

# Towards a Collaborative and Systematic Approach to Alert Verification

Xuejiao Liu

Institute of Network and Communication Technology, Huazhong Normal University, Wuhan, China

Email: liuxuejiao@gmail.com

Debao Xiao and Xi Peng

Institute of Network and Communication Technology, Huazhong Normal University, Wuhan, China

Email: {dbxiao, x.peng }@mail.cnu.edu.cn

**Abstract**—With the menace of hackers increasing every day, even well administrated networks are vulnerable to attack. Therefore, securing network assets has become a significant issue for the corporate world. Traditionally, the tools used for network security involve the firewall systems, the intrusion detection system (IDS), the anti-virus software, the vulnerability scanning software. Each of the above, used independently or in conjunction, is not capable of withstanding the onslaught of various network threats. To solve the problem, collaborative security paradigms, which integrate some security devices closely to provide accurate alerts and attain enhanced protection are emerging. In this paper we present a collaborative framework of vulnerability-based alert verification. Along with conducting systematic analyses on contextual information modeling, alert verification is applied above intrusion detection aimed at helping determine whether the attack was successful or not. Based on the confidence assigned in the process of verification, appropriate actions can be taken to deal with the attacks. That confidence enables an administrator to perform further analysis: identifying the alerts and prioritizing defense accordingly.

**Index Terms**—Vulnerability Analysis, Intrusion Detection, Alert Verification, Collaborative

## I. INTRODUCTION

In recent years, with the advances in networking technology applied in many areas, network communication has become an integral part of our daily life. Organizational dependence on networked information technology and its underlying infrastructure has grown explosively. However, the widespread reliance on network makes computer attacks launched over the network more devastating than ever before [11]. To guard against these malicious attacks, many hardware and software are designed, such as the firewall systems, the intrusion detection system (IDS), the virus protection software, the vulnerability scanning software and so on. Nevertheless, the isolated deploy of these devices can not guarantee the security of computer systems against all attacks. Moreover, employ of these devices often brings other problems such as alert flooding and false alerts.

With the advent of the Internet Era, the widespread reliance as well as CSCW (Computer Supported Cooperative Work) technology on network makes computer attacks launched over the Internet more devastating than ever before, systems administrators need more sophisticated tools to warn and direct their responses. Therefore, security mechanisms inevitably become an indispensable part of CSCW application. In order to get rid of those deficiencies, a new collaborative security paradigm, which integrates some security devices closely to provide accurate alerts and therefore achieve an enhanced protection is emerging. Intrusion detection is a domain in great need of collaboration support tools, which being an important and indispensable device as part of security mechanisms for network security.

Nowadays intrusion detection products have been widely deployed and have gained widely acceptance as a worthwhile investment. In a perfect world, IDS systems would be able to correctly detect and identify all the attacks without making any mistakes. Unfortunately, traditional IDS systems live up to meet the expectation of reliably identifying malicious activity. Moreover, one major challenge in current intrusion detection is that how to identify camouflaged intrusions more accurately from a huge amount of alerts [6]. The sheer number of IDS alerts can be overwhelming—current IDSs can trigger thousands of alarms per day, up to 99% of which are false negatives [25]. False positives correspond to those alerts generated against normal activities and system immune alerts which are attacks but not harmful to the target.

Traditional IDS system often run without any(or very limited) information of the network or host resource that they protect. This is why a great number of false alarms are so commonplace. Thus, only depending on single IDS, it is difficult to provide a comprehensive view for intrusion actions, especially for distributed and collaborative attacks. Then contextual information (also called background knowledge) including network topology, alert context (the configuration of the victim machine, e.g., operating system, open services, running application and etc) is quite important to identify successful intrusion attempts. Some researches have shown that most of the damages result from vulnerabilities existing

---

This work was part of the project — Unified Network Management Platform (200710421130) supported by a grant from Science and Technology Bureau of Wuhan Municipality in China.

on the network and hosts [13]. It is therefore demanding to apply vulnerability analysis to improve IDS for better network security.

As a matter of fact, in order to achieve high performance in detecting various intrusions, vulnerability-based alert verification we have proposed to achieve collaborative among various IDS systems as well as systematic modeling in respect to dynamic network environment is becoming crucial in winning against vicious cyber hackers. This paper presents a formalism and systematic methodology for vulnerability-based verification to identify successful intrusion attempts.

The rest of this paper is organized as follows. First of all, we explain the general issues about intrusion, vulnerability and verification in network security in Section 2. And in Section 3, a collaborative framework for vulnerability-based alert verification is proposed and thoroughly illustrated. While in Section 4, we further conclude our design goals of the collaborative model. Sequentially in Section 5, related work is briefly presented. Then we conclude and outline future work in Section 6.

## II. GENERAL ISSUES IN NETWORK SECURITY

This section presents the general issues in network security that we used to develop the framework presented in this paper.

