Human Interface for Cyber Security Anomaly Detection Systems

Denis Todd Vollmer†, Milos Manic ‡, *Members, IEEE*
†Idaho National Laboratory, ‡University of Idaho

Abstract — **Low-level network traffic information is often times beyond the understanding of common system operators (byte counts, port numbers, packet data, etc.). However, anomaly based Intrusion Detection Systems (IDS) often provide such low-level, difficult to comprehend information. This paper details a Human Interface for Security Awareness (HISA) algorithm for interpreting cyber incident information to human operators from anomaly based intrusion detections systems. A similarity algorithm mapping anomaly results to signature based intrusion system rules is developed. Categorizations of attacks found in rules created for the Snort intrusion system were used as a basis of information to present to the user. A proof of concept system was developed using Perl native functions and custom modules. Testing with generated ICMP packets resulted in an identification accuracy of 60% proving the efficacy of the presented HISA algorithm.**

Keywords — **site security monitoring, command and control systems.**

I. Introduction

Comprehensive state awareness of safety and security is a preeminent concern for critical infrastructures. Modern implementations of these infrastructures rely heavily on networked communication systems. Network intrusion attacks can lead to high financial costs and the endangerment of public safety. Therefore it is imperative that control operators receive relevant and comprehensible cyber health information. This paper provides a solution for presenting cyber incident information from anomaly based Intrusion Detection Systems (IDS) to human operators. The stored knowledge inherent in rule based intrusion systems is used to classify newly identified attack vectors. A similarity algorithm mapping novel to known attack classifications is presented.

The increased threat of cyber attacks is well documented and has been acknowledged by many governmental, commercial and academic entities world wide [1],[2]. Computer based systems used within many critical infrastructures to monitor and control physical functions are not immune to this threat and may potentially be more vulnerable than common information technology systems [3]. Despite large expenditures of effort, systems will continue to be penetrated or threatened by human failings. These issues provide incentive to develop capabilities that enhance operator's awareness and understanding of system security measures.

There are two primary types of IDS's, signature and anomaly based. The primary purpose of both is to detect and possibly react to illicit network intrusion activity. The signature based systems provide monitoring and alert services based on static rule sets. Static rule sets perform well on known signatures but rely upon human experts to recognize an issue, perform analysis and develop a detection rule. As is shown in virus protection products, small variances in behavior can bypass static rules. This necessitates constant and expensive updates by the vendors.

An anomaly IDS is based on recognizing deviations from a learned model of normal behavior [4]. A representative model is built primarily on historical data. The features of future intrusions are not assumed apriori and anomaly decisions are based on profiling current activity, in contrast to the stored normal behavior [5]. Such a system is capable of detecting previously unknown and dynamically changing intrusion instances. This is most effective when the intrusions are distinctively different from the learned model of acceptable behavior.

According to [6], IDS implementations primarily make use of the previously mentioned static rule configuration. Most of these IDS systems are generic in application and have rules designed for more typical information technology infrastructures. Control systems networks have similar communication infrastructures, but different behavior patterns. Research detailed in this paper may help drive the control industry to make use of both types of systems working cooperatively together. These systems used in combination may be robust enough to cover both novel and known intrusions.

Even though current IDS systems implement a different approach to attack recognition, they have a common base in regards to input and intent. In turn, this can provide a common basis for notification output. As is shown in this paper, the power of anomaly detection can be combined with the inherent knowledge representation of rule based systems to provide attack information to an operator. The information is presented in terms of similarity to well defined attack classes, such as denial of service, and references to known vulnerabilities.

The rest of the paper is organized as follows. Related work with intrusion systems is presented in section 2.

· Work supported by the U.S. Department of Energy under DOE Idaho Operations Office Contract DE-AC07-05ID14517, performed as part of the Instrumentation, Control, and Intelligent Systems Distinctive Signature (ICIS) of Idaho National Laboratory. Authors email: denis.vollmer@inl.gov, misko@ieee.org

Section 3 gives a more detailed description of the problems associated with rule and anomaly based systems. Section 4 introduces the Human Interface for Security Awareness (HISA) algorithm. Section 5 presents the achieved experimental results followed by the conclusion given in section 6.

