

Novel Evaluation Framework of Intrusion Detection Systems with Respect to Security Policies

Negmat Mullodzhanov
Computer Science
Graduate Center, CUNY
365 Fifth Avenue, New York, NY 10016-4309

Abstract

Intrusion Detection System is an integral component of the computer security infrastructure. It is usually put in place to detect computer security policy violations. While its role is important, its effectiveness is not easily measurable. This paper proposes a framework for assessing how well a rule-based Intrusion Detection System is performing its intended task.

1. Introduction

Computer systems are ubiquitous. Once a computer system is put in place, security issues arise and need to be addressed. Security infrastructure is a necessary consequence of the computer system installation. Intrusion Detection System (IDS) is one of the components of the security infrastructure which aims at detecting computer security policy violations.

Ideally, an IDS would be able to detect the illegal activities which are specified in the computer security policies, no more and no less. More often than not, the situation is not ideal. This results in the occurrence of False Positives (i.e. detection and alerts for activities which are not considered illegal under the given computer security policies) and/or False Negatives (i.e. certain policy violations are not detected by the given set of rules). Both situations are not desirable. In the case of False Positives, the resources of IT staff, who would need to analyze these alerts, are utilized unnecessarily. And False Negatives imply that the rule database is not rich enough thereby compromising company's security.

Given a rule-based IDS with its set of detection rules, and a set of computer security policies, it would be desirable to have a mechanism for measuring how well they fit. More specifically, the existence of such mechanism would enable us to do both tasks: assess how well the IDS with the current set of rules can be expected to detect security policy violations; and further improve the fit between rules and policies upon addition/removal of certain rules to/from the IDS rule database. The work of

this research suggests the framework for the implementation of such a mechanism.

2. Related Work

To date, many Intrusion Detection Systems have been designed and implemented [1, 16, 17, 18, 19, 20, 21, 22, 23]. However, definition of a set of criteria to evaluate the effectiveness of existing IDSs remains an open issue [13]. The issues involved in testing/evaluating IDSs are explored in [2]. Initial works where IDS evaluation environment was created are described in [13, 14]. The most comprehensive evaluation, however, was done at Lincoln Laboratory, MIT [15]. While this work is of great value, it has flaws, as described in [3]. Namely, the results of the evaluation using the ROC curve have little research value, as they are not suggestive of possible improvements to the IDS under evaluation. Moreover, since it is not shown that the data used for the evaluation has a real world distribution, the results are not indicative of how the system will perform in the wild. Considering these issues, our work proposes an evaluation of the IDSs which produces results more meaningful and with practical applicability. Namely, we are evaluating the IDS solution against the organization's security policies, violations of which the IDS aims to detect. As security policies vary across organizations, it is reasonable to assume that the value of the IDS for a specific company will depend on how well tuned it is to the company's security policies.

3. Intrusion Detection Systems

An intrusion is any illegal activity taken by an inside or outside user of the computer system. Intrusion Detection involves determining that some entity, an intruder, has performed a certain illegal action against the computer system. An Intrusion Detection System (IDS) aims at detecting suspected intrusions and alerting the system administrator [12]. All the Intrusion Detection Systems

could be roughly categorized into 2 groups: Network-based and Host-based.

Network-based IDS (NIDS) deals with data at the network level (i.e. packets). It usually sits on the network and is often referred to as a sensor. The limitation of NIDS lies in it not being able to detect intrusions which have no trace on the network. The example of such an illegal activity is sniffing.

Host-based IDS (HIDS) is present on each host. It collects data concerning each host, which is in the form of log files. It can detect local attacks and is also able to determine whether the attack was successful. The disadvantages of the host-based IDS are that it is difficult to deploy and manage for large number of hosts and it cannot detect attacks against multiple targets of the network.

Each group could be further divided into anomaly-based and misuse/rule-based. In the anomaly-based system, normal state of the system is defined. An intrusion is then identified by measuring the deviation from the normal state. Alternatively, in the misuse/rule based system, illegal activities are defined based on the known ways of penetrating the system. Then, these are matched against the currently occurring activities. Those that match are considered as attacks. Anomaly-based systems suffer from the fact that what is normal is not easily describable and not necessarily static. Misuse-based systems, on the other hand, suffer from the fact that novel attacks are constantly being created. And because such systems are unlikely to be updated every time a new attack is concocted, novel attacks could go unnoticed.

Our research is based on the misuse/rule-based NIDS called SNORT.