### A. Vulnerability

A vulnerability is a fact about the system that, on the one hand, potentially enables some exploit to be carried out, and on the other, is the result of some exploit [1]. Whereas an attack is a specific program or method used to take advantage of the vulnerability to achieve its desired end. Vulnerabilities both in system and software have been the main contributing factor to the security problems which has been the attack exploited.

Vulnerability analysis, also known as vulnerability assessment, is a process that defines, identifies, and classifies the security holes (vulnerabilities) in a computer, network, or communications infrastructure [2]. In addition, vulnerability analysis can forecast the effectiveness of proposed countermeasures and evaluate their actual effectiveness after they are put into use.

### B. Intrusion

An intrusion refers to “any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource” [5]. Intrusion detection (ID) is defined as “the problem of identifying individuals who are using a computer system without authorization (i.e., crackers) and those who have legitimate access to the system but are abusing their privileges (i.e., insider threat)” [3]. That is to say, Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions [4].

Currently, there are two basic paradigms to detect an intrusion: anomaly detection and misuse detection. Anomaly detection is to record normal behaviors of the system and creates statistical profiles of the activities based on these

records. In this way, it can discover unknown forms of attack. This advantage is paid for in terms of the large number of false positives. If there are any differences coming forth between the current activities and the recorded ones with respect to a very dynamic environment, it would be labeled as an intrusion.

Compared with anomaly detection, misuse detection takes a complementary approach, which involves characterizing known ways to match and identify attacks. In this way, it definitely improves the accuracy of detecting the common known intrusion effectively. However, it often leads to many false negatives. This happens because this kind of detection can only detect those intrusions depicted in their signature repository. Table 1 presents the detailed comparison between anomaly detection and misuse detection [21].

TABLE I.  
COMPARISON BETWEEN ANOMALY DETECTION AND MISUSE DETECTION

Facets	Anomaly Detection	Misuse Detection
another name	behavioral detection	signature detection
established profile	normal behaviors	known attacks
detection method	activity deviation	pattern matching
advantage	unknown-attack discovery	known-attack detection
disadvantage	high false positives	high false negatives

### C. Verification

Verification is the act of reviewing, inspecting, testing to establish that a product, service, or system meets the regulatory, standard, or specification requirements.

Alert verification is defined as the process of verifying the success of attacks. That is, given an attack (and a corresponding alert raised by an IDS), it is the task of the alert verification process to determine whether this attack has succeeded or not. When the attack has not succeeded, the alert can be either suppressed or its priority reduced.

The foundation for alert verification is a coherent framework of intrusions as well as vulnerabilities that is rich enough to capture the behavior of various attacks.

## III. PROPOSED MODEL

An intrusion alert is a report that describes what malicious activity was identified. The information contained in an alert may change largely depending on the type of intrusion detection technique adopted. When an IDS system outputs an alert, there are three possibilities.

1. **true positive:** the IDS has correctly identified a successful attack. This alert is most likely relevant.

2. **non-relevant positive:** the IDS has correctly identified an attack, but the attack failed to meet its objectives.

3. **false positive:** the IDS incorrectly identified an event as an attack. The alert represents incorrect information.

Our design is a layer above intrusion detection using vulnerability analysis to alert verification for collaborative design. Intrusion detection produces basic alerts for further verification; whereas vulnerability analysis provides important information to help verificate the alerts (e.g. identify which type of alert it is). It is capable of promptly propagating genuine alerts (type-1) and reducing false alerts (type-3) in time.

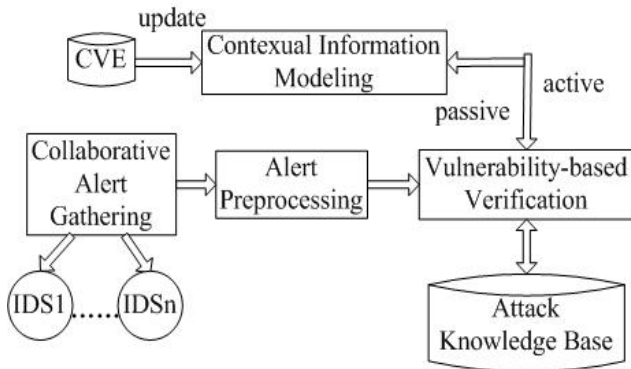


Figure 1. A collaborative framework of vulnerability-based alert verification

As shown in Fig.1, the collaborative framework consists of four main components: collaborative alert gathering, alert preprocessing, contextual information modeling and vulnerability-based verification. Each of the four parts is described in the following sections.

A. Collaborative Alert Gathering

An Intrusion Detection System (IDS) is a computer program that attempts to perform intrusion detection by either misuse or anomaly detection (as described in section 2.2), or a combination of these two techniques. The formats and contents of raw alerts generated directly by different IDS systems are application dependent. The alerts can be of different nature and level of granularity according to the detection technique and system type adopted. For example, they may represent network packets captured by network-based IDS, audit records produced by host-based IDS, or operation logs on specific application produced by application-based IDS.

