A Vulnerability-Driven Approach to Active Alert Verification

ZHIHONG TIAN[1], BINXING FANG, BIN LI, HONGLI ZHANG

Research Center of Computer Network and Information Security Technology
Harbin Institute of Technology, Harbin 150001
CHINA
1 http://pact518.hit.edu.cn/~tianzhihong

Abstract: - Intrusion detection systems are used to alert system administrators to malicious attacks. Unfortunately, running without any information of the network resources that they protect, intrusion detection systems are notorious for generating a large number of alerts that are either not related to malicious activity or not representative of a successful attack. To address this shortcoming, this paper presents a vulnerability-driven active alert verification approach that performs real-time verification of attacks detected by an intrusion detection system. By means of checking for the vulnerability that the attack attempts to exploit, we can verify whether the attack has succeeded or not. The Experimental evaluation illustrates that it is a useful tool for reducing the false positive rate.

Key-Words: - Network Security, Intrusion Detection, Alert Verification, Vulnerability

1 Introduction

The frequency of computer intrusions has been increasing rapidly for several years [1]. Internet intrusion and large-scale attacks can have catastrophic affects, including stolen or corrupted data, wide spread denial-of-service attacks, huge financial losses and even disruption of essential services. Internet security becomes more and more important. As a result, even the most diligent system administrators must cope with the reality of computer break-ins. One way system administrators combat attacks is by using intrusion detection systems (IDS). IDS has been considered the second line of defense for computer and network systems along with the prevention-based techniques such as authentication and access control. These systems analyze information about the activities performed in computer systems and networks, looking for evidence of malicious behavior. When an attack is detected, an alert that describes the type of the attack and the entities involved (e.g., hosts, processes, users) is produced.

Unfortunately, intrusion detection today faces two major challenges. Firstly, the overall number of alerts generated is overwhelming for administrators. Secondly, too many of these alerts are either not related to malicious activity (false positives) or not representative of a successful attack (non-relevant positives) [2]. The classic example is the scenario of a CodeRed attack that targets a Linux web server. It is a valid attack that is seen on the network, however, the alert that an IDS raises is of no use because the service is not vulnerable (as CodeRed can only exploit vulnerabilities in Microsoft's IIS web server). The overwhelming amount of false alerts for a system administrator to process can eventually lead to a false sense of security. Therefore, the problem of the vast imbalance between actual and false or non-relevant alerts limits the accuracy and effectiveness of IDSs.

Against this background, this paper intrduces a Vulnerability-driven Active Alert Verification approach (VAAV) that attempts to address the aforementioned shortcomings in current IDSs. Like IDSs, VAAV monitors both incoming and outgoing network packets promiscuously. Specifically, when a suspect packet is indicative of an attack on an existing network service, instead of generating the corresponding alert, VAAV further checks for the vulnerability that this attack attempts to exploit. Along the way, we can verify if the packet did in fact lead a successful break-in, and thereby help an administrator more easily focus her detective work on those real intrusions, leading to a quicker and easier identification of the threat level. Our work provides the first steps to relieve an administrator from dealing with large volumes of false alerts.

2 Related Work

As we mentioned, one thing that severely restricts the development of IDSs is to analyze a very large number of false alerts for practical attack scenarios.

There have been several proposals to address the difficult problem in recent years.

Porras et al. design a "mission-impact-based" correlation system, named M-Correlator [3]. The main idea is to evaluate alerts based on security interests and attack relevance to the protected networks and hosts. Related alerts are aggregated and clustered into a consolidated incident stream. The final result of the M-Correlator is a list of rank ordered security incidents based on the relevance and priority scores, which can be further analyzed by the system administrator. This approach focuses on the incident ranking and the system administrator needs to perform further correlation analysis.

The probabilistic alert correlation [4] and the alert clustering methods in [5,6] correlate alerts based on the similarities between alert attributes. Measures are defined to evaluate the degree of similarity between two alerts. Alert aggregation and scenario analysis are conducted by toughening or relaxing the similarity requirement in some attribute fields. Though they are effective for clustering similar alerts (e.g., alerts with the same source and destination IP addresses), they cannot correlate alerts that do not have obvious (or predefined) similarities in their attributes.

