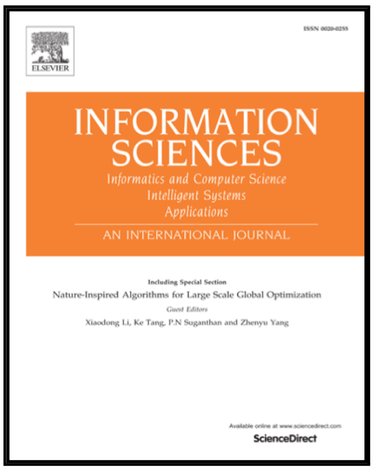

A Three-way Decision Making Approach to Malware Analysis Using Probabilistic Rough Sets

Mohammad Nauman, Nouman Azam, JingTao Yao

PII: S0020-0255(16)30896-9
DOI: 10.1016/j.ins.2016.09.037
Reference: INS 12533

To appear in: *Information Sciences*

Received date: 28 September 2015
Revised date: 17 August 2016
Accepted date: 14 September 2016

Please cite this article as: Mohammad Nauman, Nouman Azam, JingTao Yao, A Three-way Decision Making Approach to Malware Analysis Using Probabilistic Rough Sets, *Information Sciences* (2016), doi: 10.1016/j.ins.2016.09.037

Highlights

- We employ three-way decisions approach to malware analysis using probabilistic rough sets.
- Architecture for malware analysis based on three-way decisions is proposed.
- Experimental results on UNM dataset advocates for the use of three-way decisions in malware analysis

A Three-way Decision Making Approach to Malware Analysis Using Probabilistic Rough Sets

Mohammad Nauman[a], Nouman Azam[*,a] and JingTao Yao[b]

[a] *National University of Computer and Emerging Sciences, Peshawar, Pakistan*
[b] *Department of Computer Science, University of Regina, Regina, SK S4S 0A2, Canada*

Abstract

Malware analysis aims to identify malware by examining applications behaviour on the host operating system. A common issue in malware analysis is how to mitigate and handle the false decisions such as false positives. Existing approaches which are based on two-way decisions (such as acceptance and rejection) for classifying applications behaviour result in two shortcomings. Firstly, the two-way decisions are rigid and strict in the sense that they demand that a classification decision must be made irrespective of the quality of available information. This potentially leads to wrong classification decisions whenever we do not have sufficient and complete information. Secondly, two-way decisions do not involve any explicit mechanism for dealing with the false decisions at the model level. The existing approaches generally work like an add-on to learning models and are only exercised after incorrect decisions are being made by the learning models. This results in additional processing and increases the complexity of the task. In this paper, we investigate a three-way decision making approach based on decisions of acceptance, rejection or deferment. The added deferment decision option provides flexibility for delaying a certain decision whenever we do not have sufficient information. Moreover, it aims to mitigate the false decisions at the model level by determining a tradeoff between different properties of decision making such as accuracy, generality and uncertainty. We considered three-way decisions based on two probabilistic rough set models, namely, game-theoretic rough sets (GTRS) and information-theoretic rough sets (ITRS) in this study. An

[*]Corresponding Author: Email: nouman.azam@nu.edu.pk (Nouman Azam); Phone, ++92 111 128 128 143

architecture of malware analysis realized with probabilistic rough sets based three-way decisions is proposed. A new algorithm termed as sequentially stackable linux security (SSLS) based on the proposed architecture is presented. Experimental results on the system call sequences from the UNM data set advocate for the use of three-way decisions in malware analysis.

Keywords: Malware analysis, probabilistic rough sets, three-way decisions, information-theoretic rough sets, game-theoretic rough sets

1. Introduction

There is an increase in dependence on computing devices to store personal and enterprise data. Consequently, critical decisions are based on the support provided by these devices [9, 47]. While this has led to fundamental changes in many areas of human life and enabled unprecedented and novel solutions to age-old problems, it does not come without its own set of issues [21]. The protection of digital devices is a complex issue that is not generally well understood by the users of these devices [25, 31, 52, 60, 61]. What's more, it leads to several types of threats of varying levels and impact [59]. According to Symantec's Internet security threat report, there was a 91% increase in targeted attack campaigns in 2013 [59]. Moreover, they reported that no less than 38% users had experienced mobile cyber crime in 2014 and 2015 [59]. An emergent and exigent need is therefore to protect the data residing on host machines from malicious parties. The set of techniques and tools used to ensure such protection are collectively referred to as malware analysis [24]. Generally speaking, malware analysis techniques are based on deciding whether or not an application executing on the host machine is harmful to the data or the system. The applications that are harmful are termed as malicious and those that are not harmful are termed as benign [42]. This is essentially a two-way classification of applications behaviour.

An important issue in malware classification is how to deal with false positive or false alarms, i.e., incorrectly identifying an application behaviour as malicious [36]. The existing approaches are generally based on two-way decisions in the form of acceptance and rejection for classifying an application behaviour, i.e., we either decide to accept an application behaviour as malicious or we reject an application behaviour as malicious. Several malware analysis techniques and approaches have been proposed in the past based on the two-way classification decisions [13, 28, 35, 42, 55].

The two-way classification approaches result in two shortcomings when dealing with false decisions. Firstly, it strictly requires that a classification decision is always reached irrespective of the quality of available information. Since we may not have always high quality information, the requirement of always making an immediate decision may be unrealistic and will result in generating false classification decisions (including false positives). Secondly, the two-way approach does not provide any explicit mechanism for dealing with the false decisions at the model level. For instance, one of the most efficient techniques for dealing with false decisions (reporting a decrease of 25% reduction in false positive rate) is presented in [57]. It mitigates the false positives based on a postprocessing filter which is activated after false decisions are being detected and reported by the model. Moreover, from a detailed and thorough survey of existing literature related to false positive reduction, we may note that this is common in majority of the current approaches [36]. In other words, these approaches attempt to fix and reduce the false positive rate only after the learning model has reported it. In some sense, these approaches work like an add-on to learning models and involve two steps of detection and then mitigation. Another approach in the related but slightly different network intrusion detection domain is provided in [40]. While this tackles the problem from a model perspective, it is specific to network intrusion detection and is not able to handle host-based intrusion detection targeted in our work.

In this article, we present a three-way decision making approach to malware analysis based on decisions of acceptance, rejection and deferment. We argue that three-way decision approach can overcome the two limitations and shortcomings of two-way decisions. Firstly, it is more flexible compared to two-way decisions in the sense that it provides flexibility for delaying a decision whenever we do not have complete and accurate information for classifying an application behaviour. Secondly, it aims to mitigate the false decisions at the model level by implementing a tradeoff between different properties of decisions such as accuracy, generality and uncertainty thereby avoiding the need for extra components, add-ons and steps. We examine three-way decisions based on two probabilistic rough set models [65, 67]. The probabilistic rough sets are based on probabilistic association between objects and a target concept to decide the inclusion of objects into one of the three regions, namely, positive, negative and boundary regions. The three regions are used to induce three-way decisions. The three regions and the implied three-way decisions in the probabilistic rough sets are defined and controlled by a pair of thresholds [68]. There are different forms and models of probabilistic rough sets based on how these thresholds are obtained and interpreted. We consider two such models, i.e., game-theoretic rough sets (GTRS) [3, 33, 64] and information-theoretic rough sets (ITRS) [22, 23] to examine the application of probabilistic rough sets in malware analysis. Moreover, we examine and define five approaches based on the GTRS and ITRS by employing different measures and iterative methods. A target architecture realized with three-way decisions based on a typical malware analysis technique known as sequence time-delay embedding is presented. A new algorithm called sequentially stackable linux security (SSLS) based on the proposed architecture is presented. Experimental results on the UNM dataset suggest for the use of three-way decisions in malware analysis. A false positive rate as minimum as 8.5% was reported for the considered dataset which is comparable to existing approaches.

2. Background Knowledge

2.1. Malware Analysis

Malware analysis is the study aiming to develop, examine and explore approaches and techniques to classify machine executables into either harmful or non harmful [74]. Classification techniques plays a critical role in the analyzing malware. The machine executables which are harmful are called malicious and the machine executable which are not harmful are called benign. Malicious executables perform some unintended, usually harmful operations, on the host system. Benign executables are those that perform intended operations, which are not harmful, on the host system. Approaches for malware analysis are broadly categorized as signature-based and heuristics-based approaches [24].

In signature-based approaches, the binary executables are transformed to represent hashes which are matched with a database of known malicious samples [10, 15, 18, 29, 44]. If a match is found, the classifier marks the target binary as being malicious. This approach requires a huge and constantly changing database of known malware, which is highly infeasible in the face of constantly evolving malware [46]. Moreover, it has been reported in the past that it is possible to achieve malicious behaviour from benign application without making any modification to their binary files [8, 14, 54]. Due to this specific limitation, while there have been several efforts in the academia to increase the effectiveness of signature-based techniques, the industry seems

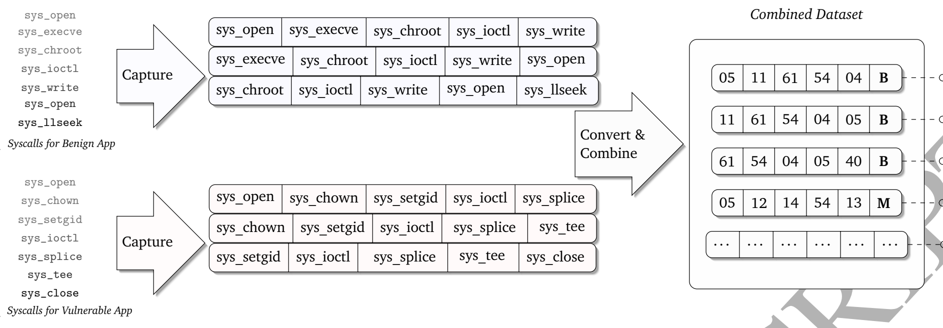
sys_open
sys_execve
sys_chroot
sys_ioctl
sys_write
sys_open
sys_llseek
Syscalls for Benign App
Capture
sys_open | sys_execve | sys_chroot | sys_ioctl | sys_write
sys_execve | sys_chroot | sys_ioctl | sys_write | sys_open
sys_chroot | sys_ioctl | sys_write | sys_open | sys_llseek
sys_open
sys_chown
sys_setgid
sys_ioctl
sys_splice
sys_tee
sys_close
Syscalls for Vulnerable App
Capture
sys_open | sys_chown | sys_setgid | sys_ioctl | sys_splice
sys_chown | sys_setgid | sys_ioctl | sys_splice | sys_tee
sys_setgid | sys_ioctl | sys_splice | sys_tee | sys_close
Convert & Combine
Combined Dataset
05 11 61 54 04 B w_1
11 61 54 04 05 B w_2
61 54 04 05 40 B w_3
05 12 14 54 13 M w_4
… … … … … … w_n

Figure 1: Capturing System Calls from Malicious and Benign Applications [45]

to be focusing more on alternatives (the interpretation of signatures as used in practice may therefore be different compared to its interpretation in the research literature). In order to mitigate and overcome this core limitation, heuristics-based approaches measure the operations performed by a target application to decide if the operations would lead to undesirable results [1, 42]. This means that heuristics-based approaches are more effective and capable of detecting actual behaviour instead of relying on static representations provided by the hashes.