The basic component of the framework is to collect alerts from various intrusion detection sensors and normalize them for later analysis. This is what collaborative alert gathering component performs. The whole process is realized and presented in Fig.2.

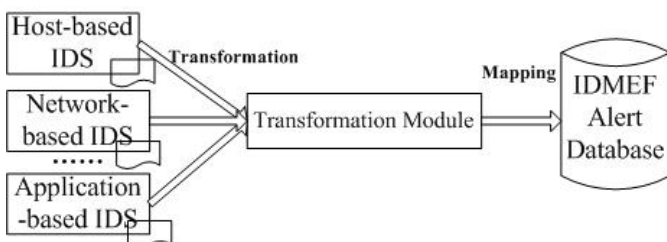


Figure 2. The process of collaborative alert gathering

Multiple IDS products (e.g., host-based IDS, network-based IDS, application-based IDS) send raw alerts to transformation module, which translates every alert that is received into a standardized representation. The format we choose is the Intrusion Detection Message Exchange Format (IDMEF) [12], which is emerging as an industry standard. IDMEF provides a standard representation for intrusion alerts. This representation defines the syntax of an alert and specifies the semantics of certain attributes.

However, the IDMEF effort is mostly concerned with syntactic rules. It is possible, and common, that different IDS systems choose different names for the same attack, provide incomplete information for certain fields. As is shown in Fig.2, the conversion of various alerts into the standard IDMEF format is performed in the transformation module. Abstract and concrete alerts emerging from various IDS systems can be expressed within the module.

Considering the great different alert outputs among multiple IDS (e.g. a Snort alert is different from a Bro alert in many aspects), a specific transformation module is designed for every IDS. The transformation module is then dedicated to that IDS product and converts its output into IDMEF format. If there is no specification for the content of most fields, they can be filled with meaningless strings (most commonly, “unknown” or “null” is used).

Whenever a new IDS is supposed to integrate into the system, we only need to provide a dedicated IDMEF transformation module to serve it. Therefore, the framework is highly adaptive.

B. Alert Preprocessing

A common limitation of existing intrusion detection systems is that they normally issue too many alerts. A lot of those alerts are duplicated in the sense that they are identical except that they are issued at a slightly different time, or issued by detectors installed at different locations [27]. It’s desirable to have as few EFFECTIVE alerts as possible when reporting the same ongoing attack. We use alert preprocessing to eliminate duplicate alerts and cluster similar ones.

Duplicate alerts may come from different IDS systems or the same ones. Whether the alerts are duplicate ones or not largely depend on certain attributes including source, target, port, classification and timestamp. For example, that two alerts generated by two network-based Intrusion Detection Systems contain an identical quadruple (source address, source port, target address, target port, classification) and timestamp attributes that are in a given time range, they are identified as duplicate ones. As a matter of fact, these alerts can be reduced as one with a count feature added in the attributes. That can be briefly demonstrated as  $[n * A] \Rightarrow B$  (there is an additional count attribute valued as n in B compared with A).

According to the similarity and dissimilarity among the attributes, alerts will be grouped into different clusters based on the principle of maximizing the intraclass similarity and minimizing the interclass similarity. For each cluster, a new alert that is representative of the information contained in the various alerts belonging to this cluster is created as a meta-alert to represent that cluster.

After the alert preprocessing process, the amount of alerts can be significantly reduced. Fig.3. shows the detailed algorithm of alert processing. In this case, the alerts are clustered according to the same classification and the count attribute for the duplicate alerts is added. Then the condensed and synthesized alerts containing the detailed information from all IDS systems are sent to the vulnerability-based verification component for further analysis.

```

global db:IDMEF_Alert;
    constant:time_window;
preprocess alerts
for all_alerts in IDMEF_Alert
    if(alert1.classification.equals(alert2_classification))
        &&(alert1.source_IP_address.equals(alert2.source_IP_address))
        &&(alert1.source_port_number==alert2.source_port_number)
        &&(alert1.target_IP_address.equals(alert2.target_IP_address))
        &&(alert1.target_port_number==alert2.target_port_number)
        &&(alert1.begintime-alert2.begintime)<=time_window)
        &&(alert1.endtime-alert2.endtime)<=time_window))
    identify duplicate alerts;
    alert1.count++;
    drop alert2;
    new cluster = alert1;

```

Figure 3. The algorithm of alert processing

### C. Contextual Information Modeling

Given an alert, it is possible to verify whether the target of the attack exists and whether a (potentially vulnerability) service is running. The classic example that Marty uses is the scenario of a Code Red 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 Code Red can only exploit vulnerabilities in Microsoft's IIS web server)[14]. A large network builds upon multiple platforms and diverse software packages and supports several modes of connectivity. Inevitably, such factors are quite important and should be considered further. When evaluating the security of a network, it is also important to consider the presence or absence of isolated vulnerabilities.

