International Journal of Semantic Computing
Vol. 5, No. 3 (2011) 281–322
©World Scientific Publishing Company
DOI: 10.1142/S1793351X11001274

MEASURING INCONSISTENCY IN A NETWORK INTRUSION DETECTION RULE SET BASED ON SNORT*

KEVIN MCAREAVEY

Centre for Secure Information Technologies
Institute of Electronics, Communications
and Information Technology (ECIT)
Queen's University Belfast
Northern Ireland Science Park, Belfast, BT3 9DT
Northern Ireland
kmcareavey01@qub.ac.uk

WEIRU LIU

School of Electronics, Electrical Engineering
and Computer Science
Queen's University Belfast, BT7 1NN
Northern Ireland
w.liu@qub.ac.uk

PAUL MILLER

Centre for Secure Information Technologies
Institute of Electronics, Communications
and Information Technology (ECIT)
Queen's University Belfast
Northern Ireland Science Park, Belfast, BT3 9DT
Northern Ireland
p.miller@ecit.qub.ac.uk

KEDIAN MU

School of Mathematical Sciences
Peking University, Beijing 100871, P. R. China
mukedian@math.pku.edu.cn

In this preliminary study, we investigate how inconsistency in a network intrusion detection rule set can be measured. To achieve this, we first examine the structure of these rules which are based on Snort and incorporate regular expression (Regex) pattern matching. We then identify primitive elements in these rules in order to translate the rules into their (equivalent) logical forms and to establish connections between them. Additional rules from background knowledge are also introduced to make the correlations among rules more explicit. We measure the degree of inconsistency in formulae of such a rule set (using the Scoring function, Shapley inconsistency values and Blame measure for prioritized knowledge) and compare the

*This is a revised and significantly extended version of [1].

informativeness of these measures. Finally, we propose a new measure of inconsistency for prioritized knowledge which incorporates the normalized number of atoms in a language involved in inconsistency to provide a deeper inspection of inconsistent formulae. We conclude that such measures are useful for the network intrusion domain assuming that introducing expert knowledge for correlation of rules is feasible.

Keywords: Network intrusion detection; inconsistency measures; Snort rules.

1. Introduction

Research into intrusion detection and prevention within the network security domain has accumulated a considerable amount of rules for detecting various intrusions and security threats. For instance, the size of Snort (a well-known open source Network Intrusion Detection System (NIDS) and Network Intrusion Prevention System (NIPS)) has expanded rapidly in recent years [2]. Such intrusion detection systems (IDSs) have provided advantages over traditional security measures in terms of accuracy and reliability [3]. Furthermore, efficient implementation of the rules used by these IDSs has proven that it is possible to integrate these systems into modern high speed networks. One such method has been to execute regular expression (Regex) pattern matching on network traffic using high performance computing facilities.

However, the current IDS rules have their limitations as identified in [4]. The most significant limitation is that for every rule there is one alert, and there is little consideration for subsequent correlation of individually detected primitive alerts. That is to say, the connections between detected suspicious messages/actions are not accurately correlated to allow understanding and to identify unusual trends and patterns in network activity. Therefore high level network intrusion events are not adequately detected and reported. At the same time new rules are continuously being added, in Snort for example [2], so it is crucial to validate any new rules to ensure they are a consistent set before being combined with the global knowledge base (the main rule set). In Snort, new rules are created in the Snort format and this set of new rules must be consistent before it is added to the main Snort rule set.

In this paper, we do not consider how network intrusion rules are obtained or whether each rule is correct. Rather, given a set of such rules (called Snort2Regex[a] rules) created using the Snort2Regex translator [5], our task is to deploy formal Artificial Intelligence (AI) approaches for analyzing these rules and to discover any (logical) inconsistencies among them, i.e. to identify rules which may conflict. Such attempts for inconsistency detection, using formal AI methods, have not been carried out within intrusion detection research. The implication of inconsistency in intrusion detection systems is that the detection of intrusion attempts will not be reliable when conflicting alerts are reported. Similarly, false alarm and

[a] We focus on the Snort2Regex rule set, rather than the original Snort rules, because it addresses the issue of creating a generic IDS rule format which will provide us with a wider range of rules for analysis in the future. It also improves on Snort by focusing on high speed implementation.

mis-detection rates are likely to increase with a greater level of inconsistency in the intrusion detection rule set.

Recent developments in inconsistency handling [6–8] have demonstrated that to effectively manage inconsistencies, it is necessary to propose measures to quantify the degree of inconsistency that individual formulae have contributed, instead of considering the inconsistency of the knowledge base as a whole. These formula-level inconsistency values, such as the Scoring function in [7] and the $S_{I_{MI}}$ measure in [8], can be considered as a degree of blame assigned to each formula in terms of its contribution to the inconsistency of the overall base. The advantage of these methods is that they identify an ordering for the degree of blame that each formula contributes to the inconsistency of the rule set.

There are a number of paths which are available to begin resolving inconsistency, the most basic of which is to simply remove inconsistent rules — one at a time — until the knowledge base is consistent. For example, given an inconsistent knowledge base $K = \{a, \neg a \wedge b, c\}$ then either a or $\neg a \wedge b$ would be removed to make K consistent. However, this approach is limited since information loss is inevitable. An alternative approach (or at least one which can be combined with removal) is to weaken inconsistent rules [6], by modification, in order to remove inconsistency while limiting information loss. For example, given the inconsistent knowledge base $K = \{a, \neg a \wedge b, c\}$ then $\neg a \wedge b$ could be weakened to $\neg a \vee b$ in order to make K consistent. By measuring the degree of blame associated with each rule for inconsistency, using inconsistency measures, it is possible to identify which rules are more inconsistent than others in order to identify the rules which should be removed or weakened first.

In many real word applications, such as in Requirements Engineering (RE) [9], some knowledge will be more important than others. Rather than assuming that all formulae are equally important, knowledge of this nature is often organized into some kind of prioritized or stratified form to differentiate more important vs. less important knowledge. For example, requirements on security would be regarded as more important than requirements on interfaces or the openness of a system. Accordingly inconsistency measures for flat knowledge bases need to be adapted for prioritized bases. The approach described in [10] demonstrates the significance of applying inconsistency measures to prioritized knowledge bases where the advantage of this approach is that the priority of a formula can be considered when measuring inconsistency.

In this preliminary study, we examine the structure of rules in the Snort2Regex rule set and then separate them into their elementary units in order to reflect their primary structure. We then apply some existing approaches to measuring inconsistency for rules in the Snort2Regex format. This study reveals that the $S_{I_{LP_m}}$ measure is most useful for non-prioritized knowledge while the $Blame_v$ prioritized measure potentially provides more useful paths for resolving inconsistency. Finally, we propose a new measure for inconsistency in prioritized knowledge which incorporates the normalized number of the atoms in a language involved in inconsistency.

Since this measure provides a deeper inspection of inconsistent formulae it is potentially more useful for identifying the most problematic rules in the Snort2Regex rule set. The investigation will lead to proposals as to how inconsistencies can be most effectively identified in an IDS rule set and, more generally, how they can be resolved.

1.1. *Network intrusion detection*

In general, the data traveling through a network (called network traffic) is comprised of network packets which are units of data usually consisting of a header and contents (or payload). The header of a packet consists of information for packet delivery such as source address, destination address and checksums to ensure transmission integrity. The contents, on the other hand, will contain the actual data to be delivered. Network intrusion detection systems are a form of network security which monitor this network traffic to detect malicious activity such as unauthorized access, misuse by authorized users or denial of service (DoS) attacks [11]. These IDSs differ from traditional forms of network security, such as firewalls and access rights, which restrict outside access to the network. Instead, IDSs apply rules to network traffic in order to detect possible intrusion attempts, originating from inside and outside the network, as they occur.

The usual method employed by modern IDSs is deep packet inspection (DPI) which has become increasingly popular since packet inspection based on packet header information only (stateful packet inspection (SPI)) is limited. In SPI for example, if a packet is received through port 80, the port used for HTTP, then SPI cannot tell whether the packet contains a web page or a virus. DPI, however, can examine the packet header, like traditional SPI, but in cases where this is not sufficient it is able to examine the packet contents as well. In addition to this, DPI is also capable of examining the contents across multiple packets. This allows DPI to determine if a packet (or series of packets) is suspicious based, not only on how the packet is being delivered, but also on what is being delivered. For example, a malware application may communicate through port 80 (the typical HTTP port) by masquerading as general HTTP traffic. In this case, SPI would permit the traffic while DPI may be able to examine the contents and detect the application initiating the communication. It may not always be necessary to examine the contents of a packet (since this is more computationally expensive), but either method can be applied by a DPI system.

Snort [12], for example, is a widely used signature-based network intrusion detection system which uses DPI. It is signature-based in the sense that intrusions are detected by searching network traffic for predefined signatures which are indicative of possible attacks. These signatures are defined as Snort rules, using the Snort language, and comprise the Snort rule set. Each Snort rule contains conditions (the attack signature), an action (e.g. an alert) and other metadata (e.g. where the rule originated). If an attack signature is detected in network traffic then an action is performed.

The Snort rule set is open source, with an extensive development community and user base, which means that it is easy to create and integrate new rules. The rule set itself is of enterprise quality because it is commercially maintained and has numerous commercial users and contributors. However, the major limitation of the Snort rule set is that it is purpose built for the Snort IDS and so can only be used, in their original format, by systems based on the Snort IDS [5]. Also, a general issue with modern IDSs is the continuing increase in network traffic and network speeds as well as the continuous addition of new detection rules. The effectiveness of current software implementations for intrusion detection and prevention, given the new 40/100Gb Ethernet standards for example, is becoming increasingly limited. Hardware implementations are proving more useful for these high speed networks, however they too are restrictive in that they need to be purpose built for a particular rule format.

The system developed in [5] aims to create a unified DPI rule set integrating, initially, the Snort rule set but later also integrating most other DPI rule sets such as Bro, L7-filter and ClamAV as well. These new generalized rules incorporate regular expression (Regex) pattern matching for DPI since Regex is particularly useful for its high speed performance. A Snort translator, called Snort2Regex, was developed to convert the existing Snort rules into Regex format for high speed hardware and software implementation. Currently, this Snort2Regex rule set contains around 8500 rules which is too large to manually identify inconsistencies between rules. Also, since this rule set is based on Snort, it will constantly expand in order to replicate new rules added to Snort. These factors mean that it will be even more difficult to maintain consistency as time passes, and so an automated approach is needed to detect and resolve conflicts.

The extent of false alarms is a well-known issue in the intrusion detection field and network security more generally. A recent study to determine the extent of false alarms caused by the default Snort IDS was carried out using a private dataset of real network traffic, from a public network, over a period of 40 days [13]. The study found that approximately 96% of total alerts generated by Snort were false alarms. These figures are particularly surprising given the popularity of Snort among IDSs. Further to this study, the false alarm rate caused by Snort was measured against the Defense Advanced Research Projects Agency (DARPA) 1999 synthetic dataset [14]. In this case, approximately 69% of total alerts generated by Snort were considered false alarms. Similar results for false alarms caused by Snort in the DARPA 1991 dataset have been found [15].

While the issue of false alarms in IDSs is as much to do with redundancy and poorly formed rules, inconsistency between rules cannot be ignored. In practice, the results found in [13] were that approximately 3000 alerts were generated per day that required manual verification. The scale of false positives detected by Snort significantly reduces its usefulness and demonstrates the difficulty in attempting to post-process alerts. While the suggestions of fine tuning Snort rules showed an improvement (resulting in a false alarm rate of 87% after fine tuning) [13], each modification has the potential to introduce new inconsistencies which can only be identified by validating the entire set.

1.2. *Related issues*

There are a number of important issues relating to the management of IDS rule sets, of which the validation of rule sets is just the first part. Validation in this case refers to the goal of ensuring that, given a single knowledge base, there are no inconsistencies inherent in it as an isolated set. This set could be the global knowledge base, such as the current Snort or Snort2Regex rule sets, or a set of new rules which are to be added to the global knowledge base. In the latter case for example, the set of new rules should be self-consistent before being added to the global knowledge base since inconsistency among new rules is a separate issue from inconsistencies between new and old rules. This is the aim of this study: that, given a set of IDS rules, how can we identify and begin to resolve inconsistencies.

A related, but separate, issue for the management of an IDS rule set deals with the case when there is a set of new rules which need to be added to the global knowledge base. From a network security perspective, and specifically for intrusion detection, this is particularly important because of the changing nature of security threats and detection mechanisms. What this means is that vulnerabilities in security systems are constantly being identified, fixes are being created, security policies are changing and new vulnerabilities are being introduced. Since old rules may deal with elapsed security threats and new rules have been introduced to deal with new threats, it is important to remove these old rules (especially if they contradict new rules) and to make sure new rules are represented in the rule set. This is the topic of revising the rule set to best represent current security threats.

Similar to the revision of rule sets, another issue which affects the management of any IDS system is how to merge multiple rule sets. These rule sets which are to be combined may simply be separate rule sets built from the same language but developed by different authors, e.g. Snort rules created separately but built from the common Snort language. Alternatively, the rule sets may be heterogenous in which case a common interpretation is need, e.g. if a Snort rule set is to be combined with another popular IDS, such as Bro. In either case the task becomes how to best represent the meaning of both rule sets. For example, if one set identifies a particular attack signature but the other does not, unless this other set explicitly defines this signature as safe, the cautious approach would be to include this attack signature in the merged rule set. In the case where two separate IDSs produce different alerts for the same signature, if the alerts are considered mutually exclusive then there is an inconsistency between both sources which needs to be resolved during the merging process. The inconsistency, in this instance, is unrelated to inconsistencies identified in the validation process since it is an inconsistency between two separate rule sets.

A final issue worth mentioning is that of data inconsistency, i.e. inconsistency in actual network traffic, such as a contradiction between the packet header field and packet contents caused by transmission errors. This type of inconsistency is separate from the issue of inconsistency when validating a rule set although it is important to consider for overall accurate intrusion detection. In contrast to rule inconsistency, data inconsistency may be beneficial for intrusion detection since problematic data could indicate an attack signature. In this case it is desirable to detect the inconsistency, rather than resolve it.

2. Related Work

The issue of understanding and correlating intrusion detection rules is important in order to begin handling inconsistency in the rule set. One such method is to formally encode the IDS knowledge base in Description Logics (DL) where rules are described in a Tbox and facts (network traffic) are described in an Abox [16]. Similarity measures could then be applied [17] to correlate rules and alerts semantically, rather than just syntactically as proposed here. This is an interesting path to address in future work.

