Analyzing Attack Strategies Against Rule-Based Intrusion Detection Systems

Pooja Parameshwarappa
University of Maryland, Baltimore County
Baltimore, MD
poojap1@umbc.edu

Zhiyuan Chen
University of Maryland, Baltimore County
Baltimore, MD
zhchen@umbc.edu

Aryya Gangopadhyay
University of Maryland, Baltimore County
Baltimore, MD
gangopad@umbc.edu

ABSTRACT

Intrusion Detection Systems (IDS) have been widely used to detect cyber attacks in Cyber-Physical Systems (CPS). However, attackers can often adapt their attacking strategies to evade detection. Many commercial IDS are rule-based systems. This paper analyzes the possible attacking strategies against a widely used rule-based IDS, Snort, using hyper graph model and clustering. We present initial results and discuss some techniques to prevent such attacks.

CCS CONCEPTS

• **Security and privacy** → **Intrusion detection systems**; • **Computing methodologies** → *Machine learning*;

KEYWORDS

Intrusion Detection Systems, Evasion, Indiscriminate Attack, Targeted Attack

ACM Reference Format:
Pooja Parameshwarappa, Zhiyuan Chen, and Aryya Gangopadhyay. 2018. Analyzing Attack Strategies Against Rule-Based Intrusion Detection Systems. In *Proceedings of Workshop on Analytics for Security in Cyber Physical Systems (ASCPS'18).* ASCPS, Varanasi, India, 4 pages.

1 INTRODUCTION

CPS are an integration of Information Technology (IT) systems and physical components [5, 12]. There are numerous vulnerabilities associated with CPS [5]. IDS [9] have been widely used to detect attacks in the cyber network of the CPS [4, 20]. IDS are used to monitor a computer system or a network of systems and detect malicious activities. IDS can be implemented as a rule-based system or a machine learning based system. Rule-based systems are composed of a set of signatures, the network traffic is parsed and matched against these signatures, and actions are taken when there is a match [7]. Machine learning based IDS are of two types [16], one based on misuse detection and the other based on anomaly detection. In misuse detection, machine learning algorithms are trained using labeled data to distinguish between benign and malicious traffic. In anomaly detection, normal network traffic is modeled and network packets that deviate from such a model are considered to be malicious and necessary actions are taken.

The complexities of IDS are constantly increasing, and so is the sophistication of the adversaries trying to evade IDS. There has been research on vulnerabilities of machine learning based IDS [11], evasions and attacks corresponding to these vulnerabilities [2], and measures to make systems robust in order to counter such attacks [10, 15]. There are also more recent studies on adversarial attacks against deep learning models [6, 13]. However, because of their commercial success, most IDS are rule-based systems [7]. One additional benefit of rule-based IDS is that they can often indicate the exact vulnerability being exploited and thus it is easier to come up with mitigation solutions.

Our study focuses on identifying potential attack strategies against rule-based IDS. We discuss two types of attacks [2]: *targeted* attack, in which an adversary's goal is to evade detection for a specific attack, and *indiscriminate* attack, in which the adversary's goal is to evade detection for any attack. These attacks are explained in detail in the context of an IDS in the subsequent sections. Our contribution can be summarized as follows:

(1) This paper proposed a method based on minimum-edge-cover to analyze the vulnerability of a rule-based IDS for indiscriminate attacks.
(2) This paper proposed a rule-clustering method to help prevent targeted attacks against a rule-based IDS.
(3) This paper presented some preliminary experimental results.

This paper is organized as follows. Section 2 explains the Minimum-edge-cover method and the Rule-clustering method. It is followed by experiments and initial results in Section 3. Section 4 presents the future work and conclusion.