The heuristics-based approaches are generally based on sequence time-delay embedding approach [42]. This approach is based on the fact that an application's behaviour is directly proportional to the sequence of system calls it makes. By capturing these system calls and training a learning algorithm on sequences of benign and malicious datasets, it is possible with a certain degree of accuracy to identify newly appearing malicious applications without having any knowledge about their executables in advance. Another important aspect is how to incorporate and implement the classifier in the existing operating systems. From an implementation perspective, the sequences of system calls can be captured through a monitoring tool such as `strace` in Linux, through API hooking mechanism `SetWindowsHookEx()` in Windows or the Linux Security Modules (LSM) framework in nix platforms.

Once the system call sequences are captured, they are transformed to windows called sliding windows. The windows are termed as sliding due to the way they are constructed [75]. Figure 1 shows the semantics of sliding window generation from traces of system calls made by malicious and benign applications. The sequences are not sliced at regular intervals, instead a moving window technique is used to capture all behaviours. The first n calls make up the first window. Then, the windows is moved forward shifting the first call out and leaving room for the $n+1$ calls on the right. Inserting these calls create the second window and the window is shifted once again to the right [26]. This process ensures that all behaviours are captured, which would otherwise have been missed if a slicing of the sequences would have been performed. How many system calls will be included in a sliding window is generally configurable by system administrator [58]. A larger value for window size will capture behaviours over a longer period of time and involves more processing and computations. A smaller value, on the other hand, involves lesser computations and is therefore faster but it may miss some behaviours which are only detected with a larger size. The best size value for the window can be empirically determined.

After the transformation of sequences of calls to sliding windows, the problem is reduced to that of training a classifier on this dataset. Several studies have been reported for this in the past. One of the most recent approaches uses the concept of extreme learning machine which uses a neural network approach on the UNM and other datasets [19]. Another highly successful idea uses the concept of n-spaced hypergrams to capture both the long-term history and the recent behaviour of an application in a single parameter using the UNM dataset [42]. A similar approach based on n-grams for analyzing HTTP attacks was presented in [48]. An adaptive threshold approach based on statistical and information theoretic analysis on host and network-based datasets (including the UNM dataset) were reported in [1]. They have reported a reduction of 50% in the false positive rate but do not report the exact false positive rate. The same UNM dataset was also used in a remote attestation setting achieving an accuracy of 82% on the remote machine [58]. Their main effort has been related to trustworthy reporting of the measurements to the remote end rather than an improvement over detection rates. Other related studies may be found in references [5, 12, 13, 19, 77].

Majority, if not all, of these previous techniques attempt to classify all application behaviours as either malicious or benign. This is reasonable as the end goal is to recognize malicious behaviours. However, it may not be always possible to confidently reach two-way classification decisions due to ambiguities in the available information. For instance, a malicious application may perform as benign in the aim to deceive the analysis engine. Forcing the classifier to produce a binary decision in such cases invariably leads to errors in classification results. Over time, this leads to the behaviour of the system users where they start to ignore warnings given by the system. This lack of trust on part of the users renders even correct results generated by the system [62].

In order to overcome this issue and other similar issues, we propose three-way decision making approach to malware analysis.

2.2. Three Way Decisions

The theory of three-way decisions emerged from the rough set notions of representing an undefinable concept using three regions [71]. The theory however is general in the sense that it considers rough set theory as one of many possible ways for constructing three-way decisions. Other theories that may be utilized for inducing three-way decisions include interval sets [41], shadowed sets [49, 50, 51], modal logic [11, 66] and orthopairs [16]. The essential idea of three-way decisions is to divide a universal set into three pair-wise disjoint regions, such as the positive, negative and boundary regions. The three regions are then processed to make decisions such as accept, reject and deferment [71]. The general framework of three-way decisions was outlined by Yao in [70, 71]. We briefly review the framework for the sake of completeness.

Considering a finite nonempty set of objects denoted by U and a set of criteria denoted by C. One may divide U based on C into three disjoint regions, POS, NEG and BND called as positive, negative and boundary regions, respectively. The three regions can be used to generate rules for three-way decisions. In particular, the POS region generates rules for decisions of acceptance, the NEG region generates rules for rejection and the BND region generates rules for decision of non-commitment or deferment. How to determine the inclusion of an object in a particular region will depend on the degree or level to which it satisfies the criteria in C. The POS region consists of objects whose satisfiability (for criteria in C) is at or above a certain level of acceptance. The NEG region consists of objects whose satisfiability is at or below a level of rejection. The BND region consists of objects whose satisfiability is above the rejection level but below the acceptance level. There are different issues or challenges for any three-way decision making model [34, 70]. They include identification and definition of set of values for measuring the satisfiability of objects and a set of values for measuring the non-satisfiability of objects, construction and meaning of evaluation functions for evaluating

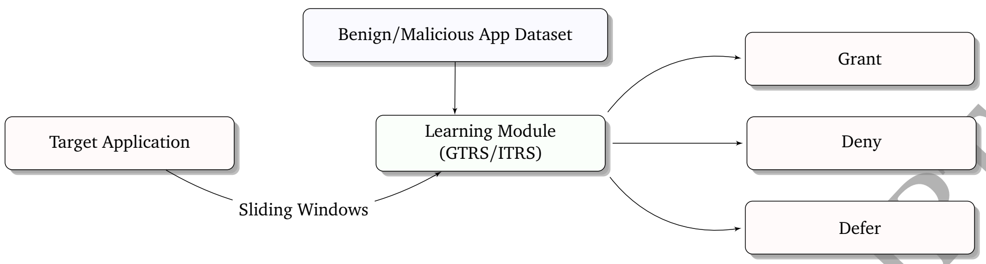

Figure 2: High-Level View of the Proposed Architecture with Three-Way Decisions

objects, the determination and interpretation of designated values for acceptance and designated values for rejection. Based on how these three issues are handled and interpreted, we may have different three-way decision making models and approaches. We focus on three-way decisions with probabilistic rough sets in this article.

Three-way decisions may provide a useful mechanism for classifying malware. Consider a scenario where a malware analysis engine is accurate most of the time but occasionally raises a red flag on a benign executable (an issue related to false alarm). Repeated cases of such incorrect classification decisions will lead to a compromised system [6]. There may be several reasons for such false classification decisions. Among several possible reasons, such as poorly designed classifier, incorrect user input, misleading application behaviour, an important reason may be that incomplete and insufficient information was available for making a classification decision. To deal with this and similar reasons, three-way decisions can play a role by exercising the deferment option which can safeguard against some of the misclassifications.

3. An Architecture of Malware Analysis with Three-way Decisions

We utilize the concept of three-way decisions in malware analysis by proposing a malware architecture. This architecture supports multiple malware analysis techniques and is based on the Linux Security Modules (LSM) framework provided by the Linux kernel. Please be noted that the approach is not limited to the Linux operating system and is portable to any operating system that supports system call hooks as described in Section 2.1. We consider Linux due to open nature of its source code and ease of incorporating our approach.

A high-level view of our proposed architecture is shown in Figure 2. The central component of the architecture is the learning module, which is based on GTRS and ITRS. The learning module is trained using a dataset gathered from benign and malicious applications. When the target application makes a system call, the updated sliding windows for this application are computed. The learning module returns an access decision according to the recent behaviour as measured by the sliding windows. The end result is a three-way decision in the form of grant, deny or deferment. The option will be exercised in cases where there is insufficient information to confidently decide whether to grant or deny the access request.

A detailed view of this proposed architecture is shown in Figure 3. The existing Linux architecture supports both Discretionary Access Control model (which allows creators of files and folders to set permissions) and Mandatory Access Control model (which allows the system administrator to enforce organization-wide policies). The latter model is being utilized in the proposed architecture. The Mandatory Access Control is implemented in Linux using Linux Security Modules (LSMs). Each LSM can enforce a particular type of policy. For instance, an antivirus module can be written as an LSM which checks the integrity of each executable before it is loaded in the system's memory. Moreover, it is possible to have multiple LSMs operating in the same machine. It is therefore possible to have extended security by enabling multiple antivirus modules each performing its own analysis. If either one of the installed modules denies the execution of the target, the application will be restricted. We extend this architecture by introducing two key features, i.e., deferment and priorities. We explain both of these by considering a specific scenario below.

A user installs two integrity-checking modules in her system to ensure that no malicious application will be able to damage her sensitive files. The first module is quite responsive and performs the analysis quite quickly. However, it is prone to a high rate of false positives and thus rejects execution of even some benign executables. The second module has a much better detection rate but is quite slow and suffers from performance and usability issues. In our proposed architecture, we resolve this dilemma by assigning a higher priority to the first module and a lower one to the second. Moreover, we reduce the rate of false positives in the first module by adding the capability of returning a deferment decision in cases where sufficient proof of being malicious is not found. The second module is called if and only if the first one results in deferment. This leads to a faster performance in cases where

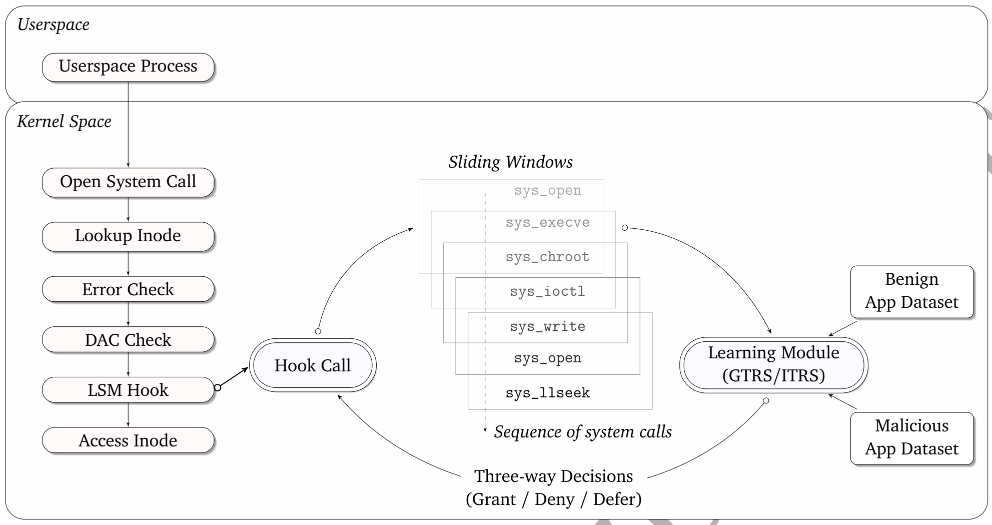

Figure 3: Detailed View of the Proposed Architecture with Three-Way Decisions [45]

the first module is successfully able to classify the target application and a better detection rate in cases where it is incapable of making an immediate right with a high level of certainty. Essentially, by incorporating three-way decisions, we can combine the strengths of more than one detection modules thus leading to better detection rates and higher runtime performance.