Dain et al. use the data mining approach to combine the alerts into scenarios in realtime [7]. The purpose of the scenarios is simply to group alerts that share a common cause. The resulting scenarios give system administrator a more complete picture of the traffic on their network rather than individual alerts. The main limitation of this method is that it relies on the attack scenarios predefined by human users, or learned from training datasets.

Some other researchers have proposed the framework of alert correlation and scenario analysis [8,9,10]. These approaches target recognition of multi-stage attacks. The assumption is that when an attacker launches a scenario, prior attack steps are preparing for later ones, and therefore, alerts are correlated if the prerequisites of some later alerts are satisfied by the consequences of some earlier alerts. Such methods can potentially uncover the causal relationship between alerts. However, these methods are time-consuming and error-prone.

Our approach differs from above studies. We check whether an attack has succeeded or not, by correlating the host vulnerability information with the attack. When the attack has not succeeded, the alert will be suppressed. This provides an effective method to lower the number of false alerts that an administrator has to deal with.

To our best knowledge, Christopher et al. originally proposed the term alert verification [11] to address the problem of false positives. However, their implementation is very complicated. Based on their idea, our active alert verification is more easy and flexible. Specially, we extend the alert verification mechanism by using some optimization method.

The rest of the paper is organized as follows. In Section 3, we introduce the VAAV state-machine model. Section 4 presents the implementation details of the proposed VAAV system. The experiments and the results are described in Section 5. Finally, Section 6 draws conclusions and outlines future work.

3 VAAV System State-Machine

To understand VAAV system more easily, we list the relation of IDS, attack signature and vulnerability in Table 1.

Table 1. Relations of each element

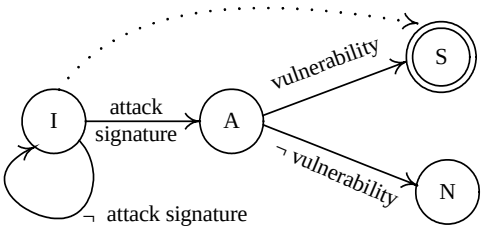
	Attack Signature	Vulnerability	Alert	Relult
1	√	√	√	True positive
2	√	X	√	Non-relevant positives
3	√	√	X	False Negative
4	X	√	√	False Positive
5	X	X	√	False Positive
6	√	X	X	True positive
7	X	√	X	X
8	X	X	X	X

As can be seen from Table 1, when an IDS sensor outputs an alert, there are eight possibilities. But only type-1 and type-6 alerts are desired scenario. Note that, in type-2, the IDS sensor has correctly identified an attack signature, but the attack failed to meet its targets. This kind of alert is called a non-relevant positive. In fact, the correct output of an ideal IDS should be type-6 alert in this condition. Thus, the key idea of VAAV is to distinguish between successful and failed intrusion attempts (non-relevant positives).

To explain pain and make it understood, the process of VAAV approach can be considered as a finite state machine, as shown in Figure 1. The nodes correspond to possible states and the edges denote state transitions. We associate transitions to the occurrence of events. These events are collections of

attack signature and vulnerability, denoted A, based on two operators =, ¬, with A = { attack-signature, vulnerability, ¬ attack-signature, ¬ vulnerability }.

In Figure 1, the solid arc leading from the initial state *I* to the alert state *A* indicates that VAAV has detected a malicious action according to the signature defined in attack-signature. Note that the state will shift from *A* to *S* if and only if the corresponding vulnerability is matched, which indicates the attack is successful. That is, VAAV determines a successful intrusion depending on two steps of actions (attack-signature and vulnerability) performed by an attacker that leads from the initial state to the successful intrusion state (the final state), as the dashed arc shows.



I—Initial State *S*—Successful Intrusion State
A—Alert State N—Non-relevant positive State

Fig.1 VAAV state transition diagram

4 The Proposed VAAV System Design

The role of VAAV is to verify the success of attacks. Which is designed on base of Snort [12]. Snort has been chosen because it is a cross-platform, lightweight network intrusion detection tool and for each new attack, the rules writing for it is very easy.