2 METHODS

This paper uses the Snort IDS [3] for demonstration purposes. The Detection Engine component of the Snort analyzes the network packets using the rules and identifies suspicious activities [8]. A Snort rule has two major components: the *rule header* and the *rule option.* A sample rule and the components are explained below.

```
alert tcp EXTERNAL_NET HTTP_PORTS -> HOME_NET any
(msg:"MALWARE-CNC Win.Trojan.BlackRev cnc allhttp
command"; flow:to_client,established;
content:allhttp|7C|; depth:8;...)
```

The rule header contains the rule action (alert), protocol (tcp), source IP (EXTERNAL_NET), source port (HTTP_PORTS), direction (->), destination IP (HOME_NET) and destination port (any). Here, HOME_NET is a user defined variable representing IPs of machines to be protected and EXTERNAL_NET is a variable representing IPs of other machines (e.g., those not in HOME_NET).

Rule options include a set of key value pairs. For example, in the above mentioned rule, *msg*, *flow*, *content* and *depth* represent keys, and *"MALWARE-CNC Win.Trojan.BlackRev cnc allhttp command"*, *to_client, established*, *allhttp|7C|* and *8* represent the corresponding values. For a given packet, if all the conditions of a rule match, then the corresponding action (here an alert) is triggered.

In an indiscriminate attack, the attacker tries to evade as many rules as possible. For example, if most of the rules contain the rule header EXTERNAL_NET -> HOME_NET, it might be possible to compromise an internal machine which belongs to HOME_NET to launch an attack. In a targeted attack, an adversary tries to evade a specific rule by modifying the packets such that one or more rule-options are not satisfied. For example, it might be possible to insert meaningless bytes in the content to evade the *content* rule option. In the next sub-sections we explain how minimum-edge-cover and clustering can be used for analyzing these attacks.

2.1 Minimum-Edge-Cover Method

Given a set of Snort rules, Minimum-Edge-Cover method analyzes vulnerability of these rules to indiscriminate attacks. This method first represents Snort rules using hyper graph.

A hyper graph [23] is a generalization of a graph, which contains a set of vertices X and subsets of X, E known as hyper edges and is denoted by (X, E). Figure 1 shows a hyper graph with five vertices $X = \{v1, v2, v3, v4, v5\}$, and three hyper edges $E = \{e1, e2, e3\} = \{\{v1, v2, v3, v4\}, \{v2, v3, v4, v5\}, \{v2, v4\}\}$.

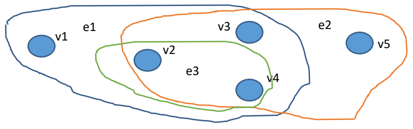

Figure 1: Hyper graph

Minimum edge cover of a graph G is a set of edges E_c, such that, each vertex in G is incident with at least one of the edges in the set [14]. In Figure 1, the minimum edge cover is $\{e1, e2\}$.

For analyzing the Snort rules, each rule (rule ID) is represented as a vertex, and each condition (rule header and rule options) is represented as a hyper edge. A hyper edge contains all the rule IDs with that condition. Minimal set of conditions that covers all the vertices (rules) or a relevant set of vertices (rules) provides insight into the possible attacking strategies. Suppose an adversary wants to make all the rules or a subset of rules in the rule set useless, the adversary needs to modify every condition in the minimum edge cover. For example, in Figure 1, suppose the adversary wants to evade all the rules. The minimum edge cover is $\{e_1, e_2\}$. So, the adversary just needs to modify the conditions corresponding to e_1 and e_2. Finding minimum edge cover is NP-hard but there exists fast approximation algorithms.

2.2 Rule-Clustering Method

For targeted attack, adversary needs to evade detection by the IDS for a specific attack (e.g., an attack to exploit a certain vulnerability). Adversary typically uses a variant of that attack to achieve this. However, it is difficult to know all possible variants of attacks. On the other hand, every Snort rule is used to detect a specific attack so we can look at similar rules rather than similar attacks.

Rule-clustering groups similar rules together, so one can view rules in the same cluster as different variants of the same attack. One can use these clusters to generalize the rules [1] and create variants of the rules that are not in the IDS in order to capture zero-day attacks.