The concise semantics of this procedure is presented as **Algorithm 1** which is also referred to as SSLS. We refer to all the modules that are capable of returning a deferment decision as "stackable". The inspiration of this name comes from the Linux kernel jargon where stackable security modules are those that are called one after the other. The difference in semantics here is that we only call those modules stackable which can return a deferment decision since they can be thought of as being placed on top of one another. The highest priority module is called first and if it fails to provide a decision (a deferment result), another one with a lower priority is called. All the other modules i.e. those which cannot return a deferment decision are termed as non-stackable. All stackable modules are assigned a priority by the system administrator at kernel compilation time. The algorithm takes all the LSMs as input and first separate out all the modules that can be stacked and arranges them in decreasing order of priority (line 2). It then calls each of the stackable modules in turn (lines 5–11). If a decision to grant or deny is

Algorithm 1 Sequentially Stackable Linux Security (SSLS)

```
Input: List of all LSMs ($\Lambda$)
Output: Decision to grant or deny $d \in \{\text{Grant, Deny}\}$
 1: Get all stackable LSMs $\Lambda_s$
 2: sortByPriority($\Lambda_s$)
 3: Set all other LSMs as unstackable $\Lambda_u \leftarrow \Lambda - \Lambda_s$
 4: $d \leftarrow$ 'Grant' ; Use default-allow policy
 5: for all $\lambda \in \Lambda_s$ do
 6:     $d_\lambda \leftarrow$ getDecision($\lambda$)
 7:     if $d_\lambda \neq$ 'Deferment' then
 8:         $d \leftarrow d_\lambda$
 9:         break
10:     end if
11: end for
12: If $d =$ 'Deferment' then $d \leftarrow$ 'Grant'
13: while $d \neq$ 'Deny' do
14:     $\lambda \leftarrow$ getNextLSM($\Lambda_u$)
15:     $d \leftarrow$ getDecision($\lambda$)
16: end while
17: return $d$
```

made by any of the modules, it is accepted as the umbrella decision for all the stackable modules (lines 8, 9). In case of a deferment, the stackable module with a lower priority is called to check if it would return a deny or grant. This module can perform a more rigorous and possibly more time-consuming analysis of the call to make the decision. In this way, all the stackable modules are called one after the other until a definite decision is made as either deny or grant. In this paper, we propose the semantics of the highest priority LSM that can make three-way decisions using the sequence-time delay embedding technique using sliding windows as explained in Section 2.1. We also note that in case there is a larger dataset available, sliced windows can also be used in place of sliding windows.

Afterwards, the non-stackable modules are called to ensure that none of them returns a denial decision (lines 13–16). This last step is necessary to ensure backwards compatibility with existing security systems. We use a default-allow policy that means that a denial will be issued only if at least one of modules returns a deny decision (line 4). If all of the modules result in deferment, the decision will stand as granted (line 12). In case of a grant decision, access to the system call is permitted and the calling application proceeds normally. In case of a deny decision, access to the system call is denied and the result is sent to the issuing application where it can be handled by the userspace logic.

For the demonstration of our proposed architecture, we have implemented an LSM called System Call Behaviour Monitor. It is configurable in the Linux kernel's `make menuconfig` screen. Moreover, the priorities of the stackable modules can also be set in the same screen thus allowing for maximum flexibility in our architecture. The details of exactly how three-way decisions are made within this module will be discussed in the rest of the paper.

Please be noted that a similar notion of two level testing or primary secondary testing is generally studied in the context of multiple classifier systems under the issue of multiple classifier topologies [38]. Although there are some studies on multiple classifier systems in the malware domain such as [43, 53, 56], there are no specific studies on the classifier topologies. The study which comes closer to the proposed idea is presented in [17] where a two level approach is suggested. In the first level, the malicious and benign behaviours are separated and then at the second level, more detailed analysis is carried out to determine the specific type of malicious behaviour. Our approach is different in the sense that in the second level, we do not restrict our analysis to a specific category. Another relevant study from a closely related domain, i.e., anomaly detection may be found in [20]. In their first level they look at some basic statistical properties to detect anomaly in network flow and in the next stage they use a classifier to investigate the anomaly. They do not employ classifiers in both the levels to make analysis at different granular levels.

It is also important to note that the proposed three-way decision making approach is different from the commonly used primary secondary testing where provisional decisions are being confirmed via secondary testing. Here are some points in this regards.

- In three-way decision making approach, we do not make any decision for the deferred cases while in case of provisional decisions, the implications are that we make preliminary decisions which are verified based on secondary testing.

- In three-way decisions, we explicitly identify the cases which need additional exploration (or secondary testing), i.e., the deferred cases. In conventional approaches, we may not have any mechanism to identify the cases for which provisional decisions may be made. Moreover, which provisional decision to make (grant/deny) is another issue.

Finally, in the suggested approach presented in **Algorithm 1**, the deferred decisions control the workload for secondary testing (secondary testing is being carried out for the deferred cases). Since secondary testing involves additional cost, the deferred decisions must be carefully controlled by selecting appropriate threshold levels in order to achieve overall cost effectiveness. One may choose them arbitrarily. However, it will not be based on scientific reasoning and the overall cost of decision making may not be effective. In the next section, we present two different three-way models such as ITRS and GTRS that seek for scientific justifications of selecting appropriate threshold values aiming to optimize overall decision making cost.

4. Realization of the Architecture with Probabilistic Rough Sets

The overall success of the proposed architecture in analyzing malware critically depends on effective strategies and approaches for obtaining three-way decisions. We consider probabilistic rough sets for this purpose which is one of the most widely used generalization of rough sets. For the sake of completeness, we cover the basic notions and results of probabilistic rough sets.

The general form of probabilistic rough sets emerged from the studies on decision-theoretic rough sets [72, 73]. Based on the general form, the probabilistic lower and upper approximations for a concept C are defined using a pair of thresholds (α, β) as [67],

$$\begin{aligned} \underline{apr}_{(\alpha,\beta)}(C) &= \{x \in U \mid P(C|[x]) \geq \alpha\}, & (1) \\ \overline{apr}_{(\alpha,\beta)}(C) &= \{x \in U \mid P(C|[x]) > \beta\}, & (2) \end{aligned}$$

where $P(C|[x])$ is the conditional probability of a concept C with an equivalence class $[x]$ containing object x and U is the universal set of objects. The conditional probability reflects the evaluation of an object x to be in C, provided that $x \in [x]$. The three rough set regions based on lower and upper approximations are defined as,

$$
\begin{aligned}
\mathrm{POS}_{(\alpha,\beta)}(C) &= \{x \in U | P(C|[x]) \geq \alpha\}, & (3)\\
\mathrm{NEG}_{(\alpha,\beta)}(C) &= \{x \in U | P(C|[x]) \leq \beta\}, & (4)\\
\mathrm{BND}_{(\alpha,\beta)}(C) &= \{x \in U | \beta < P(C|[x]) < \alpha\}. & (5)
\end{aligned}
$$

The $\mathrm{POS}_{(\alpha,\beta)}(C)$, $\mathrm{NEG}_{(\alpha,\beta)}(C)$ and $\mathrm{BND}_{(\alpha,\beta)}(C)$ in Equations (3) - (5) are referred to as positive, negative and boundary regions, respectively. In decision making context, these three regions are referred to as regions of acceptance, rejection and deferment decisions [67]. The above framework may be used for deciding the application behaviour as being malicious or otherwise. For instance, when the probabilistic evidence i.e., $P(C|[x])$ (the concept C may be realized as the overall representation of malicious behaviour and $[x]$ may be considered as the description of the malicious behaviour) is above the upper thresholds, we may accept the behaviour as being malicious. If the probabilistic evidence is below the lower threshold, we may reject the behaviour as being malicious. If the probabilistic evidence is between the two thresholds, we may not be able to accept or reject and therefore should make a deferment decision. We provide a detailed example on how to use the probabilistic framework for analyzing malware in Section 5.

A key issue in the application of probabilistic rough sets is the determination and interpretation of thresholds (α, β) [7, 22, 69]. Although one may set them arbitrarily or by hit and trail. However, such approaches are not scalable and may require several hit and trails before reaching acceptable values [32]. Moreover, in the absence of scientific justification, explaining the choice and selection of particular thresholds, one may not feel confident about the resulting decisions.

There are different models and approaches for determining the thresholds. These approaches seek effective and optimum thresholds by examining the relationship between different quality aspects of rough sets based classification and decision making and their impact on the thresholds. We focus on two such model, i.e., game-theoretic rough sets (GTRS) [3, 33, 64] and information-theoretic rough sets (ITRS) [22, 23]. The choice of using these two models are based on the following two reasons. Firstly, unlike the some of the earliest probabilistic models which were based on restricted pairs of thresholds such as, 0.5-probabilistic model and $(0.5, \beta)$ model, the GTRS and ITRS allows for an examination of thresholds based on different aspects and properties. Secondly, whereas other models rely on some user input or domain knowledge to govern the thresholds, such as, decision-theoretic rough sets and variable precision rough sets, the GTRS and ITRS are successfully used in conjunction with learning or searching mechanisms to learn the thresholds based on the data itself [3, 23]. We now briefly discuss the GTRS and ITRS models.

4.1. Game-theoretic Rough Sets

Game-theoretic rough sets (GTRS) determine the thresholds based on a tradeoff between multiple criteria [33]. A game is implemented and analyzed for this purpose. A typical game is defined as a tuple $\{P, S, u\}$ [39], where:

- P is a finite set of n players,
- $S = S_1 \times ... \times S_n$, where S_i is a finite strategies set for player i. Each $s = (s_1, s_2, ..., s_n) \in S$ is called a strategy profile where each player i plays strategy s_i, and
- $u = (u_1, ..., u_n)$ where $u_i : S \longmapsto \Re$ is a real-valued utility or payoff function for player i.

The game solution, commonly Nash equilibrium is used in GTRS to determine possible game outcome. To explain Nash solution, consider a strategy profile $s_{-i} = (s_1, s_2, ..., s_{i-1}, s_{i+1}, ..., s_n)$ without player i strategy. The strategy profile containing strategies of all the players, i.e., $(s_1, s_2, ..., s_n)$ is therefore written as (s_i, s_{-i}). The strategy profile $(s_1, s_2, ..., s_n)$ is a Nash equilibrium, if for all the players i, s_i is the best response to s_{-i}. Mathematically, this is expressed as [39],

$$\forall i, \forall s_i^{'} \in S_i, \quad u_i(s_i, s_{-i}) \geq u_i(s_i^{'}, s_{-i}), \quad \text{where } (s_i^{'} \neq s_i) \tag{6}$$

Intuitively, Equation (6) suggests that Nash equilibrium is a strategy profile in which no player would benefit from changing his strategy, provided he knows the other players strategies.