In the literature, approaches to network security have tended to focus on the avoidance of inconsistency or the application of workarounds after inconsistencies have been detected. This is in contrast to an approach of formally measuring a degree of blame associated with rules in relation to their contribution to inconsistency. The method applied in [18] for network intrusion and firewall anomalies (of which inconsistencies are a subset) differs from ours in that it seeks to reduce the impact of anomalies by decoupling of rules. They have shown that this method is effective in reducing inconsistencies since constraints are removed and rules are independent. However this leads to redundancy and, from an AI perspective, makes it difficult to construct semantic understanding in order to correlate rules.

The term 'anomaly' [19−21] in network security covers a number of issues relating to potential inconsistency, such as conflicts (mutual exclusion) and ambiguity (e.g. subsumption) [22, 23], however it rarely addresses logical inconsistency. While the term anomaly is not synonymous with inconsistency as it is applied here, it is used in the case of refining a network security rule set of which inconsistency is an important factor.

Techniques and algorithms are presented in [19] for identifying anomalies in distributed firewall policies and implemented in a software tool, 'Firewall Policy Advisor'. This approach does not offer a mechanism to automatically resolve inconsistencies, rather it requires user interaction to decide on the actions to take. If this approach were applied to an IDS, such as the Snort2Regex rule set, it would be prohibitive since any inconsistencies caused by new rules would have to be resolved manually. It could be argued however, that manual resolution is the most desirable path since the security operator should have overall control of the security system [24] but never the less, it is an expensive and often very difficult task for a security operator when dealing with a large collection of rules such as ours. In this case, the approach of formally measuring inconsistencies can be useful in identifying problematic rules so that the security operator can focus on the most promising solutions. Other work on anomalies in Firewalls can be found in [25–27].

In addition to work on inconsistencies in network intrusion systems and firewalls, there are a number of methods which address the issue of Access Control List (ACL) conflicts. In [28], conflicts between packet filter rules (used for packet classification from packet header fields) are defined as the existence of ambiguity between rules, which is a case of inconsistency from [22] not addressed in this paper. While it does not consider explicit logical inconsistency, it applies an automatic system for detecting and resolving these ambiguities. In this case, it finds that using a simple prioritizing approach to avoid inconsistencies is not enough and instead adds resolve filters (new rules which address the ambiguous region of conflicts). This type of approach is analogous to weakening of rules, which we have suggested as a possible path to resolving conflicts. Other examples of resolving conflicts in ACLs can be found in [24, 29], while further work on resolving anomalies in security policies can be found in [20].

More generally, an alternative approach for inconsistency handling in knowledge bases is the use of paraconsistency which views the occurrence of inconsistency as unavoidable. Instead of attempting to resolve inconsistency, the paraconsistent approach seeks to reason in the presence of inconsistency by isolating or ignoring inconsistent areas of a knowledge base [30]. Paraconsistency does not support the trivial assumption that inconsistent information is incorrect and so by maintaining inconsistencies, meaningful information is not lost. However, the extent of false positives (detecting a threat where no threat has occurred) in IDSs [13–15] means that it is a laborious and difficult task to identify true positives (actual attacks). A paraconsistent approach would most likely need to be run online which would further increase the burden on the detection system. For this reason an offline approach of identifying and resolving inconsistencies can prove more effective for an intrusion detection scenario.

3. Preliminaries

Let $\mathcal{L}$ denote the propositional language obtained from a finite set of propositional atoms $\mathcal{P} = \{a, b, c, \ldots\}$, using logical connectives $\{\vee, \wedge, \neg, \rightarrow\}$. Formulae in $\mathcal{L}$ are denoted as α, β, γ, etc. An interpretation w is a total function from $\mathcal{P}$ to $\{0, 1\}$ and let $\mathcal{W}$ denote the set of all interpretations. Interpretation w is a model of formula a, denoted $w \models a$, iff a is true in the usual truth-functional way. We denote set inclusion (resp. strict) by $\subseteq$ (resp. $\subset$). Let $\perp$ denote an inconsistent formula.

If K logically entails ϕ we denote this as $K \vdash \phi$ where ϕ is the logical consequence of K.

We define a knowledge (belief) base as the union of a rule base (rule set) and a fact base, though either can be empty. In the Snort2Regex example, the Snort2Regex rules comprise the rule base and the network traffic (events) comprises the fact base. Together they form the knowledge base.

We define inconsistency (contradiction) as logical inconsistency in the classical sense, i.e. that a knowledge base is unsatisfiable under any interpretation. The term inconsistency measure [6, 7] refers to a measure of the degree to which a formula contributes to the inconsistency of the knowledge base or, when applied to the knowledge base itself, the degree to which the base is inconsistent. The term is synonymous with the terms inconsistency value [8], blame measure and degree of blame [9, 10] (as well as the significance of a formula in relation to inconsistency).

Definition 1 ([7]). Let $\mathcal{D}$ be the set of databases formed from $\mathcal{L}$, where $\mathcal{D} = \wp(\mathcal{L})$. Let $\mathbb{N}$ be the set of natural numbers. For $n \in \mathbb{N}$, $\mathcal{D}^n$ is the set of databases of size n i.e. $\mathcal{D}^n = \{\Gamma \in \mathcal{D} \mid |\Gamma| = n\}$.

Definition 2 (MI and MC [7]). Let $\Delta \in \mathcal{D}$, $Con(\Delta) = \{\Gamma \subseteq \Delta \mid \Gamma \nvdash \perp\}$, and $Incon(\Delta) = \{\Gamma \subseteq \Delta \mid \Gamma \vdash \perp\}$.

$$MC(\Delta) = \{\Phi \in Con(\Delta) \mid \forall \Psi \in Con(\Delta), \Phi \not\subset \Psi\},$$
$$MI(\Delta) = \{\Phi \in Incon(\Delta) \mid \forall \Psi \in Incon(\Delta), \Psi \not\subset \Phi\}.$$

$MI(\Delta)$ is the set of minimal inconsistent subsets (MISs) of Δ, and $MC(\Delta)$ the set of maximal consistent subsets of Δ.

In logics, inconsistency is the occurrence of conflicting information. e.g. both a and $\neg a$ can be proven. However, for the rule set we are considering, we need to determine if inconsistent information will ever be produced, e.g. $\{a \to b, b \to \neg a\}$ is classically consistent but if we later learn $\{a\}$ then $\{a, a \to b, b \to \neg a\}$ is classically inconsistent.

We introduce a new definition of inconsistency below, equivalent to the classical MISs, with which we can 'pre-empt' inconsistencies.

Definition 3 (Θ-inconsistency). Let $\Theta \subseteq \mathcal{L}$ s.t. $\Theta \nvdash \perp$. A knowledge base K is said to be Θ-inconsistent (formerly preemptively-inconsistent [1]) iff $K \cup \Theta \vdash \perp$. The set of minimal Θ-inconsistent subsets of K is $MPS(K \mid \Theta) = \{\Phi \setminus \Theta \mid \Phi \in MI(K \cup \Theta)\}$.

From this definition we can see that if K is inconsistent then $MI(K) \subseteq MPS(K \mid \Theta)$ for all Θ. Θ can be taken as a kind of constraint for a given K. In the application we consider, Θ is in fact a subset of atoms which is a very small proportion of alerts generated, e.g. the most important alerts. $MPS(K \mid \Theta)$ is Θ sensitive, since given a K, there may be more than one Θ satisfying Definition 3. Due to space limitation, we do not consider how this can be addressed [31].

Note that a similar definition to Θ-inconsistency was given in [32] on requirements engineering, in which notation $R \cup \Delta_I$ was used to state that R (a set of requirements) is inconsistent relative to a scenario $\Delta = \langle \Delta_I, \Delta_E \rangle$. However, in [32], the authors retained the computation of the MI set on $R \cup \Delta_I$ whilst we propose the calculation of MPS set on K here.

Although methods deploying model-based approaches in the inconsistency handling community interpret the comma connector in the definition of knowledge bases as synonymous with the conjunctive connector, i.e. $K = \{a, b, c\}$ and $K' = \{a \wedge b \wedge c\}$ are the same. It is important to clarify that, although K and K' could be considered semantically equivalent (i.e. they have the same set of models), this is not the adopted interpretation used in this paper. The reason for this is that we are seeking to identify individual formula responsible for inconsistency and, since K and K' have different cardinalities, i.e. $|K| = 3$ and $|K'| = 1$, this would dramatically affect the formula-level inconsistency measures applied later.

4. Snort2Regex Rules

We currently have 8500 rules from [5], which have been translated into Snort2Regex format using the Snort2Regex translator. Regex pattern matching is then executed using high-speed hardware for intrusion detection on raw packet data. Rules in the Snort2Regex format are difficult to read and analyze in terms of inconsistency so we first translate them into a logical format and then attempt to measure their inconsistency.

4.1. *Regex*

Regex [33] is a formal language for pattern matching of text strings and is used extensively in computing. The Regex strings themselves are interpreted by a regular expression processor which parses input text for patterns matching the predefined Regex string. In the Snort2Regex rule set, Regex strings are used for identifying suspicious packets by examining the character encoded, raw packet data of incoming and outgoing traffic. These Regex strings will examine the header and/or contents of incoming packets for predefined patterns (attack signatures). For example, in Fig. 1 the Snort2Regex rule uses one Regex string to examine the packet header and two Regex strings to examine the packets contents. Then if the raw data of a packet matches these predefined patterns, the packet will be detected as

```
34
headers: 1
^.{1}.{1}.{2}.{2}.{2}.{1}.{1}.{2}\x8F\x75\x45\x53\x8F\x75
    \x44\x01.{2}.{2}.{4}.{4}.{1}.{1}.{2}
content: 2
^001\xACOptix\s+Pro\s+v\d+\x2E\d+\s+Connected\s+
    Successfully\x21/smi
^(?i)001\xACOptix(?-i)(?i)Pro(?-i)(?i)Connected(?-i)(?i)
    Successfully!(?-i)/
msg: "BACKDOOR optix 1.32 runtime detection - init conn"
reference: url,www3.ca.com/securityadvisor/pest/pest.aspx?id
    =453085748
```

Fig. 1. A sample Snort2Regex rule using three Regex strings.

suspicious and an alert will be generated. The syntax for defining Regex strings is complex and an understanding of it is not necessary for this paper.

4.2. *Logical representation of rules*

All rules expressed as Snort2Regex follow the same strict format as defined below.

Definition 4. Let U be a unique rule identifier, m and n integers, A a set of regular expressions for packet header matching, B a set of regular expressions for packet content matching, c a message describing the consequence of the rule and L a reference for identifying a source. A rule Π in Snort2Regex format is defined as:

$$\Pi = \begin{cases} U \\ \mathit{headers} : m \\ A = \{a_1, \ldots, a_m\} \\ \mathit{content} : n \\ B = \{b_1, \ldots, b_n\} \\ \mathit{msg} : c \\ \{\mathit{reference} : L;\} \end{cases}$$

A Snort2Regex rule, Π, says that from any of the given headers coupled with all the contents, we can infer the message. Π can be translated into a unique equivalent propositional formula denoted as $\mathcal{P}(\Pi) = (a_1 \vee \cdots \vee a_m) \wedge (b_1 \wedge \cdots \wedge b_n) \rightarrow c$, where each regular expression in A (resp. B) is represented by an atom a_i (resp. b_i).

Example 1. Given two intrusion detection rules $\mathbb{S}$ (Fig. 2) and $\mathbb{T}$ (Fig. 3) of the format Π, we can convert them to an equivalent logical form.

Both rules contain one regular expression for packet header matching but no regular expressions for packet content matching; they produce different messages for intrusion alerts; and originate from the same sources, i.e. bugtraq,9952, cve,2004-0176 and cve,2004-0367.

Let z be a unique regular expression from $\mathbb{S}$ (resp. $\mathbb{T}$) matched to packet headers:

$$\hat{}.\{1\}.\{1\}.\{2\}.\{2\}.\{2\}.\{1\}\backslash x02.\{2\}.\{4\}.\{4\}.\{2\}.\{2\}$$

```
2
headers: 1
^.{1}.{1}.{2}.{2}.{2}.{1}\x02.{2}.{4}.{4}.{2}.{2}
content: 0
msg: "EXPLOIT IGMP IGAP account overflow attempt"
reference: bugtraq,9952; reference: cve,2004-0176; reference: cve,2004-0367
```

Fig. 2. Snort2Regex rule $\mathbb{S}$.

```
3
headers: 1
^.{1}.{1}.{2}.{2}.{2}.{1}\x02.{2}.{4}.{4}.{2}.{2}
content: 0
msg: "EXPLOIT IGMP IGAP message overflow attempt"
reference: bugtraq,9952; reference: cve,2004-0176; reference: cve,2004-0367
```

Fig. 3. Snort2Regex rule $\mathbb{T}$.

Let v be the message from $\mathbb{S}$:

“EXPLOIT IGMP IGAP account overflow attempt”,

and let x be the message from $\mathbb{T}$:

“EXPLOIT IGMP IGAP message overflow attempt”,

then we have logical rules $\mathcal{P}(\mathbb{S}) = z \to v$ and $\mathcal{P}(\mathbb{T}) = z \to x$.

4.3. *Extending the rule set*

Snort2Regex treats $\mathbb{S}$ and $\mathbb{T}$ independent from each other, suggesting

$$v = \text{“}EXPLOIT\ IGMP\ IGAP\ account\ overflow\ attempt\text{”},$$
$$x = \text{“}EXPLOIT\ IGMP\ IGAP\ message\ overflow\ attempt\text{”},$$

have no connection. However there are obvious connections, e.g. both refer to the IGMP protocol, EXPLOIT attempts, etc, which are ignored by Snort2Regex. To identify connections among rules, we examine the structure of these rules and establish connections, where possible, for inconsistency analysis.

Definition 5. Let Π be a rule in Snort2Regex format and c be its message. Assume that c can be expressed as $c = \{c_1, \ldots, c_t\}$ where each c_i is a meaningful expression for intrusion detection, and is treated as an atom, then c can be expressed as a conjunctive form of atoms obtained from c_i, denoted as $\mathcal{P}(c) = c_1 \wedge \cdots \wedge c_t$.

For instance, given message v in $\mathbb{S}$ where

$$v = \{\mathsf{EXPLOIT}, \mathsf{IGMP}, \mathsf{IGAP}, \mathsf{account}, \mathsf{overflowAttempt}\},$$

its logical form is $\mathcal{P}(v) = q \wedge n \wedge t \wedge r \wedge s$, where q denotes atom EXPLOIT, and u denotes IGMP, etc. Similarly, the logical form for x in $\mathbb{T}$ is $\mathcal{P}(x) = q \wedge n \wedge t \wedge p \wedge s$. Therefore v and x share 4 out of 5 atoms, with only one atom different, i.e. account versus message.

The importance of this transformation is that it allows correlations between rules to be identified and, along with underlying knowledge constraints, makes it easier to maintain the rule set by allowing us to identify inconsistencies.

