

國立交通大學

資訊管理研究所

碩士論文

一個在雲端環境上的二階段入侵偵測合作機制

A Two-phase Collaborative Intrusion Detection Mechanism  
for Cloud Computing

研究生：陳光禹

指導教授：羅濟群教授

中華民國壹百年五月

一個在雲端環境上的二階段入侵偵測合作機制

A Two-phase Collaborative Intrusion Detection Mechanism for Cloud  
Computing

研究生：陳光禹

Student: Kuang-Yu Chen

指導教授：羅濟群

Advisor: Chi-Chun Lo



A Thesis  
Submitted to Institute of Information Management  
College of Management  
National Chiao Tung University  
in Partial Fulfillment of the Requirements  
for the Degree of  
Master of Science  
in  
Information Management  
June 2011  
Hsinchu, Taiwan, the Republic of China

中華民國 壹 百 年 五 月

# 一個在雲端環境上的二階段入侵偵測合作機制

研究生：陳光禹

指導教授：羅濟群 老師

國立交通大學資訊管理研究所

## 中文摘要

隨著雲端環境運算的進步，有許多相關的議題被熱烈討論，資訊安全是其中一項重要課題。本論文將專注於入侵攻擊的防範，並探討如何運用已建構在雲端中多個入侵偵測系統，使它們彼此合作成為一個可行方案。一個兩階段的合作機制被提出來加強雲端安全。第一階段是建構信譽管理模型，此模型被設計用來建立入侵偵測系統之間的信賴關係。它是由三個步驟的方法所構成，分別是傳送驗證訊息，鼓勵回應以及考慮信譽的遞移性。第二階段是協同合作，是利用系統之間彼此的信賴關係，來加強合作的品質；而這些信賴關係是在第一階段中被建立完成。第二階段有兩種協同合作方法，分別是警報關聯整合與攻擊徵狀的分享。入侵偵測系統能夠藉由系統間分享彼此的資訊，顯著的提升偵測的效能。最後，透過模擬結果分析，本機制在偵測系統對攻擊最敏感的情況下，平均偵測準確度98%，明顯高於不合作的情況(88%)或是其他學者提出的合作機制(90%)。

關鍵字： 入侵偵測系統、合作機制，信譽管理，雲端運算

# **A Two-phase Collaborative Intrusion Detection Mechanism for Cloud Computing**

Student: Kuang-Yu Chen

Advisor: Dr. Chi-Chun Lo

Institute of Information Management  
Nation Chiao Tung University

## **Abstract**

With the advent of cloud computing, a number of issues are discussed and among them, security is an important one. This thesis concentrates on intrusion detection. It studies how to apply the intrusion detection systems (IDS) in cloud and makes them cooperate with each other to provide a more secure solution. A two-phase collaborative mechanism is proposed to enhance the security in cloud. The first phase is constructing the trust management model. Such model is designed to establish the trustworthiness relationships between each IDS. It is contributed by three steps, sending test messages, encouraging replying, and considering the transitivity of trust. The second phase is collaborating. The trustworthiness between each system, derived at first phase, is used to strengthen the quality of collaboration. There are two ways to collaborate, alert correlation and symptoms sharing. An IDS can increase the performance obviously by sharing the information with each other. Eventually, with analyzing the simulation results, the average detection accuracy of IDSs in the proposed mechanism is 98% when the IDSs are sensitive to attacks. It is higher than the non-cooperation (88%) and the other proposal (90%).

Keyword: intrusion detection systems, collaboration, trust-based, cloud computing

## 致謝

能夠順利完成這份論文並且能夠完成碩士班的學業，首要感謝我的父母與弟妹，他們給了我一個溫暖的家庭，讓我在外衝刺學業時，不論遇到什麼挫折與困難，隨時都能夠有一個避風港，讓我拋開一切煩惱。

接著我要感謝我的指導教授羅濟群老師，交大兩年的師生情緣，老師帶給我的不是隻字片語能夠形容，感謝老師給予我在課業上的諄諄教誨，以及生活上的各種照顧，讓我在新竹的日子裡能夠從裡至外的蛻變。

再來要感謝實驗室裡的俊傑學長與志華學長，他們在課業上、計畫上的各種指導、幫助與討論，讓我能夠有機會不斷的汲取各種新知並且茁壯成長，同時了解做研究應有的態度與精神，使我能夠游刃有餘的面對各種問題。

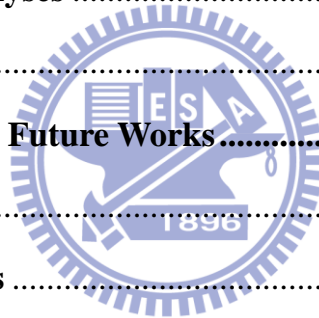
然後我還要感謝實驗室的學長姐—栩嘉學姊、鼎元學長、邦擘學長、湘婷學姊、元辰學長、世豪學長、致衡學長、冠儒學長與志健學長，以及同學們—秉賢、慕均、冠廷、哲豪、棉媛、孟儒、芳儀與靜蓉，他們在學業上與生活上帶給我的點點滴滴。當然也少不了學弟妹們—漢麟、彥似、淇奧、淳皓、御柔、佳蓁、雅晴、馨瑩、雅芬與媛如，他們帶給我各式各樣的歡樂氣氛，讓我在苦悶的研究生涯中依舊能保持愉快的心情。

最後，我要感謝我的女朋友—郁璇。沒有妳這本論文可能無法順利完成，沒有妳我的生活將失去色彩，沒有妳我的情感便會枯竭，沒有妳我的人生便不能稱做完滿，千言萬語的感謝也無法表達我心中的激動，真的十分感謝妳。

# Outline

中文摘要.....	i
Abstract.....	ii
致謝.....	iii
Outline.....	iv
Table List.....	vi
Figure List.....	vii
Notation.....	ix
Chapter 1 Introduction.....	1
1.1    Research Background and Motivation .....	1
1.2    Approach and Objective.....	3
1.3    Organization.....	3
Chapter 2 Related Works .....	4
2.1    Intrusion Detection Systems .....	4
2.2    Trust Management Model.....	7
2.2.1.  Test Message.....	7
2.2.2.  Incentive Design.....	7
2.2.3.  Trust Transitivity .....	8
2.3    Collaborative Properties .....	8
2.3.1.  Alert Correlation .....	9
2.3.2.  Symptoms Sharing .....	9
2.4    Cloud Computing Architecture .....	9
2.5    Virtual Machine Protection.....	12
Chapter 3 A Two-Phase Collaborative Intrusion Detection Mechanism ...	13
3.1    Problems in Collaborative Intrusion Detection Systems .....	13

3.2	<b>The Two-phase Collaborative Mechanism .....</b>	<b>14</b>
3.2.1.	<b>Phase 1: Constructing Trust Management Model.....</b>	<b>16</b>
3.2.2.	<b>Phase 2: Collaborating .....</b>	<b>20</b>
3.3	<b>Discussion .....</b>	<b>24</b>
3.3.1.	<b>Solutions for the Problems.....</b>	<b>24</b>
3.3.2.	<b>Constraints and Assumptions .....</b>	<b>25</b>
<b>Chapter 4 Simulation and Security Analyses .....</b>		<b>26</b>
4.1	<b>Simulation and Analyses .....</b>	<b>26</b>
4.1.1.	<b>Simulation Environment.....</b>	<b>26</b>
4.1.2.	<b>Simulation Results and Analysis .....</b>	<b>32</b>
4.2	<b>Security Analyses .....</b>	<b>49</b>
4.3	<b>Discussion .....</b>	<b>50</b>
<b>Chapter 5 Conclusion and Future Works.....</b>		<b>53</b>
5.1.	<b>Conclusion .....</b>	<b>53</b>
5.2.	<b>Future works .....</b>	<b>54</b>
<b>References .....</b>		<b>55</b>



# Table List

TABLE 1. TOP 10 OBSTACLES AND OPPORTUNITIES FOR GROWING CLOUD COMPUTING .....	2
TABLE 2. SPI SERVICES AND DELIVERY VENDORS .....	11
TABLE 3. FOUR TYPES OF CLOUD DEPLOYMENT MODELS .....	11
TABLE 4. PROBLEMS IN CIDS .....	13
TABLE 5. DIFFERENT DEFINITION OF TRUST .....	16
TABLE 6. AMOUNT OF SIMULATORS .....	27
TABLE 7. DIFFERENT TYPES OF SIGNATURE RULES IN EACH SIMULATOR .....	28
TABLE 8. DETAIL OF THE PROGRAMMING ENVIRONMENT .....	29
TABLE 9. STABLE VALUES OF PARAMETERS IN THE SIMULATION MODEL .....	30
TABLE 10. VALUES OF MULTIPLE PARAMETERS IN THE SIMULATION MODEL.....	31
TABLE 11. DETECTION ACCURACY IN THE HONEST ENVIRONMENT .....	51
TABLE 12. DETECTION ACCURACY IN THE DISHONEST ENVIRONMENT .....	52