The above game description is used in the GTRS to formulate a game. The players in the game are selected in order to highlight different criteria or aspects of rough set based decision making, such as, accuracy or applicability of decision rules [3]. Suitable measures are examined and defined to evaluate each of these criteria. The strategies are realized as direct or indirect modifications in the threshold values. Each criterion is affected in a certain way by employing different strategies. The overall objective of the game is to reach and determine an acceptable solution based on the considered criteria. The game solution leads to the determination of effective thresholds.

In this article, we considered three games in GTRS. Two of these three games are based on balancing the uncertainties of different probabilistic rough set regions while one game is based on determining a tradeoff between the properties of accuracy and generality of the rough set regions. The essential difference in these games are the consideration of different types of game players.

The objective in the games for balancing uncertainties is to reduce the overall uncertainty involved in the rough set regions. Two players are defined for this purpose, namely, the immediate decision region, denoted as I and deferred decision region, denoted as D. The player I denotes the collective uncertainty in probabilistic positive and negative regions and the player D denotes the uncertainty in the probabilistic boundary region. Changing the threshold levels to decrease the uncertainty of player I leads to an increase in uncertainty for player D [3]. A balanced solution is determined by considering a game for determining the thresholds between uncertainty of player I and player D. Based on player I and D, these two games differ in the way the uncertainty is interpreted and computed. In one game, the uncertainty is measured with the Shannon entropy. This game was previously examined in the context of text categorization and medical decision making [3, 63]. In the second game, the uncertainty is computed with Gini coefficient. This game was investigated in [76]. The third game is similar to the game discussed in the Section 5.1. This game is based on the tradeoff between the properties of accuracy and generality and was previously examined in the context of recommender systems [4]. In this study, we examine these games in the context of malware analysis. These three games will be referred to as GTRS$_E$, GTRS$_G$ and GTRS$_{(A/G)}$, respectively.

The strategies in all the three games are formulated in the same way and are considered in terms of different threshold levels. In particular, three types of strategies are considered for the players, namely, $s_1 = \alpha_\downarrow$ (decrease α), $s_2 = \beta_\uparrow$ (increase β) and $s_3 = \alpha_\downarrow\beta_\uparrow$ (decrease α and increase β). This means that strategy sets for the players are the same and given by, $\{s_1, s_2, s_3\}$.

Finally, we need to define the utility of the players. In the games based on the uncertainty analysis, i.e., GTRS$_E$ and GTRS$_G$, the uncertainties associated with different regions are used to define the payoff functions. For specific semantics of the payoff functions, please refer to [3]. In the third game, i.e.,

Table 1: Payoff table for the game

		Player 2		
		$s_1 = \alpha_\downarrow$	$s_2 = \beta_\uparrow$	$s_3 = \alpha_\downarrow\beta_\uparrow$
Player 1	$s_1 = \alpha_\downarrow$	$u_1(s_1,s_1),u_2(s_1,s_1)$	$u_1(s_1,s_2),u_2(s_1,s_2)$	$u_1(s_1,s_3),u_2(s_1,s_3)$
	$s_2 = \beta_\uparrow$	$u_1(s_2,s_1),u_2(s_2,s_1)$	$u_1(s_2,s_2),u_2(s_2,s_2)$	$u_1(s_2,s_3),u_2(s_2,s_3)$
	$s_3 = \alpha_\downarrow\beta_\uparrow$	$u_1(s_3,s_1),u_2(s_3,s_1)$	$u_1(s_3,s_2),u_2(s_3,s_2)$	$u_1(s_3,s_3),u_2(s_3,s_3)$

GTRS$_{(A/G)}$, the payoff functions are determined using Equations (20) - (21). Finally, Nash equilibrium defined in Equation (6) is used to determine the game solutions in these games.

Table 1 may be used to represent the considered games in a payoff table. The players in the payoff table, denoted as Player 1 and Player 2, may be interpreted based on a certain game. For instance, in the game between accuracy and generality, denoted by GTRS$_{(A/G)}$, the Player 1 may be realized as the measure of accuracy and the Player 2 may reflect the measure of generality. The rows of Table 1 represent the strategies of Player 1 and the columns represent the strategies of Player 2. Each cell of the table represents a strategy profile of the form (s_i,s_j), which means that Player 1 is playing strategy s_i and Player 2 is playing strategy s_j. The pair of entries in each cell denote the payoff functions for the two players corresponding to a strategy profile. We may not be able to determine effective thresholds based on a one time non-repeated game. We therefore employ a repeated learning algorithm for determining the thresholds with GTRS as proposed in [3].

4.2. Information-theoretic Rough Sets

The ITRS determines the thresholds based on minimizing the information uncertainty of the probabilistic rough set regions. Based on how the uncertainty is defined and measured, we have at least two types of information-theoretic rough sets, namely, ITRS based on Shannon entropy and ITRS based on Gini coefficient. These two types are essentially same in form but different in how uncertainty is interpreted.

Consider a partition based on a concept C, given by, $\pi_C = \{C, C^c\}$ and another partition with respect to the thresholds (α, β), given by, $\pi_{(\alpha,\beta)} = \{\text{POS}_{(\alpha,\beta)}(C), \text{NEG}_{(\alpha,\beta)}(C), \text{BND}_{(\alpha,\beta)}(C)\}$. The uncertainty in π_C with respect to the three probabilistic regions based on Shannon entropy is given as, [22]

$$\begin{aligned}
H(\pi_C|\mathrm{POS}_{(\alpha,\beta)}(C)) &= -P(C|\mathrm{POS}_{(\alpha,\beta)}(C)) \log P(C|\mathrm{POS}_{(\alpha,\beta)}(C)) \\
&\quad -P(C^c|\mathrm{POS}_{(\alpha,\beta)}(C)) \log P(C^c|\mathrm{POS}_{(\alpha,\beta)}(C)), \quad (7) \\
H(\pi_C|\mathrm{NEG}_{(\alpha,\beta)}(C)) &= -P(C|\mathrm{NEG}_{(\alpha,\beta)}(C)) \log P(C|\mathrm{NEG}_{(\alpha,\beta)}(C)) \\
&\quad -P(C^c|\mathrm{NEG}_{(\alpha,\beta)}(C)) \log P(C^c|\mathrm{NEG}_{(\alpha,\beta)}(C)), \quad (8) \\
H(\pi_C|\mathrm{BND}_{(\alpha,\beta)}(C)) &= -P(C|\mathrm{BND}_{(\alpha,\beta)}(C)) \log P(C|\mathrm{BND}_{(\alpha,\beta)}(C)) \\
&\quad -P(C^c|\mathrm{BND}_{(\alpha,\beta)}(C)) \log P(C^c|\mathrm{BND}_{(\alpha,\beta)}(C)). \quad (9)
\end{aligned}$$

The same uncertainty based on the Gini coefficient is determined as [76],

$$\begin{aligned}
G(\pi_C|\mathrm{POS}_{(\alpha,\beta)}(C)) &= 1 - P(C|\mathrm{POS}_{(\alpha,\beta)}(C))^2 \\
&\quad -P(C^c|\mathrm{POS}_{(\alpha,\beta)}(C))^2, \qquad (10) \\
G(\pi_C|\mathrm{NEG}_{(\alpha,\beta)}(C)) &= 1 - P(C|\mathrm{NEG}_{(\alpha,\beta)}(C))^2 \\
&\quad -P(C^c|\mathrm{NEG}_{(\alpha,\beta)}(C))^2, \qquad (11) \\
G(\pi_C|\mathrm{BND}_{(\alpha,\beta)}(C)) &= 1 - P(C|\mathrm{BND}_{(\alpha,\beta)}(C))^2 \\
&\quad -P(C^c|\mathrm{BND}_{(\alpha,\beta)}(C))^2. \qquad (12)
\end{aligned}$$

A particular conditional probability in above equations, say, $\mathrm{P}(C|\mathrm{POS}_{(\alpha,\beta)}(C))$ is computed as, $P(C|\mathrm{POS}_{(\alpha,\beta)}(C)) = \frac{|C \bigcap \mathrm{POS}_{(\alpha,\beta)}(C)|}{|\mathrm{POS}_{(\alpha,\beta)}(C)|}$. Other conditional probabilities are similarly obtained.

The overall uncertainty is computed as an average uncertainty of the regions [23]. The overall uncertainty based on Shannon Entropy is computed as,

$$\begin{aligned}
H(\pi_C|\pi_{(\alpha,\beta)}) &= P(\mathrm{POS}_{(\alpha,\beta)}(C))H(\pi_C|\mathrm{POS}_{(\alpha,\beta)}(C)) \\
&\quad +P(\mathrm{NEG}_{(\alpha,\beta)}(C))H(\pi_C|\mathrm{NEG}_{(\alpha,\beta)}(C)) \\
&\quad +P(\mathrm{BND}_{(\alpha,\beta)}(C))H(\pi_C|\mathrm{BND}_{(\alpha,\beta)}(C)), \qquad (13)
\end{aligned}$$

and the overall uncertainty based on the Gini coefficient is determined as [76],

$$\begin{aligned}
G(\pi_C|\pi_{(\alpha,\beta)}) &= P(\mathrm{POS}_{(\alpha,\beta)}(C))G(\pi_C|\mathrm{POS}_{(\alpha,\beta)}(C)) \\
&\quad +P(\mathrm{NEG}_{(\alpha,\beta)}(C))G(\pi_C|\mathrm{NEG}_{(\alpha,\beta)}(C)) \\
&\quad +P(\mathrm{BND}_{(\alpha,\beta)}(C))G(\pi_C|\mathrm{BND}_{(\alpha,\beta)}(C)). \qquad (14)
\end{aligned}$$

The Equations (13) - (14) was reformulated in a more readable form in [3].

Suppose that the uncertainties of the positive, negative and boundary regions are denoted by $\Delta_P(\alpha,\beta)$, $\Delta_N(\alpha,\beta)$ and $\Delta_B(\alpha,\beta)$ respectively, i.e.,

$$\Delta_P(\alpha,\beta) = P(\mathrm{POS}_{(\alpha,\beta)}(C))\delta(\pi_C|\mathrm{POS}_{(\alpha,\beta)}(C)), \tag{15}$$
$$\Delta_N(\alpha,\beta) = P(\mathrm{NEG}_{(\alpha,\beta)}(C))\delta(\pi_C|\mathrm{NEG}_{(\alpha,\beta)}(C)), \tag{16}$$
$$\Delta_B(\alpha,\beta) = P(\mathrm{BND}_{(\alpha,\beta)}(C))\delta(\pi_C|\mathrm{BND}_{(\alpha,\beta)}(C)). \tag{17}$$

where $\delta = \{H, G\}$. Using Equations (15) - (17), Equations (13) - (14) are represented in a general form as,

$$\Delta(\alpha,\beta) = \Delta_P(\alpha,\beta) + \Delta_N(\alpha,\beta) + \Delta_B(\alpha,\beta), \tag{18}$$

which represents the overall uncertainty due to a particular thresholds (α,β). The determination of optimum thresholds may be realized as the selection of (α,β) thresholds that minimizes Equation (18). This may be approached from a viewpoint of some optimization algorithm. We employ gradient descent based approach discussed in [23]. Moreover, we examine both types of information-theoretic rough sets, namely, ITRS based on Shannon entropy and ITRS based on Gini coefficient and will refer to them as ITRS$_E$ and ITRS$_G$, respectively.