II. Related Work

Rule based IDS's are being used to provide an important layer of security for computer systems and networks. In [7] the authors state that an IDS's responsibility is to detect suspicious or unacceptable system and network activity and to alert a systems administrator to this activity. Their intent was to identify a way in which SNORT® could be improved by generalizing rules to identify novel attacks. The conditions and parameters of a sample set of rules were modified to relax the conditions and parameters. The approach was proven using a set of network capture data obtained from the Massachusetts Institute of Technology. Previously undetected variants of attacks were identified.

False alert reduction is important in rule based systems. Because the definition of a rule can be general, it is increasingly likely that false positives will occur given a large volume of network traffic. Long et al. made use of Snort alerts in XML format to form clusters [8]. The clusters were formed using an XML distance measure. This proved effective in discriminating between normal sessions that raised false alerts and those that contained real attack information.

Many IDS's use rule based signature solutions. Rule formats used by most of these systems are not standard. This leads to duplication of effort making definitions for the same attack in multiple expressions. In [9] the author proposed and implemented an automated rule conversion system. The results showed that Snort rules can be generalized for use in other systems. Such a system could be used by the HISA algorithm to broaden the scope of rules.

Anomaly based IDS's usually report events via one of three different mechanisms. The first is simply if an anomaly event was detected or not [10]. This can be based on passing some threshold and may contain a confidence factor. Second, when labeled data is available, a supervised network can be trained to distinguish between different input vectors [11]. The output then can present the appropriate label if it closely matches any of the known categories. Third, a new set of rules for identifying attacks can be created [12]. These rules use a derived set of attributes from the data sets that are identified as anomalous. This requires analyzing the known attack attributes and choosing those that are general enough to identify a group of attack types.

III. Problem Description

This paper focuses on the specific manner in which the results of anomaly based systems are presented to an operator by making use of information from a signature based IDS. Rule based systems can identify specific intrusions and report human friendly messages. The rules themselves are created by people allowing for the opportunity to embed relevant information in the rule definition such as an attack category or well known vulnerability reference. These rules are hence easy to comprehend as they present an abstract view of the information that is typically beyond the knowledge of the common control system operator (byte counts, IP addresses, port numbers, network protocols and packet data).

Network traffic that is outside the norm can be identified by anomaly based systems. This feature provides the ability to detect new attacks. Systems such as these can report on the peculiarities that caused identification of traffic determined to be outside the norm. They do not have the information or generally the capability to express the attack alert in human terms. In contrast, rule based systems have a built in categorization mechanism. The human creation of rules provides the opportunity for general categorization of attack features. In addition to the rules primary purpose as an attack recognition repository, they are a valuable source of community knowledge. They provide a useful historical database of information and characteristics. This database of attack features can be used for similarity comparisons to help describe previously unseen attacks.

Anomaly based systems can identify new attacks and alert users to the situation. However, there have been few proposed mechanisms for delivering alert information in a meaningful way to system operators. Specifically, as illustrated in section 2, anomaly systems can report whether or not traffic is anomalous or if it closely matches a known signature. The signature match requires either training a system with tagged data or generating rules based on known attack characteristics. The later method is similar to our proposal. However the HISA algorithm makes use of community defined rule sets, instead of sample network data, which is unique.

A. *Snort Rule Analysis*

Snort is an open source IDS created by Martin Roesch. It is capable of performing protocol analysis, content searching/matching and many other abilities including using rule sets. Several rule sets are available for use including those officially approved by the Sourcefire Vulnerability Research Team (VRT) and those contributed by other communities [13], [14]. Snort supports a simple rule language that matches against network packets, generating alerts or log messages. Rules are broken into two logical sections: rule headers and rule options. Because of these factors it was chosen as a starting point for our algorithm implementation.