# Figure List

FIGURE 1. ORGANIZATION OF A GENERALIZED IDS .....	5
FIGURE 2. CHARACTERISTICS OF IDSS .....	6
FIGURE 3. SPI EVOLUTION THROUGH VIRTUALIZATION .....	10
FIGURE 4. ARCHITECTURE OF PROTECTING HOST-BASED INTRUSION DETECTORS THROUGH VM.....	12
FIGURE 5. DISTRIBUTED CIDSS ARCHITECTURE [21] .....	15
FIGURE 6. TWO PHASES IN THE CORRELATION UNIT .....	15
FIGURE 7. SCENARIO OF ALERT CORRELATION .....	22
FIGURE 8. SCENARIO OF SYMPTOMS SHARING .....	23
FIGURE 9. DATA FLOW OF SIMULATION MODEL .....	27
FIGURE 10. AVERAGE TRUST VALUES FOR THREE TYPES OF SIMULATORS IN OSSEC ( $\gamma=1$ ).....	32
FIGURE 11. AVERAGE TRUST VALUES FOR THREE TYPES OF SIMULATORS IN SNORT ( $\gamma=1$ ) .....	33
FIGURE 12. AVERAGE TRUST VALUES FOR THREE TYPES OF SIMULATORS IN BRO ( $\gamma=1$ ).....	33
FIGURE 13. AVERAGE TRUST VALUES FOR THREE TYPES OF SIMULATORS IN OSSEC ( $\gamma=0.1$ ).....	33
FIGURE 14. AVERAGE TRUST VALUES FOR THREE TYPES OF SIMULATORS IN SNORT ( $\gamma=0.1$ ).....	34
FIGURE 15. AVERAGE TRUST VALUES FOR THREE TYPES OF SIMULATORS IN BRO ( $\gamma=0.1$ ).....	34
FIGURE 16. T AND AW CONVERGENCE CURVE WITH “DO NOT KNOW” RESPONSES .....	35
FIGURE 17. T CONVERGENCE CURVE WITH “DO NOT KNOW” RESPONSES (FUNG ET AL.) .....	35
FIGURE 18. TRUST VALUES FOR OTHERS IN OSSEC ( $\gamma=1$ , FUNG ET AL.).....	36
FIGURE 19. TRUST VALUES FOR OTHERS IN SNORT ( $\gamma=1$ , FUNG ET AL.).....	36
FIGURE 20. TRUST VALUES FOR OTHERS IN BRO ( $\gamma=1$ , FUNG ET AL.) .....	36
FIGURE 21. DETECTION ACCURACY UNDER DIFFERENT $PA$ AFTER LEARNING ( $th3=6$ ).....	37
FIGURE 22. DETECTION ACCURACY UNDER DIFFERENT $PA$ AFTER LEARNING ( $th3=6$ , FUNG ET AL.).....	38
FIGURE 23. DETECTION ACCURACY UNDER DIFFERENT $PA$ AFTER LEARNING ( $th3=3$ , FUNG ET AL.).....	38
FIGURE 24. DETECTION ACCURACY UNDER DIFFERENT $PA$ AFTER LEARNING ( $th3=1$ , FUNG ET AL.).....	38
FIGURE 25. DETECTION ACCURACY UNDER DIFFERENT $PA$ ( $th3=6$ , NON-COOPERATION).....	39
FIGURE 26. DETECTION ACCURACY UNDER DIFFERENT $PA$ ( $th3=3$ , NON-COOPERATION).....	39
FIGURE 27. DETECTION ACCURACY UNDER DIFFERENT $PA$ ( $th3=1$ , NON-COOPERATION).....	39
FIGURE 28. DETECTION ACCURACY UNDER DIFFERENT $PA$ WITHOUT LEARNING ( $th3=6$ ) .....	40
FIGURE 29. DETECTION ACCURACY UNDER DIFFERENT $PA$ WITHOUT LEARNING ( $th3=3$ ) .....	41
FIGURE 30. DETECTION ACCURACY UNDER DIFFERENT $PA$ WITHOUT LEARNING ( $th3=1$ ) .....	41
FIGURE 31. TRUST VALUES FOR THREE DISHONEST NODES IN OSSEC ( $Err=1$ , $\beta=0.01$ ).....	42
FIGURE 32. TRUST VALUES FOR THREE DISHONEST NODES IN SNORT ( $Err=1$ , $\beta=0.01$ ) .....	42
FIGURE 33. TRUST VALUES FOR THREE DISHONEST NODES IN BRO ( $Err=1$ , $\beta=0.01$ ).....	43
FIGURE 34. TRUST VALUES FOR THREE DISHONEST NODES IN OSSEC ( $Err=0.2$ , $\beta=0.01$ ).....	43
FIGURE 35. TRUST VALUES FOR THREE DISHONEST NODES IN SNORT ( $Err=0.2$ , $\beta=0.01$ ) .....	43
FIGURE 36. TRUST VALUES FOR THREE DISHONEST NODES IN BRO ( $Err=0.2$ , $\beta=0.01$ ).....	44

FIGURE 37. TRUST VALUES FOR THREE DISHONEST NODES IN OSSEC ( $Err=0.2, \beta=0.1$ )..... 44

FIGURE 38. TRUST VALUES FOR THREE DISHONEST NODES IN SNORT ( $Err=0.2, \beta=0.1$ )..... 44

FIGURE 39. TRUST VALUES FOR THREE DISHONEST NODES IN BRO ( $Err=0.2, \beta=0.1$ )..... 45

FIGURE 40. TRUST VALUES FOR THREE DISHONEST NODES IN OSSEC (FUNG ET AL.) ..... 45

FIGURE 41. TRUST VALUES FOR THREE DISHONEST NODES IN SNORT (FUNG ET AL.) ..... 46

FIGURE 42. TRUST VALUES FOR THREE DISHONEST NODES IN BRO (FUNG ET AL.) ..... 46

FIGURE 43. DETECTION ACCURACY UNDER DIFFERENT  $PA$  WITH 3 DISHONEST NODES ( $th3=6$ ) ..... 47

FIGURE 44. DETECTION ACCURACY UNDER DIFFERENT  $PA$  WITH 15 DISHONEST NODES ( $th3=6$ ) ..... 47

FIGURE 45. DETECTION ACCURACY UNDER DIFFERENT  $PA$  WITH 27 DISHONEST NODES ( $th3=6$ ) ..... 48

FIGURE 46. DETECTION ACCURACY UNDER DIFFERENT  $PA$  WITH 3 DISHONEST NODES ( $th3=3$ ) ..... 48

FIGURE 47. DETECTION ACCURACY UNDER DIFFERENT  $PA$  WITH 15 DISHONEST NODES ( $th3=3$ ) ..... 48

FIGURE 48. DETECTION ACCURACY UNDER DIFFERENT  $PA$  WITH 27 DISHONEST NODES ( $th3=3$ ) ..... 49



## Notation

- $j, i, h$  : Represents the IDSs and  $j \neq i \neq h$   
 $k$  : A serial number of a test message  
 $n$  : A serial number of the most current test message  
 $Sat_k^{j,i}$  : A satisfaction value of the  $k_{th}$  test message for node  $i$  evaluated at node  $j$   
 $AR_k^{i,j}$  : An actual response of the  $k_{th}$  test message from node  $i$  to node  $j$   
 $FR_k^j$  : A forecasted response of the  $k_{th}$  test message at node  $i$   
 $Err$  : A tolerable error range when evaluating  $Sat$   
 $tw_i^j$  : Preliminary trust value for node  $i$  evaluated at node  $j$   
 $T_i^j$  : Conclusive trust value for node  $i$  evaluated at node  $j$   
 $a_k$  : A type of described attack in the  $k_{th}$  test message  
 $aw_a^{j,i}$  : Preliminary ability value about  $a$  type of attack for node  $i$  evaluated at node  $j$   
 $AW_{a_k}^{j,i}$  : Conclusive ability value about  $a_k$  type of attack for node  $i$  evaluated at node  $j$   
 $F_k$  : A forgetting factor of the  $k_{th}$  test message  
 $d_k$  : A difficulty level of  $k_{th}$  test message  
 $T_{stranger}$  : A default trust value for new comer or inexperienced node  
 $AW_{stranger}$  : A default ability value for new comer or inexperienced node  
 $syp$  : A symptom  
 $R_j^{new}(syp)$  : An alert ranking of a symptom evaluated at node  $j$   
 $f_j(syp)$  : a occurring frequency of a symptom at node  $j$   
 $p$  : A probability of a node replying “do not know”  
 $\alpha$  : An amount of considered  $sat$

- $\beta$  : A weight for conclusive ability value when evaluating  $tw$
- $\gamma$  : A positive constant controlling severity of penalty when evaluating  $T$
- $\varphi$  : A positive constant controlling severity of penalty when evaluating  $AW$
- $\delta$  : An amount of consulted nodes about the trust value and ability value for the other node
- $\rho$  : An amount of consulted nodes about the alert ranking of a suspicious symptom.
- $th1_j$  : A threshold for trust value when consulting other nodes about trust value for the other node at node  $j$
- $th2_j$  : A threshold for trust value when consulting other nodes about ability value for the other node at node  $j$
- $th3_j$  : A threshold for occurring frequency of a symptom at node  $j$
- $th4_j$  : A threshold for trust value when consulting a symptom at node  $j$
- $th5_j$  : A threshold for the ranking of a symptom when adding a new symptom at node  $j$
- $th6_j$  : A threshold for trust value when adding a new symptom at node  $j$
- $N$  : Total amount of the simulators in the simulation model
- $P_A$  : A probability of occurring an attack
- $P_{TEST}$  : A probability of a simulator sending out test messages
- $P_{TSYP}$  : A probability of a simulator selecting a symptom from symptom table when sending out test messages
- $R_T$  : A true response an honest simulator should reply to another consulting node

# Chapter 1 Introduction

## 1.1 Research Background and Motivation

In recent years, there has been a dramatic proliferation of research concerned with cloud computing, which is the long-held dream of computing as a utility. Mell and Grance (2009) [13] points that “cloud computing is a model for enabling convenient and on-demand network access to a shared pool of configurable and reliable computing resources, which can be rapidly provisioned and released with minimal management effort or service provider interaction.”

Cloud computing has a huge potential to transform an immense part of information technology (IT) industry. With it, developers with innovative ideas for network services no longer require the large capital outlays (e.g. hardware, and human expense) to operate it. They have no necessary to concern about overprovisioning or underprovisioning for the services of which popularity does not match their prediction. Moreover, companies benefit from cloud computing by speeding up programs (e.g. large batch-oriented tasks or tasks with large data processing). As a result, cloud computing is becoming a very popular theme for blogs, white papers, and has been featured in the title of workshops, conferences, and even magazines and journals [3]. It's not inconceivable to anticipate that cloud computing becomes an indispensable infrastructure in our life such as internet.