5. Three-way decisions for Malware Analysis using UNM Dataset

To demonstrate the application of three-way decisions for malware analysis, we considered a subset of the system call sequences from the UNM dataset for the `xlock` application [27]. Detailed experimental results on the full dataset is discussed in the next section. The traces of malicious behaviour are based on the system calls of `xlock` exploited with the `CA-97.13.xlock` vulnerability, which are also obtained from the UNM dataset. The system call sequences are converted to sliding windows with a custom bash script. Each of these windows form the rows of Table 2 represented by $w_1, w_2, ..., w_{32}$. The system calls (each system call is identified by a number) in each window form the columns of Table 2 and represented by $s_1, s_2, ..., s_5$. The last column of the Table 2 indicate the application behaviour as malicious or benign represented by M and B, respectively. Let X_i represents an equivalence class which is the set of windows having the same sequence of system calls, i.e., set of windows having the same description. Nine equivalence classes are formed based on Table 2. These equivalence classes are shown in Table 3.

From rough sets perspective, Table 2 is essentially an information table since the set of objects, (i.e., perceived as windows) are described by a set of attributes (i.e., considered as system calls). The target concept (or concept of interest) in this case is to determine the malicious behaviour. We may not be unable to exactly specify this concept based on the equivalence classes $X_1, ..., X_9$. For instance, it is hard to decide whether or not X_3 belongs to this concept as three behaviours in X_3 are malicious and one behaviour is benign. We therefore approximate this concept using the probabilistic rough set model. In order to apply probabilistic rough set model, we need to calculate the conditional probability between an equivalence class X_i and a concept C, i.e., $P(C|X_i)$. Since the concept of interest in this case is the

Table 2: System call sequences and their corresponding identification

Window	s_1	s_2	s_3	s_4	s_5	Behaviour
w_1	106	106	106	106	106	M
w_2	125	5	5	3	90	B
w_3	125	5	5	3	90	M
w_4	106	5	90	6	5	B
w_5	106	106	106	106	106	M
w_6	125	5	5	3	90	M
w_7	3	90	90	90	6	B
w_8	3	90	90	90	6	M
w_9	106	5	90	6	5	M
w_{10}	125	5	5	3	90	M
w_{11}	125	5	5	3	90	M
w_{12}	5	108	3	19	6	B
w_{13}	108	3	19	6	33	B
w_{14}	108	3	19	6	33	M
w_{15}	3	90	90	90	6	M
w_{16}	3	90	90	90	6	M
w_{17}	106	5	90	6	5	M
w_{18}	3	6	5	108	3	B
w_{19}	5	108	3	19	6	B
w_{20}	5	108	3	19	6	M
w_{21}	125	5	5	3	90	B
w_{22}	45	45	5	108	45	B
w_{23}	45	45	5	108	45	M
w_{24}	3	6	5	108	3	B
w_{25}	3	6	5	108	3	B
w_{26}	3	6	5	108	3	M
w_{27}	125	5	5	3	90	B
w_{28}	6	5	108	3	19	B
w_{29}	3	6	5	108	3	M
w_{30}	45	45	5	108	45	B
w_{31}	5	108	3	19	6	B
w_{32}	6	5	108	3	19	B

Table 3: Equivalence classes based on Table 2

$X_1 = \{w_1, w_5\}$	$X_2 = \{w_7, w_8, w_{15}, w_{16}\}$
$X_3 = \{w_4, w_9, w_{17}\}$	$X_4 = \{w_2, w_3, w_6, w_{10}, w_{11}, w_{21}, w_{27}\}$
$X_5 = \{w_{13}, w_{14}\}$	$X_6 = \{w_{18}, w_{24}, w_{25}, w_{26}, w_{29}\}$
$X_7 = \{w_{22}, w_{23}, w_{30}\}$	$X_8 = \{w_{12}, w_{19}, w_{20}, w_{31}\}$
$X_9 = \{w_{28}, w_{32}\}$	

malicious behaviour, i.e., Behaviour = M, the conditional probability of an equivalence class X_i is determined as,

$$P(C|X_i) = P(\text{Behaviour} = \text{M}|X_i) = \frac{|\text{Behaviour} = \text{M} \bigcap X_i|}{|X_i|}. \tag{19}$$

The conditional probabilities of equivalence classes $X_1, ..., X_9$ using Equation (19) are determined as 1.0, 0.75, 0.67, 0.57, 0.5, 0.4, 0.33 0.25 and 0.0, respectively. These conditional probabilities represent the level of agreement between similar patterns of system calls to be representative of malicious behaviour. The probability of an individual equivalence class, i.e., X_i is determined as $P(X_i) = |X_i|/|U|$. This means that the probability of X_1 is $|X_1|/|U| = 2/32 = 0.0625$. The probabilities of the remaining equivalence classes $X_2, ..., X_9$ are similarly computed as 0.125, 0.09375, 0.21875, 0.0625, 0.15625, 0.09375, 0.125 and 0.0625, respectively. We now elaborate the application of probabilistic rough set models, such as, GTRS and ITRS (explained in Section 4) for classifying application behaviours.

5.1. Three-way decisions with Game-theoretic Rough Sets

Three-way decisions with GTRS are obtained by employing a game between multiple criteria as outlined in Section 4.1. We consider a game in GTRS discussed in [4]. The players considered in the game are the measures of accuracy and generality. The accuracy reflects the relative number of correct classification decisions for objects compared to the total classification decisions. The generality highlights the relative number of objects for whom classification decisions can be made. The measures of accuracy and generality can be defined as [67],

$$Accuracy(\alpha,\beta) = \frac{|(\mathrm{POS}_{(\alpha,\beta)}(C)\cap C)\bigcup(\mathrm{NEG}_{(\alpha,\beta)}(C)\cap C^c)|}{|\mathrm{POS}_{(\alpha,\beta)}(C)\bigcup \mathrm{NEG}_{(\alpha,\beta)}(C)|}, \quad (20)$$

$$Generality(\alpha,\beta) = \frac{|\ \mathrm{POS}_{(\alpha,\beta)}(C)\bigcup \mathrm{NEG}_{(\alpha,\beta)}(C)|}{|U|}, \quad (21)$$

In order to compute these measures based on certain thresholds, for instance, (α, β) = (0.8,0.4), we need to determine the three regions. The three regions in this case are determined as $\mathrm{POS}_{(0.8,0.4)}(C) = \bigcup\{X_1\}$, $\mathrm{BND}_{(0.8,0.4)}(C) = \bigcup\{X_2, X_3, X_4, X_5\}$, and $\mathrm{NEG}_{(0.8,0.4)}(C) = \{X_6, X_7, X_8, X_9\}$. The measure of accuracy and generality are now computed as,

$$Accuracy(\alpha,\beta) = \frac{|(X_1\cap C)\bigcup((X_6\bigcup X_7\bigcup X_8\bigcup X_9)\cap C^c)|}{|X_1\bigcup X_6\bigcup X_7\bigcup X_8\bigcup X_9|}$$

$$= \frac{|\{w_1,w_5,w_{18},w_{24},w_{25},w_{22},w_{30},w_{12},w_{19},w_{31},w_{28},w_{32}\}|}{|\{w_1,w_5,w_{18},w_{24},w_{25},w_{26},w_{29},w_{22},w_{23},w_{30},w_{12},w_{19},w_{20},w_{31},w_{28},w_{32}\}|} = 0.75 \quad (22)$$

$$Generality(\alpha,\beta) = \frac{|(X_1\bigcup X_6\bigcup X_7\bigcup X_8\bigcup X_9)|}{|U|}$$

$$= \frac{|\{w_1,w_5,w_{18},w_{24},w_{25},w_{26},w_{29},w_{22},w_{23},w_{30},w_{12},w_{19},w_{20},w_{31},w_{28},w_{32}\}|}{|\{w_1,w_2,\ldots,w_{32}\}|} = 0.5 \quad (23)$$

Each player compete in the game by choosing from three possible strategies, i.e., $s_1 = \alpha\downarrow$ = 20% decrease in α, $s_2 = \beta\uparrow$ = 20% increase in β and $s_3 = \alpha\downarrow\beta\uparrow$ = 20% decrease in α and 20% increase in β. To compute a threshold pair corresponding to a strategy profile, we consider two possible cases in the game. During a game, if only a single player suggests a change in a threshold value, the value will be determined as an increase or decrease suggested by that player. On the other hand, if both the players suggest a change, the value will be decided based on the sum of the two changes. Using these two cases, a threshold pair corresponding to a strategy, say, $(s1, s2) = (\alpha_\downarrow, \beta_\uparrow)$ are determined as $(\alpha, \beta) = (0.75, 0.25)$.

Table 4 represents the payoff table for the considered game based on the data in 2. The cell containing bold values, i.e., **(0.8191,0.3438)**, represents the game solution based on Equation (6). The strategies profile based on the game solution is (s_1, s_3) which leads to thresholds $(\alpha, \beta) = (0.6, 0.2)$. This suggest that by setting the acceptance level at 0.6 and the rejection level at 0.2, we are able to make 81.91% correct decisions in 34.38% of the cases.

Table 4: Payoff table for the game between accuracy and generality

		Generality		
		$s_1 = \alpha_{\downarrow}$ = 20% dec. α	$s_2 = \beta_{\uparrow}$ = 20% inc. β	$s_3 = \alpha_{\downarrow}\beta_{\uparrow}$ = 20% (dec. α & inc. β)
Accuracy	$s_1 = \alpha_{\downarrow}$ = 20% dec. α	(0.8191,0.3438)	(1.0,0.1250)	**(0.8191,0.3438)**
	$s_2 = \beta_{\uparrow}$ = 20% inc. β	(1.0000,0.1250)	(0.7500,0.5000)	(0.7500,0.5000)
	$s_3 = \alpha_{\downarrow}\beta_{\uparrow}$ = 20% (dec. α & inc. β)	(0.8191,0.3438)	(0.7500,0.5000)	(0.7400,0.7188)

5.2. Three-way decisions with Information-theoretic Rough Sets

Three-way decisions with ITRS is based on minimization of the overall uncertainty. In this example we consider the ITRS based on the measure of Shannon entropy, i.e., ITRS$_E$. The results can be easily extended to ITRS based on the measure of Gini coefficient, i.e., ITRS$_G$.