For this study, hierarchical agglomerative clustering [18] was used to cluster the rules, in which, each rule is in its own cluster to begin with, and clusters are merged based on the distance, as one moves up the hierarchy. A customized distance measure involving Levenshtein distance [21] was used, which is explained below. Only the rule options were used for the distance computation. Consider the following two rules as an example for the distance computation.

```
alert tcp HOME_NET any -> EXTERNAL_NET [82,1087]
(msg:MALWARE-BACKDOOR Win.Backdoor.Rebhip.A
variant outbound connection type B;
flow:to_server,established; content:|70 C0 E4 28 02 26 11
3C 63 2F 8F 76 B4 55 DA 05|; fast_pattern:only;
metadata:impact_flag red, policy security-ips drop;
classtype:trojan-activity; sid:21969; rev:5;)

alert tcp HOME_NET any -> EXTERNAL_NET [82,1087]
(msg:MALWARE-BACKDOOR Win.Backdoor.Rebhip.A
variant outbound connection type A;
flow:to_server,established; content:32|7C 0A|; depth:4;
metadata:impact_flag red, policy security-ips drop;
classtype:trojan-activity; sid:21968; rev:5;)
```

- The rule options are converted to key-value format. Some keys such as sid (rule ID) and rev (revision number) are not considered, since their contribution to the measure of the difference between the rules is not very significant
- Union of the keys (msg, flow, content, fast_pattern, metadata, classtype, depth) and their intersection (msg, flow, content, metadata, classtype) are computed
- *key_distance* is computed as shown below.
  $key\_distance = length(union(k1, k2)) - length(intersect(k1, k2))$
  In the above example, $key\_distance = 2$
- For each key that is common to both the rules, Levenshtein distance between the corresponding values is computed (*value_distance*). In this example, there are 5 common keys,

Table 1: Coverage for Rule Headers

\$HOME_NET any − > \$EXTERNAL_NET \$HTTP_PORTS	2487
\$HOME_NET any − > \$EXTERNAL_NET any	346
\$HOME_NET any − > \$EXTERNAL_NET 25	211
\$EXTERNAL_NET \$HTTP_PORTS − > \$HOME_NET any	171
\$EXTERNAL_NET \$FILE_DATA_PORTS − > \$HOME_NET any	164
\$EXTERNAL_NET any − > \$HOME_NET any	157
\$EXTERNAL_NET any − > \$HOME_NET \$HTTP_PORTS	128
\$EXTERNAL_NET any − > \$SMTP_SERVERS 25	120

and the Levenshtein distance between the corresponding values are: 1, 0, 43, 0, 0

- Final distance is computed as $final\_distance = w_1 \cdot key\_distance + w_2 \cdot sum(value\_distance) = 46$, where w_1 and w_2 are weights. In the experiments, $w_1 = w_2 = 1$.

3 EXPERIMENTS AND RESULTS

3.1 Hyper Graph Representation of the Rules

For this experiment, malware rules from Snort rule version 2.9.9.0 [19] were used. The rules directory consists of a set of files, where each file has rules with common characteristics. We combined four malware files (backdoor, cnc, tools and other) and there were 5030 rules in total. The experiments were conducted using R statistical environment on a computer with 8 GB RAM and 2.5 GHz processor running the Windows 10 operating system.

For the first experiment, each rule ID was represented as a vertex and each rule header was represented as a hyper edge that included all the rules (rule IDs) with that header. Table 1 shows the rule headers for which the coverage (cardinality of the hyper edge, in other words, number of rules associated with a certain rule header) was greater than 100 rules. Figure 2 shows the cumulative coverage (number of rules covered by more than one rule header. Eg: the first two headers in Table1 together cover 2833 rules). For the second experiment, each rule ID was represented as a vertex and each rule option was represented as a hyper edge. Table 2 shows the coverage for certain relevant rule options and Figure 3 shows their cumulative coverage.

The results show that a small number of conditions have very high coverage, which indicates a potential for indiscriminate attacks. For example, we found that 96% of rules use EXTERNAL_NET in rule header. So attackers can compromise an internal machine and launch attacks from that machine, which will evade the detection of 96% of rules. This means that in real life users should be careful at setting EXTERNAL_NET. Setting it to represent IPs not in HOME_NET will lead to fewer false alarms but adversary may launch the above evasion attack. Setting it to represent any IP (both in or not in HOME_NET) will prevent such attacks, but may lead to more false alarms.