Armbrust et al. (2009) [3] have noted that there are ten principal obstacles and opportunities for growing cloud computing. They are listed in Table 1. It is simple to recognize that almost all of these opportunities are fraught with security risks. Similarly obstacles cannot avoid security problems (e.g. data storage security, data transmission security, application security, and security related to third party

resources). And it is near impossible to clear up all the problems with one solution. Thus this thesis concentrates on defending intrusion attacks. It studies how to apply the intrusion detection systems (IDS) in cloud and makes them cooperate with each other to be a proper solution.

Table 1. Top 10 obstacles and opportunities for growing cloud computing

<i>Obstacle</i>	<i>Opportunity</i>
Availability / Business Continuity	Use multiple cloud providers to support business continuity; Use elasticity to protect from DDOS.
Data Lock-In	Standardize APIs; Make compatible software available to enable Surge or Hybrid computing
Data Confidentiality and Auditability	Deploy Encryption, VLANs, Firewalls Accommodate national laws via geographical data storage
Data Transfer Bottlenecks	FedExing disks, data backup / archival; Lower WAN router costs, and higher bandwidth LAN switches
Performance Unpredictability	Improved virtual machine support, and flash memory; Gang Scheduling virtual machines for applications
Scalable Storage	Invent scalable store
Bugs in Large Distributed Systems	Invent debugger that relies on distributed virtual machines
Scaling Quickly	Invent auto-scaler that relies on machine learning; Snapshots to encourage cloud computing conservationism
Reputation Fate Sharing	Offer reputation-guarding services like those for email
Software Licensing	Pay-for-use licenses Bulk use sales

Source: [3]

## 1.2 Approach and Objective

As a result of that intrusions over the internet are getting more sophisticated and dynamic with advancing in technology. This thesis combines and improves some proposals [1][9][12][22], to present a solution, a two-phase collaborative intrusion detection mechanism. It associates a trust management model and collaborative properties with IDSs for cloud computing. Such mechanism has the potential to resolve the weakness of individual IDS, since it is able to identify wide attacks and reduce false negative rate by aggregating and sharing evidence of attacks from multiple IDSs.

There are three critical issues in the two-phase collaborative mechanism. They are IDSs, trust management, and collaborative properties. An IDS collects data from the environment, and analyzes it. Consequently it reports to defenders while detecting malicious attacks. With trust management and collaborative properties, an IDS can consolidate other IDSs to promote the performance, and avoid being suffered from betrayal attacks and collusion attacks. With such mechanism, the IDSs in cloud can make the environment much securer than before.

## 1.3 Organization

The rest of this thesis is structured as follows. Chapter 2 contains the related works, including IDSs, trust management, collaborative properties, cloud computing, and virtual machine protection. Chapter 3 describes the proposed mechanism, composed of phase 1, constructing trust management model and phase 2, collaborating. The simulation and security analysis are presented in Chapter 4. Chapter 5 concludes this work and proposes some future works. Eventually, the references are attached at the end.

## Chapter 2 Related Works

Many previous attempts on intrusion detection systems (IDSs) had been focused on collecting data, analyzing data, and reporting to its defenders. In this chapter, IDSs, trust management model, as well as some of other IDSs collaborative properties developed for large scale networks will be introduced. In addition, both cloud computing and virtual machine (VM) protection are also included.

### 2.1 Intrusion Detection Systems

Briefly, the main purpose of IDSs is to detect misuse, unauthorized use, and abuse of computer systems by internal or external users [2]. An IDS could be any software, hardware, or combination of both, which monitors the activities of a given environment (e.g. a single system or networks), and determines whether they are malicious attacks or normal behaviors [16].

Typically, an IDS is that of burglar alarm with sensors, which can detect the potential or ongoing intrusions by comparing with policies, so the IDS can notify or alert the defender [10]. Wu and Banzhaf (2010) [17] have noted that there are many issues, such as data collection, data pre-processing, intrusion recognition, reporting, and response, in building an IDS. Among them, intrusion recognition is the most fundamental. And intrusion models have a huge impact on the performance of recognition. Automatically constructing intrusion detection models from data is significant. This is because intrusion detection faces the problems such as highly imbalanced data distribution, large network traffic, the difficulty of finding a way to figure out decision boundaries between normal and abnormal behaviors, and a demand for continuous adaption to a frequently changing environment. Figure 1 illustrates the organization of an IDS where solid lines indicate responses to intrusive



activities, while dashed lines indicate data / control flow.

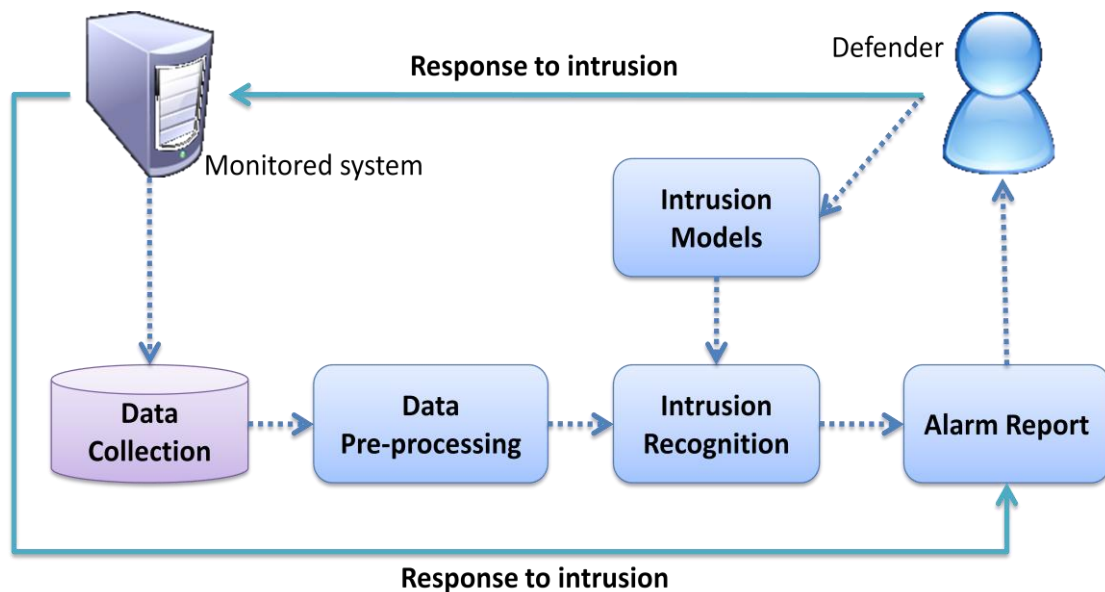


Figure 1. Organization of a generalized IDS

Source: [17]

Nowadays, IDSs are widely deployed for securing important IT-Infrastructures, and there are several characteristics can use to classify IDSs [17]. They are detection method, response to intrusion, audit data source and locus of detection, as shown in Figure 2.

Depending on the detection methods, IDSs are grouped into two detection principles, one is misuse-based (or signature-based) IDS and the other one is anomaly-based IDS. A misuse-based IDS recognizes intrusions by matching observed data with pre-defined descriptions of intrusions. On the contrary, an anomaly-based IDS does by matching data with normal behaviors, which are modeled before. According to the data source that an IDS collects, it can be either network-based IDS (NIDS) or host-based IDS (HIDS). A NIDS detects malicious activities by analyzing network flow, which is different from a HIDS that gathers information inside the host. Furthermore, the differences between distributed IDSs and central IDSs are the scale

of an intrusion recognition unit handling. In Distributed architecture, each IDS has its own intrusion recognition unit, while in central architecture, IDSs have the same one.

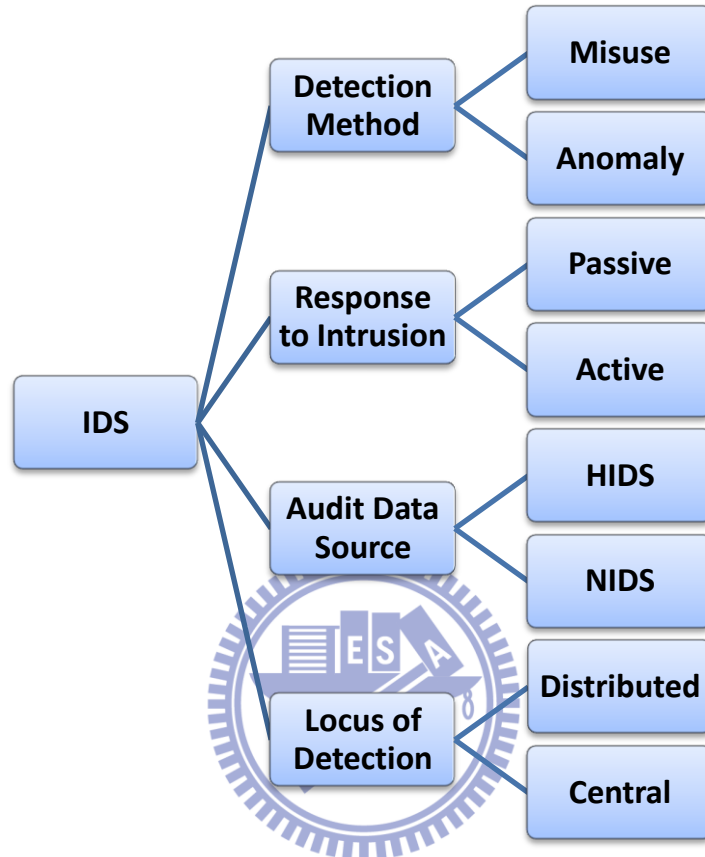


Figure 2. Characteristics of IDSs

Source: [17]

After a number of research, Elshoush and Osman (2010) [8] revealed that IDSs with collaborative approach are more powerful, and also provide better flexibility and performance over individual approach. However, Zhou and Morin et al. (2010, 2009) [20][14] pointed that several challenges, such as trust, alert correlation, system architecture, etc., in any collaborative intrusion detection system (CIDS) must be noticed. Such challenges will be discussed and solved by the proposed cooperative mechanism.