Considering a majority oriented model, defined by the conditions $0 \leq \beta < 0.5 \leq \alpha \leq 1.0$. We have the following domains for the thresholds α and β based on the data in Table 2.

$$D_\alpha = \{1.0, 0.7, 0.6, 0.5\}, \quad D_\beta = \{0.0, 0.3, 0.4\}$$

To determine the uncertainty with respect to certain thresholds, say $(\alpha, \beta) = (1.0, 0.0)$, we need to determine the positive, negative and boundary regions given in Equations (3) - (5). In this case the three regions as computed as $\mathrm{POS}_{(1.0,0.0)}(C) = \{X_1\}$, $\mathrm{NEG}_{(1.0,0.0)}(C) = \{X_9\}$ and $\mathrm{BND}_{(1.0,0.0)}(C) = \bigcup\{X_2, X_3, ..., X_8\}$. The probabilities of the three regions are determined as $P(\mathrm{POS}_{(\alpha,\beta)}(C)) = P(X_1) = 0.0625$, $P(\mathrm{NEG}_{(\alpha,\beta)}(C)) = P(X_9) = 0.0625$ and $P(\mathrm{BND}_{(\alpha,\beta)}(C)) = P(X_2) + P(X_3) + ... + P(X_8) = 0.875$, respectively. The conditional probability with respect to the positive region $P(C|\mathrm{POS}_{(\alpha,\beta)}(C))$ is,

$$P(C|\mathrm{POS}_{(1,0)}(C)) = \frac{\sum_{i=1}^{1} P(C|X_i) * P(X_i)}{\sum_{i=1}^{1} P(X_i)} = \frac{1 * 0.0625}{0.0625} = 1.0 \quad (24)$$

The conditional probability $P(C^c|\mathrm{POS}_{(1,0)}(C))$ is computed as $1 - P(C|\mathrm{POS}_{(1,0)}(C)) =$

$1-1=0$. The uncertainty with respect to the positive region based on Equation (7) is,

$$H(\pi_C|\mathrm{POS}_{(1,0)}(C)) = -1 * log1 - (0 * log0) = 0. \tag{25}$$

Based on Equation (15), the average uncertainty of the positive region is $\Delta_P(1,0) = P(\mathrm{POS}_{(1,0)}(C))H(\pi_C|\mathrm{POS}_{(1,0)}(C)) = 0$. Equations (16) and (17) may be used in the same way to determine the average uncertainties due to negative and boundary regions. They are computed as $\Delta_N(1,0) = 0$ and $\Delta_B(1,0) = 0.875$, respectively. According to Equation (18), the total uncertainty is therefore $\Delta(1,0) = 0.875$.

We can compute the overall uncertainties of all other possible threshold pairs. The results are summarized in the form of the following matrix.

$$\begin{array}{cc} & \begin{array}{cccc} \alpha=1.0 & \alpha=0.7 & \alpha=0.6 & \alpha=0.5 \end{array} \\ \begin{array}{c} \beta=0.0 \\ \beta=0.3 \\ \beta=0.4 \end{array} & \begin{pmatrix} 0.875 & 0.8680 & 0.8607 & 0.8606 \\ 0.8682 & 0.8688 & 0.8661 & 0.8768 \\ \mathbf{0.8544} & 0.8665 & 0.8704 & 0.8937 \end{pmatrix} \end{array}$$

We can identify the minimum uncertainty in the matrix and the corresponding thresholds. In this case, the minimum value is **0.8544** which corresponds to thresholds $(\alpha, \beta) = (1.0, 0.4)$.

The above examples demonstrate the application and use of probabilistic rough sets based three-way decisions for determining malicious behaviour using two models, i.e., ITRS and GTRS. The essential difference between these two models may be explained based on the type and nature of the solution. The the GTRS provides a tradeoff based solution to determine the thresholds and the ITRS provides an optimization based solution in computing the thresholds. The other possible difference is that GTRS is based on multiple criteria while the ITRS is based on a single criterion, i.e., the uncertainty. The choice of using a certain model may depend on the intended requirements and usage in the application.

6. Experimental Results and Discussion

In Section 5, we demonstrated the application of three-way decisions on a subset of the system call sequences from the UNM dataset for the `xlock` application [27]. In this section, we present detailed experimental results on the full dataset. The structure of the UNM dataset is such that each

Table 5: Results of accuracy for different window sizes

Window	GTRS$_{(A,G)}$	GTRS$_E$	GTRS$_G$	ITRS$_E$	ITRS$_G$	Pawlak
w=3	0.6966	0.7157	0.7171	0.7070	0.7426	0.6733
w=4	**0.7339**	0.7187	0.6762	0.7541	0.7055	0.7195
w=5	0.7267	0.7203	0.7258	**0.7977**	0.6706	0.6965
w=6	0.7194	0.7256	**0.7390**	0.7943	0.6524	0.6582
w=7	0.6747	**0.7463**	0.6470	0.6540	**0.7532**	**0.7643**

application has two traces of execution: one with a benign executable and another with an infected malicious vulnerability. As such, each trace contains only one malware family. From this trace, different data points are extracted. The data points in the training set are not i.i.d. as is the case with the usual setting. We therefore, do not assume i.i.d. sampling of data points which is much closer to the real life dataset. The system call sequences of the malicious and benign applications are converted to information table using the sliding window approach discussed in Section 2.1.

6.1. Experimental Results and Discussion

As discussed in Section 2.1, the size of sliding windows is an important parameter for sequence-time delay approach [30]. The parameter influences the detection rate and performance of the model. If the size of sliding windows is too small, a very small slice of an application's actions will be captured and would be insufficient to learn the behaviour. On the other hand, if the size is too large, it might overfit the training data and also lead to performance bottlenecks. The tradeoff is generally dependent on the usage scenario and has to be computed empirically. The useful values of the window size in the previous studies are in the range of 3 to 7 [2, 19]. We therefore report our experimental results for different window size in the same range.

Along with the five three-way decision approaches discussed in Section 4, we also report experiments with the standard Pawlak model. The Pawlak model is defined by setting $(\alpha, \beta) = (1.0)$ in Equations (1) - (5). For testing purpose, we use 10 fold cross validation in all the experiments.

Table 5 shows the results of accuracy for the considered approaches. The best results for each approach corresponding to different windows sizes are shown in bold. Considering the GTRS based approaches, the best results for GTRS$_{(A/G)}$, GTRS$_E$ and GTRS$_G$ are based on window sizes of 4, 7 and

Table 6: Test results for 10 folds cross validation with GTRS

Folds	GTRS$_{(A,G)}$		GTRS$_E$		GTRS$_E$	
	Acc.	Gen.	Acc.	Gen.	Acc.	Gen.
1.	0.7059	0.7473	0.7347	0.9423	0.6842	0.9344
2.	0.7097	0.6813	0.7273	0.8462	0.7347	0.8033
3.	0.7049	0.6703	0.7381	0.8077	0.7551	0.8033
4.	0.7377	0.6703	0.7561	0.7885	0.7500	0.7869
5.	0.7719	0.6264	0.7561	0.7885	0.7400	0.8197
6.	0.7586	0.6374	0.7561	0.7885	0.7551	0.8033
7.	0.7586	0.6374	0.7561	0.7885	0.7551	0.8033
8.	0.7500	0.6154	0.7442	0.8269	0.7547	0.8689
9.	0.7000	0.6593	0.7561	0.7885	0.7551	0.8033
10.	0.7414	0.6304	0.7381	0.7925	0.7059	0.8226

6 with accuracy values of 73.39%, 74.63% and 73.90%, respectively. Generally speaking, the GTRS$_E$ performance is better than the other GTRS based approaches with no accuracy value being lesser than 70.0%. Compared to Pawlak model, the GTRS approaches provide better results except for window sizes of 4 and 7. Let us now consider the the results with ITRS based approaches. The best results for ITRS$_E$ and ITRS$_G$ are based on window sizes of 5 and 7 with values of 79.77% and 75.32%, respectively. Overall, ITRS$_E$ shows better results compared to ITRS$_G$ with only one value below 70% for ITRS$_E$. Generally speaking, ITRS based approaches provide similar results to that of the Pawlak model. However, in some cases the difference is quite noticeable compared to Pawlak model, such as, ITRS$_E$ for window sizes of 5 and 6 with differences of around 10% and 14% respectively. The results of 10 folds cross validation based on the best window size results of accuracy for each approach is shown in Tables 6 and 7.

Table 8 shows the results of generality for the considered approaches. Since three-way decisions involve deferring some decisions. This may affect the accuracy as some hard to classify cases may be deferred [63]. The results of generality are therefore important in knowing the relative number of cases for which we are making useful decisions. Some researchers refer to the same property of making useful decisions as commitment rate while some others use the terminology of deferment rate, which is essentially (1-generality) [68].

The best generality results for each approach corresponding to different

Table 7: Test results for 10 folds cross validation with ITRS and Pawlak

Folds	ITRS_E		ITRS_G		Pawlak	
	Acc.	Gen.	Acc.	Gen.	Acc.	Gen.
1.	0.6977	0.5890	0.6250	0.9231	0.6739	0.8846
2.	0.8525	0.8356	0.7778	0.8654	0.7609	0.8846
3.	0.8000	0.6164	0.7826	0.8846	0.7778	0.8654
4.	0.7872	0.6438	0.7674	0.8269	0.7778	0.8654
5.	0.8043	0.6301	0.7556	0.8654	0.7778	0.8654
6.	0.8000	0.6164	0.7674	0.8269	0.7660	0.9038
7.	0.8085	0.6438	0.7500	0.8462	0.7826	0.8846
8.	0.8085	0.6438	0.7727	0.8462	0.7778	0.8654
9.	0.8525	0.8356	0.7778	0.8654	0.7826	0.8846
10.	0.7660	0.6351	0.7556	0.8491	0.7660	0.8868

Table 8: Results of Generality for different window sizes

Window	$\mathrm{GTRS}_{(A,G)}$	GTRS_E	GTRS_G	ITRS_E	ITRS_G	Pawlak
w=3	0.7256	0.6822	0.6855	0.6790	0.7453	0.6265
w=4	0.6575	0.6696	0.7893	0.7454	0.7761	0.7354
w=5	0.6744	0.7361	0.6184	0.6690	0.6839	0.6580
w=6	**0.8886**	**0.8658**	0.8249	0.8298	**0.8903**	**0.8887**
w=7	0.8312	0.8158	**0.9251**	**0.8503**	0.8599	0.8791

windows sizes are shown in bold in Table 8. Considering the GTRS based approaches, there is no significant difference between the results for different window sizes. In other words, there is no single approach which provides better generality for all window sizes. However, it is noted that for the best overall results against any GTRS approach is 92.51%, i.e., GTRS_G with window size 7, for the same window size, the difference between GTRS_G and other GTRS approaches are about 9-11% which is quite significant. The difference between the two ITRS based approaches is also insignificant. It is however important to note that the ITRS_G always provide better generality compared to the Pawlak model.

We also note that in general, the generality of the approaches improve as we increase the window size. This may be explained from rough set perspective. Recall from Section 5, the window size determine the number of attributes in our information table. For instance, a window size of 5 means that we are having five attributes in our information table. By increasing the window size, we are increasing the number of attributes in our information table. Since the attributes control the information about objects, more attributes mean more information about the objects. From rough sets perspective, when we have more additional information about objects, we may be able to make more certain decisions. This means that some of the previously deferred cases can now be certainly decided. As a result, our generality improves and the deferment decisions are reduced.