When a Snort rule is fired and an attack is detected, it is the task of VAAV to use a vulnerability scanner to check for the vulnerability that this attack attempts to exploit.

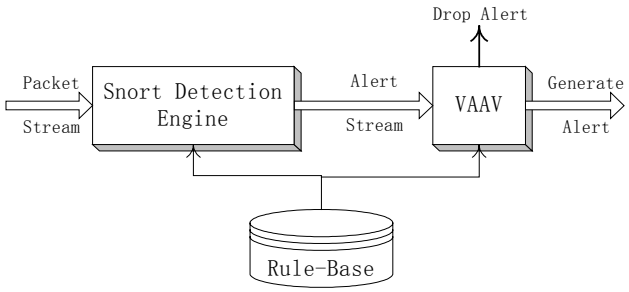


Fig.2 The relation between VAAV and Snort

Figure 2 describes the relations between Snort and VAAV. When a Snort rule is triggered, the associated alert is passed to VAAV for verification. Then, VAAV utilizes the information provided by Nessus [13], a popular vulnerability scanner to flag the alert as non-relevant or not.

4.1 Implementation

In this section, we introduce the implementation of VAAV in detail.

From the above discussion, we can seen that VAAV is implemented an extension for Snort. Figure 3 describes all the modules of Snort. Each packet captured by Snort will be processed by decoder, preprocessors, detection-engine and output plug-ins respectively. Note that the detection plug-ins modules. Those modules are referenced from its definition in the rules files, and they are intended to identify patterns whenever a rule is evaluated (i.e. the detection engine makes use of the detection plug-ins to match each packet). Therefore, the detection plug-in mechanism provides additional detection functions on the packets. In practice, we modified Snort by adding some active alert verification plug-ins and extending the Snort rule-base at the same time.

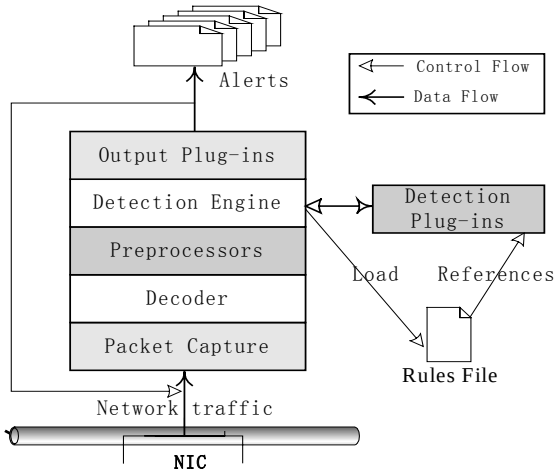


Fig.3 Snort component diagram

4.2 Active Alert Verification Plug-ins

In most Snort rule, there is a field, named CVE-ID, a unique identifier for vulnerabilities which is assigned by the Common Vulnerabilities and Exposures project [14]. Considering the following example of a Snort rule:

Table 2. Snort rule for Linux mountd overflow

alert udp $EXTERNAL_NET any -> $HOME_NET 635 (msg:"EXPLOIT x86 Linux mountd overflow"; content:"|5eb0 0289 06fe c889 4604 b006 8946|"; reference:cve,**CVE-1999-0002**; reference:bugtraq,121; classtype:attempted-admin; sid:315; rev:3;)

This rule augments the standard Snort rule for the Linux mountd overflow. Some implementations of the Network File System (NFS) on Linux systems use a vulnerable version of mountd that is subject to a buffer overflow condition in the logging subsystem. It is possible that intruders escalate privileges remotely using the vulnerability in mountd. Note that the CVE-ID of this vulnerability is CVE-1999-0002.

On the other hand, as a great tool designed to automate the testing and discovery of known security problems, Nessus security scanner has a strong script language, named NASL (Nessus Attack Scripting Language). NASL plug-ins typically test by sending very specific code the target and comparing the results against stored vulnerable values. Table 3 lists the NASL script corresponding to the **Linux mountd overf**low vulnerability. Similarly for every NASL script there is also a CVE-ID associated with it by Nessus.