In order to objectively analyze the network security, analysis tools should be able to automatically establish the verification process based on the target network vulnerabilities, network services, connection relations and access authorization. As the number of system vulnerabilities multiplies in recent years, vulnerability analysis has emerged as a powerful system security administration tool that can identify vulnerabilities in existing systems before they are exploited.

To have an in-depth understanding of the network and host, vulnerability scanning tool is necessary to be applied. A vulnerability scanning tool performs scanning with its database in which vulnerabilities are specified. It is a program specifically designed to search a given target (piece of software, computer, network, etc.).

However, there is a drawback for using vulnerability scanner is that it may have an adverse effect on the hosts of the protected network. To some extent, it may consume network bandwidth and resources at the scanned host. To

minimize the impact, we should balance between gathering information and performing scanning.

Nessus is developed with Nessus Attack Scripting Language (NASL) for system administrators to scan vulnerability on these systems they maintain and manage [13]. By applying the real-time updating plugins in Nessus, Nessus can get the up-to-the-minute vulnerability information about the scanned target. Nessus was chose as a vulnerability analysis component because of the high quality of its vulnerability checks, its minimal impact on production networks, and the ease with which it could be integrated into IDS system.

The information-gathering process can check the current configuration of the victim host, or scan for attack traces .The contextual information (coming from vulnerability analysis) related to alerts may includes the host information (e.g. operating system, open ports, running services, vulnerability information), network information (e.g. connectivity, network traffic, topology), the attacker' ability and etc. Therefore, based on the information that alert verification requires to verify IDS alerts, the contextual information can be modeled in three aspects: host, connectivity and attacker.

The specific representation is described as follows.

**Host (H):** a set of hosts connected to the network.

Hosts are the main hubs within a network. They run services, process network requests, and maintain data. With rare exceptions, every action in an attack scenario will target a host in some way. Typically, an action takes advantage of vulnerable or mis-configured software to gain information or access privileges for the attacker [23]. The main goal in modeling hosts is to capture as much information as possible about components that may contribute to creating an exploitable vulnerability.

A host  $h \in H$  is a tuple  $(id, os, sv, app, cve)$ , where

*id* is a unique host identifier (typically, name and network address).

*os* is the operating system type and version that is configured and running on the host and the vulnerabilities that may exist. Most attacks are directed to specific system or server, such as Code Red, Syslogd.

*sv* is a list of service name/port number pairs describing each service that is active on the host and the port on which the service is listening. Services so often serve as a conduit for exploits. For instance, the alert of "WEB-IIS iissamples access" requires that an attempt is made to exploit a potential weakness on the victim running Microsoft Internet Information Server (IIS). When the victim's configuration shows that it is running an Apache Server on Linux, the exploit cannot possibly succeed.

*app* is a list of application software on the host, including installed software with exploitable security flaws (e.g., a *setuid* program with a buffer overflow problem), or mis-configured environment settings (e.g., existing user shell for system-only users, such as *ftp*). It is not enough to know which operating system the host is using, but it is also required to know which application is running and which patches have been applied.

*cve* is a list of host-specific vulnerabilities. Common Vulnerabilities and Exposures (CVE) [20] is a standard list of names for vulnerabilities and other information security exposures. A CVE identifier is an eight-digit string prefixed with the letters “CVE” (for accepted vulnerabilities) or “CAN” (for candidate vulnerabilities).

Generally speaking, vulnerabilities are categorized and sorted according to CVE. For each vulnerability reference, there is a corresponding security solution to facilitate security decision-making. Through the updating of vulnerability information referenced to CVE (Common Vulnerabilities and Exposures), some new intrusions can be captured and therefore can be defended in time.

Host information is static and isolated in nature, additional specification is needed to give to dynamic connectivity between the hosts. For remote attacks, it is also possible to check whether malicious packets can possibly reach the target, given the network topology and the firewall rule configuration. So we model it as connectivity relation.

**Connectivity (C):** a connectivity relation expressing the network topology and inter-host reachability [23].

Following Ritchey and Ammann [24], connectivity is expressed as a ternary relation  $C \subseteq H \times H \times P$ , where  $P$  is a set of integer port numbers.  $C(h1, h2, p)$  means that host  $h2$  is reachable from host  $h1$  on port  $p$ . By checking whether unknown ports accept connections on the host, verification system can get evidences of some attacks, such as installation of a back-door.

An attacker must have the corresponding privilege and information when he exploit some vulnerability, and will obtain the higher privilege and more information after the successful attack.