An important issue in malware analysis is to decide the relative importance of using different window sizes. It may be noted that based on the accuracy results of Table 5, three out of six approaches, i.e., GTRS$_E$, ITRS$_G$ and Pawlak, show their best results for windows size of 7. According to generality results of Table 8, the best generality results for different approaches are either based on window sizes of 6 or 7. These observations advocate for the use of higher size of window, i.e., either window size of 6 or 7 to achieve effective and optimal performance.

Summarizing the experimental results, the accuracy values for the considered ITRS and GTRS based three-way approaches are ranging between 65% to 80% and the generality values are ranging between 65% and 92%. Moreover, majority of the best results are noted for large window sizes. The experimental results discussed in this section suggest for the use of three-way decisions in analyzing and examining malicious behaviours of applications.

The proposed three-way approach to malware analysis may be extended in different aspects. One interesting aspect is to be deploy the three-way approach on the production systems for gathering live data traces. This will enable us to measure efficiency on large scale. Another worth exploring aspect is to examine it in the context of smartphones where the conventional approaches to malware analysis have failed to produce acceptable results. Last but not the least, various three-way decision models and approaches may be examined to find the best match for the domain.

6.2. Analysis of ROC

In this section, we analyze the three-way approaches using the ROC graph. The ROC graph is based on the true positive rate and false posi-

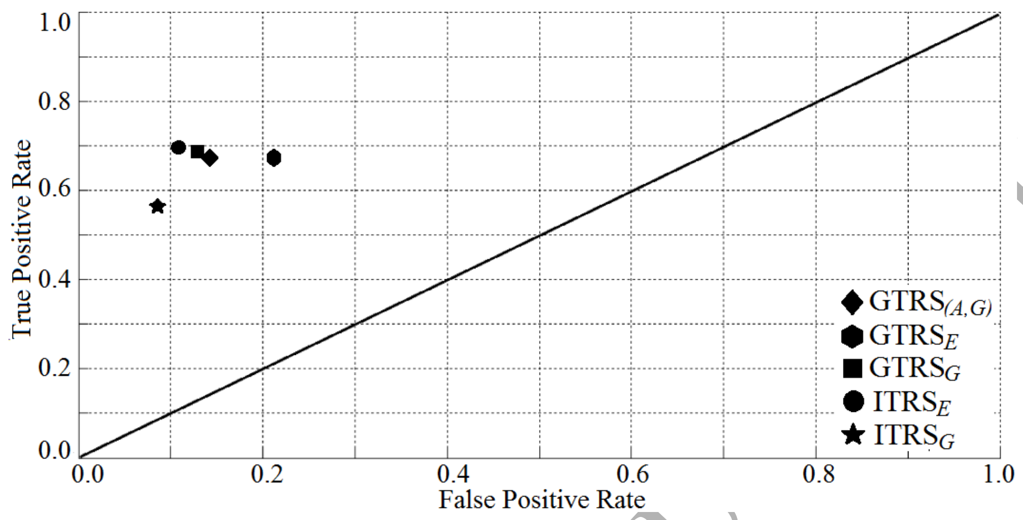

Figure 4: Points in the ROC Space Corresponding to Three-way Approaches

tive rate which are defined as follow.

$$\begin{aligned} \text{true positive rate} &= \frac{\text{Positives correctly classified}}{\text{Total positives}} \\ \text{false positive rate} &= \frac{\text{Negatives incorrectly classified}}{\text{Total negatives}} \end{aligned} \qquad (26)$$

It is important to note that each three-way approach produces a single pair of thresholds (α, β). Each of these thresholds can be used to classify all the instances in the dataset to obtain a pair of (true positive rate, false positive rate) values. This means that these approaches will produce a single point in the ROC space rather than a curve. Figure 4 shows the performance of the five three-way approaches on the ROC graph. We note that from perspective of false positive rate, the ITRS$_E$ may be more suitable choice. However, the overall performance of ITRS$_G$ may more better due to higher true positive rate and little difference in false positive rate compared to ITRS$_E$. The GTRS based approaches tend to have comparatively more false positive rates. The minimum false positive rate is noted against ITRS$_E$ with a value as low as 0.085.

Another important aspect of ROC graphs in the context of three-way decisions was recently being highlighted in [37]. It is argued that the conventional classifiers which are based on binary or two-way decisions do not pro-

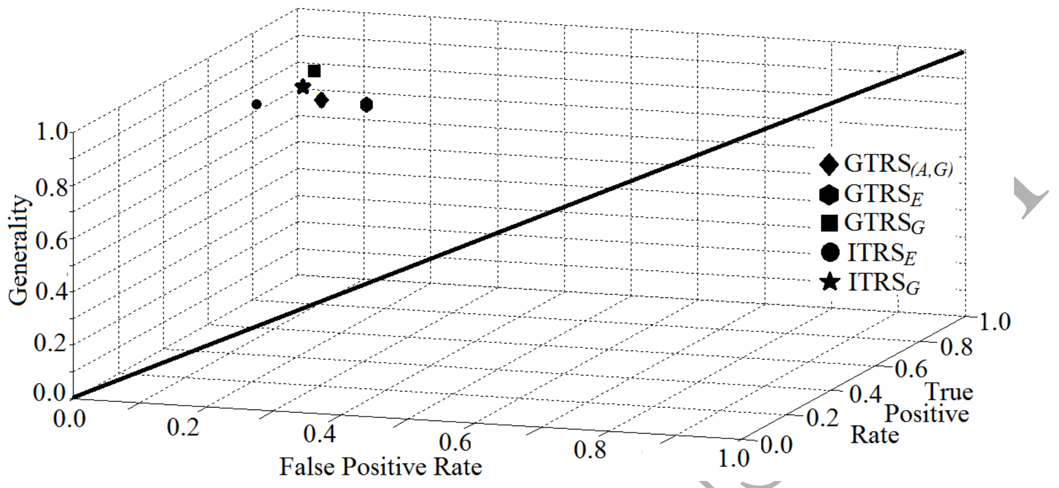

Figure 5: Points in the ROC Space Corresponding to Three-way Approaches

duce boundary region or deferment decisions. Therefore, the two-dimensional ROC graph can depict the tradeoff between true positives and false positives. However, for a classifier based on three-way decisions, the true positives and false positives are affected by the deferment decisions. Therefore, the tradeoff among three aspects should be considered in the ROC graph. To incorporate this, we include Figure 5. We have false positive rate on the x-axis, true positive rate on the y-axis and the generality on the z-axis. A good classifier will produce a point with lower x value and higher y and z values. We may note that the results are a bit different as compared to Figure 4. The GTRS$_G$ provides better overall results when the three parameters are considered.

7. Conclusion

Dealing with false classification decisions is a key issue in malware detection. Existing approaches suffer from two limitations. Firstly, they are based on two-way decisions which require that a classification decision must be made irrespective of the quality of available information. Secondly, they lack explicit mechanisms to deal with false decisions at the level of learning model. In this paper, we examine a three-way decision making approach for overcoming these two limitations. In particular, we considered three-way decisions based on two probabilistic rough set models, i.e., game-theoretic rough sets and information theoretic rough sets. An architecture of malware analysis realized with probabilistic rough sets based three-way decisions is proposed. A new algorithm called sequentially stackable linux security (SSLS) based on the proposed architecture is presented. Experimental results on the UNM dataset suggest that a false positive rate as minimum as 8.5% may be achieved with the three-way approaches which is comparable to some of the most promising existing approaches. Moreover, it is also noted that by increasing the windows size, which is an important parameter in capturing application behaviour, the results are improved both in terms of accuracy and generality. The insights from this study suggest that the three-way approach can be a useful alternative for malware classification.

8. Acknowledgement

This work was partially supported by a Discovery Grant from NSERC Canada.

References

[1] Ali, M. Q., Al-Shaer, E., Khan, H., Khayam, S. A., 2013. Automated anomaly detector adaptation using adaptive threshold tuning. ACM Transactions on Information and System Security (TISSEC) 15 (4), 17.

[2] Ali, T., Nauman, M., Zhang, X., 2011. On leveraging stochastic models for remote attestation. In: Trusted Systems. pp. 290–301.

[3] Azam, N., Yao, J. T., 2014. Analyzing uncertainties of probabilistic rough set regions with game-theoretic rough sets. International journal of approximate reasoning 55 (1), 142–155.

[4] Azam, N., Yao, J. T., 2014. Game-theoretic rough sets for recommender systems. Knowledge-Based Systems 72, 96–107.

[5] Bauman, E., Ayoade, G., Lin, Z., 2015. A survey on hypervisor-based monitoring: Approaches, applications, and evolutions. ACM Computing Surveys 48 (1), 10.

[6] Biddle, R., van Oorschot, P. C., Patrick, A. S., Sobey, J., Whalen, T., 2009. Browser interfaces and extended validation ssl certificates: an empirical study. In: Proceedings of the 2009 ACM workshop on Cloud computing security. pp. 19–30.

[7] Bing, Z., 2012. A cost sensitive approach to ternary classification. Ph.D. thesis, University of Regina, Regina, Saskatchewan.

[8] Buchanan, E., Roemer, R., Shacham, H., Savage, S., 2008. When good instructions go bad: Generalizing return-oriented programming to risc. In: Proceedings of the 15th ACM conference on Computer and communications security. ACM, pp. 27–38.

[9] Burford, S., Park, S., 2014. The impact of mobile tablet devices on human information behaviour. Journal of Documentation 70 (4), 622–639.

[10] Caballero, J., Grier, C., Kreibich, C., Paxson, V., 2011. Measuring pay-per-install: The commoditization of malware distribution. In: Usenix security symposium.

[11] Cattaneo, G., Ciucci, D., Dubois, D., 2011. Algebraic models of deviant modal operators based on de morgan and kleene lattices. Information Sciences 181 (19), 4075–4100.

[12] Chandola, V., Banerjee, A., Kumar, V., 2009. Anomaly detection: A survey. ACM Computing Surveys (CSUR) 41 (3), 15:1–15:58.

[13] Chandola, V., Mithal, V., Kumar, V., 2014. A reference based analysis framework for understanding anomaly detection techniques for symbolic sequences. Data Mining and Knowledge Discovery 28 (3), 702–735.

[14] Checkoway, S., Davi, L., Dmitrienko, A., Sadeghi, A.-R., Shacham, H., Winandy, M., 2010. Return-oriented programming without returns. In: Proceedings of the 17th ACM conference on Computer and communications security. pp. 559–572.

[15] Chen, P. S., Lin, S. C., Sun, C. H., 2015. Simple and effective method for detecting abnormal internet behaviors of mobile devices. Information Sciences 321, 193–204.

[16] Ciucci, D., 2011. Orthopairs: A simple and widely usedway to model uncertainty. Fundamenta Informaticae 108 (3-4), 287–304.

[17] Comar, P. M., Liu, L., Saha, S., Tan, P.-N., Nucci, A., 2013. Combining supervised and unsupervised learning for zero-day malware detection. In: Proceedings of INFOCOM. pp. 2022–2030.