Example 2. Given expert knowledge suggesting $IGAP \rightarrow \neg EXPLOIT$, denoted as $t \rightarrow \neg q$, then we have a new rule set $K = \{z \rightarrow q \wedge n \wedge t \wedge r \wedge p \wedge s, t \rightarrow \neg q\}$ where z is a Regex string for packet headers. If z is successfully matched to the header of a packet then an inconsistency will occur, i.e., $MPS(K \mid \{z\}) = \{K\}$.

The Θ-inconsistency in K occurs if we define $\Theta = \{z\}$, because we know the IGAP variant will not be involved in EXPLOIT attempts but we have previous IGAP related EXPLOIT rules. When the Regex string z is matched (the relevant attack signature is found) the first rule is triggered, which in turn triggers the second rule and the inconsistency occurs. The main advantage of identifying formulae at this elementary level is that it allows connections among rules to be identified.

Definition 6. Let Π be a rule in Snort2Regex format where r is the collection of Regex strings for finding attack signatures in Π, then $\mathcal{R}(r)$ is the set of unique elementary tasks in r expressed as a set of atoms.

The function $\mathcal{R}(r)$ lets us identify the unique tasks carried out by the collection of Regex strings contained in a Snort2Regex rule where possible. For example, if a rule is searching for an attack signature by examining the protocol field and the source address from the packet header, then instead of considering this as a single task (which could be carried out by one Regex string) we divide this into two tasks, i.e. finding the protocol and finding the source address.

Definition 7. Let Π be a rule in Snort2Regex format where c is the the message in Π and r is the collection of Regex strings in Π. We define $\mathcal{M}_\Pi$ as a mapping $\mathcal{M}_\Pi : 2^{\mathcal{R}(r)} \rightarrow 2^c$ assigning each $r_i \subseteq \mathcal{R}(r)$ to a subset $c_i \subseteq c$, i.e. $\mathcal{M}_\Pi(r_i) = c_i$.

The function $\mathcal{M}_\Pi$ allows us to assign the unique tasks in Π to their relevant conclusions. Continuing the previous example, we can now say that if the two tasks (finding the source address and finding the protocol) are completed, then we carry out the final task of producing the alert.

Example 3. Given a Snort2Regex rule S where $\mathcal{P}(S) = w \rightarrow q \wedge n \wedge t \wedge r \wedge \neg p \wedge s$ and $\mathcal{R}(w) = \{a, b, c\}$ then, after identifying the relevant tasks,

$$\mathcal{M}_S = \{a \rightarrow n, b \rightarrow r \wedge s, c \rightarrow p \wedge s\}.$$

Let K be a set of expert knowledge (constraints) for Snort2Regex rules where $K = \{n \wedge r \wedge \neg p \wedge s \rightarrow q, n \wedge y \rightarrow \neg d\}$. Let E by the resulting knowledge base where $E = \mathcal{M}_S \cup K$ then

$$E = \left\{ \begin{array}{c} a \rightarrow n, \\ b \rightarrow r \wedge s, \\ c \rightarrow p \wedge s, \\ n \wedge r \wedge \neg p \wedge s \rightarrow q, \\ n \wedge y \rightarrow \neg d \end{array} \right\}.$$

These rules are therefore defined in terms of low-level tasks which are not unique to any single rule (the task of finding port 80, for example, can be identified between rules) and where tasks are delegated between interdependent rules. As such, redundancy is removed because the method used for detection of particular exploits is subdivided into the tasks of detecting contributing aspects of the exploit, e.g. detecting the protocol, detecting an exploit attempt and defining the consequence of an exploit attempt for a given protocol.

Regular expressions in Snort2Regex rules are implemented as a whole in a high-performance environment, individual primitive elements are not considered at the regular expression or message level. However, individual primitive elements in regular expressions and in messages do have delicate connections which domain experts can identify in order to make links between what is being detected and how it is being detected. It is also possible to introduce constraints to limit the scope of rules and to identify possibly incorrect rules, where they exist, because they will be inconsistent with the constraints. In this sense, expert knowledge forms the meta-knowledge of the intrusion detection rule set which correlates rules and introduces constraints based on negation (forcing logical inconsistency).

The method for analyzing and correlating rules presented in this section is potentially limited because, while the rules themselves are standardized, rule components are not. This means that while all rules are created using conditions (Regex strings) and actions (such as alerts) in a predefined format, how these actions are created is not predefined, i.e. two alerts referring to the same threat could be written differently where neither is incorrect. For example, given the alert "*Samba send_mailslot buffer overflow attempt*" from an actual Snort2Regex rule, this could potentially be written as "*Samba send_mailslot tried exceed buffer*". In practice however, alert definitions tend to follow the same format, e.g. the application/service name will be stated, the description will refer to known keywords, etc.

It is important to stress that in attempting to reduce false alarms in IDSs, the question of how to correlate rules and alerts needs to be addressed whether a rule inconsistency resolution or an alert post-processing approach is taken. For this reason other methods for correlating rules and alerts may prove useful.

5. Approaches to Inconsistency Detection and Measurements

Generally speaking, we can refer to three types of inconsistent rules: logically inconsistent, conflicting and ambiguous [22]. Logical inconsistency refers to a rule which is unsatisfiable under any interpretation (such as $a \wedge \neg a$), while a conflicting set of rules are those which can be simultaneously triggered but whose conclusions (consequents) are mutually exclusive (such as $a \rightarrow b$ and $a \rightarrow c$ where b and c cannot be simultaneously true). Finally, ambiguous rules are those which can be simultaneously triggered but whose conclusions are different (such as $a \rightarrow b$ and $a \rightarrow c$ where $b \neq c$). When we refer to inconsistency in this paper we understand this to mean logical inconsistency only, however the case of conflicting rules can also be considered as long as it is encoded in a logical format and contained in the knowledge base (since this reverts to logical inconsistency). Ambiguous rules will not be considered since they are an intentional feature of this rule set and are not problematic in relation to intrusion detection.

Some more explanation about the motivation for Definition 3 is in order. Although a subset of rules K translated from rules in the SnortsRegex format may be consistent in the sense of classical logic, when an event (denoted as a) is reported, $K \cup \{a\}$ could be inconsistent. There are potentially millions of events which may be generated by these Snort2Regex rules. However, if we do not detect inconsistencies in the rules until events are detected then actual intrusion detection will be unreliable. Therefore, subsets of rules for detecting intrusion attempts should be consistent given any eventuality, especially for major events (alerts).

Consider this simple example: we have *a blacklist of IP addresses from which exploit attempts are known to originate*; we have a rule stating that *if the source address is blacklisted then an exploit attempt is detected*; we have another rule, originating from expert knowledge, stating that *if the packet uses the IGMP protocol and the source address is blacklisted then an exploit attempt is not detected.* These rules are classically consistent but if we later receive *an IGMP packet originating from a blacklisted IP address* then they will become classically inconsistent. However this inconsistency detection can only be achieved at runtime, and since we need to ensure that inconsistencies will never occur, Θ-inconsistency is needed.

Let us look at this example, given $K_1 = \{a \rightarrow b, b \rightarrow \neg a\}$ then $K_1 \cup \{a\}$ is classically inconsistent. However we cannot simply add formula to create an inconsistency and apply inconsistency measures in the traditional way to the result (such as $K_1 \cup \{a\}$) since there are potentially a large number of variations which could create an inconsistency. Given $K_2 = \{a \vee b \wedge c \rightarrow d, d \rightarrow \neg a \wedge \neg b\}$ there are two alternatives which create an inconsistency, i.e. $K_2 \cup \{a\}$ and $K_2 \cup \{b, c\}$ where $|K_2 \cup \{a\}| = 3$ and $|K_2 \cup \{b, c\}| = 4$. These will produce different inconsistency measures because the number of formulae in the inconsistency will vary. Therefore we consider K_2 as a Θ-inconsistent subset where $|K_2| = 2$. We are only interested in K_2, not $K_2 \cup \{a\}$ or $K_2 \cup \{b, c\}$.

Given the case of conflicting rules, suppose we have a knowledge base $K_3 = \{a \rightarrow b, a \rightarrow c\}$. If b and c are considered mutually exclusive and jointly exhaustive ($a \rightarrow b$ and $a \rightarrow c$ are conflicting) then this concept would need to be represented in K_3, e.g. by containing the rule $(b \wedge \neg c) \vee (\neg b \wedge c)$ s.t. $K_3 = \{a \rightarrow b, a \rightarrow c, (b \wedge \neg c) \vee (\neg b \wedge c)\}$. In this case, K_3 would be Θ-inconsistent since $K_3 \cup \{a\} = \{a, a \rightarrow b, a \rightarrow c, (b \wedge \neg c) \vee (\neg b \wedge c)\}$ and $MI(K_3 \cup \{a\}) = \{K_3 \cup \{a\}\}$.

Suppose we have another knowledge base $K_4 = \{a \rightarrow b, a \rightarrow \neg c\}$. If b and c are considered synonymous ($a \rightarrow b$ and $a \rightarrow \neg c$ are conflicting) then this concept would again need to be represented in K_4 for it to be included in our understanding. In this case, K_4 could include the rules $b \rightarrow c$ and $c \rightarrow b$ s.t. $K_4 = \{a \rightarrow b, a \rightarrow \neg c, b \rightarrow c, c \rightarrow b\}$. K_4 would then be Θ-inconsistent since $K_4 \cup \{a\} = \{a, a \rightarrow b, a \rightarrow \neg c, b \rightarrow c, c \rightarrow b\}$ and $MI(K_4 \cup \{a\}) = \{a, a \rightarrow b, a \rightarrow \neg c, b \rightarrow c\}$.

5.1. *Identifying inconsistencies*

A Snort2Regex parser was developed to convert the rules to abstract propositional logic. While a purely syntactic analysis is beneficial to identify patterns and repetition in the language, we found that negation does not occur and so logical inconsistency is impossible. This is not to say that logical inconsistency does not exist, merely that it cannot be discovered without a semantic understanding. For this preliminary study, we will introduce constraints assumed from expert knowledge to further correlate Snort2Regex rules.

Example 4. Let E be the abstract propositional Snort2Regex rule set K from Example 3:

$$E = \left\{ \begin{array}{c} a \to n, \\ b \to r \wedge s, \\ c \to p \wedge s, \\ n \wedge r \wedge \neg p \wedge s \to q, \\ n \wedge y \to \neg d \end{array} \right\}.$$

Let Z and H be rule sets built from the same Snort2Regex language:

$$Z = \left\{ \begin{array}{c} a \to \neg q, \\ y \to d \wedge s, \\ q \to \neg a, \\ m \vee t \to g, \\ \neg w \to q \end{array} \right\} \qquad H = \left\{ \begin{array}{c} s \to \neg b \wedge \neg c, \\ a \to q, \\ m \to \neg e, \\ k \to l \end{array} \right\}.$$

Given the rule set $\Gamma = E \cup Z \cup H$ (where $|\Gamma| = 14$), then the MPSs of Γ (w.r.t. different Θ) are:

$$\begin{aligned} \mathit{MPS}(\Gamma \mid \{a, r, \neg p, s\}) &= \{M_1, M_3\}, \\ \mathit{MPS}(\Gamma \mid \{n, y\}) &= \{M_2\}, \\ \mathit{MPS}(\Gamma \mid \{b\}) &= \{M_4\}, \\ \mathit{MPS}(\Gamma \mid \{c\}) &= \{M_5\}, \\ \mathit{MPS}(\Gamma \mid \{a\}) &= \{M_6, M_7\} \end{aligned}$$

where

$$\begin{aligned} M_1 &= \{a \to n, n \wedge r \wedge \neg p \wedge s \to q, a \to \neg q\}, \\ M_2 &= \{n \wedge y \to \neg d, y \to d \wedge s\}, \\ M_3 &= \{a \to n, n \wedge r \wedge \neg p \wedge s \to q, q \to \neg a\}, \\ M_4 &= \{b \to r \wedge s, s \to \neg b \wedge \neg c\}, \\ M_5 &= \{c \to p \wedge s, s \to \neg b \wedge \neg c\}, \\ M_6 &= \{a \to q, q \to \neg a\}, \\ M_7 &= \{a \to \neg q, a \to q\}. \end{aligned}$$

Using Θ-inconsistency, we can therefore identify 7 inconsistencies among this sample of Snort2Regex and expert rules given a set of facts (alerts). While identifying minimal inconsistencies is computationally hard, algorithms in [34] have shown that a practical application is possible.

5.2. *Scoring function*

The first inconsistency measure we will look at is the Scoring function since it is closely related to the concept of minimal inconsistent subsets. As described in [7] the Scoring function assigns a value to each subset in terms of its contribution to the overall inconsistency of a rule set based on how many inconsistencies would be resolved if the subset was removed. This provides an intuitive ordering to inconsistent subsets where the greater the score the greater the inconsistency and a value of 0 indicates the subset is entirely consistent.

Definition 8 (Scoring function [7]). Let $\Delta \in \mathcal{D}$ and S be the Scoring function for Δ defined as follows, where $S : \wp(\Delta) \mapsto \mathbb{N}$ and $\Gamma \in \wp(\Delta)$, then $S(\Gamma) = |MI(\Delta)| - |MI(\Delta - \Gamma)|$.

The Scoring function can be seen as the most natural inconsistency measure since for $\alpha \subseteq K$, it measures the number of inconsistencies in K caused by α.

Example 5. Consider $K_3 = \{\neg a, b, a, c, \neg a \vee \neg d, \neg c, d\}$. Then $MI(K_3) = \{M_1, M_2, M_3\}$, where

$$M_1 = \{\neg a, a\}, \quad M_2 = \{c, \neg c\}, \quad M_3 = \{a, \neg a \vee \neg d, d\}.$$

So,

$$S(\{a\}) = 2, \quad S(\{b\}) = 0, \quad S(\{\neg a \vee \neg d\}) = 1,$$

and

$$S(K_3) = 3.$$

With the Scoring function a is most inconsistent because it is involved in two inconsistencies, while b is completely consistent. The Scoring function considers the remaining formulae to be equally inconsistent since they are all involved in a single inconsistency each.

Example 6. The result of applying the Scoring function on formulae in Θ-inconsistent subsets in Γ w.r.t. $\Theta = \{a, r, \neg p, s, n, y, b, c\}$ is:

$$\begin{array}{rrr} S(\{a \rightarrow n\}) = 2 & S(\{b \rightarrow r \wedge s\}) = 1 & S(\{c \rightarrow p \wedge s\}) = 1 \\ S(\{n \wedge r \wedge \neg p \wedge s \rightarrow q\}) = 2 & S(\{n \wedge y \rightarrow \neg d\}) = 1 & S(\{a \rightarrow \neg q\}) = 2 \\ & S(\{y \rightarrow d \wedge s\}) = 1 & S(\{q \rightarrow \neg a\}) = 2 \\ S(\{s \rightarrow \neg b \wedge \neg c\}) = 2 & S(\{a \rightarrow q\}) = 2. & \end{array}$$

The combined scores for subsets E, Z and H would be $S(E) = 6$, $S(Z) = 5$ and $S(H) = 4$. The overall inconsistency score for the rule set Γ is $S(\Gamma) = 7$.

