore,

we apply KS-IFSS in real applications and compare with

other popular scanners(including KS) to testify its malware

detection ability and e±ciency on the daily data collection.

8.1 Detection Ability of KS-IFSS

In this section, we apply KS-IFSS in real applications to

testify its detection ability of the daily data collection. Table

3 illustrates the daily data collection obtained from Kingsoft

Anti-virus lab for the week of Jan. 25th, 2009 to Jan. 31st,

2009.

1391

Figure 7: The interface of IFSS system.

Day Date All Files Malware Benign Files

1 2009/01/25 407,882 42,608 51,595

2 2009/01/26 516,715 44,204 59,245

3 2009/01/27 551,120 44,813 50,297

4 2009/01/28 597,767 47,796 38,982

5 2009/01/29 312,372 49,077 65,113

6 2009/01/30 520,761 57,144 66,022

7 2009/01/31 705,620 55,523 54,489

Sum 3,612,237 341,165 385,723

Table 3: Daily data collection for the week of Jan.

25th, 2009 to Jan. 31st, 2009.

3,612,237 ¯le samples are collected in total: 2,885,349 of

which are gray ¯les, 385,723 of which are benign ¯les, and

341,165 of which are malware samples detected by all of the

four anti-virus scanners. For the gray ¯les, we just review

the malware samples detected by KS-IFSS. We examine KS-

IFSS's malware detection ability and FP(False Positive)rate

which is the ratio of benign ¯les misclassi¯ed as malicious

in comparison with some of the popular scanning tools like

NOD32, Kaspersky and KS. For comparison purpose, we

use all of the Anti-virus scanners' newest versions of the

base of signature on the same day. Table 4 and Figure 8

show that KS-IFSS outperforms other Anti-virus scanners

on malware detection ability, since it can detect the mal-

ware from the gray list while all the popular scanners fail.

Figure 9 shows that KS-IFSS outperforms other Anti-virus

scanners on FP(False Positive)rate.

Day Perf. KS-IFSS KS NOD32 Kaspersky

1 DRate 91.53% 87.33% 81.56% 72.85%

2 DRate 91.98% 87.72% 82.69% 66.09%

3 DRate 91.28% 88.24% 83.42% 65.72%

4 DRate 92.84% 89.21% 84.95% 62.95%

5 DRate 92.89% 89.97% 86.27% 75.47%

6 DRate 93.02% 89.68% 88.66% 79.12%

7 DRate 91.76% 89.40% 85.37% 86.36%

Table 4: Malware detection results of di®erent Anti-

Virus Scanners. Remarks: DRate means the detec-

tion rate of the Anti-Virus Scanner which is the ratio

of malware correctly classi¯ed.

8.2 Detection Efficiency of KS-IFSS

In this set of experiments, we compare the e±ciency of

our KS-IFSS with di®erent Anti-virus scanners. We also

use the daily malware data collection, from Jan. 25th, 2009

to Jan. 31st, 2009, described in Section 8.1 to testify the de-

tection e±ciency of each Anti-virus scanner. The results in

Figure 10 illustrate that KS-IFSS achieves higher e±ciency

than other scanners when being executed in the same envi-

ronment.

Figure 8: Comparisons of malware detection ability

for di®erent Anti-Virus Scanners.

Figure 9: Comparisons of FP rate for di®erent Anti-

Virus Scanners.

9. CONCLUSION

In this paper, we present an an intelligent ¯le scoring sys-

tem (IFSS) for malware detection from gray list. IFSS uses

an ensemble framework and it has several favorable traits

including diverse feature representations, dynamic training

sets, heterogeneous base classi¯ers and human-in-the-Loop.

In addition, IFSS performs simultaneous model construction

and testing. With these properties, IFSS makes it practical

for virus analysts to identify malware samples from the huge

gray list and improves the detection ability of anti-virus soft-

ware.

IFSS has already been incorporated into the scanning tool

of Kingsoft's Anti-Virus software. The case studies on large

data collections on the the gray list and real daily data col-

lection obtained from the Anti-virus Lab of Kingsoft cor-

poration demonstrate the e®ectiveness and e±ciency of our

IFSS system.

1392

Figure 10: Malware detection e±ciency of di®erent

Anti-Virus Scanners

Acknowledgments

The authors would also like to thank the members in the

Anti-virus Lab at Kingsoft Corporation for their helpful dis-

cussions and suggestions. The work of Y. Ye an Q. Jiang

is partially supported by the National Science Foundation

of China under Grant #10771176 and Guangdong Province

Foundation under Grant #2008A090300017. The work of T.

Li is supported in part by the US National Science Founda-

tion Under grant IIS-0546280 and by multiple IBM Faculty

Awards.

10. REFERENCES

[1] R. Agrawal and T. Imielinski. Mining association rules

between sets of items in large databases. In *Proceedings of*

*SIGMOD*, 1993.

[2] A.Kolcz, X.Sun, and J.Kalita. E±cient handling of

high-dimensional feature spaces by randomized classi¯er

ensembles. In *Proceedings of KDD'02*, 2002.

[3] E. Baralis and P. Torino. A lazy approach to pruning

classi¯cation rules. In *Proceedings of ICDM'02*, 2002.

[4] R. Bayardo, R. Agrawal, and D. Gunopulos.

Constraint-based rule mining in large, dense databases. In

*Proceedings of ICDE'99*, 1999.