Table 3. NASL script for Linux mountd overflow

```
if(description)
{
script_id(11337);
script_version ("$Revision: 1.3 $");
script_cve_id("CVE-1999-0002");

name["english"] = "mountd overflow";
script_name(english:name["english"]);

desc["english"] = "The remote mount daemon seems
to be vulnerable to a buffer overflow when it receives
a request for an oversized share. An attacker may use
this flaw to gain root access on this host. Risk factor :
High";
… … …
```

In practice, according to the "sid-msg.map" file in the Snort distribution and the NASL scripts, we create Snort-to-Nessus mappings using CVE-IDs. At the same time, Snort rule-base is extended and most rules are modified to accommodate the active alert verification plug-ins. Table 4 shows the new rule for Linux mountd overflow attack.

Table 4. New rule for Linux mountd overflow

```
alert udp $EXTERNAL_NET any -> $HOME_NET 635
(msg:"EXPLOIT x86 Linux mountd overflow";
content:"|5eb0 0289 06fe c889 4604 b006 8946|";
vulcorrelation: "$FULLNAME";
reference:cve,CVE-1999-0002; reference:bugtraq,121;
classtype:attempted-admin; sid:315; rev:3;)
```

The *vulcorrelation* is keyword to the processing function of active alert verification. The variable *$FULLNAME* is the parameter and indicates the directory that contains the appropriate NASL script associating with this rule, which can be obtained according to the Snort-to-Nessus mappings.

The advantage of using *$FULLNAME*, instead of extracting CVE-ID automatically like the method in [11], as the index to NASL scripts is that our approach can process the non-CVE signatures. For the signatures that don't match a NASL script with the CVE-ID, we can write corresponding NASL script and inform active alert verification plug-in by the variable *$FULLNAME*.

Thus, once a detection rule is triggered, the active alert verification plug-ins will dynamically call the *execute_nasl_script()* routine (defined in the exec.c of the Nessus project) to execute the appropriate NASL script against the victim host to check whether there is the corresponding vulnerability. If the target has the vulnerability, the output plug-ins will be called to generate an alert, which shows the attack is successful. However, if there is no corresponding security hole, the attack will be tagged as non-relevant, which shows the attack cannot cause security threat to the target host or network.

4.3 Optimization Method

As one can see, VAAV must process each suspicious packet. However, if intruder launches hostile attack such as alert flooding [15] to VAAV itself, the performance of VAAV will be reduced significantly. This is because when attacker launches attacks such as repeated Ping of Death or Teardrop, alerts usually arrive in batches of 50 alerts or more.

For performance consideration, we define two additional rule options *THRESHOLD* and *QUANTUM*, which provide a mechanism to handle alert flooding. We aggregate these alerts by the "*THRESHOLD*" keyword. That is, such alerts will only be processed when the count on the alert reaches the given value of $*THRESHOLD* in the time window of $*QUANTUM*.

4.4 False Positives

By design, VAAV is able to verify most known vulnerability, and hence should have very low false positives.

However, some viruses, such as Cheese worm, will patch the corresponding security hole after it exploits the vulnerability. In this case, even though VAAV matches the attack signature, it cannot find the vulnerability on the target host. We will consider it as a future work for further analyzing.

5 Evaluation

This section presents the evaluation of the presented techniques using the data generated by the test network constructed for this purpose. The topology of the test network is shown in Figure 4. Three machines were present on this test:

1. an attacker machine;
2. a victim host with Linux Redhat 7.3 installed;
3. a machine with VAAV deployed.

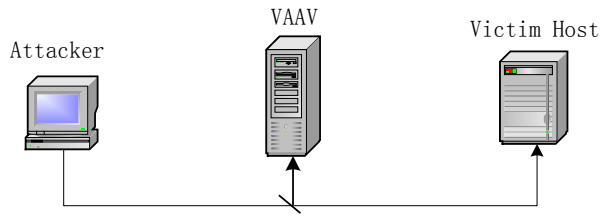


Fig.4 Topology of test network

In our experiments the attacks were performed using various tools and techniques. All the attacks we used are shown in the Table 5. For convenience, we updated the victim host with the patches for all of these attacks listed in Table 5.