## **2.2 Trust Management Model**

As other distributed systems, trust management is a very important element in this work. Since the overall detection accuracy of an IDS stands on the correctness of the alert information, which is provided by other participating IDSs [20]. A trust management model establishes relationships between IDSs in the group. With such model, any IDS can figure out that whether another participating IDS is trustworthy or not. Fung et al. (2008) [9] proposed a robust and scalable trust model between IDSs in p2p network with two mechanisms, test message and incentive design.

### **2.2.1. Test Message**

This mechanism helps in recognizing inexperienced and malicious IDSs. Each IDS sends out either requests for consulting about suspicious activities, or test messages for examining others on trustworthiness. A test message has the same format as a real consulting request. And a test message is sent out in a way that the tested IDS are difficult to distinguish from a normal consulting request. The testing IDS knows the severity of the activity which is described in test message. By measuring the feedback and the original severity, it derives a satisfaction value, when sending out a test message each time. With such values, the testing IDS can evaluate a trust value for others

### **2.2.2. Incentive Design**

Incentive mechanism is designed to motivate cooperation. IDSs which are consulted about suspicious activities will not reply to all requesters in a period of time, because of the limited bandwidth or computational resources. Thus, it only replies to the IDSs with higher trust values. In such way, a participating IDS is encouraged to build up and maintain higher trust values held in other IDSs. In addition, an IDS is

allowed to reply “do not know.” Certainly, the frequency of replying “do not know” affects the trust value. If an IDS always replies “do not know,” its trust value is becoming the default value as a new comer.

### **2.2.3. Trust Transitivity**

Abdulai-Rahman and Hailes (1997) [1] pointed a view of trust in their proposal. There is a common assumption that trust is transitive. For instance, Bob trusts Mary, and Mary trusts Jenny. This represents that Bob trusts Jenny in a way. In many cases, it's not simple to verify a person (or an IDS) is trustworthy or not by oneself. With considering the transitivity of trust, people (or IDSs) can recognize others in a more comprehensive way.

Their proposal is not without its flaws. Fong et al. merely concern about HIDS. If there are a few types of IDSs, they will not believe others of different types as the same type. Moreover, if an IDS unable to reply a satisfying feedback, it will obtain the same severe penalty as the other lying deliberately. In addition, although trust transitivity is a common assumption, it is not generally true.

## **2.3 Collaborative Properties**

An IDS suffers from several problems such as that it may flag a large number of alerts every day, and it may miss certain attacks [14][18]. In recent years, several studies have worked on how to handle alarms. They aim to reduce the false alarms and try to provide a congruous response to attacks by comprehending the relationship between different alarms [23]. Alert correlation [9] and symptoms sharing [22] are proposed to overcome such challenges.

### **2.3.1. Alert Correlation**

According to Fung et al. (2008) [9], each IDS sends out consulting requests to others for alert correlation, while they are trustworthy. Moreover, besides trustworthiness, this research considers the physical distance between IDSs. Both of them affect the result of aggregation. The closer or more trustworthy one gains a higher weight on its feedback. Conversely, the longer or less trustworthy one acquires the lower weight. With such mechanism, IDSs can support each other either preventing unknown intrusions or allowing normal activities which are suspected.

### **2.3.2. Symptoms Sharing**

An algorithm is proposed by Zhou et al. (2007) [22] for sharing suspicious evidence (symptom) between IDSs in order to detect attacks at an early stage. Each one of them subscribes the symptoms collected from its own subnetwork (responsible domain). Then, they exchange all symptoms periodically. if the number of IDSs which subscribes a symptom is exceeds a threshold, each IDS obtains a notification alert about the symptom. In such way, an IDS can obtain symptoms of attacks while they are detected in other domains. Thus, it can avoid suffering from such attacks.

Similarly there are some drawbacks in both. The feedback of an able IDS is not strengthened when aggregating. This gives the chance of that other incorrect feedback compromise the correct ones. In addition, there is no trust management when sharing symptoms. Cooperating with deceitful IDSs is helpless even pernicious.

## **2.4 Cloud Computing Architecture**

Krutz and Vines (2010) [11] have noted that cloud computing architecture is still evolving, and it will continue evolving and changing. For some time now, there is a

classification upon which is generally agreed, has been coined the software-platform-infrastructure (SPI) model. Such model represents the three major services provided through the cloud, which are infrastructure-as-a service (IaaS), platform-as-a-service (PaaS), and software-as-a-service (SaaS). Figure 3 illustrates the progression of evolution from traditional datacenter to a full SPI framework with increasing the flexibility and lowering cost.

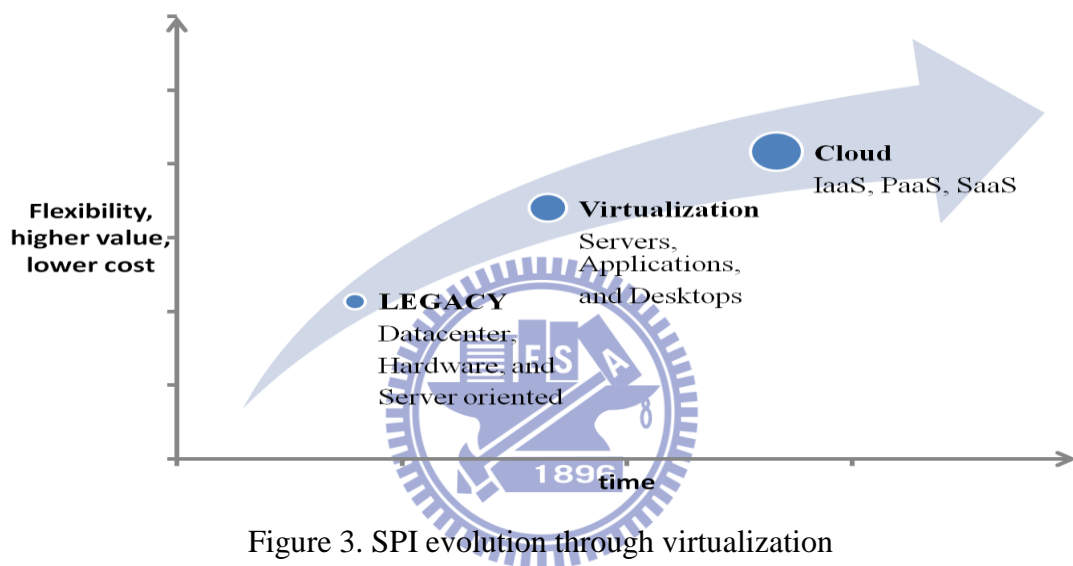


Figure 3. SPI evolution through virtualization

Source: [11]

Even though there are a few other concepts, SPI model is the most widely accepted classification in cloud computing. A number of famous companies (e.g. Amazon, Google and Microsoft) provide such services in the recent world. Table 2 lists the three major services paired with the vendors supplying for that layer.

Each of the three services can be deployed with several different models. Such deployment models are technically or functionally unrelated to the service model. It means that any of service models can exist in any of deployment models. According to Mell and Grance (2009) [13], there are four types, private cloud, community cloud, public cloud, and hybrid cloud listed in Table 3.

Table 2. SPI services and delivery vendors

<i>SPI Services</i>	<i>Descriptions</i>	<i>Vendor Example</i>
<b>SaaS</b>	Stateless cloud-enabled multiple-instance applications on a pay-per-use pricing model	Zoho Suite, Apple's MobileMe, Google Docs
<b>PaaS</b>	Application platform that provides developers with quick deployment	Google App Engine, force.com, Microsoft Azure
<b>IaaS</b>	Shared Internet infrastructure, such as servers and storage	Amazon EC2 and S3, Sun Microsystems Cloud Services, Terremark, Dropbox

Source: [11]

Table 3. Four types of cloud deployment models

<i>Models</i>	<i>Descriptions</i>
Private	This is operated singly for an organization.
Community	This is shared by several organizations and supports a specific community which has shared concerns.
Public	This is made available to the general public or a large industry group and is owned by an organization selling cloud services.
Hybrid	This is a composition of two or more clouds that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability

Source: [13]

Ultimately, all types of clouds were affected by a number of technology and architectural development over the past decades. With such causes (e.g. virtualization, open source software, universal connectivity, and excess capacity), they generate many influence (e.g. horizontal scaling, high-performance computing, utility and enterprise grid computing, and autonomic computing) on clouds [11]. The way to apply the characteristics of cloud is worth trying.

## 2.5 Virtual Machine Protection

Although in theory the infrastructure with VMs might be able to address most of security issues, in practice there are still plenty of security problems [4]. Because an attacker may have a chance to compromise the system through a VM, servers have to be protected from such intrusions. Laureano et al. (2007) [12], they proposed a way to monitor the VM through a VM monitor, which is illustrated in Figure 4.