Rule headers contain necessary protocol fields that every rule must have and rule options contain a list of optional information used to refine a match. The rule field format and an example rule is as follows:

<action> <protocol> <source IP> <source port> <direction> <destination IP> <destination port> (<rule options>)

Alert tcp any any -> 192.168.1.0/24 any (content:"ELHO")

The rule action tells Snort what to do when a match occurs. A common action is to log information to an alert file. The protocol field specifies one of four possible values: TCP, UDP, ICMP and IP. Each value has options specific to the protocol available for use in the option section. The source IP address section can contain the keyword any, a single IP address or a CIDR (Classless Inter-Domain Routing) block. CIDR blocks allow for specifying ranges of IP addresses. Port numbers may be specified as a single static port, a range or use the keyword any.

There are two direction operators. One specifies that the source and destination sections of a rule must match the appropriate items from a packet. The bidirectional operator indicates that the source and destination sections can match either portion of a packet. This allows for tracking two way conversations (i.e. FTP sessions).

Rule options provide further refinement of matching parameters and tie the rule to a rule identification system. There are four major categories of rule options: general, payload, non-payload and post-detection. General options provide information about the rule such as reference information, rule identification and specific log messages. Payload options examine data contained in the packet data such as content matching expressions. Non-payload options provide matching specifications against packet header data outside of ports and IP addresses. Options include fragment offsets, time-to-live values and specific IP options.

An important option keyword is classtype. This keyword is used to mark a rule as belonging to a specific attack class. The attack classes are predefined in a configuration file. This makes for a consistent description and format for the attack information. Although it is an optional field, all rules provided for general consumption include a classification. These classifications can be provided to operators as they convey meaningful information without requiring a lot of in-depth technical knowledge about cyber incident specifics. The default classifications are show in Table I. The classtype names are descriptive but further details can be found in [15].

B. *Snort Rule Processing Description*

Snort builds a tree structure used to compare against packet features. Each mandatory field in a Snort rule is stored in a Rule Tree Node (RTN). An OTN (Optional Tree Node) is associated with an RTN and used to store optional rule fields. If multiple rules have the same RTN fields they are only represented by a single node. This optimization feature allows for removal of multiple rules from consideration once a negative match occurs. This is a detrimental aspect to the proposed HISA algorithm and it is important to recognize the impact on the processing time.

Table I: Snort Classifications.

Classtype	***Classtype***
attempted-admin	rpc-portmap-decode
attempted-user	successful-dos
kickass-porn	successful-recon-largescale
policy-violation	successful-recon-limited
shellcode-detect	suspicious-filename-detect
successful-admin	suspicious-login
successful-user	system-call-detect
trojan-activity	unusual-client-connection
unsuccessful-user	web-application-activity
web-application-attack	icmp-event
attempted-DOS	misc-activity
attempted-recon	network-scan
bad-unknown	not-suspicious
default-login-attempt	protocol-command-decode
denial-of-service	string-detect
misc-attack	unknown
non-standard-protocol	tcp-connection

The intrusion rule can be seen as a Boolean truth statement. In order for Snort to identify a match, a logical *and* of all positive field matches is necessary. Upon discovery of a false condition in the *and* evaluation, further processing of that rule is halted. The set of all rules would then be equivalent to a logical *or*.

Short circuit of rule evaluation prevents using a modified Snort source as a base. As the Snort engine processes a rule section that proves false for a rule that rule is no longer considered for a possible match. This is a logical performance enhancement that speeds the execution of the rule engine. However this prohibits the intent of the proposed algorithm.

In the HISA processing, all parts of a rule will be evaluated to see if it matches the packet data regardless of a previous rule section match. A value of 1 for each header field indicates a match, 0 otherwise. The header values are a necessary match condition but are not always a strong indicator of similarity. For instance, a TCP packet on port 80 is a very common packet as this is traditionally where the http protocol takes place. However, a large amount of attacks of varying types can take place via http. A stronger indicator of an attack type is in the packet header and payload details (i.e. Snort rule option fields). Consequently, these details are required for a stronger response upon match. The more detailed and specific a rule proves to be a more relevant match indicator it should prove to be.