4. State-of-art IDS: SNORT

SNORT is an open-source, lightweight Network Intrusion Detection and Prevention System. It is capable of performing real-time traffic analysis and packet logging on TCP/IP networks. It can perform protocol analysis, content searching/matching and can be used to detect a variety of attacks and probes, such as buffer overflows, denial of service, stealth port scans, and unauthorized system access [1].

4.1 Rules and their components

Rules are the heart of SNORT. Each rule is designed to detect an instance of a certain type of an attack. Detection is performed by matching a rule with a packet.

Each SNORT rule is divided into two logical sections: *rule header*, which contains the rule's action, protocol, source and destination IP addresses and

netmasks, and the source and destination ports information; *rule option*, which contains alert messages and information on which parts of the packet should be inspected to determine if the rule action should be taken [4]. Below is an example of a SNORT rule, designed to detect a type of a DOS (Denial of service) attack:

```
alert ip $EXTERNAL_NET any->
$HOME_NET any (msg:"DOS IGMP dos
attack"; fragbits:M+; ip_proto:2;
reference:bugtraq,514;
reference:cve,1999-0918;
reference:url,www.microsoft.com/technet
/security/bulletin/MS99-034.mspx;
classtype:attempted-dos;
sid:272;rev:11;)
```

In the rule above, all the text preceding the open parenthesis is considered as rule header and text enclosed in the parenthesis are rule options. The above rule will alert on any packet coming from the external network where *protocol = ip* and *More fragments bit* in the IP header is set. The important rule component for our research is **reference**, which specifies the vulnerability database(s) which contain(s) information about the vulnerability this rule was designed to detect.

5. Vulnerability Databases

In general, a vulnerability database contains information about different types of vulnerabilities and exposures. The information usually includes items such as vulnerability name, description of how it is exploited, and patches, if available. There are currently many different vulnerability databases [9]. The vulnerability information source we find most useful is Common Vulnerabilities and Exposures (CVE) [8]. CVE is a dictionary which aims to standardize the names for all publicly known vulnerabilities and security exposures. Currently, the dictionary contains 3052 entries, which come from multiple vulnerability databases [9]. An example of a CVE entry is shown below:

```
Name:CVE-1999-0918
Description:Denial of service in
various Windows
systems via malformed,
fragmented IGMP packets.
```

CVE is sponsored by the US-CERT at the U.S. Department of Homeland Security. US-CERT is the operational arm of the National Cyber Security Division at the

U.S. Department of Homeland Security. US-CERT incorporates CVE names into its security advisories whenever possible and advocates the use of CVE and CVE-compatible products and services to the U.S. government and all members of the information security community [11].

Many organizations are currently working to make their product/service CVE compatible [10]. This implies that as time goes on CVE will become more complete with respect to the space of all the known vulnerabilities. Therefore, CVE can be used to provide a baseline for evaluating the coverage of organization's security tools (e.g. IDS).

6. Security Policies

A policy is typically a document that outlines specific requirements or rules that must be met. In the information/network security realm, policies are usually point-specific, covering a single area. For example, an "Acceptable Use" policy would cover the rules and regulations for appropriate use of the computing facilities [5]. As long as an organization has computing devices, it should have computer security policies. And as mentioned in the previous sections, in order to detect security policy violations, an IDS is installed, as a component of the computer security infrastructure. It should be noted, however, that IDS cannot detect the violations of all the parts of the security policies. Therefore, if one is to judge the effectiveness of an IDS at detecting security policy violations, one first needs to identify the parts of the security policies, violations of which are IDS-detectable and make the evaluation within that space.

More specifically, security policies could be divided into items, violations of which are detectable by an IDS and those that are not. And those violations which are IDS-detectable could be further divided into those detectable by NIDS and those detectable by HIDS. Network scanning, for example, could be detected by NIDS, while sniffing could only be detected by HIDS, since this type of an activity leaves no trace on the network.

For the purpose of this research work, we used the policy templates from the SANS SECURITY POLICY PROJECT [5]. These are sanitized security policies from large organizations. They were developed by a group of experienced security professionals with experience in government and commercial organizations, and each policy went through a vigorous approval process. Two examples of the NIDS enforceable security policies from the source mentioned above are:

1. *Interfering with or denying service to any user other than the employee's host*

2. *Introduction of malicious programs into the network or server (e.g., viruses, worms, Trojan horses, e-mail bombs, etc.)*

7. Assessment Framework

7.1 Why is there no straightforward solution?

As explained in the previous sections, rules are very granular while the policies are coarse. That is, the rule specifies the values certain parts of the analyzed packet need to have, in order for this packet to trigger an alert. The policies, however, state in plain English which activities are disallowed. Representational differences make the comparison between the two not straightforward. To bridge between two representations, a meta-space had to be found, as the representational space for both.