**Attacker (A):** According to this characteristic, we model *pvl* as the attacker’s ability on a specific host.

*pvl* refers to the privilege the attacker has gained with the intrusion going along. The privilege level required to launch the attack is also quite necessary to specify. According to [10], the privilege categories that are applied within this thesis are *remote access*, *local access*, *user access*, *root access*. There is a total order on the privilege levels: *remote access* < *local access* < *user access* < *root access*.

#### D. Vulnerability-based Verification

As is illustrated in section 2.2, one of the main weaknesses of existing IDS is that they often generate too many false alerts, be it false positives or non-relevant alerts, for various reasons. Excessive false positive alerts often cause alert flooding. Unfortunately, the flooding is prone to make system security operator miss the real vicious attacks. That is what false negative alerts refer to.

There are several reasons leading to that weakness. According to section 2.2, the IDS systems have their own drawbacks. The delay to enrich the signature sets and the difficulty to define the boundary between abnormal and normal activities are two significant ones. However, another important reason is attributable to the fact that current IDS systems are often unaware of the system context they are protecting.

To improve IDS performance, our proposed solution is the use of vulnerability-based verification. The process makes best use of system environment information provided from contextual information modeling to further analyze and optimize the alerts generated by IDS. The idea of the verification component is to discriminate between successful and failed intrusion attempts, then to remove alerts that do not represent successful attacks and response to the harmful intrusions in real time.

The most important requirement for the alert verification process is accuracy. The verification result of an attack can be accomplished by the attack knowledge base which is updating and adaptive with the alerts of IDS. It is established to describe various attacks in the form of precondition and effect. Every attack description has an identifier of AlertID, which is quite convenient to encode each alert to describe the necessary conditions and the possible consequence of the attack that the alert implies.

There remains a problem that what level of abstraction is appropriate for attack description in attack knowledge base. Too coarse or too fine granularity is not suitable. Though analyzing the relationships between alerts from Snort and CVE in the database, we find that (a) many attacks don’t correspond to any vulnerability, such as port scan, ping and etc. (b) an alert may refer to many CVE numbers or the combination of the CVEs. (c) a vulnerability may refer to many intrusions. For example, for “DELETED EXPLOIT bootp x86 linux overflow” attack, there is CVE-1999-0799, CVE-1999-0389, CVE-1999-0798 to be matched with it. And to CVE-1999-0798, there are “MISC bootp hardware address length overflow” and “MISC bootp invalid hardware type” corresponding to it. The approach of trying to find the relation between CVE and IDS name is not suitable and applicable.

In order to model the attack appropriately, we categorize the atomic attack into five different classes {Probe, Buffer Overflow, Compromises, DOS, Worm/Trojan horse}.

The detailed description is specified as follows.

**Probe** denotes pinging, probing and scanning, such as host or port scanning [22]. An attacker may probe to collect the list of valid IP addresses within a network, and the basic information about the system such as OS, the services it runs, the port it opens and so on.

**Buffer Overflow** denotes storing more data in a buffer than it was intended to hold. Since buffers are created to contain a finite amount of data, the extra information which has to go somewhere, may overflow into adjacent buffers, corrupting or overwriting the valid data held in them, or even trigger specific action to the computer. A buffer overflow attack exploits the fact that oftentimes programs do not check for boundary conditions in operations such as accessing arrays. This can be used to overwrite parts of the stack such as the return address and cause malicious code to be executed.

**Compromises** denote using known vulnerabilities such as buffer overflows and weak security to gain privileged access to hosts. There are two ways of compromises. One is **R2L** (Remote to Login), which refers to unauthorized access from a remote machine, such as guessing a password. The other is

**U2R** (User to Root), which means unauthorized access to local super user privileges, such as various buffer overflow attacks.

**DoS (Denial-of-Service)** attacks (e.g., smurf, land) usually attempt to shut down a network, computer, process and otherwise deny the use of resources or services to the authorized users.

**Worm/Trojan horse** is well-know to us, which can aggressively replicate on other hosts. The difference between worm and Trojan horse is that worms are self-replicating, while trojan horses are downloaded by users.

The instantiate process is accomplished after alert preprocessing. As illustrated in Fig.4., the meta-alerts are then to be mapped into a specific attack type depending on its own type. According to the atomic attack classification in the attack knowledge base, meta-alert can find its type. Appropriate verification is then triggered after that.

As regards for an attack, only if all of the conditions and steps have been satisfied can the attack be successful. As a matter of fact, combing with the system and vulnerability information provided from contextual information modeling module, the alerts are further identified.

Verification process can be performed using both passive and active techniques, and each requires different support from the vulnerability analysis.

before an alert is received (running in advance or a regular intervals), while active mechanisms perform verification in real-time, as a reaction to a received alert [14]. Both passive and active techniques are complementary to each other, helping verifying the alerts.

The key idea of alert verification is to distinguish between successful and failed intrusion attempts (both false and non-relevant positives). In the verification process, the alerts provided by alert preprocessing component with the vulnerability requirement information are represented and processed synthetically.