Three sources for acquiring rules were used. Sourcefire VRT certified rules and community rules are available online from the Snort repository [13]. The third set was obtained from emerging threats and is available online at [15]. All of these sets combined to define 16,181 rules covering 31 of the 34 class categories. Table II describes the protocol and number of related rule sets available in these sets.

TABLE II: SNORT CLASSIFICATIONS.

Protocol (number)	*Protocol (number)*
TCP (12,325)	UDP (890)
IP (2,808)	ICMP (158)

IV. HISA ALGORITHM DESCRIPTION

The goal of HISA is to present information characterizing an unknown attack to a human user. An approach utilizing prior rule definitions to find a close match is described. The computational process necessary to do so is described next in a pseudo coding style. Each major functional area is described in detail in the sections following.

```
Initialization:
find rules and load them into a rule structure.
open network packet file.

Check for matches:
Loop through packets
Decode packet;
Loop through rules
    initialize matches to 0;
    check for match on IP address;
    check for match  on protocol;
    switch on protocol type
        check matches on protocol specific;
    store match information;
end rule loop
end packet loop

Process results:
Loop through packet records
Loop through rule match record
    sum match results;
end rule match loop
identify largest match count rule(s);
end packet loop
```

The network packet data is identified by an outside anomaly detection routine and is passed to the HISA algorithm for possible identification. The identification process consists of traversing all rule sets looking for those rules that match as closely as possible by matching each part of a rule.

A. Initialization

The rules are defined in several different file sets. These files are opened and loaded into memory. The Snort specific rule format is well defined in [16] and is the only rule format currently supported. Each rule is parsed into its component parts and stored in a record for fast search and retrieval. The record itself is a hash or associative array of rule part names to textual values. These rule parsing and record population routines were implemented using a Perl module called Net::Snort::Parse [17].

The record matches are stored in the records as text by default. An additional field is included to handle IP addresses which can be stored using CIDR notation. The CIDR record field maintains an object reference to handle these types of checks. The Perl module Net::CIDR available on Cpan has functions for dealing with IP address range checks.

Special handling of port and IP sections is required. Snort rules allow for variables or the keyword ‘any’ in these sections. The variables can be defined once with specific values that are replaced in the rules when encountered. When processing rules, these variables need to be accounted for, in both destination and source, by replacing them with legal values. Tables III and IV illustrate the mapping from variable name to replacement value that is used.

TABLE III: PORT VARIABLE DEFINITIONS.

Variable	*Definition*
$HTTP_PORTS	80
$SHELLCODE_PORTS	!80
$ORACLE_PORTS	1521
$SSH_PORTS	22

TABLE IV: IP VARIABLE DEFINITIONS.

Variable	*Definition*
any	0.0.0.0/0
$AIM_SERVERS	!0.0.0.0/0
$*_NET	0.0.0.0/0

The anomaly data is stored in pcap format which is an industry standard file format for captured network information. The packet data consists of anomalous packet data only. The system assumes that each packet is of interest and attempts to match each packet to the stored rule information. The packet can be seen as a feature vector $\vec{v}$ where each feature v is a unique data point in the vector such that $\vec{v} = v_0 ... v_i$. The set **S** contains all feature vectors $\vec{v}$ in a collection of network packet data.

B. Match Check

The match check is a O(n^2) portion of the algorithm as is shown in (1) where *n* is the number of packets and rules (*n* approaches infinity). The match function maps the attributes of rule *(r)* to those of packets *(p)* . The function *T(n)* is a constant time operation with a fixed set of comparison operations.

$$T(n) = \sum_{p=1}^{n} \sum_{r=1}^{n} match(r, p) \qquad (1)$$

Each packet in the stored file is compared against the specifics of each rule definition. A match record stores an integer value for each match item defined. If a match occurs, a 1 is stored 0 otherwise. This is a simple method to indicate a match. The value is stored as an integer despite the current storing of a binary value. A future improvement may be to weigh specific matches differently with a unique value.

The current system matches on the following items: source IP address, source port, destination IP address, destination port, protocol, payload content, ICMP id, ICMP sequence, ICMP type, ICMP code. These are a subset of the possible match fields with a focus on ICMP values as is explained in the results section.