The CVE dictionary is used as the metaspace. The idea is to identify the portion of the CVE space covered by the IDS rules and the portion covered by the policies. This information is then used to evaluate the portion of rules and policies which match and/or mismatch.

To summarize the assessment framework: The rules and policies are mapped to the CVE space through the reference component and keywords, respectively. Once both are in the CVE space, a decision is made on which rules are to be added and/or removed to/from the IDS rule database.

7.2 Mapping Rules to CVE items

Mapping rules to CVE items is straightforward. The evaluator will take each rule, and looking at the reference component identify the CVE name for it. This name will be the rule's representative in the CVE space. For example, the rule shown below has a CVE representative identified as **CVE-2000-0474**:

```
alert tcp $EXTERNAL_NET any ->
$HOME_NET 8080 (msg:"DOS Real Server
template.html";
flow:to_server,established;
content:"/viewsource/template.html?";
nocase;reference:bugtraq,1288;
reference:cve,2000-0474;
classtype:attempted-dos; sid:278; rev:6)
```

Once all the rules are processed, the result a list of rules, each assigned a CVE name. It should be noted that this step can be easily automated.

7.3 Mapping Policies to CVE items: Step 1

Policies appear in the form of English sentences, stating what activities are not allowed. CVE items also appear in the form of English sentences, describing a certain vulnerability. A person performing a match/mapping between policy items and CVE items would have to analyze each policy and all the CVE items, making a judgement on where a match occurs. The judgement will obviously be subjective, as there is ambiguity involved.

To assist in the assignment procedure, and decrease the ambiguity, we suggest to define a set of keywords, which are to guide the evaluation. As such, the evaluator will be provided with the policies, CVE items and the keywords. First, the CVE items will be categorized. Namely, the evaluator will scan through each CVE item, and upon a keyword match, in the description associated with a CVE item, assign this item to the category associated with this keyword. Assuming that we have a category *denial of service*, CVE item shown in section 5 will be assigned to this category.

Depending on the nature of the keywords, a single CVE item could be mapped into one or multiple keywords. The result of this evaluation can be represented by a table with 2 columns where the first column specifies the keyword and the second column, of the same row, lists all the CVE items assigned to this keyword category.

Second, categorization of the policy items will be performed in a similar fashion. For each policy item, the set of categories it belongs to will be identified by scan and word match. At the end of this procedure, we will have a set of policy items, each assigned to one or more categories. Assume that we have categories: *network monitoring*, *unauthorized system access*, and *denial of service*. Below are two examples of policy categorization:

(1) Item 13 in section 4.2 in the **InfoSec Acceptable Use Policy** [5]: *Interfering with or denying service to any user other than the employee's host*. This policy will be assigned to category *denial of service*.

(2) A policy can cover multiple categories: *Effecting security breaches or disruptions of network communication*. *Security breaches include, but are not limited to, accessing data of which the employee is not the intended recipient or logging into a server or account that the employee is **not expressly authorized to access**, unless duties are within the scope of regular duties*. For purposes of this section, "disruption" includes, but is not limited to **network sniffing**, pinged floods, packet spoofing, **denial of service**, and forged routing information for malicious purposes. From keywords bolded above we can claim that this policy will be assigned to at least the following categories: *network monitoring*, *unautho-*

rized system access, and *denial of service*.

A policy item belonging to category c is taken to cover all the CVE entries in the table belonging to category c .

7.4 Mapping Policies to CVE items: Step 2

Assignment of policies to a set of categories could be represented by a binary string. Assume that we have categories indexed 1, 2, ... k . For any given policy item, some of these k categories will be ON (i.e. 1 in the i^{th} position of the binary string) and others will be OFF (i.e. 0 in the i^{th} position of the binary string).

For example, let $a = '001001'$ be a binary string produced from a policy item. The meaning of this binary string is the following: *This policy covers categories 3 and 6*.

In order to obtain the complete information about the categories all the policy items cover, we will apply a logical OR to the set of binary strings produced from each policy item. The result is a Policy Final Binary String, which will be further used in the evaluation procedure.