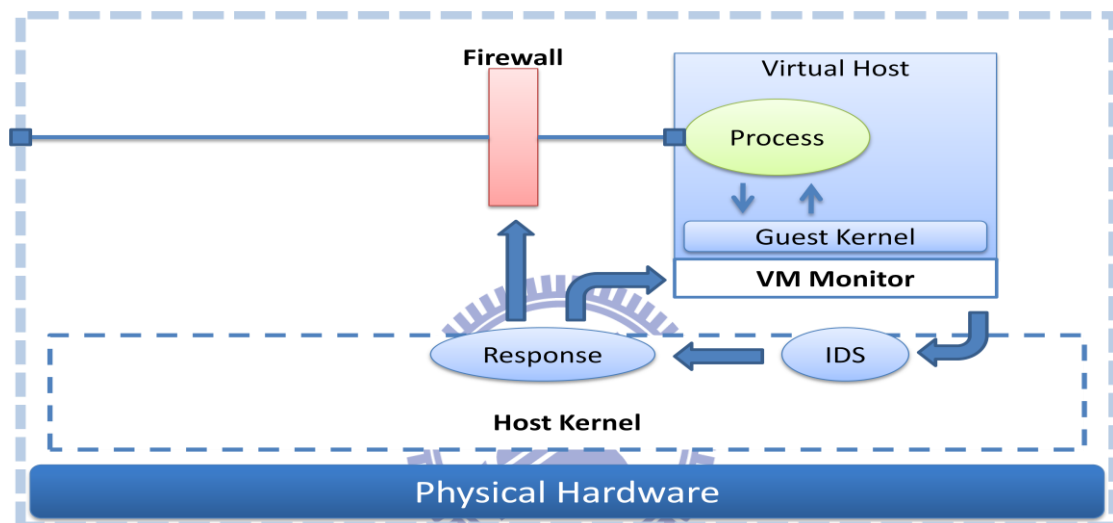


Figure 4. Architecture of Protecting Host-based Intrusion Detectors through VM

Source: [12]

Their proposal's main idea is to monitor a VM from outside. An IDS and the response unit are implemented outside the VM. This approach makes it easy to be implemented, because both IDS and response unit can be implemented as ordinary processes on the real host. But it needs to modify the code of VM monitor manually, or it has no capability to support this service originally.



## Chapter 3 A Two-Phase Collaborative Intrusion Detection

### Mechanism

A trust-based cooperation between IDSs is designed to raise the performance and reduce the possibility of being attacked in cloud. In this chapter, it will introduce the definition of problems, proposed work and discussions gradually.

### 3.1 Problems in Collaborative Intrusion Detection Systems

As mentioned in Chapter 2, IDSs with collaborative approach outperform the isolated ones. Despite this fact, there are a number of problems needed to solve. The problems in collaborative intrusion detection systems (CIDS) are listed in Table 4.

Table 4. Problems in CIDS

<i>Problems</i>	<i>Descriptions</i>
Security and Trust	Security and trust are significant factor for any CIDS, since the correctness of the alert information contributes the detection accuracy of the CIDS. A trustworthy partner can perform better than the deceitful one.
Alert Correlation	How the alerts from several individual IDSs are correlated advantageously is a critical issue. Detecting the network wide attacks efficiently and reducing the irrelevant alerts precisely is the main goal of a CIDS.
System Architecture	A CIDS is distributed intrusion detection system. The places of detection units and detection units influence the scalability and performance of the CIDS.
Trade-off	How to balance the trade-off problems (e.g. the trade-off between expressiveness of a correlation algorithm and corresponding computational complexity or between detection rate and false alarm rate) is worth discussed.

Source: [8][14]

Although other researchers provided several solutions [1][9][22] to solve such problems, they cannot without their flaws. For instance, IDSs treat different types of others with different faith. The severity of penalties for liars and other confused ones is the same. Even there is no trust management between each other. And it is weak that an IDS merely concentrates on its own domain as well as the load unbalance problem exists in distributed architecture.

### **3.2 The Two-phase Collaborative Mechanism**

In this thesis, a collaborative intrusion detection mechanism based on the negotiation results among IDSs is proposed for cloud computing. As mentioned in Chapter 2.4, virtualization is an important factor in cloud and VMs are certainly the protected targets. Each VM has a corresponding HIDS monitoring the virtual operating system kernel. And several NIDSs monitor the network flow in cloud. Both of them are in the distributed architecture. In such architecture, each IDS has its own detection unit and correlation unit. It is shown in Figure 5.

The first phase is constructing trust management model and the second phase is collaborating. Each correlation unit is composed of such two phases, as shown in Figure 6. Trust model is in order to establish the relationships between IDSs. There are several steps and mathematical equations inside. Hence, an IDS can evaluate trustworthiness for others. And collaborative properties are in order to apply such trustworthiness between each other to facilitate the cooperation. There are two ways to cooperate IDSs. One is alert correlation and the other one is symptoms sharing. These properties make IDSs detect more attacks with better accuracy. In the rest of this section, it will introduce the detail of the two phases, constructing trust management model and collaborating separately.

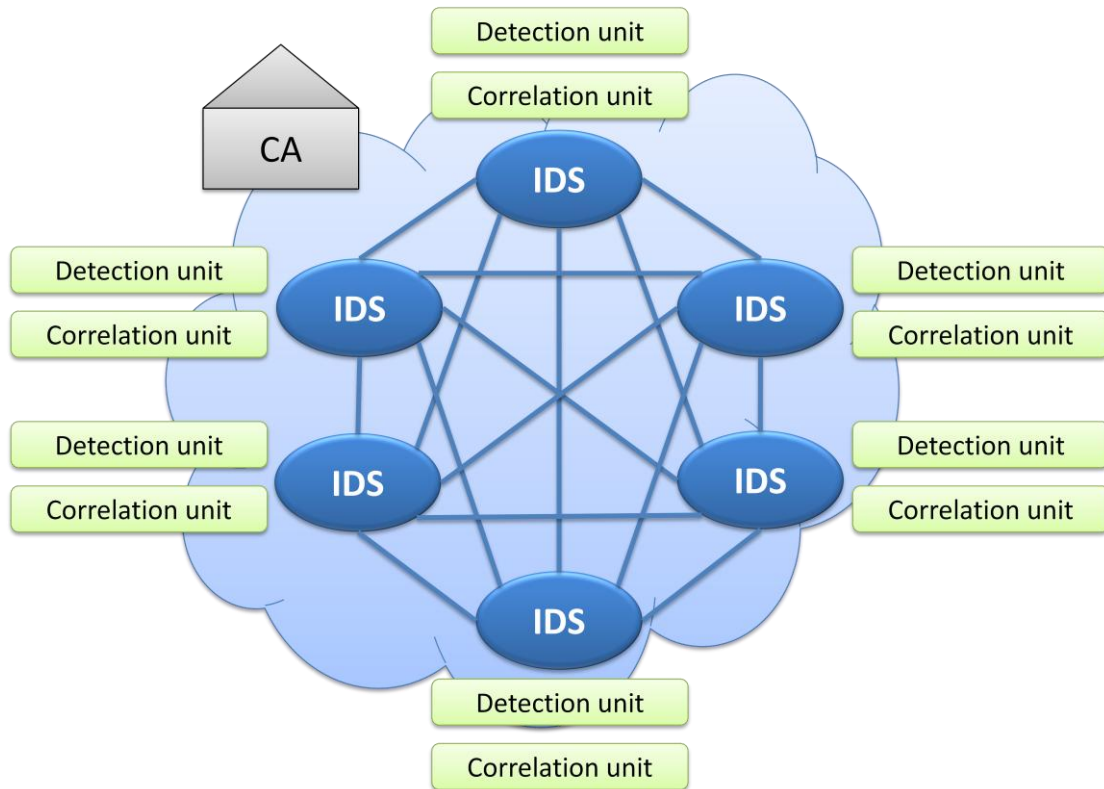


Figure 5. Distributed CIDSs architecture [21]

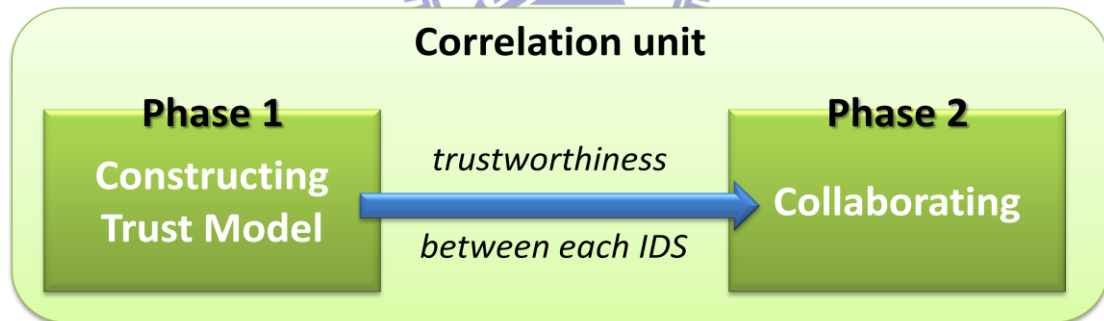




Figure 6. Two phases in the correlation unit

In addition, before the beginning of introducing the proposed mechanism, trust must be clarified in advance. In this thesis, trust is different from the one mentioned by Fung et al. (2008) [9]. The difference is illustrated in Table 5. In their proposal, honesty can represent trust fully. An IDS replies incorrect responses, signifying that it is must lying. But in this thesis, it is either lying or confusing, since it probably has no ability to reply precisely. It is unjustifiable that an IDS pays the same penalty for all

conditions. Thus, the trust in the following involves two essential factors, honesty and ability. Honesty is indiscernible and ability is appreciable.

Table 5. Different definition of trust

<i>Trust in Fung et al.</i>	<i>Trust in this thesis</i>
	

### 3.2.1. Phase 1: Constructing Trust Management Model

The trust management model is developed to establish trustworthiness relationships between IDSs. The model is composed of three gradual stages. They are (1) test message, (2) incentive design and (3) trust transitivity. With such model, any IDS can avoid the difference in believing different types of IDSs and give severe penalties for lying IDSs. Each of them can evaluate ability value and trust value for others. And the values will be updated when it goes through the stages each time.