C. Process Results

The match check records are processed individually. Each match item value is summed and tracked. The resulting sums are sorted and grouped from largest value to smallest. The top matching item is then considered to be the closest match. The class type, as previously defined in Table 1, and message ID of the winning sum is presented as the closest match to the user. When the largest sum value has multiple rule matches, a weighted class list is presented.

The weighted class list shows the percentage of each class that matched. For instance, if five rules match and three of them are of the denial-of-service class and two are policy-violation the resulting values of 0.60 and 0.40 respectively are presented. The intent is to provide the user with high level information about the nature of the attack.

V. Experimental Results

ICMP rules and characteristics were the primary focus of the test data set creation to show a proof of concept. ICMP packets have fewer options in both packet details and rule match items. In addition, as can be seen in Table V, the number and class type representations are reduced. This simplification could have had a negative impact on the results. With fewer data points to work from, the similarity measure may not have produced meaningful results. However this does not appear to have been the case as is shown in this section.

Table V: ICMP Rule Classifications.

ClassType	*Count (total 145)*
denial-of-service	1
misc-activity	103
bad-unknown	3
attempted-recon	11
attempted-user	1
trojan-activity	9
attempted-dos	16
network-scan	1

As a base case, to prove program correctness, the system was exercised with test packets against all ICMP rules. These test packets were crafted to cause specific rules from different classes to exactly match the rule parameters. It was surmised that if the system could not match known vectors than it might be fundamentally flawed. The system correctly identified 100% of test packets with the appropriate static rule definition.

Nemesis is a network packet crafting and injection tool [18]. Implemented as a command line tool, it is well suited for reproducing test scenarios. Nemesis can create and inject ARP, DNS, ETHERNET, ICMP, IGMP, IP, OSPF, RIP, TCP and UDP packets. It was used to produce the ICMP test packets. An example command line used to create a packet is shown below.

```
> nemesis -i 8 -s 0 -d 666 -d lo
```

This command will create an ICMP packet with a type of 8 sequence number of 0 and ICMP-ID within the header of 666. Subsequently the packet will be placed on the lo interface of the machine.

The packets created using the nemesis tool were captured and stored as a pcap data file. Five different test packets were created to trigger five different Snort rules. Each rule was chosen for its membership in a different class type. The packet nemesis command line specifics and class types are presented in Table VI.

Table VI: ICMP Packet Details.

Packet Details	*Class Type*
-i 0 -s 0 -e 667	attempted-dos
-i 8 -s 0 -e 666	attempted-recon
-i 5 -c 0	bad-unknown
-i 3 -c 2	attempted-user
-i 8 -s 14611 -c 123	misc-activity

The 145 ICMP rules had representation from all three sources mentioned in section 3. After proving the correctness via the previously mentioned base case, the matching rule definitions were removed from operation. The intent was to run the test packets through the system without the matching rules. This simulates the availability of unknown attack vectors and tests the systems ability to identify similarities with known attacks. It also provides a known set of results to compare the categorization results against. The results are shown in Table VII.

Table VII: ICMP Rule Classifications.

Correct ClassType	*Identified ClassType*	*% Match*
attempted-dos	attempted-dos(3)	100%

attempted-recon	attempted-recon(4)
	network-scan(1)
		80%

bad-unknown	bad-unknown(2)
	attempted-recon(3)
	misc-activity (28)
		6%

trojan-activity	attempted-user(1)	0%
misc-activity	misc-activity(1)	100%

The first column in the table is the class type that the test packet originally matched before removal of the associated rule. The second column shows the system classification output. The class type is followed by the number of rule matches that had the highest similarity score. As is illustrated by the number in parenthesis, it was possible to have multiple hits with the same score. In the final column a percentage value is recorded. This value represents a match percentage of the output results. For example, the attempted-dos test had three results with the same class type. All of the class types matched the correct value resulting in a 100% score. The attempted-recon test had four correct hits and one miss. This resulted in a score of 80%.