As another example, the rule option *depth:4*, indicates that the IDS only scans the first 4 bytes of the payload and the attackers may insert meaningless bytes in the beginning and evade the rules with this rule option if insertion of these bytes does not change the meaning of the packet.

Table 2: Coverage for Rule Options

flow:to_server,established	3935
distance:0	1543
flow:to_client,established	763
flowbits:noalert	303
content:POST	242
depth:4	178
depth:5	159
distance:1	136
depth:8	127
depth:6	122

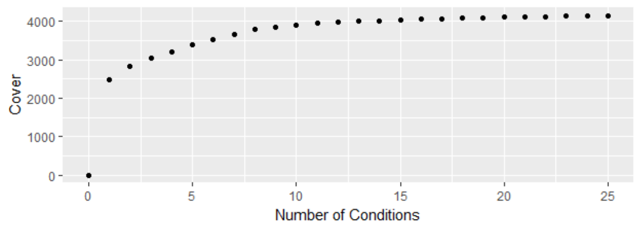

Figure 2: Cumulative Coverage for Rule Headers

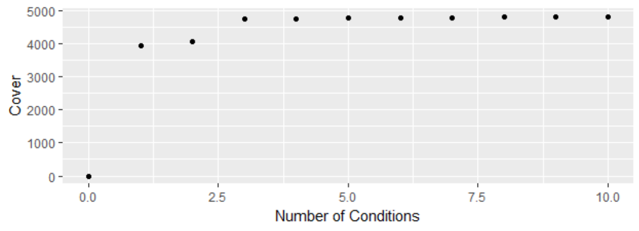

Figure 3: Cumulative Coverage for Rule Options

3.2 Clustering of Rules

For this experiment, for demonstration purposes, first 200 rules from file-office rules were considered. Agglomerative clustering was performed using the the distance measure mentioned in Section 2.2. The computation of the initial distance matrix took 27.82 seconds. The distance matrix was provided as input to the hclust function in R [17], and the time taken for clustering was 0.72 seconds. For different number of clusters, variance explained [22] was computed and is shown in Fig 4. For k=125, sample rules from one of the clusters is shown below.

```
# alert tcp EXTERNAL_NET any -> SMTP_SERVERS 25 (msg:"FI
LE-OFFICE Microsoft Office Excel with embedded Flash file
attachment attempt"; flow:to_server,established; flowbits:
isset,file.xls; content:"RldTC";fast_pattern:only;
pcre:"RldTC[ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqr
stuv][A-Za-z0-9\\x2b\x2f][A-Za-z0-9\\x2\x2f]/";metadata:
service smtp; classtype:attempted-user; sid:19067; rev:7;)

# alert tcp EXTERNAL_NET any -> SMTP_SERVERS 25 (msg:"FI
LE-OFFICE Microsoft Office Excel with embedded Flash file
attachment attempt"; flow:to_server,established;flowbits:
isset,file.xls; content:"Q1dTC";fast_pattern:only;
pcre:"Q1dTC[ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqr
stuv][A-Za-z0-9\\x2b\x2f][A-Za-z0-9\\x2\x2f]/";metadata:
service smtp; classtype:attempted-user; sid:19070; rev:7;)
```

The above rules in the same cluster are very similar. They only differ on content ("RldTC" vs. "Q1dTC") and first part of pcre (signature). It can help in generalization [1] of the rules which can be used to capture new attacks. For example, the string *RldTC* and *QldTC* in the *content* and *pcre* in the above example could be replaced with **ldTC* to capture similar attacks.