7.5 Deriving keywords

While the advantage of introducing keywords into the mapping procedure is clear, the derivation of these keywords is not an easy task. We believe that the chosen keywords should have at least the following properties:

1. Most/all policy/CVE items could be categorized using them
2. The keyword carries most of the information/meaning which was in the item that mapped into it

Property 1 ensures that policies can be evaluated against CVE items and vice versa. Property 2 ensures that if a policy and a CVE item are classified into the same category, then they carry the same meaning. In our study, we used the types of illegal activities as keywords. The source we used for derivation were the 34 class types defined in `/etc/classification.config` file which comes with latest SNORT release [7]. As described in [4], a class type categorizes alerts into attack classes. Some class types, however, are too specific and policy/CVE items would not map to them (e.g. string-detect), while some others are too general, resulting in any policy/CVE item mapping into them (e.g. policy violation). These class types are removed, grouping the remaining into the 7 new categories:

1. **Denial of service**: attempted-dos, successful-dos, denial-of service

2. **Information leak:** attempted-recon, successful-recon-limited, successful-recon-largescale
3. **Unauthorized system access:** attempted-admin, successful-admin, attempted-user, successful-user, suspicious-login, default-login attempt
4. **Trojan activity:** trojan-activity
5. **Scan:** network-scan
6. **Web attack:** web-application-activity, web-attack
7. **Miscellaneous:** misc-activity, misc-attack

We believe it is reasonable to use types of illegal activities as keywords. From the CVE side, the description of the vulnerability always contains information on the illegal activity/attack type this vulnerability belongs to. From the policy side, the policy items state exactly what activities are illegal. Thus, Property 1 should be satisfied as long as the list of illegal activities is detailed and exhaustive enough. No test was yet designed for examining whether Property 2 is satisfied.

7.6 Evaluation

At this stage of the procedure we will have a set of CVE items from the analyzed rules (i.e. those covered by all the rules taken together), a table of CVE items categorized by keywords, and a binary string produced from policies, where the i^{th} bit specifies whether any policy covers the i^{th} category.

At the final evaluation step, the results from these three sources are analyzed to suggest what rules need to be added to or removed from the IDS database as to tune it towards the company's security policy.

STEP 1: Take the CVE items covered by all the rules, CVE_{rules}^{all} , and the CVE table, and mark the items in the table that match CVE_{rules}^{all} . The marked items in the table are the portion of the CVE space covered by the rule set of the IDS under evaluation.

STEP 2: Analyze the CVE table with marked items against the Policy Final Binary String. Below are 3 possible scenarios which could occur during this analysis and the suggested action to be taken under each:

Case 1: The i^{th} bit in the policy string contains a zero and nothing is marked in the i^{th} row of the table or the i^{th} bit in the policy string contains a one and everything is marked in the i^{th} row of the table. In either case we will do nothing, as this is an exact match between rules and policies for the CVE subspace represented by the i^{th} category.

Case 2: The i^{th} bit in the policy string contains a zero, while some of the items are marked in the i^{th} row

of the table. This is a case where we have rules covering a CVE sub-space not covered by the current policy. The rules associated with the marked items should be removed to minimize the generation of false positives.

Case 3: The i^{th} bit in the policy string contains a one and nothing or only some vulnerabilities are marked in the i^{th} row of the table. When the i^{th} bit in the policy binary string is set, it implies that all the CVE items associated with i^{th} keyword are covered by the current policy. Therefore, for the vulnerabilities in the i^{th} row of the CVE table which are not marked, we suggest to write/add the rules to detect them, as to minimize the false negatives.

8. Discussions

The proposed framework assesses the effectiveness of an IDS with respect to a certain set of security policies. This approach is more practically meaningful than other types of evaluations, as it suggests improvements to the IDS rule set so that they better fit the company's security policies. There is, however, at least one issue which needs to be addressed. Namely, more than 50% of the SNORT rules do not provide a reference to the CVE dictionary. This implies that we would not be able to map them into the CVE space. We believe that this issue needs to be addressed from the side of the writers of SNORT rules. If the rule is detecting a certain type of a publicly known vulnerability, the rule should reference one of the public vulnerability databases, which would further be referenced by CVE. Considering that SNORT is CVE compatible, this issue will most likely be resolved in the future.