[18] Coron, J.-S., 2000. On the exact security of full domain hash. In: Advances in Cryptology (CRYPTO 2000). pp. 229–235.

[19] Creech, G., Hu, J., 2014. A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns. IEEE Transactions on Computers 63 (4), 807–819.

[20] Dainotti, A., Pescapé, A., Ventre, G., 2009. A cascade architecture for dos attacks detection based on the wavelet transform. Journal of Computer Security 17 (6), 945–968.

[21] Demchenko, Y., Ngo, C., de Laat, C., Membrey, P., Gordijenko, D., 2014. Big security for big data: Addressing security challenges for the big data infrastructure. In: Secure Data Management. pp. 76–94.

[22] Deng, X. F., Yao, Y. Y., 2012. An information-theoretic interpretation of thresholds in probabilistic rough sets. In: Proceedings of Rough Sets and Current Trends in Computing (RSCTC'12), Lecture Notes in Computer Science 7413. pp. 232–241.

[23] Deng, X. F., Yao, Y. Y., 2014. A multifaceted analysis of probabilistic three-way decisions. Fundamenta Informaticae 132, 291–313.

[24] Egele, M., Scholte, T., Kirda, E., Kruegel, C., 2012. A survey on automated dynamic malware-analysis techniques and tools. ACM Computing Surveys (CSUR) 44 (2), 6:1–6:42.

[25] Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., Wagner, D., 2012. Android permissions: User attention, comprehension, and behavior. In: Proceedings of the Eighth Symposium on Usable Privacy and Security. pp. 3–16.

[26] Forrest, S., Hofmeyr, S. A., Somayaji, A., Longstaff, T. A., 1996. A sense of self for unix processes. In: IEEE Symposium on Security and Privacy. pp. 120–128.

[27] Forrest, S., Hofmeyr, S. A., Somayaji, A., Longstaff, T. A., 2015. UNM dataset. Available: `http://www.cs.unm.edu/~immsec/data-sets.htm`, accessed: April 6, 2015.

[28] Gandotra, E., Bansal, D., Sofat, S., 2014. Malware analysis and classification: A survey. Journal of Information Security 2014.

[29] Griffin, K., Schneider, S., Hu, X., Chiueh, T.-C., 2009. Automatic generation of string signatures for malware detection. In: Recent advances in intrusion detection. pp. 101–120.

[30] Gupta, M., Gao, J., Aggarwal, C., Han, J., 2014. Outlier detection for temporal data. Synthesis Lectures on Data Mining and Knowledge Discovery 5 (1), 1–129.

[31] Hashizume, K., Rosado, D. G., Fernández-Medina, E., Fernandez, E. B., 2013. An analysis of security issues for cloud computing. Journal of Internet Services and Applications 4 (1), 1–13.

[32] Herbert, J. P., 2010. Investigating machine learning decision problems with game theory. Ph.D. thesis, Regina, Saskatchewan, Canada.

[33] Herbert, J. P., Yao, J. T., 2011. Game-theoretic rough sets. Fundamenta Informaticae 108 (3-4), 267–286.

[34] Hu, B. Q., 2014. Three-way decisions space and three-way decisions. Information Sciences 281, 21–52.

[35] Huang, G.-B., Zhou, H., Ding, X., Zhang, R., 2012. Extreme learning machine for regression and multiclass classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 42 (2), 513–529.

[36] Hubballi, N., Suryanarayanan, V., 2014. False alarm minimization techniques in signature-based intrusion detection systems: A survey. Computer Communications 49, 1–17.

[37] Jia, X. Y., Shang, L., 2015. How to evaluate three-way decisions based binary classification? In: Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing. pp. 366–375.

[38] Lam, L., 2000. Classifier combinations: implementations and theoretical issues. In: Multiple classifier systems. pp. 77–86.

[39] Leyton-Brown, K., Shoham, Y., 2008. Essentials of Game Theory: A Concise Multidisciplinary Introduction. Morgan & Claypool Publishers.

[40] Li, W., Meng, W., Luo, X., Kwok, L. F., 2016. Mvpsys: Toward practical multi-view based false alarm reduction system in network intrusion detection. Computers & Security 60, 177–192.

[41] Liang, D., Liu, D., 2014. Systematic studies on three-way decisions with interval-valued decision-theoretic rough sets. Information Sciences 276, 186–203.

[42] Mehdi, B., Ahmed, F., Khayyam, S. A., Farooq, M., 2010. Towards a theory of generalizing system call representation for in-execution malware detection. In: 2010 IEEE International Conference on Communications. pp. 1–5.

[43] Menahem, E., Shabtai, A., Rokach, L., Elovici, Y., 2009. Improving malware detection by applying multi-inducer ensemble. Computational Statistics & Data Analysis 53 (4), 1483–1494.

[44] Mohaisen, A., Alrawi, O., Larson, M., McPherson, D., 2014. Towards a methodical evaluation of antivirus scans and labels. In: Information Security Applications. pp. 231–241.

[45] Nauman, M., Azam, N., Yao, J. T., 2015. A three-way decision making approach to malware analysis. In: Proceedings of International Conference on Rough Sets and Knowledge Technology (RSKT'15), Lecture Notes in Computer Science 9436. pp. 273–284.

[46] Nissim, N., Moskovitch, R., Rokach, L., Elovici, Y., 2014. Novel active learning methods for enhanced pc malware detection in windows os. Expert Systems With Applications 41 (13), 5843–5857.

[47] Oldham, G. R., Da Silva, N., 2015. The impact of digital technology on the generation and implementation of creative ideas in the workplace. Computers in Human Behavior 42, 5–11.

[48] Oza, A., Ross, K., Low, R. M., Stamp, M., 2014. Http attack detection using n-gram analysis. Computers & Security 45, 242–254.

[49] Pedrycz, W., 1998. Shadowed sets: representing and processing fuzzy sets. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 28 (1), 103–109.

[50] Pedrycz, W., 2009. From fuzzy sets to shadowed sets: interpretation and computing. International Journal of Intelligent Systems 24 (1), 48–61.

[51] Pedrycz, W., 2013. Granular computing: analysis and design of intelligent systems. CRC press.

[52] Peng, J., Choo, K.-K. R., Ashman, H., 2016. User profiling in intrusion detection: A review. Journal of Network and Computer Applications 72, 14–27.

[53] Perdisci, R., Lanzi, A., Lee, W., 2008. Mcboost: Boosting scalability in malware collection and analysis using statistical classification of executables. In: Proceedings of Annual Computer Security Applications Conference. pp. 301–310.

[54] Prandini, M., Ramilli, M., 2012. Return-oriented programming. IEEE Security & Privacy 10 (6), 84–87.

[55] Rehman, A., Saba, T., 2014. Evaluation of artificial intelligent techniques to secure information in enterprises. Artificial Intelligence Review 42 (4), 1029–1044.

[56] Shahzad, R. K., Lavesson, N., 2012. Veto-based malware detection. In: Proceedings of Seventh International Conference on Availability, Reliability and Security (ARES), 2012. pp. 47–54.

[57] Spathoulas, G. P., Katsikas, S. K., 2010. Reducing false positives in intrusion detection systems. computers & security 29 (1), 35–44.

[58] Syed, T. A., Ismail, R., Musa, S., Nauman, M., Khan, S., 2012. A sense of others: behavioral attestation of unix processes on remote platforms. In: Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication. p. 51.

[59] Symantec Inc., 2015. Internet security threat report, volume 19. Accessed: Apr 12, 2015, http://www.symantec.com/security_response/publications/threatreport.jsp.

[60] Tang, W., Jin, G., He, J., Jiang, X., 2011. Extending android security enforcement with a security distance model. In: Internet Technology and Applications (iTAP), 2011 International Conference on. IEEE, pp. 1–4.

[61] Wright, J., Dawson Jr, M. E., Omar, M., 2012. Cyber security and mobile threats: The need for antivirus applications for smart phones. Journal of Information Systems Technology and Planning 5 (14), 40–60.

[62] Yahyaoui, H., Al-Mutairi, A., 2016. A feature-based trust sequence classification algorithm. Information Sciences 328, 455–484.

[63] Yao, J. T., Azam, N., 2014. Three-way decision making in Web-based medical decision support systems with game-theoretic rough sets. IEEE Transactions on Fuzzy Systems 23 (1), 3–15.

[64] Yao, J. T., Herbert, J. P., 2008. A game-theoretic perspective on rough set analysis. Journal of Chongqing University of Posts and Telecommunications (Natural Science Edition) 20 (3), 291–298.

[65] Yao, J. T., Yao, Y. Y., Ziarko, W., 2008. Probabilistic rough sets: Approximations, decision-makings, and applications. International Journal of Approximate Reasoning 49 (2), 253–254.

[66] Yao, Y., Lin, T. Y., 1996. Generalization of rough sets using modal logics. Intelligent Automation & Soft Computing 2 (2), 103–119.

[67] Yao, Y. Y., 2008. Probabilistic rough set approximations. International Journal of Approximate Reasoning 49 (2), 255–271.

[68] Yao, Y. Y., 2011. The superiority of three-way decisions in probabilistic rough set models. Information Sciences 181 (6), 1080–1096.

[69] Yao, Y. Y., 2011. Two semantic issues in a probabilistic rough set model. Fundamenta Informaticae 108 (3-4), 249–265.

[70] Yao, Y. Y., 2012. An outline of a theory of three-way decisions. In: Proceedings of Rough Sets and Current Trends in Computing (RSCTC'12), Lecture Notes in Computer Science 7413. pp. 1–17.

[71] Yao, Y. Y., 2015. Rough sets and three-way decisions. In: Proceedings of 10th International Conference on Rough Sets and Knowledge Technology (RSKT'15), Lecture Notes in Computer Science 9436. pp. 62–73.

[72] Yao, Y. Y., Greco, S., Slowinski, R., 2015. Probabilistic rough sets. In: Handbook of Computational Intelligence, Projektorganisation und Management im Software Engineering. pp. 315–339.

[73] Yao, Y. Y., Wong, S. K. M., Lingrass, P., 1990. A decision-theoretic rough set model. Methodologies for Intelligent Systems 35, 17–24.

[74] Yin, H., Song, D., Egele, M., Kruegel, C., Kirda, E., 2007. Panorama: capturing system-wide information flow for malware detection and analysis. In: Proceedings of the 14th ACM conference on Computer and communications security. pp. 116–127.

[75] Yolacan, E. N., Dy, J. G., Kaeli, D. R., 2014. System call anomaly detection using multi-hmms. In: 2014 IEEE 18 International Conference on Software Security and Reliability-Companion. pp. 25–30.

[76] Zhang, Y., 2013. Optimizing gini coefficient of probabilistic rough set regions using game-theoretic rough sets. In: Proceedings of 26th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE'13). pp. 699–702.

[77] Zhang, Y., Lee, W., 2000. Intrusion detection in wireless ad-hoc networks. In: Proceedings of the 6th annual international conference on Mobile computing and networking. pp. 275–283.