From these results we can identify an inconsistency order based on the Scoring value, i.e. those formulae with a score of 2 would be considered more inconsistent than those with a score of 1 because their removal would resolve more inconsistencies. However it is not sufficiently discriminatory because variations in inconsistency scores are limited (e.g. 0, 1, 2). This means identifying inconsistency ordering will also be limited because the Scoring function does not assign each formula a proportion of blame for the overall inconsistency.

The ability to assign an inconsistency score to the entire rule set allows comparisons to be made between multiple sets, e.g. when merging multiple rule sets, a set could be ignored if it was particularly inconsistent.

The use of the Scoring function for automatic resolution of inconsistent Snort 2Regex rules is limited, however manual resolution may be more desirable anyway and in this case the Scoring function would be useful. Once minimal inconsistencies have been identified in the Snort2Regex rule set, application of the Scoring function becomes trivial. From this, the most problematic rules can be highlighted for the security operator. A further benefit of the Scoring function is that it can be applied to any subset which would allow the identification of consistent subsets of IDS rules. This would be useful to allow clustering of rules in order to avoid inconsistencies, rather than resolve them.

5.3. *Shapley inconsistency value*

The next set of methods we will look at, which were proposed in [8], extend the approach applied by the Scoring function. They address the issue of identifying a proportion of blame associated with each formula in terms of its contribution to the overall inconsistency of the base. Both methods take an inconsistency measure as a payoff function in coalitional form and, using the Shapley value from coalitional game theory, determine the proportional inconsistency for each formula in a base called the Shapley Inconsistency Value (SIV). We first present the definition for the SIV, since it will be combined with the I_{MI} and I_{LP_m} measures, to provide a proportional representation of inconsistency.

Proportional inconsistency measures, such as the SIVs, are significantly more useful for identifying the most problematic rules in an IDS set. By focusing on a rules responsibility for inconsistency in the whole set in relation to other rules, a more representative and discriminatory value can be determined. This in turn allows more precise identification of problematic rules for resolution. During manual resolution for example, it means that fewer rules need to be examined by the security operator.

Definition 9 (Shapley Inconsistency Value [8]). Let I be a basic inconsistency measure. Let K be a knowledge base where $C \subseteq K$. We define the corresponding

Shapley Inconsistency Value (SIV), denoted S_I, as the Shapley value of the coalitional game defined by the function I, i.e. let $a \in K$:

$$S_I^K(a) = \sum_{C \subseteq K} \frac{(c-1)!(n-c)!}{n!}(I(C) - I(C \setminus \{a\})),$$

where n is the cardinality of K and c is the cardinality of C.

The SIV is a proportional measure because it represents the inconsistency of each formula in relation to the inconsistency of the base. It works by calculating the sum of the inconsistency of every subset of the power set of the knowledge base. As a result, a practical application of the SIV is prohibitive for an complete IDS rule set, since for $\alpha \in K$ where $|K| = 14$ then $|\mathcal{P}(K)| = 2^{14}$. If rules are clustered in terms of attack type, attack severity or origin it may be feasible to automatically determine the SIV for the particular subset instead of the set as a whole. Alternatively, the concepts of the SIVs could simplified for performance.

The SIV for a knowledge base is represented as a vector of the SIV for every formula in the base. These vectors are defined below.

Definition 10 (SIV Vector [8]). $S_I(K)$ denotes the vector of the corresponding SIV for each formula of the base K where $a \in K$, i.e. $S_I(K) = (S_I^K(a_1), \ldots, S_I^K(a_n))$.

Definition 11 (SIV Vector for Θ-inconsistency). $S_I(K|\Theta)$ denotes the vector of the corresponding SIV for each formula of the base K w.r.t. Θ where $a \in K$, i.e. $S_I(K|\Theta) = (S_I^{K\cup\Theta}(a_1), \ldots, S_I^{K\cup\Theta}(a_n))$.

Since the Shapley inconsistency measures produces a vector value for the knowledge base (representing the degree of blame for each individual formula) it is difficult to make comparisons with similar measures. When attempting to produce a single inconsistency value to compare with other measures, a simple sum of the Shapley values for each formula is not suitable since this results in the basic inconsistency measure when applied to the knowledge base, i.e. the sum of $S_{I_{MI}}^K(\alpha)$ for $\alpha \in K$ is equal to $I_{MI}(K)$ thereby removing the benefit of the Shapley value. For this reason, the Max SIV was defined in [8] to provide a single inconsistency measure for a knowledge while retaining a representation of the Shapley value. Intuitively this is based on the maximum inconsistency value produced for a formula in the knowledge base.

Definition 12 (Max SIV [8]). Let K be a belief base, $\hat{S}_I(K) = \max S_I^K(a)$, $a \in K$.

Definition 13 (Max SIV for Θ-inconsistency). Let K be a belief base, $\hat{S}_I(K|\Theta) = \max S_I^{K\cup\Theta}(a)$, $a \in K$.

We will consider two functions presented in [35]: the I_{MI} inconsistency measure which, similar to the Scoring function, only takes into account the formulae which contribute to an inconsistency; and the I_{LP_m} inconsistency measure which considers the number of atoms, as well as formulae, contributing to an inconsistency. These measures are then combined with the Shapley inconsistency value to provide a proportional measure.

Firstly, we define the I_{MI} measure [8], which takes into account the extent to which a formula contributes to an inconsistency.

Definition 14 (MI [8]). The MI inconsistency measure is defined as the number of minimal inconsistent subsets of K, i.e.: $I_{MI}(K) = |MI(K)|$.

The I_{MI} measure simply represents the number of inconsistencies in a given K.

Example 7. Consider $K_3 = \{\neg a, b, a, c, \neg a \vee \neg d, \neg c, d\}$. Then $MI(K_3) = \{M_1, M_2, M_3\}$, where

$$M_1 = \{\neg a, a\}, \quad M_2 = \{c, \neg c\}, \quad M_3 = \{a, \neg a \vee \neg d, d\}.$$

So,

$$I_{MI}(K_3) = 3.$$

We now apply the I_{MI} measure as a Shapley inconsistency value, denoted $S_{I_{MI}}$ [8], which takes into account the proportion that a formula contributes to an inconsistency.

Example 8. Consider $K_3 = \{\neg a, b, a, c, \neg a \vee \neg d, \neg c, d\}$. Then $MI(K_3) = \{M_1, M_2, M_3\}$, where

$$M_1 = \{\neg a, a\}, \quad M_2 = \{c, \neg c\}, \quad M_3 = \{a, \neg a \vee \neg d, d\}.$$

So,

$$S^{K_3}_{I_{MI}}(\{a\}) = \frac{1}{2} + \frac{1}{3} = \frac{5}{6},$$

and

$$S_{I_{MI}}(K_3) = \left(\frac{1}{2}, 0, \frac{5}{6}, \frac{1}{2}, \frac{1}{3}, \frac{1}{2}, \frac{1}{3}\right),$$

$$\hat{S}_{I_{MI}}(K_3) = \frac{5}{6}.$$

Taking the formula $\neg a$, we can see that it has a $S_{I_{MI}}$ value of $\frac{1}{2}$ since it is a member of the inconsistency M_1 which has a cardinality of 2. This value is therefore the proportion that $\neg a$ contributes to the inconsistencies that it is involved in, i.e. M_1.

Example 9. Given the rule set Γ in Example 3, the result of applying the $S_{I_{MI}}$ measure on formulae in Θ-inconsistent subsets of Γ w.r.t. $\Theta = \{a, r, \neg p, s, n, y, b, c\}$ is (*to reduce notation in the calculations below, we simply write* Γ *instead of* $\Gamma \cup \Theta$, *this applies to the remaining examples in which* Θ *is defined*):

$$S^{\Gamma}_{I_{MI}}(\{a \to n\}) = \frac{2}{3} \quad S^{\Gamma}_{I_{MI}}(\{b \to r \wedge s\}) = \frac{1}{2} \quad S^{\Gamma}_{I_{MI}}(\{c \to p \wedge s\}) = \frac{1}{2}$$

$$S^{\Gamma}_{I_{MI}}(\{n \wedge r \wedge \neg p \wedge s) \to q\}) = \frac{2}{3} \quad S^{\Gamma}_{I_{MI}}(\{n \wedge y \to \neg d\}) = \frac{1}{2} \quad S^{\Gamma}_{I_{MI}}(\{a \to \neg q\}) = \frac{5}{6}$$

$$S^{\Gamma}_{I_{MI}}(\{y \to d \wedge s\}) = \frac{1}{2} \quad S^{\Gamma}_{I_{MI}}(\{q \to \neg a\}) = \frac{5}{6}$$

$$S^{\Gamma}_{I_{MI}}(\{s \to \neg b \wedge \neg c\}) = 1 \quad S^{\Gamma}_{I_{MI}}(\{a \to q\}) = 1.$$

Giving a vector value

$$S_{I_{MI}}(\Gamma) = \left(\frac{2}{3}, \frac{1}{2}, \frac{1}{2}, \frac{2}{3}, \frac{1}{2}, \frac{5}{6}, \frac{1}{2}, \frac{5}{6}, 0, 0, 1, 1, 0, 0\right).$$

The Max SIV for subsets E, Z and H would be: $\hat{S}^{\Gamma}_{I_{MI}}(E) = \frac{2}{3}$, $\hat{S}^{\Gamma}_{I_{MI}}(Z) = \frac{5}{6}$, and $\hat{S}^{\Gamma}_{I_{MI}}(H) = 1$. And a Max SIV for the rule set $\hat{S}_{I_{MI}}(\Gamma) = 1$.

This function produces more useful results than the Scoring function since it is proportional, and so has a greater degree of discrimination in the inconsistency measures (e.g. $0, \frac{1}{2}, \frac{2}{3}, \frac{5}{6}, 1$) which allows us to identify more significant inconsistent formulae. The $S_{I_{MI}}$ measure also produces a notably different inconsistency ordering, e.g. $\{s \to \neg b \wedge \neg c\}$ has the maximum inconsistency value by $S_{I_{MI}}$ but is less significant by the Scoring function.

The benefit of the $S_{I_{MI}}$ measure is clear from this example. Firstly, it is more specific about which rules are most problematic (in an IDS rule set with hundreds of rules, for example, the difference between identifying 10 or 100 rules as most problematic is significant). Secondly, it can distinguish between a rule which plays a small role in an inconsistency (such as an inconsistency caused by the union of 10 rules) and a rule which plays a large role (such as an inconsistency caused by the union of only two rules).

As an alternative to the I_{MI} measure we introduce the I_{LP_m} measure [8], from LP models.

Definition 15 (I_{LP_m} [8]). Let $\mathcal{P}$ be the set of atoms in a language and K be a belief base from the language. Let ω be a total function from $\mathcal{P}$ to $\{T, F, B\}$, where B indicates an interpretation 'both true and false'. 'Truth values' are ordered as $F <_t B <_t T$. Then

$$I_{LP_m} = \frac{min_{\omega \in Mod_{LP}(K)}\{|\omega!|\}}{|\mathcal{P}|}.$$

Here

$$\omega! = \{x \in \mathcal{P} \mid \omega(x) = B\}$$

which is the set of 'inconsistent' variables in an interpretation. Also

$$Mod_{LP}(\varphi) = \{\omega \in 3^{\mathcal{P}} \mid \omega(\varphi) \in \{T, B\}\}$$

which is the set of models of formula φ.

$$min(Mod_{LP}(\varphi)) = \{\omega \in Mod_{LP}(\varphi) \mid \nexists\, \omega' \in Mod_{LP}(\varphi) \text{ s.t. } \omega'! \subset \omega!\}.$$

The minimum models of a formula are the 'most classical' ones. The LP_m consequence relation is then defined by:

$$K \models_{LP_m} \varphi \text{ iff } min(Mod_{LP}(K)) \subseteq Mod_{LP}(\varphi)$$

So φ is a consequence of K if all the 'most classical' models of K are models of φ.

The I_{LP_m} measure differs from the concepts discuss so far since it represents variable-based inconsistency instead of formula-based inconsistency. Here an inconsistency occurs if there is an interpretation in which a variable is 'both true and false'.

Example 10. Consider $K_3 = \{\neg a, b, a, c, \neg a \vee \neg d, \neg c, d\}$. Then $MI(K_3) = \{M_1, M_2, M_3\}$, where

$$M_1 = \{\neg a, a\}, \quad M_2 = \{c, \neg c\}, \quad M_3 = \{a, \neg a \vee \neg d, d\}.$$

So,

$$I_{LP_m}(M_1) = \frac{1}{1} = 1,$$

since $min(Mod_{LP})(M_1) = \{w_1\}$ where $w_1(a) = B$,

$$I_{LP_m}(M_2) = \frac{1}{1} = 1,$$

since $min(Mod_{LP})(M_2) = \{w_2\}$ where $w_2(c) = B$,

$$I_{LP_m}(M_3) = \frac{1}{2},$$

since $min(Mod_{LP})(M_3) = \{w_3, w_4\}$, where $w_3(a) = T, w_3(d) = B$ or $w_4(a) = B$, $w_4(d) = T$.

Then

$$I_{LP_m}(K_3) = \frac{2}{4} = \frac{1}{2}.$$

In this example the atom d is only inconsistent in ω_3 so the proportion of inconsistent atoms in K_3 is $\frac{1}{2}$.

We now apply the I_{LP_m} measure as a Shapley inconsistency value, denoted $S_{I_{LP_m}}$ [8], which takes into account the proportion that a formula contributes to an inconsistency as well as the proportion of inconsistent atoms in the formula in relation to the language.

Example 11. Consider $K_4 = \{a, \neg a, b, c \wedge \neg c\}$. Then $MI(K_4) = \{M_1, M_2\}$, where

$$M_1 = \{a, \neg a\}, \quad M_2 = \{c \wedge \neg c\}.$$

So,

$$S_{I_{LP_m}}(K_4) = \left(\frac{\frac{1}{3}}{2}, \frac{\frac{1}{3}}{2}, 0, \frac{\frac{1}{3}}{1}\right) = \left(\frac{1}{6}, \frac{1}{6}, 0, \frac{1}{3}\right).$$

Consider $K_4' = \{a \wedge \neg a \wedge b \wedge c \wedge \neg c\}$. Then $MI(K_4') = K_4$. So,

$$S_{I_{LP_m}}(K_4') = \left(\frac{\frac{2}{3}}{1}\right) = \left(\frac{2}{3}\right).$$

Then

$$\hat{S}_{I_{LP_m}}(K_4) = \frac{1}{3}, \quad \hat{S}_{I_{LP_m}}(K_4') = \frac{2}{3}.$$

This example highlights the difference between the $S_{I_{LP_m}}$ value and previous measures in that it does not simply identify K_4' as completely inconsistent since the atom b does not contribute to inconsistency.