9. Conclusions and Future Work

There often exists a gap between what the security policies forbid, and the violations detected by the security infrastructure. An effective Intrusion Detection System, being a component of this infrastructure, will be able to detect security policy violations which are IDS-detectable. The value of this work lies in accentuating the need for and utility of a mechanism for evaluating the implemented IDS solution with respect to the security policies whose violations it claims to detect, and noting the difficulties encountered in doing so due to different levels of granularity between rules and policies. Although the proposed mechanism is still being designed, we show that such mechanism is feasible and practically applicable. In the future we plan to implement this framework, to test its practicality, automating it in the process. We will also explore approaches for choosing good keywords and design methods for testing

that they satisfy the two properties, as described in the paper.

References

- [1] Martin Roesch. SNORT - Lightweight Intrusion Detection For Networks. *Proceedings of LISA '99: 13th Systems Administration Conference*, Vol. 17, pp. 1-100, 1987.
- [2] P. Mell, V. Hu, R. Lippmann, J. Haines, M. Zissman. An Overview of Issues in Testing Intrusion Detection Systems. No. NIST IR 7007, National Institute of Standards and Technology, August 2003.
- [3] John McHugh. Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory. *ACM Transactions on Information and System Security*, Vol. 3, No. 4, November 2000, 262-294.
- [4] Martin Roesch, Chris Green. Snort Users Manual: 2.4.0RC1. *Snort Documentation*, 2005.
- [5] The SANS Security Policy Project, 2006. <http://www.sans.org/resources/policies/>.
- [6] Sourcefire VRT Certified Rules, February 2006. <http://www.snort.org/rules/>.
- [7] SNORT Downloads. <http://www.snort.org/dl/>.
- [8] Common Vulnerabilities and Exposures Dictionary. <http://www.cve.mitre.org/cve>
- [9] Common Vulnerabilities and Exposures Data Sources. <http://www.cve.mitre.org/cve/datasources.html>
- [10] CVE Compatible Organizations. <http://www.cve.mitre.org/compatible/>
- [11] CVE and US-CERT. <http://www.cve.mitre.org/advisory/uscert.html>
- [12] Anita K. Jones and Robert S. Sielken. Computer System Intrusion Detection: A Survey, 1999
- [13] Debar, H., Dacier, M., Wespi, A., and Lampart, S. An experimental workbench for intrusion detection systems. *Res. Rep. RZ 2998 (93044)*. Research Division, IBM, New York, NY, September, 1998.
- [14] Puketza, N., Chung, M., Olsson, R. A., and Mukherjee B. A software platform for testing intrusion detection systems. *IEEE Software*, 14, 5, September 1997, 43-51.
- [15] Lippmann, R.P., Fried, D., Graf, I., Haines, J., Kendall, K., McClung, D., Webber, D., Webster, S., Wyschograd, D., Cunningham, R., and Zissman, M. Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. *IEEE Computer Society Press*, January 2000, 12-26.
- [16] Anderson, D., T. Frivold and A. Valdes. Next-generation Intrusion Detection Expert System (NIDES): A Summary. *Technical Report SRI-CSL-95-07*, May 1995.
- [17] Hochberg, J., K. Jackson, C. Stallings, J.F. McClary, D. DuBois, and J. Ford. NADIR: An Automated System for Detecting Network Intrusion and Misuse. *Computers and Security*, 12.3 (1993) 235-248.
- [18] Ilgun, K. USTAT: A Real-Time Intrusion Detection System for UNIX. *Proceedings of the IEEE Symposium on Research in Security and Privacy*, May 1993
- [19] Kemmerer, R.A. NSTAT: A Model-based Real-time Network Intrusion Detection System. *University of California-Santa Barbara Technical Report TRCS97-18*, November, 1997
- [20] V.Paxon. Bro: A system for detecting network intruders in real-time. *In Proceedings of the 7th USENIX Security Symposium*, 1998
- [21] Smaha, S.E.. Haystack: An Intrusion Detection System. *Fourth Aerospace Computer Security Applications Conference*, December 1988.
- [22] Snapp, S.R., J. Brentano, G.V. Dias, T.L. Goan, L.T. Heberlein, C. Ho, K.N. Levitt, B. Mukherjee, S.E. Smaha, T. Grance, D.M. Teal and D. Mansur. DIDS (Distributed Intrusion Detection System) Motivation, Architecture, and An Early Prototype. *Proceedings of the 14th National Computer Security Conference*, October 1991
- [23] Staniford-Chen, S., S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. GrIDS. A Graph Based Intrusion Detection System for Large Networks. *20th National Information System Security Conference (NISSC)*, October 1996