Overall, if a match score of 80% or greater is considered as successful, the system correctly identified three of the five (60%) of the test cases. Veracity and detail of rules makes a difference. The quality of the answer is only as good as the basis from which it is drawn. In this case, the Snort rule set and matching algorithm are the basis features. It can be observed that 71% of the ICMP rules are in the misc-activity category. There may be an opportunity for refining these rules into a more useful category. However even with this limitation the results are positive and provide useful information.

VI. Conclusion

An algorithm for presenting anomaly based intrusion detection alerts based on similarity to static rules was presented. Rules developed for the Snort IDS and their default classifications were used. A similarity algorithm was developed that utilized a simple match summation process. This process utilized native Perl functionality and custom modules. For each match found in a rule, a positive value was noted. The rule(s) with the max sum was used to identify the attack class membership. This result was subsequently presented to the user as an indicator of system cyber security status. An identification rate of 60% demonstrated the effectiveness of the proposed algorithm on test ICMP data.

References

[1] B. Gellman, "Cyber-Attacks by Al Qaeda Feared." Washington Post. Online: ,http://www.washingtonpost.com/ac2/wp-dyn/A50765-2002, Jun26

[2] J. Meserve, “Sources: Staged cyber attack reveals vulnerability in power grid.” CNN. Online http://www.cnn.com/2007/US/09/26/power.at.risk/.

[3] C. Taylor, P Oman, A. Krings, “Assessing Power Substation Network Security and Survivability: A Work in Progress Report”, Proceedings of the International Conference on Security and Management (SAM '03), Las Vegas, pp 23-26, 2003

[4] A. K. Gosh, A. Schwartzbard, M. Schatz, “Learning Program Behavior Profiles for Intrusion Detection”, In Proceedings of the 1st USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, CA, April 1999, pp. 51-62.

[5] O. Linda, T. Vollmer, M. Manic, “Neural Network Based Intrusion Detection For Critical Infrastructures”, *IJCNN09*, Int. Joint INNS-IEEE Conf. on Neural Networks, Atlanta, Georgia, June 14-19, 2009.

[6] M. Analoui, B. Bidgoli, M. Rezvani, “Hierarchical Two-Tier Ensemble learning: A new Paradigm for Network Intrusion Detection.”

[7] U. Aickelin, J. Twycross and T. Hesketh-Roberts ‘Rule generalisation in intrusion detection systems using SNORT’, Int. J. Electronic Security and Digital Forensics, Vol. 1, No. 1, pp.101–116, 2007

[8] J. long, D. Schwartz, S. Stoecklin, ‘Distinguishing False from True Alerts in Snort by Data Mining Patterns of Alerts’, SPIE Defense and Security Symposium 2006, Orlando, FL, USA, 17 April 2006

[9] S. T. Eckmann, “Translating Snort rules to STATL scenarios”, 2001

[10] S. Zanero, S. Savaresi, ‘Unsupervised learning techniques for an intrusion detection system’, SAC 04 March 14-17, Nicosia, Cyprus

[11] L. Khan, M. Awad, B Thuraisingham, ‘A new intrusion detection system using support vector machines and hierarchical clustering’, The VLDB Journal, Vol. 16 pp. 507-521, 2007.

[12] Y. Huang, L. Wenke, ‘A Cooperative Intrusion Detection System for Ad Hoc Networks’, in *Proc. 1st ACM workshop on security of ad hoc and sensor networks*, Fairfax, Virginia, 2003 pp.135-147

[13] M. Roesch. Writing Snort Rules: How To write Snort rules and keep your sanity. http://www.snort.org.

[14] http://www.emergingthreats.net

[15] M. Roesch. Snort – lightweight intrusion detection for networks. In *Proceedings of USENIX LISA ′99*, November 1999

[16] Sourcefire, Inc. Snort Users Manual 2.8.3, September 15, 2008, pp 102-103

[17] B. Caswell, Perl-Net-Snort-Parser online: http://projects.honeynet.org

[18] J. Nathan, Nemisis online: http://nemesis.sourceforge.net