#### (1) Test Message

Each IDS sends out either test messages or real consulting requests to others with the described activities in messages or requests. It sends out test messages randomly following Binomial Distribution [5]. Such test message is difficult to distinguish from a normal request. And a test message can be either fundamental or unusual. The former is effortless to reply satisfyingly. The latter is arduous to reply precisely. After an IDS receives the feedback from the replying ones, it will evaluate the satisfaction

values (*Sat*) for them by measuring the actual responses (*AR*) and forecasting response (*FR*). Both *AR* and *FR* are between 1 and 0. 1 represents that the described activity is very dangerous, and 0 represents it is very safe.

$$Sat_k^{j,i} = 1 - (|AR_k^{i,j} - FR_k^j|)^{\frac{1}{[|AR_k - FR_k|/Err]^{+1}}}, \quad 0 \leq Sat, AR, FR, Err \leq 1 \quad (1)$$

Equation (1) shows that how an IDS (node *j*) derives *Sat* for the other (node *i*). *k* is the serial number of test messages.  $Sat_k^{j,i}$  is the satisfaction value of  $k_{th}$  test message for node *i* evaluated at node *j*. It is in the interval between 1 and 0. 1 represents very satisfied, and 0 represents very unsatisfied.  $|AR_k^{i,j} - FR_k^j|$  denotes the error between *AR* from node *i* to *j* and *FR* at node *j*. *Err* is a tolerable error range. If the error is higher than *Err*, the  $Sat_k^{j,i}$  will drop dramatically. This can be perceived by observing the exponent,  $\frac{1}{[|AR_k - FR_k|/Err]^{+1}}$ . It controls the severity of penalty about  $Sat_k^{j,i}$ . When error is lower than *Err*, the exponent is equal to 1. However when error is higher than *Err*, the exponent is higher than 1.

In most of the time, an IDS receives the replies, and then derives *Sat*. With such values, it can evaluate both preliminary trust (*tw*) and ability (*aw*) value for the replying nodes.

$$tw_i^j = \frac{\sum_{k=n-\alpha}^n Sat_k^{j,i} (F_k - \beta AW_{a_k}^{j,i}) \frac{1}{d_k}}{\sum_{k=n-\alpha}^n (F_k - \beta AW_{a_k}^{j,i}) \frac{1}{d_k}}, \quad 0 \leq tw, F, AW, d \leq 1 \quad (2)$$

Equation (2) shows that how node *j* derives *tw* for node *i*. According to the time, all  $Sat_k^{j,i}$  are ordered from the most recent to the oldest. *n* is a serial number

of the most current test message.  $\alpha$  is a positive constant, controlling the amount of  $Sat^{j,i}$  to adjust  $tw_i^j$ .  $F_k$  is a forgetting factor [19] between 1 and 0, used to handle the possible changes in the behavior of IDSs over time. It gives less weight on the older  $Sat^{j,i}$ , and higher on more recent ones.  $a_k$  is a type of the attack described in the  $k_{th}$  test message.  $AW_{a_k}^{j,i}$  is the conclusive ability value at node  $j$ , indicating that node  $i$  is either good at the attacks of  $a_k$  or not. It matches onto the interval between 0 and 1 where 0 means the worst and 1 means the best.  $\beta$  is the positive constant affecting the influence of ability upon trust.  $d_k$  is the difficulty level of  $k_{th}$  test message. It is also between 0 and 1 where 0 means the simplest and 1 means the most difficult. Those simple test messages ( $\frac{1}{d_k}$  is closer to 1) have larger impact when evaluating  $tw$ .

$$aw_a^{j,i} = \frac{\sum_{k=n-\alpha}^n Sat_k^{j,i} F_k d_k}{\sum_{k=n-\alpha}^n F_k d_k}, \quad 0 \leq aw \leq 1 \quad (3)$$

Equation (3) shows that how node  $j$  derives  $aw$  for node  $i$ . The symbols are similar with Equation (2). But there are several differences in the content. Although  $Sat^{j,i}$  are ordered from the most recent to the oldest, all of them belong to one type of attacks,  $a$ . Thus, the most  $\alpha$  recent  $Sat^{j,i}$  are used in such equation. In addition, it is unnecessary to consider  $AW_a^{j,i}$  while deriving  $aw_a^{j,i}$ . In addition, the reason for using  $d_k$  instead of  $\frac{1}{d_k}$  is that difficult test messages ( $d_k$  is closer to 1) have larger impact when evaluating  $aw$ . It is opposite to Equation (2).

## (2) Incentive Design

In this thesis, it is possible that an IDS replies “do not know,” because it probably

has no experience or confidence with the ranking decision of such activities described in messages or requests. Or it may do this deliberately in order to maintain their conclusive trust value ( $T$ ) or ability value ( $AW$ ) without any contribution to others. For encouraging IDSs to provide satisfying responses whenever possible,  $T$  as well as  $AW$  is adjusted each time while it replies “do not know.”

$$T_i^j = (tw_i^j - T_{stranger})(1 - p)^\gamma + T_{stranger}^{1+p}, \quad 0 \leq T, p \leq 1 \quad (4)$$

$$AW_a^{j,i} = (aw_a^{j,i} - AW_{stranger})(1 - p)^\varphi + AW_{stranger}^{1+p} \quad (5)$$

Equation (4) and (5) shows that how node  $j$  handle “do not know” responses from node  $i$ . First of all, both equations are executed after evaluating  $tw_i^j$  and  $aw_a^{j,i}$ . Thus both values are used to adjust  $T_i^j$  and  $AW_a^{j,i}$ .  $T_{stranger}$  and  $AW_{stranger}$  are the default value for a new comer or an inexperience node.  $p$  is the percentage of that node  $i$  replied “do not know” to node  $j$  from the very beginning to present dividing the total amount of test messages between two nodes.  $\gamma$  and  $\varphi$  are positive constants, controlling either the severity of penalties or the increasing rate on  $T_i^j$  and  $AW_a^{j,i}$ . No matter what values  $tw$  and  $aw$  are in the original, all of them are approaching the square of the default values ( $T_{stranger}$  and  $AW_{stranger}$ ) when  $p$  is closing to 1. Moreover, it will approach more rapidly if  $\gamma$  or  $\varphi$  is higher. In addition, if  $tw$  or  $aw$  as well as  $\gamma$  or  $\varphi$  is lower than boundaries and the node keeping replying “do not know,”  $T$  and  $AW$  will become negative. In such of cases, it is regarded as 0.

### (3) Trust Transitivity

Conditional transitivity of trust is proposed to make an IDS provides trust value

or ability value for the other IDS to the consulting one. After following the two stages above, an IDS derives  $T$  and  $AW$  for others by itself. For more accuracy, it is better to take opinions of trustworthy nodes. But, it is not appropriate and practical to consult all others each time since it decreases the efficiency and increases the overload. Therefore, an IDS can decide thresholds for trust value and the amount of the consultants.

$$T_i^{newj} = \frac{\sum_{T_h^j \geq th1} T_h^j T_i^h + T_i^{oldj}}{\sum_{T_h^j \geq th1} T_h^j + 1}, \quad h = 1, 2, \dots, \delta, \quad \forall h \neq j \cap \forall h \neq i \quad (6)$$

$$AW_a^{newj,i} = \frac{\sum_{T_h^j \geq th2_j} T_h^j AW_a^{h,i} + AW_a^{oldj,i}}{\sum_{T_h^j \geq th2_j} T_h^j + T_j^j} \quad (7)$$

Equation (6) and (7) shows that how node  $j$  adjusts  $T_i^j$  and  $AW_a^{j,i}$  for node  $i$  by consulting others (node  $h$ ). All consultants are ordered by their  $T_h^j$  from the highest to the lowest. Node  $j$  can decide three thresholds,  $th1_j$ ,  $th2_j$ , and  $\delta$ .  $th1_j$  and  $th2_j$  set the boundaries for  $T_h^j$  when consulting  $T_i^j$  and  $AW_a^{j,i}$ .  $\delta$  is a positive constant, controlling the amount of consulting nodes. With multiplying the feedback ( $T_i^h$  or  $AW_a^{h,i}$ ) by the weight ( $T_h^j$ ), node  $j$  aggregates all opinions including its to derive new trust value ( $T_i^{newj}$ ) and ability value ( $AW_a^{newj,i}$ ). In addition, the weight ( $T_j^j$ ) on itself is 1.

### 3.2.2. Phase 2: Collaborating

In this thesis, there are two collaborative properties, (1) alert correlation and (2) symptoms sharing. They are proposed to take advantages of the trustworthiness



relationships between each IDS. As mentioned in the beginning of Chapter 3.2, both honesty and ability are also considered when correlating alerts. And it abandons the factor, physical distance, since it is insignificant in virtualization. On the other side, it adopts the trust management model when sharing symptoms. Thus, an IDS can avoid consulting malicious nodes and ignore the sharing requests from them for saving the computational resources and network bandwidth.

### (1) Alert Correlation

Each IDS in this mechanism consults others about suspicious activities, which cannot be ranked determinately. It demands the definite ranking of the activities by sending out consulting requests in which the symptoms of activities are described.

$$f_j(syp) \geq th3_j$$



(8)

Equation (8) shows that one criterion of a suspicious symptom at node  $j$ .  $f_j(syp)$  is the occurring frequency of the symptom ( $syp$ ) and  $th3_j$  is the threshold for that frequency at node  $j$ . If  $syp$  occurs more often than  $th3_j$ , it will be regarded as a suspicious symptom. And the other criterion is that any symptom recorded in symptom table is suspicious as well. In addition, there is a similar problem that it is inefficient to consult all IDSs in the group. Hence, the trust values for consulted IDSs and the amount of them are limited when sending requests.

$$R_j(syp) = \frac{\sum_{T_i^j \geq th4_j} T_i^j R_i(syp) AW_{a_{syp}}^{j,i}}{\sum_{T_i^j \geq th4_j} T_i^j AW_{a_{syp}}^{j,i}}, \quad 0 \leq R \leq 1, \quad i = 1, 2, \dots, \rho \quad (9)$$

Equation (9) shows that how node  $j$  aggregates feedback from other nodes

(node  $i$ ).  $R_j(syp)$  is the ranking of a suspicious symptom detected by node  $j$ . Similarly as the other equations before, all consultants are ordered by their  $T_i^j$  from the highest to the lowest. Node  $j$  can decide two thresholds,  $th4_j$  and  $\rho$ .  $th4_j$  sets the boundary for  $T_i^j$  in consulting  $R_j(syp)$ , and  $\rho$  is a positive constant, controlling the amount of consulting nodes. With multiplying the feedback ( $R_i(syp)$ ) by the weight ( $T_i^j$  and  $AW_{a_{syp}}^{j,i}$ ), it aggregates all opinions to derive new ranking. Such scenario is illustrated in Figure 7.