TABLE II.  
CONFIDENCE AND CORRESPONDING ACTION GIVEN TO ALERTS AFTER VERIFICATION

Confidence	Alert	Action
=0	false positive	drop
0<confidence<1	undetermined alert	postprocessing
=1	true positive	alarm

Based on the match degree of the instantiated alerts' conditions and the contextual information, a relevance score is attached to each alert. As is shown in Table II, the relevance score is defined as confidence ranging from 0 to 1, which shows the likelihood of alerts belonging to their assigned classes. Intuitively, confidence of the alerts represents the likelihood that the atomic attack will succeed. According to table 2, with referring to the confidence, certain defense actions are triggered by the rules to deter intrusions. The score 1 means an affirmation of the alert therefore the alert should be alarmed in time, so it is send to the real-time alarm engine to bring the administrator's attention, while 0 means a complete negative of the alert and the alert should be discarded, accordingly it is sent to the determined-false alert engine to be filtered out. If these alerts are not eliminated, not only precious bandwidth and computation resources would be wasted, but also the reactions of users towards genuine alerts would be disturbed[27]. When there may be some vulnerabilities existing but do not lead to attack in the present or no such appropriate precondition is found, the alert is flagged as undetermined and be given confidence between 0 and 1. They need to be sent to alert postprocessing component to be analyzed further.

In addition, there are also some alerts that do not correspond to any known vulnerability (e.g. those from a newly discovered attack that has not yet been included in the vulnerability database), in such circumstance, the alerts are also tagged as undetermined, then it is sent to alert postprocessing directly. That is what we will focus on in our future research.

The message exchanging and information sharing among the intrusion alerts and basic information in protected system and vulnerability-based verification methods enable us

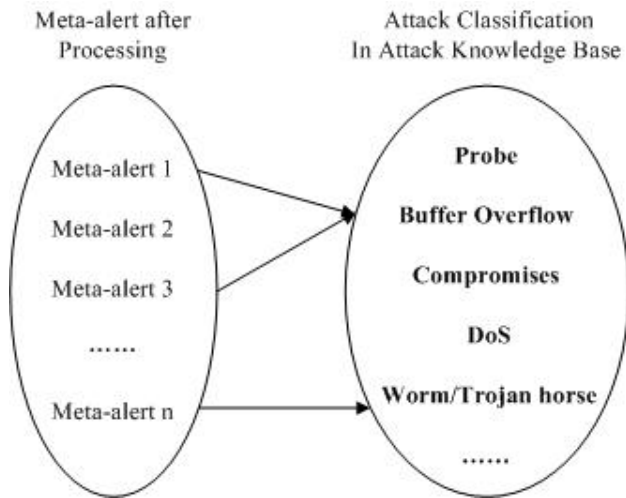


Figure 4. The mapping relation between Meta-relation and attack classification

Passive techniques depend on a priori information gathered about the hosts, the network, and the installed services. By relying on scanning all hosts and devices in the network periodically, it is possible to verify whether the target of the attack exists and whether a (potentially vulnerable) service is running. Active evaluation techniques need to look for evidence of the success of an attack by checking information at the victim machine. When an attack has been detected, a scanner can be used to check for the vulnerability that this attack attempts to exploit. It enables the vulnerability analysis engine to check whether unknown ports accept connections, which could be evidence that a backdoor has been installed. Passive mechanisms collect data about a protected network

efficiently and effectively evaluate generated alerts. Predefined defense rules can take actions based on the confidence, thus leading to automation of some defense process.

#### IV. DESIGN GOALS

To overcome limitations of traditional IDSs and enhance the efficiency of intrusion detection, we have provided a systematic and detailed explanation of our framework about alert verification.

As stated above, our framework is designed to achieve three major goals: (a) *timely*: reacting to security alerts as soon as possible; (b) *efficient*: filtering false positives as much as possible; and (c) *accurate*: analyzing uncertain and spiteful attacks as accurate as possible. To achieve these goals, we attach great importance to **collaboration** in the framework.

(a) collaborative among different IDSs (such as network-based IDS, host-based IDS as well as application-based IDS and etc). These relatively independent systems generate alerts based on their own knowledge base, such as the information from local hosts(e.g., Snort) or the information from local networks(e.g., Bro[26]). Accordingly, they generate lots of alerts promptly with different levels of precisions, and they even may generate false alerts. Collaboration work can integrate these alerts to standard alerts among IDSs.

(b) collaborative between intrusion alert from IDS systems and other security-relevant information mainly from vulnerability scanning tool. In fact, security products including firewall, network management systems and etc can provide useful information for alert verification. In this paper, we only focus on contextual information from vulnerability analysis. How to realize collaborative among all kinds of the security products is what we should make further research on.