We can see this function introduces the concept that the more atoms of a language affected by inconsistency, the more inconsistent the base, e.g. $S^{\Gamma}_{I_{MI}}(K_4') = (1)$ but $S^{\Gamma}_{I_{LP_m}}(K_4') = \left(\frac{2}{3}\right)$ since the atom b does not contribute to inconsistency. It is also more discriminatory than both the Scoring function and $S_{I_{MI}}$ measure when there are variations in the number of atoms contributing to inconsistency.

Example 12. Given the rule set Γ, the result of applying the $S_{I_{LP_m}}$ measure on formulae in Θ-inconsistent subsets of Γ w.r.t. $\Theta = \{a, r, \neg p, s, n, y, b, c\}$ is:

$$S^{\Gamma}_{I_{LP_m}}(\{a \rightarrow n\}) = \frac{2}{51} \qquad S^{\Gamma}_{I_{LP_m}}(\{b \rightarrow r \wedge s\}) = \frac{1}{34} \quad S^{\Gamma}_{I_{LP_m}}(\{c \rightarrow p \wedge s\}) = \frac{1}{34}$$

$$S^{\Gamma}_{I_{LP_m}}(\{n \wedge r \wedge \neg p \wedge s \rightarrow q\}) = \frac{2}{51} \quad S^{\Gamma}_{I_{LP_m}}(\{n \wedge y \rightarrow \neg d\}) = \frac{1}{34} \quad S^{\Gamma}_{I_{LP_m}}(\{a \rightarrow \neg q\}) = \frac{5}{102}$$

$$S^{\Gamma}_{I_{LP_m}}(\{y \rightarrow d \wedge s\}) = \frac{1}{34} \quad S^{\Gamma}_{I_{LP_m}}(\{q \rightarrow \neg a\}) = \frac{5}{102}$$

$$S^{\Gamma}_{I_{LP_m}}(\{s \rightarrow \neg b \wedge \neg c\}) = \frac{1}{17} \qquad S^{\Gamma}_{I_{LP_m}}(\{a \rightarrow q\}) = \frac{1}{17}.$$

Giving a vector value

$$S_{I_{LP_m}}(\Gamma) = \left(\frac{2}{51}, \frac{1}{34}, \frac{1}{34}, \frac{2}{51}, \frac{1}{34}, \frac{5}{102}, \frac{1}{34}, \frac{5}{102}, 0, 0, \frac{1}{17}, \frac{1}{17}, 0, 0\right).$$

The Max SIV for subsets E, Z and H would be $\hat{S}^{\Gamma}_{I_{LP_m}}(E) = \frac{2}{51}$, $\hat{S}^{\Gamma}_{I_{LP_m}}(Z) = \frac{5}{102}$, and $\hat{S}^{\Gamma}_{I_{LP_m}}(H) = \frac{1}{17}$. And a Max SIV for the rule set $\hat{S}_{I_{LP_m}}(\Gamma) = \frac{1}{17}$.

The reason there is less variance than the $S_{I_{LP_m}}$ measure, in this case, is because the Θ-inconsistencies in Γ only relate to one atom in each instance. In other cases, where inconsistencies are caused by multiple atoms, we would expect a greater degree of variation in inconsistency values.

The obvious benefit of the $S_{I_{LP_m}}$ for an intrusion detection rule set is that in can further discriminate between how problematic rules are, based the amount of the language they incorporate, i.e. more complicated rules are more/less problematic than less complicated rules.

5.4. *Blame$_v$ prioritized measure*

In network intrusion detection systems there are a number of ways in which the concept of prioritization could be applied to rules. These include the age of a rule, the severity/significance of an attack that a rule relates to, the reliability of the rule author and a confidence or certainty rating for a rule. In most cases these priority ratings will be a form of meta-knowledge where multiple interpretations of priority may be applied at once, for example a rule relating to a serious attack developed by a reputable author would be more important than a rule relating to a serious attack but developed by a unknown author. In terms of inconsistency, the application of this meta-knowledge is not only more accurate than simply considering all rules as equally important, but is useful in deciding on how to begin resolving conflicts when they arise by incorporating these priority levels.

There are two types of prioritized knowledge bases described in [10], namely the Type-I and the Type-II prioritized knowledge base, which provide an intuitive ordering for formulae within a base. The approach of Type-I knowledge bases is to assign a numerical significance value to each formula. Its limitation is that it is a quantitative approach to a qualitative concept, i.e. it is difficult to determine a significance value for a formula in relation to other formulae in the base. The exception to this would be when representing uncertainty by probability values, for example more certain rules could be considered more important than less certain rules. On the other hand, Type-II prioritized knowledge bases use a qualitative approach by dividing formulae into subsets based on their priority instead of assigning numerical values to formulae. A knowledge base is regarded as a collection of subsets with the first subset containing the most important formulae.

In this subsection, we use $\tilde{K}$ to represent a classical (flat) knowledge base and K to represent a Type-II prioritized knowledge base. We use $\tilde{\mathcal{K}}_{\mathcal{L}}$ to denote the set of classical knowledge bases definable from the language $\mathcal{L}$[10]. The relevant prioritized knowledge base can then be defined.

Definition 16 (Type-II prioritized knowledge base). Let T be a Type-II prioritized knowledge base [10] with k priority levels. Let $T = \langle \tilde{A}_1, \ldots, \tilde{A}_k \rangle$ containing subsets of formulae assigned to priority levels $1, \ldots, k$ respectively, where $\tilde{A}_1$ has the most important formulae, and $\tilde{A}_k$ the least.

Example 13. Let Γ be a Type-II prioritized knowledge base and E, Z and H be the sets of formulae from Example 3. Let E be most significant and H be least significant, i.e. $\Gamma = \langle E, Z, H \rangle$. The set of MPSs of Γ from Example 3 w.r.t.

$\Theta = \{a, r, \neg p, s, n, y, b, c\}$ are now:

$$\begin{aligned}
M_1 &= \langle\{a \to n, n \wedge r \wedge \neg p \wedge s \to q\}, \{a \to \neg q\}, \emptyset\rangle,\\
M_2 &= \langle\{n \wedge y \to \neg d\}, \{y \to d \wedge s\}, \emptyset\rangle,\\
M_3 &= \langle\{a \to n, n \wedge r \wedge \neg p \wedge s \to q\}, \{q \to \neg a\}, \emptyset\rangle,\\
M_4 &= \langle\{b \to r \wedge s\}, \emptyset, \{s \to \neg b \wedge \neg c\}\rangle,\\
M_5 &= \langle\{c \to p \wedge s\}, \emptyset, \{s \to \neg b \wedge \neg c\}\rangle,\\
M_6 &= \langle\emptyset, \{q \to \neg a\}, \{a \to q\}\rangle,\\
M_7 &= \langle\emptyset, \{a \to \neg q\}, \{a \to q\}\rangle.
\end{aligned}$$

The same MISs (or MPSs) of a prioritized knowledge base are equivalent to those from a classical knowledge bases except that formulae are prioritized.

The $Blame_v$ inconsistency measure for prioritized knowledge bases, from [10], is divided into a number of components. To begin defining the $Blame_v$ measure we will introduce each of these components, starting with the concept of Opposed formulae.

Definition 17 (Opposed formulas [10]). Let M be a minimal inconsistent prioritized knowledge base and α^P a formula being attached with a priority level. Then the set of opposed formulae to α^P w.r.t. M, denoted $Opp(M, \alpha^P)$, is defined as:

$$Opp(M, \alpha^P) = \begin{cases} \{\alpha^P\}, & \text{if } M = \{\alpha^P\},\\ M - \{\alpha^P\}, & \text{if } \{\alpha^P\} \subset M,\\ \emptyset, & \text{if } \alpha^P \notin M. \end{cases}$$

This is the set of remaining formulae from a MIS resulting from the removal of a single formula from the MIS at each priority level. The cardinality of this set is used in determining a degree of blame for the formulae in an inconsistency. When a formula is a minimal inconsistency (internally inconsistent) the Opposed formula is itself.

Example 14. Consider $K_5 = \langle\{\neg a, b\}, \{a, c, \neg a \vee \neg d\}, \{\neg c, d\}\rangle$. Then $MI(K_5) = \{M_1, M_2, M_3\}$, where

$$M_1 = \langle\{\neg a\}, \{a\}, \emptyset\rangle, \quad M_2 = \langle\emptyset, \{c\}, \{\neg c\}\rangle, \quad M_3 = \langle\emptyset, \{a, \neg a \vee \neg d\}, \{d\}\rangle.$$

So,

$$\begin{aligned}
Opp\,(M_1, a) &= \langle\{\neg a\}, \emptyset, \emptyset\rangle,\\
Opp\,(M_1, \neg a) &= \langle\emptyset, \{a\}, \emptyset\rangle.
\end{aligned}$$

Example 15. Given the MPS M_1 from $MPS(\Gamma|\{a, r, \neg p, s\})$, the result of applying the $Opp(M_1, m)$ for $m \in M_1$ is:

$$\begin{aligned}
Opp(M_1, a \to n) &= \langle\{n \wedge r \wedge \neg p \wedge s \to q\}, \{a \to \neg q\}, \emptyset\rangle,\\
Opp(M_1, n \wedge r \wedge \neg p \wedge s \to q) &= \langle\{a \to n\}, \{a \to \neg q\}, \emptyset\rangle,\\
Opp(M_1, a \to \neg q) &= \langle\{a \to n, n \wedge r \wedge \neg p \wedge s \to q\}, \emptyset, \emptyset\rangle.
\end{aligned}$$

The next step in defining the *Blame*$_v$ measure is to introduce the Significance function. The Significance function alone is not an inconsistency measure, it merely determines a weighting for a given set.

Definition 18 (Significance function *Sig*$_c$ [10]). The significance function for classical knowledge bases, denoted Sig_c is a function $Sig_c : \tilde{\mathcal{K}}_{\mathcal{L}} \mapsto \mathbb{N}$ s.t. $\forall \tilde{K} \in \tilde{\mathcal{K}}_{\mathcal{L}}$,

$$Sig_c(\tilde{K}) = |\tilde{K}|.$$

The Significance function can be applied to each priority level to determine a weighting for each level, i.e. the number of formulae at each level.

Example 16. Consider $\tilde{K}_3 = \{\neg a, b, a, c, \neg a \vee \neg d, \neg c, d\}$. Then $MI(\tilde{K}_3) = \{\tilde{M}_1, \tilde{M}_2, \tilde{M}_3\}$, where

$$\tilde{M}_1 = \{\neg a, a\}, \quad \tilde{M}_2 = \{c, \neg c\}, \quad \tilde{M}_3 = \{a, \neg a \vee \neg d, d\}.$$

So,

$$Sig_c(\tilde{M}_1) = 2, \quad Sig_c(\tilde{M}_2) = 2, \quad Sig_c(\tilde{M}_3) = 3,$$

and

$$Sig_c(\tilde{K}_3) = 7.$$

Next, we define the Inc_c measure which is another component of the *Blame*$_v$ measure.

Definition 19 (The inconsistency measure *Inc*$_c$ [10]). The inconsistency measure for classical knowledge bases, denoted as Inc_c, is a function $Inc_c : \tilde{\mathcal{K}}_{\mathcal{L}} \mapsto \mathbb{R}$ s.t. $\forall\, \tilde{K} \in \tilde{\mathcal{K}}_{\mathcal{L}}$,

$$Inc_c(\tilde{K}) = \sum_{\tilde{M} \in MI(\tilde{K})} Inc_c(\tilde{M}),$$

where $Inc_c(\tilde{M}) = \frac{1}{|\tilde{M}|}$ for each $\tilde{M} \in MI(\tilde{K})$.

Example 17. Consider $\tilde{K}_3 = \{\neg a, b, a, c, \neg a \vee \neg d, \neg c, d\}$. Then $MI(\tilde{K}_3) = \{\tilde{M}_1, \tilde{M}_2, \tilde{M}_3\}$, where

$$\tilde{M}_1 = \{\neg a, a\}, \quad \tilde{M}_2 = \{c, \neg c\}, \quad \tilde{M}_3 = \{a, \neg a \vee \neg d, d\}.$$

So,

$$Inc_c(\tilde{M}_1) = \frac{1}{2}, \quad Inc_c(\tilde{M}_2) = \frac{1}{2}, \quad Inc_c(\tilde{M}_3) = \frac{1}{3},$$

and

$$Inc_c(\tilde{K}_3) = \frac{8}{6} = \frac{4}{3}.$$

We continue defining the structure of the $Blame_v$ measure by introducing the Inc_v measure [10]. This, along with the Opposed formula, determines the $Blame_v$ value of a formula.

Definition 20 (The Significance Vector Sig_v [10]). Let $K = \langle K(1), \ldots, K(n) \rangle$ be a Type-II prioritized knowledge base. The significance vector for K, denoted $Sig_v(K)$, is defined as

$$Sig_v(K) = (Sig_c(K(1)), \ldots, Sig_c(K(n))).$$

The significance vector Sig_v allows us to define the significance of a knowledge base at each priority level.

Example 18. Consider $K_5 = \langle \{\neg a, b\}, \{a, c, \neg a \vee \neg d\}, \{\neg c, d\} \rangle$. Then $MI(K_5) = \{M_1, M_2, M_3\}$, where

$$M_1 = \langle \{\neg a\}, \{a\}, \emptyset \rangle, \quad M_2 = \langle \emptyset, \{c\}, \{\neg c\} \rangle, \quad M_3 = \langle \emptyset, \{a, \neg a \vee \neg d\}, \{d\} \rangle.$$

So,

$$Sig_v(K_5) = (2, 3, 2).$$

In this example, K_5 has the highest significance value at the second priority level.

Definition 21 (Inc_v inconsistency measure [10]). Let $K = \langle K(1), \ldots, K(n) \rangle$ be a Type-II prioritized knowledge base. The inconsistency measure for K, denoted as $Inc_v(K)$, is defined as:

$$Inc_v(K) = \sum_{M \in MI(K)} Inc_v(M),$$

where $Inc_v(M) = \frac{Sig_v(M)}{|M|} \times Inc_c(M^*)$ for each $M \in MI(K)$.