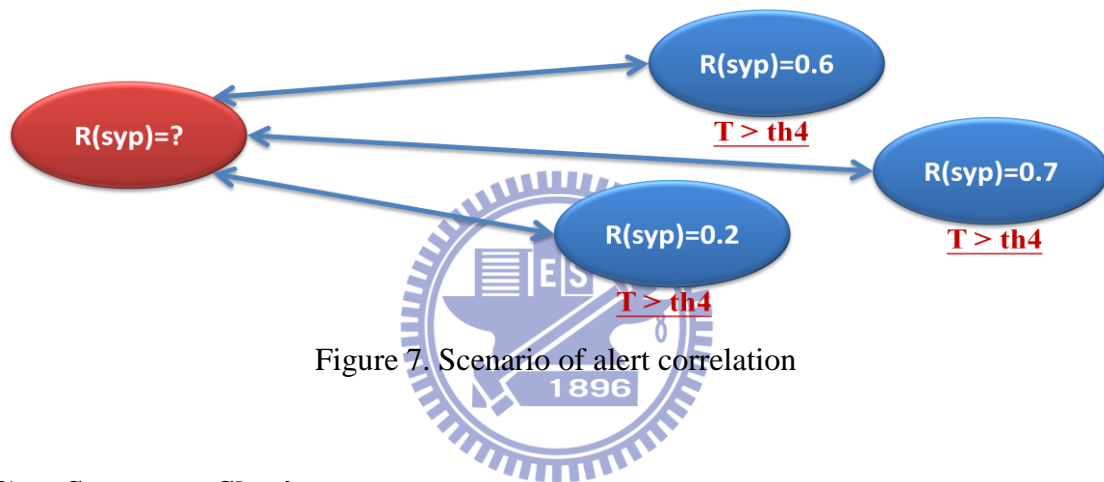


Figure 7. Scenario of alert correlation

## (2) Symptoms Sharing

Each IDS in this mechanism shares the symptoms of malicious attacks with each other. Thus, other IDSs can detect such attacks in the early stage. But, there is a difficulty in this property, the balance between efficiency and completeness. If IDSs share all symptoms to each other, it makes sure that each IDS obtains the whole suspicious symptoms. But it also increases the loading to network and computing components.

Therefore, in this thesis, IDSs merely share symptoms conditionally in two conditions. First, an IDS observes a suspicious symptom itself, and then consults others as mention in alert correlation. After ascertaining the symptom is dangerous, the IDS updates its symptom table. Second, an IDS is consulted by the other one with

a symptom, but the consulted one has no information about the symptom. Then it adds the symptom without its ranking to symptom table. Both conditions are based on the source IDSs are trustworthy..

$$R_j(syp) \geq th5_j \tag{10}$$

$$T_i^j \geq th6_j \tag{11}$$

Equation (10) shows one criterion of that how node  $j$  makes a decision to add the symptom to symptom table in first condition.  $th5_j$  is the threshold for the ranking of a symptom. After correlating, node  $j$  derives the ranking ( $R_j(syp)$ ) and then it compares  $R_j(syp)$  with  $th5_j$ . If  $R_j(syp)$  is higher, it adds  $syp$  to symptom table. The reason for this is that  $syp$  is safe when  $R_j(syp)$  is too low. It is unnecessary to record safe symptom automatically in signature-based IDS.

Equation (11) shows that the other criterion of node  $j$  adding  $syp$  to symptom table in second condition.  $th6_j$  is the threshold for trust value. Node  $j$  adds the symptom from node  $i$  while  $T_i^j$  is higher than  $th6_j$ . In such way, it can avoid adding incorrect symptoms as possible. Both scenarios are illustrated in Figure 8.

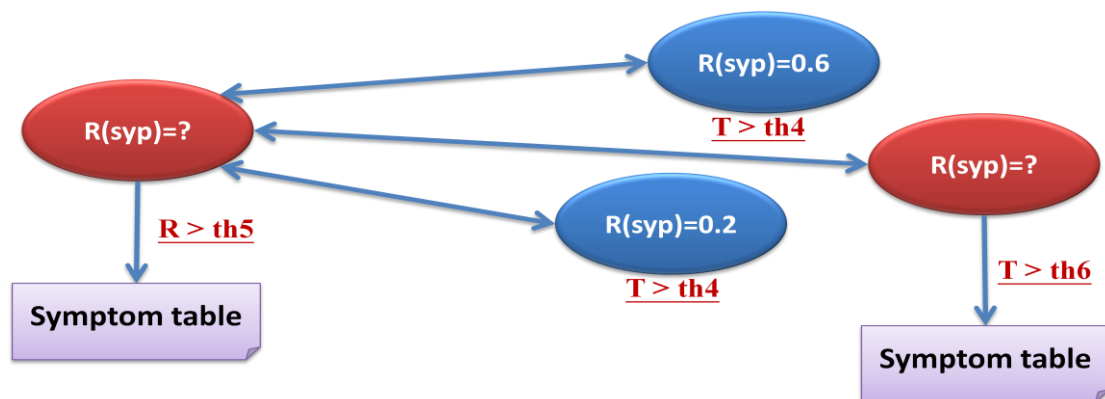


Figure 8. Scenario of symptoms sharing

### 3.3 Discussion

The proposed collaborative intrusion detection mechanism aims at solving the problems in CIDS. Moreover, in order to achieve this goal, there are several constraints and assumptions. All the discussions are in the following.

#### 3.3.1. Solutions for the Problems

There are four aspects can be discussed. First the proposed mechanism enhances the trust management model proposed by Fung et al [9]. With considering ability and honesty in trust, it gives different penalties for IDSs with different ability. If an IDS replies unsatisfying feedback, it will receive severe penalty when it is good at such type of attacks. Since this represents that the IDS has a higher probability being a liar. In addition, with considering the transitivity of trust, each IDS obtains trust value and ability value for an IDS from others. Thus, an IDS can know the other one is malicious even it performs well, because it merely lies to others. The IDS can prevent the malicious one in an early stage. On the contrary, if an IDS merely lies to the other one, this cannot be discovered easily by the IDS in the fraud. The reason is that the trust value for dishonest IDS is compromised by the high value feedback of others. And there is no appropriate mechanism restrains the fake responses when consulting others about trust values or ability values.

Second the proposed mechanism combines and modifies two collaborative properties. With considering ability in alert correlation, it offers more weights on the feedback of IDSs with better ability. Thus, their opinions will have less chance being compromised by other IDSs with lower ability even malicious IDSs. With sharing symptoms with each IDS, an IDS can detect some attacks even it have no information about them. But it merely accepts the sharing from trustworthy IDSs. This can avoid adding incorrect symptoms.

Third the proposed mechanism is built in cloud computing. With the virtualization, horizontal computing and high performance of cloud, CIDS can overcome the limitation, which is balancing the loading for different IDSs with different correlation units in the distributed architecture. In such way, it can not only avoid the single point failure problem in centralize CIDS but also overcome the load balancing difficulty in distributed CIDS.

Fourth the proposed mechanism tries to balance some trade-off problems between efficiency and effectiveness by considering the thresholds mentioned above. With those thresholds, it has no necessary to consult or accept all other participating IDSs. It can save the network bandwidth and computational resources but relinquish little accuracy or completeness.

### **3.3.2. Constraints and Assumptions**

There are several constraints and assumptions in the proposed mechanism. First for simplifying the mechanism, all participating IDSs are signature-based IDSs. Only the administrators can edit the white list including safe symptoms. And the cloud merely has a VM control node. It represents that all VMs are in the same group. Second the participating IDSs have no communication with others outside the group. They purely exchange information with the IDSs having registered to CA. Third it assumes that the existence of a function  $G$ , which normalizes the cognition of all symptoms for different IDSs. It maps all alert ranking onto the interval between 0 and 1 where 0 denotes benign activities and 1 indicates dangerous symptoms. Forth there is an acknowledged information format for transferring messages between IDSs and it should be encrypted. IDMEF probably is an appropriate solution for such assumption [7]. Fifth each IDS knows all the types of attacks. Thus, an IDS can calculate the ability value of all types of attack for others.

## Chapter 4 Simulation and Security Analyses

This chapter presents the simulation and analysis at first. It is composed of simulation environment and results analysis. And then security analysis shows how the proposed mechanism defends some common threats. Moreover, there is a discussion in the end of this chapter.

### 4.1 Simulation and Analyses

This section describes the simulation environment and the results analysis. Simulation environment contains programming environment, assumptions, definition of cases and performance metrics. And then each metric in different cases are analyzed.

#### 4.1.1. Simulation Environment

There four parts in this section. It describes how the simulation is designed and the programming environment behind. And it also introduces the assumptions and the definition of cases and performance metrics.

##### (1) Simulation Design

Totally there are  $N$  nodes and an event generator in the simulation model. Each node represents an IDS simulator. The event generator sends fictitious events to all simulators. A fictitious event is contributed by a symptom, which can be either dangerous or safe. And the simulators are simulated from three open source IDSs, OSSEC, snort, and bro by using authentic signature rules. They are widely used to protect network servers or workstations from intrusions in recent years. The amounts of simulators are listed in Table 6.