[5] U. Bayer, A. Moser, C. Kruegel, and E. Kirda. Dynamic

analysis of malicious code. *J Comput Virol*, 2:67{77, May

2006.

[6] N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P.

Kegelmeyer. SMOTE: Synthetic minority over-sampling

technique. *Journal of Arti¯cial Intelligence Research*,

16:321{357, 2002.

[7] M. Christodorescu, S. Jha, and C. Kruegel. Mining

speci¯cations of malicious behavior. In *Proceedings of*

*ESEC/FSE'07*, pages 5{14, 2007.

[8] Thomas G. Dietterich. Machine-learning research: Four

current directions. *AI Magazine*, 18(4):97{136, 1997.

[9] Thomas G. Dietterich. Ensemble methods in machine

learning. In *Proceedings of the First International*

*Workshop on Multiple Classi¯er Systems*, pages 1{15.

[10] D.K.S.Reddy and A.K.Pujari. N-gram analysis for

computer virus detection. *J Comput Virol*, 2:231{239,

November 2006.

[11] Saso D·zeroski and Bernard ·Zenko. Is combining classi¯ers

with stacking better than selecting the best one? *Mach.*

*Learn.*, 54(3):255{273, 2004.

[12] Eric Filiol, Gregoire Jacob, and Michael Le Liard.

Evaluation methodology and theoretical model for antiviral

behavioural detection strategies. *Journal in Computer*

*Virology*, 3(1):27{37, 2007.

[13] J. Han, J. Pei, and Y. Yin. Mining frequent patterns

without candidate generation. In *Proceedings of SIGMOD*,

pages 1{12, May 2000.

[14] C. Hsu and C. Lin. A comparison of methods for multiclass

support vector machines. *IEEE Trans. Neural Networks*,

13:415{425, 2002.

[15] H.Yu, J.Yang, and J.Han. Classifying large data sets using

svms with hierarchical clusters. In *Proceedings of KDD'03*,

2003.

[16] I.W.Tsang, J.T Kwok, and P.M.Cheung. Core vector

machines: Fast svm training on very large data sets.

*Journal of Machine Learning Research*, 6:363{392, 2005.

[17] A. Jain, R. Duin, and J. Mao. Statistical pattern

recognition: A review. *IEEE Trans. Pattern Analysis and*

*Machine Intelligence*, 22:4{37, 2000.

[18] N. Japkowicz and S. Stephen. The class imbalance problem:

A systematic study. In *Intelligent data Analysis 6(5)*, pages

429{450, 2002.

[19] J. Kolter and M. Maloof. Learning to detect malicious

executables in the wild. In *Proceedings of KDD'04*, 2004.

[20] P. Langley. Selection of relevant features in machine

learning. In *Proceedings of AAAI Fall Symp.*, 1994.

[21] W. Li, J. Han, and J. Pei. CMAR: Accurate and e±cient

classi¯cation based on multiple class-association rules. In

*Proceedings ICDM'01)*, pages 369{376, 2001.

[22] B. Liu, W. Hsu, and Y. Ma. Integrating classi¯cation and

association rule mining. In *Proceedings of KDD'98*, pages

80{86, 1998.

[23] M. Mahta, R. Agrawal, and J. Rissanen. SLIQ: A fast

scalable classi¯er for data mining. In *EDBT-96*, 1996.

[24] Christopher J. Merz. Using correspondence analysis to

combine classi¯ers. *Machine Learning*, 36(1-2):33{58, 1999.

[25] H. Peng, F. Long, and C. Ding. Feature selection based on

mutual information: Criteria of max-dependency,

max-relevance, and min-redundancy. *IEEE Trans. Pattern*

*Analysis and Machine Intelligence*, 27, 2005.

[26] M. Schultz, E. Eskin, and E. Zadok. Data mining methods

for detection of new malicious executables. In *Proceedings.*

*2001 IEEE Symposium*, pages 38{49, 2001.

[27] A. Sung, J. Xu, P. Chavez, and S. Mukkamala. Static

analyzer of vicious executables (save). In *Proceedings of the*

*20th Annual Computer Security Applications Conference*,

2004.

[28] G.J. Tesauro, J.O. Kephart, and G.B. Sorkin. Neural

networks for computer virus recognition. *IEEE Expert*,

11:5{6, 1996.

[29] F. Thabtah. A review of associative classi¯cation mining.

In *The Knowledge Engineering Review, Vol. 22:1*, pages

37{65, 2007.

[30] J. Wang, P. Deng, Y. Fan, L. Jaw, and Y. Liu. Virus

detection using data mining techniques. In *Proceedings*

*ICDM'03*, 2003.

[31] Y. Wang, Q. Xin, and F. Coenen. A novel rule ordering

approach in classi¯cation association rule mining. In

*Proceedings of ICDM Workshop*, 2007.

[32] David H. Wolpert. Stacked generalization. *Neural*

*Networks*, 5(2):241{259, 1992.

[33] Y. Ye, D. Wang, T. Li, and D. Ye. IMDS: Intelligent

malware detection system. In *Proccedings of KDD'07)*,

2007.

[34] X. Yin and J. Han. CPAR: Classi¯cation based on

predictive association rules. In *Proceedings of SIAM*

*International Conference on Data Mining (SDM-03)*, pages

331{335, 2003.

1393