Especially, we call $Inc_v^{(i)}(M) = \frac{Sig_v^{(i)}(M)}{|M|} \times Inc_c(M^*)$ the ith level inconsistency amount of M, and abbreviate it as $Inc_v^{(i)}(M)$.

The Inc_v measure combines the Sig_v vector with the Inc_c value to produce an inconsistency measure for each priority level.

Example 19. Consider $K_5 = \langle \{\neg a, b\}, \{a, c, \neg a \vee \neg d\}, \{\neg c, d\} \rangle$. Then $MI(K_5) = \{M_1, M_2, M_3\}$, where

$$M_1 = \langle \{\neg a\}, \{a\}, \emptyset \rangle, \quad M_2 = \langle \emptyset, \{c\}, \{\neg c\} \rangle, \quad M_3 = \langle \emptyset, \{a, \neg a \vee \neg d\}, \{d\} \rangle.$$

So,

$$Inc_v(M_1) = \left(\frac{1}{2^2}, \frac{1}{2^2}, \frac{0}{2^2} \right) = \left(\frac{1}{4}, \frac{1}{4}, 0 \right),$$

$$Inc_v(M_2) = \left(\frac{0}{2^2}, \frac{1}{2^2}, \frac{1}{2^2} \right) = \left(0, \frac{1}{4}, \frac{1}{4} \right),$$

$$Inc_v(M_3) = \left(\frac{0}{3^2}, \frac{2}{3^2}, \frac{1}{3^2} \right) = \left(0, \frac{2}{9}, \frac{1}{9} \right),$$

and

$$Inc_v(K_5) = \left(\frac{1}{4}, \frac{13}{18}, \frac{13}{36}\right).$$

Again, K_5 has the highest inconsistency value at the second priority level.

Example 20. Given the Type-II prioritized knowledge base Γ w.r.t. $\Theta = \{a, r, \neg p, s, n, y, b, c\}$ and the MPS M_1, the result of applying the Inc_v inconsistency measure is:

$$Inc_v(M_1) = \left(\frac{2}{3^2}, \frac{1}{3^2}, \frac{0}{3^2}\right) = \left(\frac{2}{9}, \frac{1}{9}, 0\right).$$

The Inc_v result for Γ is $Inc_v(\Gamma) = (\frac{43}{36}, \frac{35}{36}, 1)$.

We now present the full definition of the $Blame_v$ measure from [10].

Definition 22 (The Blame of each formula for the κth level inconsistency [10]). Let $K = \langle K(1), \ldots, K(n)\rangle$ be a Type-II prioritized knowledge base. Then for each $1 \leq k \leq n$, the blame of each formula of K for the kth level inconsistency of K, denoted $Blame_v^{(k)}$, is defined as follows:

$$\forall\, \alpha \in K, Blame_v^{(k)}(K, \alpha) = \sum_{M \in MI(K)} Blame_v^{(k)}(M, \alpha),$$

where

$$Blame_v^{(k)}(M, \alpha) = \begin{cases} \dfrac{Sig_v^{(k)}(Opp(M, \alpha))}{\sum_{\beta \in M} Sig_v^{(k)}(Opp(M, \beta))} \times Inc_v^{(k)}(M), & \text{if } |M(k)| > 0, \\ 0, & \text{if } |M(k)| = 0, \end{cases}$$

for each $M \in MI(K)$.

The $Blame_v$ measure for the formula α can be summarized as the sum of the blame of α, at priority level K, in each inconsistency in which it is involved.

Example 21. Consider $K_5 = \langle\{\neg a, b\}, \{a, c, \neg a \vee \neg d\}, \{\neg c, d\}\rangle$. Then $MI(K_5) = \{M_1, M_2, M_3\}$, where

$$M_1 = \langle\{\neg a\}, \{a\}, \emptyset\rangle, \quad M_2 = \langle\emptyset, \{c\}, \{\neg c\}\rangle, \quad M_3 = \langle\emptyset, \{a, \neg a \vee \neg d\}, \{d\}\rangle.$$

So,

$$Blame_v^{(1)}(M_1, a) = \frac{|\{\neg a\}|}{|\{\neg a\}| + |\emptyset|} \times \frac{1}{4} = \frac{1}{4},$$

$$Blame_v^{(2)}(M_1, a) = \frac{|\emptyset|}{|\emptyset| + |\{\neg a\}|} \times \frac{1}{4} = 0,$$

$$Blame_v^{(3)}(M_1, a) = \frac{|\emptyset|}{|\emptyset| + |\emptyset|} \times 0 = 0,$$

and

$$Blame_v(M_1, a) = \left(\frac{1}{4}, 0, 0\right).$$

This example indicates the formula a is to blame for the inconsistency M_1 at the first priority level only.

Example 22. Given the results of $Opp(M_1, m)$ for $m \in M_1$ and $Inc_v(M_1)$, the result of $Blame_v^{(k)}$ for the formula $a \to n$ in each priority level is:

$$Blame_v^{(1)}(M_1, a \to n) = \frac{|\{n \wedge r \wedge \neg p \wedge s \to q\}|}{\begin{array}{c}|\{n \wedge r \wedge \neg p \wedge s \to q\}| + |\{a \to n\}| \\ + \ |\{a \to n, n \wedge r \wedge \neg p \wedge s \to q\}|\end{array}} \times \frac{2}{9} = \frac{1}{18},$$

$$Blame_v^{(2)}(M_1, a \to n) = \frac{|\{a \to \neg q\}|}{|\{a \to \neg q\}| + |\{a \to \neg q\}| + |\emptyset|} \times \frac{1}{9} = \frac{1}{18},$$

$$Blame_v^{(3)}(M_1, a \to n) = \frac{|\emptyset|}{|\emptyset| + |\emptyset| + |\emptyset|} \times 0 = 0.$$

Giving an overall $Blame_v$ value for $a \to n$ in M_1:

$$Blame_v(M_1, a \to n) = \left(\frac{1}{18}, \frac{1}{18}, 0\right).$$

A vector value for a formula represents the total blame at each priority level for the inconsistency of the knowledge base.

Definition 23 (The Blame of each Formula for Inconsistency [10]). Let $K = \langle K(1), \ldots, K(n)\rangle$ be a Type-II prioritized knowledge base. The blame of each formula of K for the inconsistency of K, denoted $Blame_v$, is defined as follows:

$$\forall\, \alpha \in K, Blame_v(K, \alpha) = (Blame_v^{(1)}(K, \alpha), \ldots, Blame_v^{(n)}(K, \alpha),$$

where $Blame_v^{(k)}(K, \alpha)$ is the blame of α to the kth level inconsistency of K for each $1 \leq k \leq n$.

Example 23. Consider $K_5 = \langle\{\neg a, b\}, \{a, c, \neg a \vee \neg d\}, \{\neg c, d\}\rangle$. Then $MI(K_5) = \{M_1, M_2, M_3\}$, where

$$M_1 = \langle\{\neg a\}, \{a\}, \emptyset\rangle, \quad M_2 = \langle\emptyset, \{c\}, \{\neg c\}\rangle, \quad M_3 = \langle\emptyset, \{a, \neg a \vee \neg d\}, \{d\}\rangle.$$

So,

$$Blame_v(M_1, a) = \left(\frac{1}{4}, 0, 0\right),$$

$$Blame_v(M_3, a) = \left(0, \frac{1}{18}, \frac{1}{18}\right),$$

and

$$Blame_v(K_5, a) = \left(\frac{1}{4}, \frac{1}{18}, \frac{1}{18}\right).$$

Given the result of the $Blame_v$ measure on each formula of the knowledge base, an ordering for the degree of blame between formulae can then be defined.

Definition 24 (The relation of less blameful than, $\leq_B$ [10]). Let K be a Type-II prioritized knowledge base. A binary relation on K, denoted as $\leq_B$, is defined as follows: $\alpha \leq_B \beta$ if and only if $Blame_v(K, \alpha) \preceq Blame_v(K, \beta)$.

Further, $\alpha <_B \beta$ if $\alpha \preceq_B \beta$ and $\beta_B \alpha$. $\alpha \simeq_B \beta$ if $\alpha \leq_B \beta$ and $\beta \leq_B \alpha$. We say that α is less blameful for the inconsistency in K than β if $\alpha <_B \beta$.

Example 24. Given the Type-II prioritized rule set Γ, the result of applying the $Blame_v$ measure on formulae in Θ-inconsistent subsets of Γ w.r.t. $\Theta = \{a, r, \neg p, s, n, y, b, c\}$ would be:

$$Blame_v(\Gamma, a \to n) = \left(\frac{1}{9}, \frac{1}{9}, 0\right) \qquad Blame_v(\Gamma, b \to r \wedge s) = \left(0, 0, \frac{1}{4}\right)$$

$$Blame_v(\Gamma, c \to p \wedge s) = \left(0, 0, \frac{1}{4}\right) \qquad Blame_v(\Gamma, n \wedge r \wedge \neg p \wedge s \to q) = \left(\frac{1}{9}, \frac{1}{9}, 0\right)$$

$$Blame_v(\Gamma, n \wedge y \to \neg d) = \left(0, \frac{1}{4}, 0\right) \qquad Blame_v(\Gamma, a \to \neg q) = \left(\frac{1}{9}, 0, \frac{1}{4}\right)$$

$$Blame_v(\Gamma, y \to d \wedge s) = \left(\frac{1}{4}, 0, 0\right) \qquad Blame_v(\Gamma, q \to \neg a) = \left(\frac{1}{9}, 0, \frac{1}{4}\right)$$

$$Blame_v(\Gamma, s \to \neg b \wedge \neg c) = \left(\frac{1}{2}, 0, 0\right) \qquad Blame_v(\Gamma, a \to q) = \left(0, \frac{1}{2}, 0\right).$$

This produces an ordering for the distribution of blame between inconsistent formulae of Γ, i.e.

$$\begin{aligned}(\{b \to r \wedge s\} \simeq_B \{c \to p \wedge s\}) &<_B \{n \wedge y \to \neg d\} <_B \{a \to q\} \\ &<_B (\{a \to \neg q\} \simeq_B \{q \to \neg a\}) <_B (\{a \to n\} \\ &\simeq_B \{n \wedge r \wedge \neg p \wedge s \to q\}) <_B \{y \to d \wedge s\} <_B \{s \to \neg b \wedge \neg c\},\end{aligned}$$

where $s \to \neg b \wedge \neg c$ is considered as most to blame for the inconsistency of Γ.

The results for the $Blame_v$ measure are striking since the formula $a \to q$ (considered one of the most problematic formulae by the Scoring function, $S_{I_{MI}}$ measure and $S_{I_{LP_m}}$ measure) is here considered to much less problematic. The reason for this is because $a \to q$ is considered a low priority rule in the original definition of Γ. This result clearly demonstrates the significance of integrating prioritization in determining the degree of blame to associate with a formula for inconsistency.

The implication for a practical IDS rule set is that the degree of importance for rules, which can be determined by numerous factors, can have a significant impact on assign an inconsistency rating. This, along with the extra discrimination provided by the $Blame_v$ measure, will assist resolution whether a manual or automatic mechanism is used.

5.5. *Comparison of current inconsistency measures*

We have applied a number of approaches to measuring the degree of inconsistency in knowledge bases. The informativeness of these measures can be summarized as:

- the $S_{I_{MI}}$ measure is more representative than the Scoring function because it considers the proportion that a formula contributes to the base;
- the $S_{I_{LP_m}}$ measure is an extension of the $S_{I_{MI}}$ measure because it also considers the proportion of a language affected by inconsistency;
- when formulae are prioritized the $Blame_v$ measure, unlike the $S_{I_{MI}}$ measure, can take into account this ordering. However it cannot represent the proportion of language affected by inconsistency.

For all the inconsistency measures discussed, an ordering can be defined in terms of a formula's contribution to the inconsistency of a base where the greater the inconsistency value, the greater the degree of blame. This ordering can then be used in the process of resolving inconsistency because the most inconsistent formulae have been identified.

Reducing or even eliminating inconsistencies in an intrusion detection system such as the Snort2Regex rule set, would be a significant step in reducing false alarms and mis-detection. In the end this will improve the performance of the IDS for network security. In terms of this formal inconsistency measure, their potential benefit for resolving these inconsistencies is based on how they can be practically implemented. As we have seen, the Scoring function is trivial to calculate while the $Blame_v$ measure would not have a significant impact on performance. However it is unlikely that the SIVs could be applied in practice without a large amount of refinement or simplification. In this case it would still be possible to incorporate their basic concepts such as language inconsistency.

6. $Blame_L$ Measure for Prioritized Knowledge Bases

When knowledge is prioritized it is intuitive that incorporating this relative priority level is beneficial when measuring the degree of inconsistency of formulae, i.e. if formulae are considered more or less important then they will contribute to inconsistency to a greater or lesser degree. While the $Blame_v$ measure offers a solution to this problem, it treats all formulae equally regardless of the proportion of the language involved. Similarly, the $S_{I_{LP_m}}$ measure is able to provide an analysis of inconsistency in terms of the number of inconsistent variables in an interpretation, but this measure can only be applied to classical knowledge bases. To overcome this problem, we propose a solution which extends the $Blame_v$ measure using an approach similar to the $S_{I_{LP_m}}$ value. In this way, we can incorporate the normalized number of the atoms in a language involved in inconsistency to provide a deeper inspection of inconsistent formulae to define a new measure, denoted $Blame_L$, for inconsistency in prioritized knowledge bases.

The advantage of this approach for intrusion detection rules is that it can distinguish between rules based on their complexity, i.e. the proportion of the language involved in each rule. This is common variation between IDS rules which is not considered by the $Blame_v$ measure and the extra discrimination is useful for identifying the most problematic rules for resolution.

The $Blame_L$ measure is broken up into components in the same way as the $Blame_v$ measure. These components will be redefined for the new measure where necessary. We begin by introducing the Sig_P significance function which represents the normalized number of the atoms in a language involved in inconsistency. This function is also able to take into account the priority level of atoms in the language, i.e. for the set of atoms in K, the number of atoms at any $K(i)$ may be different.

Definition 25 (Significance function Sig_P). Let $\mathcal{P}$ be the set of atoms in a language and $K = \langle K(1), \ldots, K(n)\rangle$ be a Type-II prioritized knowledge base from the language. Let $\mathcal{P}_K$ be the set of atoms in K. The significance value for K, denoted $Sig_P(K)$, is defined as

$$Sig_P(K) = \sum_{M \in M(K)} Sig_P(M \mid K),$$

where $Sig_P(M \mid K) = \frac{|\mathcal{P}_M|}{|\mathcal{P}_K|}$ for each $M \in MI(K)$.

Especially, we call $Sig_P^{(i)}(M \mid K) = \frac{|\mathcal{P}_{M(i)}|}{|\mathcal{P}_{K(i)}|}$ the ith level significance of M in relation to K, and abbreviate it as $Sig_P^{(i)}(M \mid K)$.