Table 5. The attacks used for experiment

Name	CVE-ID
WEB-IIS Unicode directory traversal attempt	CVE-2000-0884
SMTP sendmail 8.6.9 exploit	CVE-1999-0204
EXPLOIT x86 Linux samba overflow	CVE-1999-0811
EXPLOIT x86 Linux mountd overflow	CVE-1999-0002
WEB-IIS ISAPI .ida access	CVE -2000-0071
WEB-IIS ISAPI .idq attempt	CVE -2000-0071
FTP EXPLOIT wu-ftpd 2.6.0 site exec format string overflow Linux	CVE -2000-0573
FTP EXPLOIT format string	CVE -2000-0573

The results are shown in Table 6. As one can see, traditional Snort will report 8 alerts. However, because no vulnerability was actually present on the victim host, these attacks could not have been successful and can thus be considered non-relevant. Clearly, VAAV can correctly identify this condition and none of alert was reported. Thus, the true positive of VAAV reach to 100%.

Table 6. Evaluation results

	Alerts	True Positives
Snort	8	0
VAAV	0	8

6 Conclusions and Future Work

In the above study, a vulnerability-driven approach to active alert verification was presented, which help an intrusion detection system to check whether the target host has the corresponding vulnerability or not. With the aid of detection plug-ins of Snort, we have implemented a prototype based on Snort and Nessus. Experimental evaluation illustrates that it is a useful tool for reducing the false positive rate of Snort.

In the future, we plan to do further analyzing about the false positive introduced in subsection4.4. Furthermore, more optimization methods should be found to increase the overall performance of VAAV.

References:

[1] CERT/CC Overview Incident and Vulnerability Trends. Technical report, CERT Coordination Center, APRIL 2002.

[2] Benjamin Morin, Ludovic Mé, Hervé Debar and Mireille Ducassé: “M2D2: A Formal Data Model for IDS Alert Correlation”; in Proceedings of Recent Advances in Intrusion Detection 2002, LNCS 2516, pp. 115-137; Springer-Verlag; 2002.

[3] P. A. Porras, M. W. Fong, and A. Valdes. A Mission-Impact-Based approach to INFOSEC alarm correlation. In Proceedings of the 5th International Symposium on Recent Advances in Intrusion Dtetection (RAID), October 2002.

[4] A. Valdes and K. Skinner. Probabilistic alert correlation. In Proceedings of the 4th International Symposium on Recent Advances in Intrusion Dtetection (RAID), October 2001.

[5] Cuppens, F. Managing alerts in a multi-intrusion detection environment. In Proceedings of 17th Annual Computer Security Applications Conference (ACSAC). 2001.

[6] Julisch, K. Mining alarm clusters to improve alarm handling efficiency. In Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC). 2001.

[7] Dain, O. and Cunningham, R.. Fusing a heterogeneous alert stream into scenarios. In Proceedings of the 2001 ACM Workshop on Data Mining for Security Applications. 1–13. 2001.

[8] S. Cheung, U. Lindqvist, and M. W. Fong. Modeling multistep cyber attacks for scenario recognition. In Proceedings of the Third DARPA Information Survivability Conference and Exposition (DISCEX III), Washington, D.C., April 2003.

[9] F. Cuppens and A. Miege. Alert correlation in a cooperative intrusion detection framework. In Proceedings of 2002 IEEE Symposium on

Security and Privacy, pages 202-215, Oakland, CA, May 2002.
[10] P.Ning, Y. Cui and D. S. Reeves. Constructing attack scenarios through correlation of intrusion alerts. In 9th ACM Conference on Computer and Communications Security, November 2002.
[11] Christopher Kruegel and William Robertson, Alert Verification - Determining the success of intrusion attempts. http://www.infosecwriters. com/hhworld /hh8/ava.txt.
[12] Snort - The Open Source Network Intrusion Detection System. http://www.snort.org.
[13] Nessus Vulnerability Scanner. http://www.ness-us.org/.
[14] Common Vulnerabilities and Exposures. http://www.cve.mitre.org/.
[15] T. H. Ptacek and N. N. Newsham. Insertion, evasion and denial of service: eluding network intrusion detection. January 1998.