#### V. RELATED WORK

Realizing the limitations of traditional IDS, researchers began to explore new ways to improve IDS performance. These years, some exciting and important advances have been made. Research results have been published and some prototype systems have been established.

Reference [6] illustrates a comprehensive approach to intrusion detection alert correlation, presents a general correlation model including a set of specific components. At the same time, they have applied a tool to a larger number of various data sets to analyze how each component contributes to the overall correlation process. Reference [7] implements an Aggregation and Correlation Console (ACC) based on the Tivoli Enterprise Console (TEC). Reference [8] constructs attack scenarios by correlating alerts on the basis of prerequisites and consequences of attacks. C. Kruegel et al [14] [15] describes two techniques for alert verification: the active and the passive, which take a complementary approach of information gathering

The main objective of these studies is to provide an accurate and condensed alert report in the way of correlating and verifying alerts generated by IDS systems.

As regards to vulnerability analysis, although it is still in its infancy in the networked environment and Internet infrastructure analysis [9], several methods have also been proposed [1] [10]. Several vendors and researchers have proposed to include vulnerability analysis data when processing IDS alerts [17] [18] as well as improving network security [19]. The idea is to utilize previously gathered vulnerability information in the aspects of IDS alerts optimization and risk assessment.

To the best of our knowledge, all the above researchers focus on only one aspect of network security, either intrusion-detection optimization, or vulnerability-analysis representation. But no one has given a systematic approach to realize that. Our collaborative and systematic approach is an example to correlate results from multiple IDSs by integrating vulnerability analysis information for collaborative design. After the verification, confidence is assigned and the corresponding actions are triggered based on that confidence.

#### VI. CONCLUSIONS

A comprehensive view of the overall security of a network requires analysis not only of alerts on a single host, but must also take into account contextual information such as vulnerability, configuration and so on.

In this paper, we have presented the great need to integrate security devices for collaborative design. After analyzing the weaknesses in traditional IDS systems, we propose a systematic approach to enable correlation among multiple intrusion detection systems applying contextual information. The ultimate goal of the vulnerability-based verification is to make intrusion detection more accurate, more efficient, and easier to use by system administrators. By combining the strengths of the various IDS systems, vulnerability analysis can be a valuable adjunct to IDS systems, helping to verificate the alerts in the system environment.

Applying contextual information to IDS systems, it does improve IDS performance to some extent. It is able to react to ongoing network intrusions in time and filter those false positive alerts out effectively. This project is carried through step by step. As of this paper, we are deliberately implementing on the proposed architecture of four components we have discussed. However, after vulnerability-based verification process, how to better deal with alerts with confidence between 0 and 1 for postprocessing, much work still remains to be done. There are potential limitations with our method. However, we believe that the approach we have presented is an advance in network-vulnerability modeling and will ultimately help network security if implemented in a reasonable way. In the near future, we will make further research on this problem.

#### ACKNOWLEDGMENT

This work was part of the project — Unified Network Management Platform (200710421130) supported by a grant from Science and Technology Bureau of Wuhan Municipality.

We would like to thank the members in our group for their valuable comments.