Informally speaking, $Sig_P^{(i)}(M \mid K)$ describes the normalized number of variables of $M(i)$ involved in inconsistency within the context of $K(i)$. It provides a more fine-grained inspection on variables involved in inconsistency of K. Note that if K is a minimal inconsistent knowledge base, then $Sig_P^{(i)}(K \mid K) = 1$ for all i s.t. $K(i) \neq \emptyset$.

Example 25. Consider $K_3 = \langle\{\neg a, b\}, \{a, c, \neg a \vee \neg d\}, \{\neg c, d\}\rangle$. Then $MI(K_3) = \{M_1, M_2, M_3\}$, where

$$M_1 = \langle\{\neg a\}, \{a\}, \emptyset\rangle, \quad M_2 = \langle\emptyset, \{c\}, \{\neg c\}\rangle, \quad M_3 = \langle\emptyset, \{a, \neg a \vee \neg d\}, \{d\}\rangle.$$

So,

$$Sig_P(M_1 \mid K_3) = \frac{1}{4},$$

$$Sig_P(M_2 \mid K_3) = \frac{1}{4},$$

$$Sig_P(M_3 \mid K_3) = \frac{1}{2},$$

and

$$Sig_P(K_3) = 1.$$

If there exists $M \in MI(K)$ such that $Sig_P(M|K) = 1$, this indicates that all atoms in the language are involved in inconsistency.

Definition 26 (The significance vector $\boldsymbol{Sig_L}$). Let $K = \langle K(1), \ldots, K(n)\rangle$ be a Type-II prioritized knowledge base. The significance vector for K, denoted $Sig_L(K)$, is defined as

$$Sig_L(K) = \sum_{M \in MI(K)} Sig_L(M \mid K),$$

where $Sig_L(M \mid K) = (Sig_P^{(1)}(M \mid K), \ldots, Sig_P^{(n)}(M \mid K))$ for each $M \in MI(K)$.

Especially, we call

$$Sig_L^{(i)}(K) = \sum_{M \in MI(K)} Sig_P^{(i)}(M \mid K)$$

the ith level significance of K, and abbreviate it as $Sig_L^{(i)}(K)$.

The significance vector therefore represents the normalized number of the atoms in a language, at each priority level, involved in inconsistency.

Example 26. Consider $K_3 = \langle\{\neg a, b\}, \{a, c, \neg a \vee \neg d\}, \{\neg c, d\}\rangle$. Then $MI(K_3) = \{M_1, M_2, M_3\}$, where

$$M_1 = \langle\{\neg a\}, \{a\}, \emptyset\rangle, \quad M_2 = \langle\emptyset, \{c\}, \{\neg c\}\rangle, \quad M_3 = \langle\emptyset, \{a, \neg a \vee \neg d\}, \{d\}\rangle.$$

So,

$$Sig_L(M_1 \mid K_3) = \left(\frac{1}{2}, \frac{1}{3}, 0\right),$$

$$Sig_L(M_2 \mid K_3) = \left(0, \frac{1}{3}, \frac{1}{2}\right),$$

$$Sig_L(M_3 \mid K_3) = \left(0, \frac{2}{3}, \frac{1}{2}\right),$$

and

$$Sig_L(K_3) = \left(\frac{1}{2}, \frac{4}{3}, 1\right).$$

In this example, the highest proportion of the language in the second priority level is involved in inconsistency.

Definition 27 (The inconsistency measure $\boldsymbol{Inc_L}$). Let $K = \langle K(1), \ldots, K(n)\rangle$ be a Type-II prioritized knowledge base. The inconsistency measure for K, denoted as $Inc_L(K)$, is defined as

$$Inc_L(K) = \sum_{M \in MI(K)} Inc_L(M \mid K),$$

where $Inc_L(M \mid K) = \frac{Sig_L(M \mid K)}{|M|}$ for each $M \in MI(K)$.

Especially, we call $\frac{Sig_L^{(i)}(M \mid K)}{|M|}$ the ith level inconsistency amount of M, and abbreviate it as $Inc_L^{(i)}(M \mid K)$.

Note that $Inc_L(M \mid K)$ takes into account the size of the minimal inconsistent subset M as well as the priority level of each formula of M within the context of K.

Example 27. Consider $K_3 = \langle \{\neg a, b\}, \{a, c, \neg a \vee \neg d\}, \{\neg c, d\}\rangle$. Then $MI(K_3) = \{M_1, M_2, M_3\}$, where

$$M_1 = \langle \{\neg a\}, \{a\}, \emptyset\rangle, \quad M_2 = \langle \emptyset, \{c\}, \{\neg c\}\rangle, \quad M_3 = \langle \emptyset, \{a, \neg a \vee \neg d\}, \{d\}\rangle.$$

So,

$$Inc_L(M_1 \mid K_3) = \left(\frac{\frac{1}{2}}{2}, \frac{\frac{1}{3}}{2}, 0\right) = \left(\frac{1}{4}, \frac{1}{6}, 0\right),$$

$$Inc_L(M_2 \mid K_3) = \left(0, \frac{\frac{1}{3}}{2}, \frac{\frac{1}{2}}{2}\right) = \left(0, \frac{1}{6}, \frac{1}{4}\right),$$

$$Inc_L(M_3 \mid K_3) = \left(0, \frac{\frac{2}{3}}{3}, \frac{\frac{1}{2}}{3}\right) = \left(0, \frac{2}{9}, \frac{1}{6}\right),$$

and

$$Inc_L(K_3) = \left(\frac{1}{4}, \frac{5}{9}, \frac{5}{12}\right).$$

In this example, the second priority level is most inconsistent because the greatest proportion of language and formulae are affected by inconsistency.

Example 28. Given the Type-II prioritized knowledge base Γ w.r.t. $\Theta = \{a, r, \neg p, s, n, y, b, c\}$ and the Θ-inconsistent subset M_1, the result of applying the Inc_L inconsistency measure is:

$$Inc_L(M_1) = \left(\frac{\frac{6}{10}}{3}, \frac{\frac{2}{9}}{3}, 0\right) = \left(\frac{1}{5}, \frac{2}{27}, 0\right).$$

Next we define the $Blame_L$ measure, which is the $Blame_v$ measure adapted for the new Sig_P, Sig_L and Inc_L values.

Definition 28 (The Blame of each formula for the *k*th level inconsistency *Blame$_L$*). Let $K = \langle K(1), \ldots, K(n)\rangle$ be a Type-II prioritized knowledge base. Then for each $1 \leq k \leq n$, the blame of each formula of K for the kth level inconsistency of K, denoted $Blame_L^{(k)}$, is defined as follows:

$$\forall \alpha \in K, Blame_L^{(k)}(K, \alpha) = \sum_{M \in MI(K)} Blame_L^{(k)}(M \mid K, \alpha).$$

where

$$\mathit{Blame}_L^{(k)}(M \mid K, \alpha) = \begin{cases} \dfrac{\mathit{Sig}_v^{(k)}(\mathit{Opp}(M, \alpha))}{\sum\limits_{\beta \in M} \mathit{Sig}_v^{(k)}(\mathit{Opp}(M, \beta))} \times \mathit{Inc}_L^{(k)}(M \mid K), & \text{if } |M(k)| > 0, \\ 0, & \text{if } |M(k)| = 0, \end{cases}$$

for each $M \in \mathit{MI}(K)$.

Example 29. Consider $K_3 = \langle \{\neg a, b\}, \{a, c, \neg a \vee \neg d\}, \{\neg c, d\} \rangle$. Then $\mathit{MI}(K_3) = \{M_1, M_2, M_3\}$, where

$$M_1 = \langle \{\neg a\}, \{a\}, \emptyset \rangle, \quad M_2 = \langle \emptyset, \{c\}, \{\neg c\} \rangle, \quad M_3 = \langle \emptyset, \{a, \neg a \vee \neg d\}, \{d\} \rangle.$$

So,

$$\begin{aligned} \mathit{Opp}\,(M_1, a) &= \langle \{\neg a\}, \emptyset, \emptyset \rangle, \\ \mathit{Opp}\,(M_1, \neg a) &= \langle \emptyset, \{a\}, \emptyset \rangle, \end{aligned}$$

and

$$\begin{aligned} \mathit{Blame}_L^{(1)}(M_1 \mid K_3, a) &= \frac{|\{\neg a\}|}{|\{\neg a\}| + |\emptyset|} \times \frac{1}{4} = \frac{1}{4}, \\ \mathit{Blame}_L^{(2)}(M_1 \mid K_3, a) &= \frac{|\emptyset|}{|\emptyset| + |\{a\}|} \times \frac{1}{6} = 0, \\ \mathit{Blame}_L^{(3)}(M_1 \mid K_3, a) &= \frac{|\emptyset|}{|\emptyset| + |\emptyset|} \times 0 = 0, \\ \mathit{Blame}_L(M_1 \mid K_3, a) &= \left(\frac{1}{4}, 0, 0 \right). \end{aligned}$$

In this example, the formula a is to blame for the inconsistency M_1 in K_3 at the first priority level only.

Example 30. Given the results of $\mathit{Opp}(M_1, m)$ for $m \in M_1$ and $\mathit{Inc}_L(M_1)$, the result of $\mathit{Blame}_L^{(k)}$ for the formula $a \to n$ in each priority level is:

$$\mathit{Blame}_L^{(1)}(M_1, a \to n) = \frac{|\{n \wedge r \wedge \neg p \wedge s \to q\}|}{\begin{array}{c}|\{n \wedge r \wedge \neg p \wedge s \to q\}| + |\{a \to n\}| \\ + |\{a \to n, n \wedge r \wedge \neg p \wedge s \to q\}|\end{array}} \times \frac{1}{5} = \frac{1}{20},$$

$$\mathit{Blame}_L^{(2)}(M_1, a \to n) = \frac{|\{a \to \neg q\}|}{|\{a \to \neg q\}| + |\{a \to \neg q\}| + |\emptyset|} \times \frac{2}{27} = \frac{1}{27},$$

$$\mathit{Blame}_L^{(3)}(M_1, a \to n) = \frac{|\emptyset|}{|\emptyset| + |\emptyset| + |\emptyset|} \times 0 = 0.$$

Giving an overall $Blame_L$ value for $a \to n$ in M_1:

$$Blame_L(M_1, a \to n) = \left(\frac{1}{20}, \frac{1}{27}, 0\right).$$

A vector value for a formula represents the total blame at each priority level for the inconsistency of the knowledge base.

Definition 29 (The Blame of each Formula for Inconsistency $Blame_L$). Let $K = \langle K(1), \ldots, K(n)\rangle$ be a Type-II prioritized knowledge base. The blame of each formula of K for the inconsistency of K, denoted $Blame_L$, is defined as follows:

$$\forall\, \alpha \in K, Blame_L(K, \alpha) = (Blame_L^{(1)}(K, \alpha), \ldots, Blame_L^{(n)}(K, \alpha)),$$

where $Blame_L^{(k)}(K, \alpha)$ is the blame of α to the kth level inconsistency of K for each $1 \le k \le n$.

Example 31. Consider $K_3 = \langle\{\neg a, b\}, \{a, c, \neg a \vee \neg d\}, \{\neg c, d\}\rangle$. Then $MI(K_3) = \{M_1, M_2, M_3\}$, where

$$M_1 = \langle\{\neg a\}, \{a\}, \emptyset\rangle, \quad M_2 = \langle\emptyset, \{c\}, \{\neg c\}\rangle, \quad M_3 = \langle\emptyset, \{a, \neg a \vee \neg d\}, \{d\}\rangle.$$

So,

$$Blame_L(M_1, a) = \left(\frac{1}{4}, 0, 0\right),$$

$$Blame_L(M_3, a) = \left(0, \frac{1}{18}, \frac{1}{12}\right),$$

and

$$Blame_L(K_3, a) = \left(\frac{1}{4}, \frac{1}{18}, \frac{1}{12}\right).$$

Example 32. Consider $K_4 = \langle\{a \wedge \neg a \wedge b, c \wedge \neg c\}, \{d\}, \{\neg d\}\rangle$. Then $MI(K) = \{M_1, M_2, M_3\}$, where

$$M_1 = \langle\{a \wedge \neg a \wedge b\}, \emptyset, \emptyset\rangle, \quad M_2 = \langle\{c \wedge \neg c\}, \emptyset, \emptyset\rangle, \quad M_3 = \langle\emptyset, \{d\}, \{\neg d\}\rangle.$$

So,

$$Opp(M_1, a \wedge \neg a \wedge b) = \langle\{a \wedge \neg a \wedge b\}, \emptyset, \emptyset\rangle,$$

$$Opp(M_2, c \wedge \neg c) = \langle\{c \wedge \neg c\}, \emptyset, \emptyset\rangle,$$

$$Opp(M_3, d) = \langle\emptyset, \emptyset, \{\neg d\}\rangle,$$

$$Opp(M_3, \neg d) = \langle\emptyset, \{d\}, \emptyset\rangle.$$

Then by applying $Blame_v$

$$Inc_v(M_1) = \left(\frac{1}{1^2}, \frac{0}{1^2}, \frac{0}{1^2}\right) = (1, 0, 0),$$

$$Inc_v(M_2) = \left(\frac{1}{1^2}, \frac{0}{1^2}, \frac{0}{1^2}\right) = (1, 0, 0),$$

$$Inc_v(M_3) = \left(\frac{0}{2^2}, \frac{1}{2^2}, \frac{1}{2^2}\right) = \left(0, \frac{1}{4}, \frac{1}{4}\right),$$

and

$$Blame_v(K, a \wedge \neg a \wedge b) = (1, 0, 0),$$
$$Blame_v(K, c \wedge \neg c) = (1, 0, 0).$$

Alternatively, by applying $Blame_L$

$$Inc_L(M_1) = \left(\frac{\frac{2}{3}}{1}, \frac{0}{1}, \frac{0}{1}\right) = \left(\frac{2}{3}, 0, 0\right),$$

$$Inc_L(M_2) = \left(\frac{\frac{1}{3}}{1}, \frac{0}{1}, \frac{0}{1}\right) = \left(\frac{1}{3}, 0, 0\right),$$

$$Inc_L(M_3) = \left(\frac{0}{2}, \frac{\frac{1}{1}}{2}, \frac{\frac{1}{1}}{2}\right) = \left(0, \frac{1}{2}, \frac{1}{2}\right).$$

and

$$Blame_L(K, a \wedge \neg a \wedge b) = \left(\frac{2}{3}, 0, 0\right),$$

$$Blame_L(K, c \wedge \neg c) = \left(\frac{1}{3}, 0, 0\right).$$

This example illustrates the benefit of the $Blame_L$ measure, over the $Blame_v$ measure, since it maintains the general approach of $Blame_v$ (d and $\neg d$ are considered less to blame than $a \wedge \neg a \wedge b$ and $c \wedge \neg c$). However, it is more discriminating and as such allows more distinction to made between the blame of formulae. In this case, $Blame_v$ is able to identify $a \wedge \neg a \wedge b$ and $c \wedge \neg c$ as most to blame but is unable to make a distinction between the blame of the two. On the other hand, $Blame_L$ is able to identify $a \wedge \neg a \wedge b$ as more to blame since more of the language is involved in inconsistency.