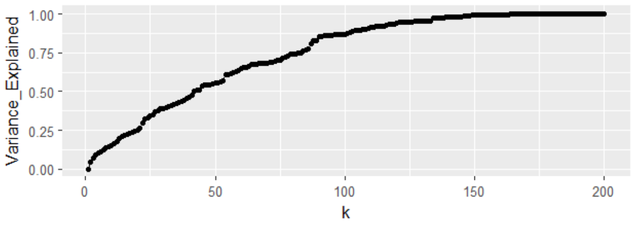

Figure 4: Variance Explained vs Number of Clusters

4 CONCLUSION

In this paper, we analyzed attacks against rule-based IDS. We examined two types of attacks and presented the minimum-edge-cover method and a clustering technique with custom distance measure to analyze the strategies of the adversaries. As a part of the future work, we intend to work on techniques for generalizing the rules by using the clusters from the proposed method.

REFERENCES

[1] Uwe Aickelin, Jamie Twycross, and Thomas Hesketh-Roberts. 2007. Rule generalisation in intrusion detection systems using SNORT. *International Journal of Electronic Security and Digital Forensics* 1, 1 (2007), 101–116.

[2] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. 2006. Can machine learning be secure?. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*. ACM, 16–25.

[3] Jay Beale, Andrew R Baker, and Joel Esler. 2007. *Snort: IDS and IPS toolkit*. Syngress.

[4] Alvaro Cardenas, Saurabh Amin, Bruno Sinopoli, Annarita Giani, Adrian Perrig, and Shankar Sastry. 2009. Challenges for securing cyber physical systems. In *Workshop on future directions in cyber-physical systems security*, Vol. 5.

[5] Alvaro A Cárdenas, Saurabh Amin, and Shankar Sastry. 2008. Research Challenges for the Security of Control Systems.. In *HotSec*.

[6] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. 2016. Adversarial perturbations against deep neural networks for malware classification. *arXiv preprint arXiv:1606.04435* (2016).

[7] Sailesh Kumar. 2007. Survey of current network intrusion detection techniques. *Washington Univ. in St. Louis* (2007).

[8] Vinod Kumar and Om Prakash Sangwan. 2012. Signature based intrusion detection system using SNORT. *International Journal of Computer Applications & Information Technology* 1, 3 (2012), 35–41.

[9] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. 2013. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications* 36, 1 (2013), 16–24.

[10] Wei Liu and Sanjay Chawla. 2010. Mining adversarial patterns via regularized loss minimization. *Machine learning* 81, 1 (2010), 69–83.

[11] Daniel Lowd and Christopher Meek. 2005. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 641–647.

[12] Robert Mitchell and Ing-Ray Chen. 2014. A survey of intrusion detection techniques for cyber-physical systems. *ACM Computing Surveys (CSUR)* 46, 4 (2014), 55.

[13] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 372–387.

[14] Sriram Pemmaraju and Steven Skiena. 2003. *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica®*. Cambridge university press.

[15] Roberto Perdisci, Guofei Gu, and Wenke Lee. 2006. Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*. IEEE, 488–498.

[16] Leonid Portnoy, Eleazar Eskin, and Sal Stolfo. 2001. Intrusion detection with unlabeled data using clustering. In *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001*. Citeseer.

[17] R Core Team. 2016. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/

[18] Lior Rokach and Oded Maimon. 2005. Clustering methods. In *Data mining and knowledge discovery handbook*. Springer, 321–352.

[19] Snort Rules. [n. d.]. ([n. d.]). Retrieved November 13, 2017 from https://www.snort.org/downloads/registered/snortrules-snapshot-2990.tar.gz

[20] Yanan Sun, Xiaohon Guan, Ting Liu, and Yang Liu. 2013. A cyber-physical monitoring system for attack detection in smart grid. In *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*. IEEE, 33–34.

[21] Robert A Wagner and Michael J Fischer. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)* 21, 1 (1974), 168–173.

[22] Xiaobei Zhao and Albin Sandelin. [n. d.]. CRAN GMD: UserâĂŹs Guide (0.3. 3). ([n. d.]).

[23] Alexander Aleksandrovich Zykov. 1974. Hypergraphs. *Russian Mathematical Surveys* 29, 6 (1974), 89–156.