#### REFERENCES

- [1] P Ammann, D Wijesekera, S Kaushik. Scalable. Graph-based Network Vulnerability Analysis[C]. Proc. of the 9th ACM Conference on Computer and Communications Security. 217-224. 2002.
- [2] [http://searchsecurity.techtarget.com/sDefinition/0,290660,sid14\\_gci1176511,00.html](http://searchsecurity.techtarget.com/sDefinition/0,290660,sid14_gci1176511,00.html).
- [3] B. Mukherjee, T.L. Heberlein, and K.N. Levitt. Network Intrusion detection. IEEE Network, 8(3): 26–41, May/ June 1994.
- [4] Aleksandar Lazarevic, Jaideep Srivastava, Vipin Kumar. Data Mining for Intrusion Detection, Tutorial at the Pacific-Asia Conference on Knowledge Discovery in Databases, Seoul, April 30, 2003.
- [5] R. Heady, G. Luger, A. Maccabe, and M. Servilla. The Architecture of a Network Level Intrusion Detection System. Technical report, University of New Mexico, Department of Computer Science, Aug 1990.
- [6] Fredrik Valeur, Giovanni Vigna, Christopher Kruegel, Richard A. Kemmerer. A Comprehensive Approach to Intrusion Detection Alert Correlation. IEEE Transactions on Dependable and Secure Computing, Vol. 1, No. 3: 146-169. 2004.
- [7] Herve Debar, Andreas Wespi. Aggregation and correlation of intrusion-detection alerts. The 4th Int'l Symp on Recent Advance in Intrusion Etection, UC Davis, 2001.
- [8] P. Ning, Y. Cui, S. Reeves, D. Xu. Techniques and Tools for Analyzing Intrusion Alerts. ACM Transactions on Information and System Security, 7(2), pp.274–318. 2004.
- [9] Guangzhi Qu, Jayaprakash, Ramkishore Modukuri, Salim Hariri, C.S Raghavendra. A Framework For Network Vulnerability Analysis. in the Proceedings of 1<sup>st</sup> IASTED International Conference on Communications, Internet, Information Technology (CIIT), St. Tomas, Virgin Islands, USA. 2002.
- [10] C Ramakrishnan, R Seka. Model-based Vulnerability Analysis of Computer Systems[C]. Proceedings of the 2nd International Workshop on Verification, Model Checking and Abstract Interpretation. 1998.
- [11] Jinqiao Yu, Y.V. Ramana Reddy, Sentil Selliah, Sumitra Reddy, Vijayanand Bharadwaj, Srinivas Kankanahalli. TRINETR: An architecture for collaborative intrusion detection and knowledge-based alert evaluation. Advanced Engineering Informatics 19(2): 93-101. 2005.
- [12] Intrusion detection message exchange message format (IDMEF), <http://tools.ietf.org/html/draft-ietf-idwg-idmef-xml-16#page-4>, 2006.
- [13] Yeu-Pong Lai, Po-Lun Hsia. Using the vulnerability information of computer systems to improve the network security. Compute Communications. 2007.3.
- [14] C. Kruegel, W. Robertson, and G. Vigna. Using Alert Verification to Identify Successful Intrusion Attempts. Practice in Information Processing and Communication (PIK), Vol. 27, No. 4, pp. 219-227. 2004.
- [15] C. Kruegel, W. Robertson. Alert Verification: Determining the Success of Intrusion Attempts[A]. Proceedings of the DIMVA'04, Dortmund[C]. Germany, pp. 25-38. 2004.
- [16] Shiping Chen, Dongyu Liu, Songqing Chen, Sushil Jajodia. V-COPS: A Vulnerability-Based Cooperative Alert Distribution System. ACSAC. pp. 43-56. 2006.
- [17] R. Gula. Correlating IDS Alerts with Vulnerability Information. Technical report, Tenable Network Security, 2002.
- [18] N. Desai. IDS Correlation of VA Data and IDS Alerts. <http://www.securityfocus.com/infocus/1708>. 2003.
- [19] Yeu-Pong Lai, Po-Lun Hsia. Using the vulnerability information of computer systems to improve the network security. Computer Communications. 2007.
- [20] Common Vulnerabilities and Exposures. <http://www.cve.mitre.org>.
- [21] Xuejiao Liu, Debao Xiao, Xin Zhuang. An Adaptive Architecture of Applying Vulnerability Analysis to IDS Alerts. Proceedings of International Conference on Advanced Infocomm Technology (ICAIT) July-Aug, 2008, Shenzhen, China. pp. 318-32. 2008.
- [22] S. J. Stolfo, W. Fan and W. Lee. Cost-Based modeling and evaluation for data mining with application to fraud and intrusion detection: Results from the JAM project. In Proceedings of the DARPA Information Survivability Conference, 2000.
- [23] Sheyner O. Scenario graphs and attack graphs [Ph.D. Thesis]. Pittsburgh: Carnegie Mellon University, 2004.
- [24] R.W. Ritchey and P. Ammann. Using model checking to analyze network vulnerabilities. In Proceedings of the IEEE Symposium on Security and Privacy, pp. 156–165, May 2001.
- [25] Julisch, K. & Dacier, M. Mining intrusion detection alarms for actionable knowledge. Proceedings of ACM Conference on Knowledge Discovery and Data Mining, pp. 366-375. 2002.
- [26] V. Paxson. Bro: a system for detecting network intruders in real time. In *Computer Networks*, volume 31 December 1999.
- [27] Shiping Chen, Dongyu Liu, Songqing Chen, and Sushil Jajodia. V-COPS: a Vulnerability-based Cooperative Alert Distribution System. Computer Security Applications Conference. ACSAC '06. pp. 43-56. 2006.

**Xuejiao Liu** is currently a PhD student in Institute of Network and Communication Technology of Huazhong Normal University, China. She serves as a project manager of the project—Unified Network Management Platform(200710421130). Her current research interests include network security, vulnerability analysis, where she published some papers addressing intrusion detection, alert verification.

**Debao Xiao** is currently a professor and supervisor of PhD Candidates in Huazhong Normal University, China. In 1993, he served as a senior visiting scholar in Multimedia Communications Research Laboratory (MCRLab) at University of Ottawa, Canada. His current research interests include computer network, network management, protocol engineering. He is the author/co-author of about 100 papers in refereed journals and conferences in the areas of communications technology and network management.

**Xi Peng** is currently a teacher and PhD student in Department of Computer Science of Huazhong Normal University, China. He received his Master of Engineering Degree of Huazhong Normal University in 2004. His current research interests include network management, network security.