The benefit for inconsistency detection in intrusion detection rules is that it can distinguish between rules based on their complexity as well as the core characteristics examined by the $Blame_v$ measure. When beginning the inconsistency resolution process it provides a more specific starting point.

Given the result of the $Blame_L$ measure on each formula of the knowledge base, an ordering for the degree of blame between formulae can then be defined.

Definition 30. (The relation of less blameful than, $\leq_B$) Let K be a Type-II prioritized knowledge base. A binary relation on K, denoted as $\leq_B$, is defined as follows: $\alpha \leq_B \beta$ if and only if $Blame_L(K, \alpha) \preceq Blame_L(K, \beta)$.

Further, $\alpha <_B \beta$ if $\alpha \preceq_B \beta$ and $\beta _B \alpha$. $\alpha \simeq_B \beta$ if $\alpha \leq_B \beta$ and $\beta \leq_B \alpha$. We say that α is less blameful for the inconsistency in K than β if $\alpha <_B \beta$.

Example 33. Given the Type-II prioritized rule set Γ, the result of applying the Θ-inconsistent subsets of Γ w.r.t. $\Theta = \{a, r, \neg p, s, n, y, b, c\}$ would be:

$$\begin{array}{ll}
\mathit{Blame}_L(\Gamma, a \to n) = \left(\frac{1}{10}, \frac{2}{27}, 0\right) & \mathit{Blame}_L(\Gamma, b \to r \wedge s) = \left(0, 0, \frac{1}{6}\right) \\
\mathit{Blame}_L(\Gamma, c \to p \wedge s) = \left(0, 0, \frac{1}{6}\right) & \mathit{Blame}_L(\Gamma, n \wedge r \wedge \neg p \wedge s \to q) \\
\mathit{Blame}_L(\Gamma, n \wedge y \to \neg d) = \left(0, \frac{1}{6}, 0\right) & \quad = \left(\frac{1}{10}, \frac{2}{27}, 0\right) \\
\mathit{Blame}_L(\Gamma, a \to \neg q) = \left(\frac{1}{10}, 0, \frac{1}{9}\right) & \mathit{Blame}_L(\Gamma, y \to d \wedge s) = \left(\frac{3}{20}, 0, 0\right) \\
\mathit{Blame}_L(\Gamma, q \to \neg a) = \left(\frac{1}{10}, 0, \frac{1}{9}\right) & \mathit{Blame}_L(\Gamma, s \to \neg b \vee \neg c) \\
\mathit{Blame}_L(\Gamma, a \to q) = \left(0, \frac{2}{9}, 0\right) & \quad = \left(\frac{3}{10}, 0, 0\right).
\end{array}$$

This produces an ordering for the distribution of blame between inconsistent formulae of Γ, i.e.

$$\begin{aligned}
(\{b \to r \wedge s\} \simeq_B \{c \to p \wedge s\}) &<_B \{n \wedge y \to \neg d\} <_B \{a \to q\} \\
&<_B (\{a \to \neg q\} \simeq_B \{q \to \neg a\}) <_B (\{a \to n\} \\
&\simeq_B \{n \wedge r \wedge \neg p \wedge s \to q\}) <_B \{y \to d \wedge s\} <_B \{s \to \neg b \wedge \neg c\},
\end{aligned}$$

where $s \to \neg b \wedge \neg c$ is considered as most to blame for the inconsistency of Γ.

In this instance, the inconsistency ordering between the Blame_v and Blame_L inconsistency measures are the same. However, from these results we can see that the inconsistency values are different since the Blame_L measure also takes into account the proportion of the language involved in inconsistency. In this way, the Blame_L measure is more discriminatory as it has a greater degree of variation in inconsistency values. It is also capable of distinguishing between formulae, which would be considered as contributing equally to inconsistency by Blame_v, if the number of atoms in the formula are different.

For IDS rules, we can see that given the worse case (complexity is equal among rules) Blame_L is at least as discriminatory, and at least as useful, as the Blame_v measure. However in any IDS rule set, such as Snort or Snort2Regex, rule complexity is a common variation between rules, e.g. Snort2Regex rules may contain one or more Regex strings to search for attack signatures and alerts do not have a fixed length. Just as it is useful to consider rule priorities when identifying problematic rules, it is also useful to consider this degree of complexity.

7. Conclusion

Effective management of inconsistencies in intrusion detection rules is essential for reliable detection of intrusion attempts, especially with a large set of rules such as Snort. The ability to not only identify these inconsistencies, but to distribute the degree of blame among contributing rules is important in the process of resolving inconsistencies. Also, the reality of varying degrees of importance of information in real world applications means that considering prioritized rules will produce more representative result. Snort, for example, is open source and so the rules originate from multiple sources — if we are able to say that some sources are more important than others, then we can also say that some inconsistencies are more important than others.

In this preliminary study, we investigated how Snort2Regex rules, based on Snort, can be translated into logical formulae. We began by identifying elementary units in the rules because in the current Snort2Regex system, each rule is looked at as a whole, so it is difficult to correlate them. We then extended this rule set to include some expert knowledge to more explicitly correlate rules. We then applied a number of approaches for measuring inconsistency in this extended rule set. From this work we were able to define a new measure of inconsistency for prioritized knowledge which incorporates the proportion of the language involved in inconsistency. In summary of our results:

- We have demonstrated a method for analyzing a set of Snort2Regex IDS rules, converting them to a propositional logic format and introducing expert knowledge to further correlate rules. This is essential to begin the process of inconsistency analysis;
- Measuring inconsistencies in intrusion detection rules is potentially very useful for this domain since ensuring accurate and reliable intrusion detections is dependent on a consistent detection system;
- The inconsistency measures presented here are effective for quantifying the inconsistency of formula in an extended, knowledge-based, system. Using these values they can determine an inconsistency ordering for the degree of blame associated with each formula in order to begin resolving inconsistency;
- Our proposed measure, denoted $Blame_L$, is able to consider the proportion of the language involved in inconsistency at each priority level. This is useful in many cases since it is more discriminatory than the $Blame_v$ measure and can be applied to prioritized knowledge, unlike the Scoring function and SIVs;
- The major problem identified by this paper is that current rules used for intrusion detection are vulnerable to inconsistencies, but because they do not incorporate any domain specific knowledge, it is difficult to correlate rules in order to identify these inconsistencies.
- Our study has revealed weaknesses in the current Snort2Regex rule set, in terms of redundancy and the correlation of rules, and we have demonstrated approaches

to addressing these issues. This study provides a promising way forward for constructing and maintaining a scalable set of consistent rules over time.

Further work will aim at using practical methods to apply these measures of inconsistency to identify inconsistent Snort2Regex rules. We then aim to propose strategies for removing minimal numbers of Snort2Regex rules in order to resolve inconsistency. Experimentally, we aim to develop a tool to automatically identify inconsistent rules in order to measure the impact of inconsistency on false alarm and mis-detection rates. Following this, automatic resolution methods can be applied and compared with the aid of experimental results.

Acknowledgements

We would like to thank Philippe Besnard for his very helpful comments, and thank Antonio Munoz and Sakir Sezer for the Snort2Regex rule set.

References

[1] K. McAreavey, W. Liu and P. Miller, Measuring inconsistency in network intrusion rules, in *Proceedings of the 22nd International Workshop on Database and Expert Systems Applications (DEXA11) (the 1st International Workshop on Data, Logic and Inconsistency (DALI '11 at DEXA11)*, IEEE Computer Society, 2011, pp. 339−344.

[2] B. Kim, S. Yoon and J. Oh, Multihash based pattern matching mechanism for high-performance intrusion detection, *International Journal of Computers* **3**(1) (2009) 45−124.

[3] R. Agarwal and M. Joshi, PNrule: A new framework for learning classifier models in data mining, *Technical Report TR 00-015*, 2000.

[4] S. Beg, U. Naru, M. Ashraf and S. Mohsin, Feasibility of intrusion detection system with high performance computing: A survey, *International Journal for Advances in Computer Science* **1**(1) (2010) 26−35.

[5] A. Munoz, S. Sezer, D. Burns and G. Douglas, An approach for unifying rule based deep packet inspection, *IEEE ICC*, 2011, pp. 5428−5432.

[6] A. Hunter and S. Konieczny, Approaches to measuring inconsistent information, in *Inconsistency Tolerance* LNCS 3300. Springer, 2004, pp. 189−234.

[7] A. Hunter, Logical comparison of inconsistent perspectives using scoring functions, *Knowledge and Information Systems* **6**(5) (2003) 528−543.

[8] A. Hunter and S. Konieczny, Shapley inconsistency values, in *Proceedings of the 10th International Conference on Knowledge Representation (KR-06)*, AAAI Press, 2006, pp. 249−259.

[9] K. Mu, Z. Jin, R. Lu and W. Liu, Measuring inconsistency in requirements specifications, *Proceedings of the Eighth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU05).* LNAI 3571, Springer, 2005, pp. 440−451.

[10] K. Mu, W. Liu and Z. Jin, Measuring the blame of each formula for inconsistent prioritized knowledge bases, *Journal of Logic and Computation* (doi: 10.1093/logcom/exr002), 2011.

[11] B. Mukherjee, L. Heberlein and K. Levitt, Network intrusion detection, *IEEE Network* **8**(3) (1994) 26−41.

[12] M. Roesch, Snort-lightweight intrusion detection for networks, in *Proceedings of 13th USENIX Systems Administration Conference (LISA 99)*, 1999, pp. 229−238.

[13] G. C. Tjhai, M. Papadaki, S. M. Furnell and N. L. Clarke, Investigating the problem of IDS false alarms: An experimental study using snort, in *Proceedings of the IFIP TC 11 23rd International Information Security Conference*, IFIP International Federation for Information Processing, Vol. 278, 2008, pp. 253−267.
[14] G. C. Tjhai, M. Papadaki, S. M. Furnell and N. L. Clarke, The problem of false alarms: Evaluation with snort and DARPA 1999 dataset, *5th International Conference: Trust, Privacy and Security in Digital Business* (*TrustBus 2008*), Lecture Notes in Computer Science, Vol. 5185/2008, 2008, pp. 139−150.
[15] G. H. Kayacik and A. N. Zincir-Heywood, Using intrusion detection systems with a firewall: Evaluation on DARPA 99 dataset, *NIMS Technical Report #062003*, 2003.
[16] G. De Giacomo and M. Lenzerini, Tbox and Abox reasoning in expressive description logics, in *Proc. of KR-96*, 316−327, Los Altos, 1996.
[17] A. Borgida, T. J. Walsh and H. Hirsh, Towards measuring similarity in description logics, in *International Workshop on Description Logics (DL2005).* Edinburgh, Scotland, 2005.
[18] J. G. Alfaro, N. Boulahia-Cuppens and F. Cuppens, Complete analysis of configuration rules to guarantee reliable network security policies, in *International Journal of Information Security* **7**(2) (2008) 103−122.
[19] E. Al-Shaer, H. Hamed, R. Boutaba and M. Hasan, Conflict classification and analysis of distributed firewall policies. *IEEE Journal on Selected Areas in Communications (JSAC)* **23**(10) (2005) 2069−2084.
[20] E. Al-Shaer and H. Hamed, Discovery of policy anomalies in distributed firewalls, in *Proc. IEEE Infocomm*, Hong Kong, 2004, pp. 2605−2616
[21] L. Yuan, J. Mai, Z. Su, H. Chen, C. N. Chuah and P. Mohapatra, FIREMAN: A toolkit for FIREwall modeling and ANalysis, in *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, 2006, pp. 199−213.
[22] A. Ligeza, *Logical Foundations for Rule-Based Systems* (Springer-Verlag, Berlin/ Heidelberg, 2006).
[23] G. J. Nalepa and A. Ligęza, Designing reliable web security systems using rule-based systems approach, in *Advances in Web Intelligence: First International Atlantic Web Intelligence Conference* (*AWIC 2003*), 2003, pp. 124−133.
[24] L. Bauer, S. Garriss and M. K. Reiter, Detecting and resolving policy misconfigurations in access-control systems, in *Proc. of the 13th ACM Symposium on Access Control Models and Technologies*, 2008, pp. 185−194.
[25] V. Capretta, B. Stepien, A. Felty and S. Matwin, Formal correctness of conflict detection for firewalls, in *Proc. of the 2007 ACM workshop on Formal Methods in Security Engineering* (*FMSE 1907*), 2007, pp. 22−30.
[26] M. G. Gouda and X-Y. A. Liu, Firewall design: Consistency, completeness and compactness, in *Proc. ICDCS* 24, 2004.
[27] H. Haded and E. Al-Shaer, Taxonomy of conflicts in network security policies, *IEEE Communications Magazine* **44**(3) (2006) 134−141.
[28] A. Hari, S. Suri and G. Parulkar, Detecting and resolving packet filter conflicts, in *Proceedings of IEEE Infocom*, 2000, pp. 1203−1212.
[29] R. Ceballos and R. M. Gasca, Fast algorithms for local inconsistency detection in firewall ACL updates, in *Proc. 1st International Workshop on Dependability and Security in Complex and Critical Information Systems* (*DEPEND*), 2008.
[30] H. Decker, Historical and computational aspects of paraconsistency in view of the logic foundation of databases, in *Proceedings of Semantics in Databases, 1981.* LNCS 2582 (Springer, 2003), pp. 63−81.

[31] M.-O. Cordier and S. Loiseau, Validation of first-order rule-based systems, *Computational Intelligence* **12** (1996) 523−540.

[32] K. Mu, Z. Jin, R. Lu and Y. Peng, Handling non-canonical software requirements based on Annotated Predicate Calculus, *Knowl. Inf. Syst.* **11**(1) (2007) 85−104.

[33] K. A. Hargreaves and K. Berry, Regex. Free Software Foundation, 675 Mass Ave., Cambridge, MA 02139.

[34] E. Grégoire, B. Mazure and C. Piette, Using local search to find MSSes and MUSes, *European Journal of Operational Research* **199**(3) (2009) 640−646.

[35] A. Hunter and S. Konieczny, On the measure of conflicts: Shapley inconsistency values, *Artificial Intelligence* **174**(14) (2010) 1007−1026.