Table 6. Amount of simulators

	<i>OSSEC</i>	<i>snort</i>	<i>bro</i>	<i>Total (N)</i>
<b>Amount</b>	14	8	8	30

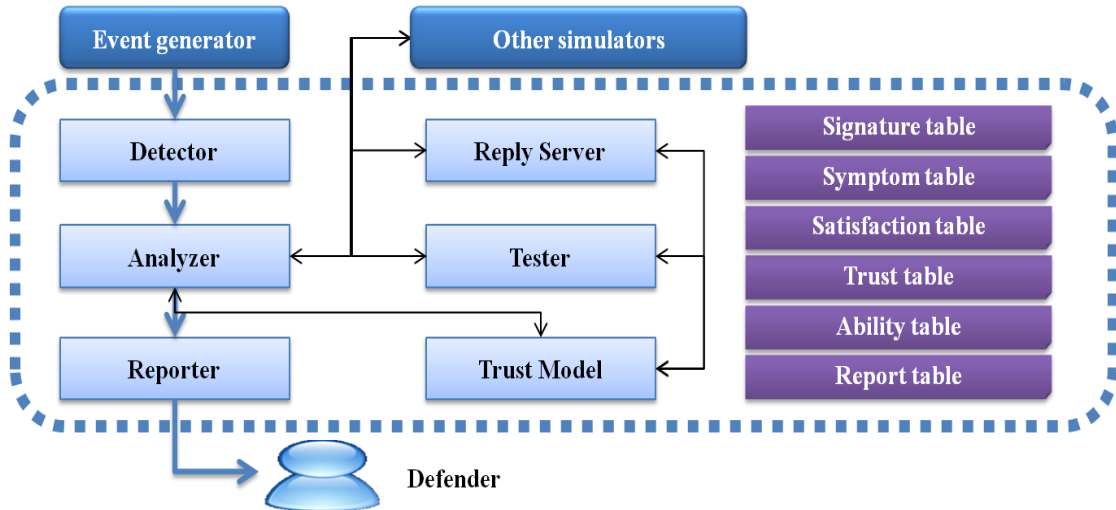


Figure 9. Data flow of simulation model

Each simulator is composed of six functional modules and its own database with several tables. First module is detector, which detects events from the simulation environment. Second module is analyzer, which analyzes the symptoms of events having been detected by detector. It sends out consulting requests to other simulators when meeting unknown symptoms. Third module is reporter, reporting the malicious events to its defenders. Forth module is reply server, replying the consult requests of other simulators. Fifth module is tester, testing other simulators by sending out test messages in which known symptoms described. It selects symptoms from signature table or symptom table. Sixth module is trust model, evaluating trust values and ability values for other simulators. The data flow of such simulation model is illustrated in Figure 9. The thick lines indicate that the data is unidirectional, and the slim lines indicate that it is bidirectional.

Furthermore, there are signature table, symptom table, satisfaction table, trust

table, ability table, and report table in the database. All the signature rules are in the signature table. Symptom table stores the suspicious symptoms which may cause threats. The satisfaction values for all other simulators are stored in satisfaction table. Trust table and ability table record trust values and ability values for other simulators. In addition, one simulator has one trust value, but it has several ability values corresponding to different types of attacks. Eventually, Report table records the reporting logs.

Table 7. Different types of signature rules in each simulator

<i>Simulators</i>	<i>Types of Signature Rules</i>
<b>OSSEC</b>	apache, arpswatch, asterisk, attack, cimserver, cisco-ios, courier, dovecot, firewall, ftpd, hordeimp, ids, impad, mailsniffer, mcafee_av, msauth, mysql, named, netscreenfw, nginx, ossec, pam, php, pix, policy, postfix, postgresql, proftpd, pure-ftpd, raccoon, roundcube, sendmail, smbd, solaris_bsm, sonicwall, spamd, squid, sshd, symantec-av, symantec-ws, syslog, telnetd, trend-osce, vmpop3d, vmware, vpn_concentrator, vpopmail, vsftpd, web, wordpress, zeus, blacklist, whitelist
<b>snort</b>	<u>Preproc</u> , attack-responses, <u>backdoor</u> , blacklist, botnet-cnc, chat, dns, dos, exploit, finger, ftp, icmp-info, imap, misc, multimedia, mysql, <u>netbios</u> , nntp, oracle, p2p, policy, pop3, <u>rpc</u> , rservices, scada, scan, shellcode, smtp, <u>specific-threats</u> , <u>spyware-put</u> , sql, tftp, voip, <u>web-activex</u> , web-cgi, web-client, web-iis, web-misc, web-php, x11, dpd, whitelist
<b>bro</b>	dpd, <u>ex.web-rules</u> , http-bots, snort-default, <u>ssl-worm</u> , <u>worm</u> , blacklist, whitelist

Source: [24][25][26]

Table 7 shows that different simulator has different types of rules. Although simulators have several the same types of rules, it does not represent that they have the same ability on the types. According to the actual rules of each type in the IDSs,



the underline ones show the type that simulators are better at than others in the same type of IDSs (HIDS or NIDS). OSSEC is the only one type of HIDS in the simulation model, so there is no special type of rules.

## (2) Programming Environment

All the simulators and the event generators are programmed with C language. And MySQL database is adopted. The operating system (OS) is Ubuntu 10.04 Long-term Support (LTS) 64 bit. And the virtual machine monitor is Xen 4.0.1. All the programs are in Dom 0. The detail of such programming environment is listed in Table 8.

Table 8. Detail of the programming environment

<i>Programming Environment</i>	
Programming language	C
Compiler	gcc
Database	MySQL 5.5.12
OS	Ubuntu 10.04 LTS 64bit
VM monitor	Xen 4.0.1
VM kernel	Linux 2.6.32.32

## (3) Simulation Assumption

For making the simulation not too sophisticated, there are several assumptions in this simulation model. First is that all dangerous events are contributed by the malicious symptoms defined in the signature rules. It represents that a dangerous event can be known by one type of simulators at least. In this thesis, there are 96 different types of rules and it assumes that each type had 5 malicious symptoms except blacklist and whitelist. There were 27 symptoms in blacklist and 3 in whitelist. Thus the total amount of malicious symptoms is 500. In addition, there are 4,500

normal symptoms for safe events.

Second no matter the events are dangerous or safe, they will be detected when sent to the simulators. A simulator will not miss any events but it may allow a dangerous event since it has no rules about it. And each simulator can merely detect the event in its domain. A simulator is unable to know what events are sent to others. In addition, the ranking of rules inside the simulators corresponding to malicious symptoms are in the interval between 0.8 and 1 randomly.

Third some values of parameters in the simulation model are listed in Table 9. They are stable in the all situations.  $P_{TEST}$  is the probability of a simulator sending a test message and  $P_{TSYP}$  is the probability of that when sending a test message the symptoms is selected from symptom table. Others are mentioned before.

Table 9. Stable values of parameters in the simulation model

<i>Parameters</i>	<i>Values</i>	<i>Parameters</i>	<i>Values</i>
$P_{TEST}$	0.01	$T_{stranger}$	0.5
$P_{TSYP}$	0.1	$AW_{stranger}$	0.5
$\alpha$	10	$th1$	0.5
$\varphi$	0.01	$th2$	0.5
$\delta$	2	$th4$	0.5
$\rho$	6	$th5$	0.8
$F_k$	0.1, 0.2, ... 1	$th6$	0.8

#### (4) Definition of Cases and Performance Metrics

In such simulation two performance metrics are observed in two main cases. Both two cases and performance metrics are introduced in the following.

- **Cases**

Case 1 is honest environment representing that all simulators are honest. They follow the steps as mentioned above and reply the requests from others without fraud.

The other case is dishonest environment representing that a number of dishonest simulators are in such environment. In order to decrease the performance of others, the dishonest nodes send no test messages or requests to others and they reply the contrary responses ( $1-R_T$ ) to the consulting nodes.  $R_T$  is the true response.

In addition, with different values of multiple parameters, both main cases can be divided into several small cases. And three mechanisms, the proposed mechanism, proposal of Fung et al. and non-cooperation, are deployed to be compared in such cases. The multiple parameters are listed in Table 10.  $P_A$  is the probability of generating a dangerous event. Others are also mentioned before.

Table 10. Values of multiple parameters in the simulation model

<i>Parameters</i>	<i>Values</i>
$P_A$	0.2, 0.5, 0.8
$Err$	0.2, 1
$\beta$	0.01, 0.1
$\gamma$	0.01, 0.1, 1
$th3$	1, 3, 6

- **Performance Metrics**

Trust values and detection accuracy are observed in this simulation model. The trust values are defined in the above. The trend of the trust values in different cases shows the performance of the trust management model. And detection accuracy is defined in Equation (12). By observing detection accuracy, it is plain to perceive the performance of the collaborative mechanism is effective or not. Each performance metric in each main case is analyzed in the following.

$$Detection\ Accuracy = \frac{Total\ amount\ of\ reported\ attacks}{Total\ amount\ of\ occurred\ attacks} \quad (12)$$

#### 4.1.2. Simulation Results and Analysis

In this section, each performance of the proposed mechanism in two main cases (e.g. honest and dishonest environment) is presented and analyzed in the following.

##### (1) Honest Environment (Case 1)

There are two performance metrics will be analyzed in such case. They are trust value and detection accuracy and the results of them are in the following.

- Trust value

In the proposed mechanism, with fixing the multiple parameters ( $P_A=0.5$ ,  $\beta=0.01$ ,  $th3=3$ ,  $Err=1$ ) and multi values of  $\gamma$ , the trust values for other simulators in three different types of nodes (OSSEC, snort and bro) are illustrated as follows.

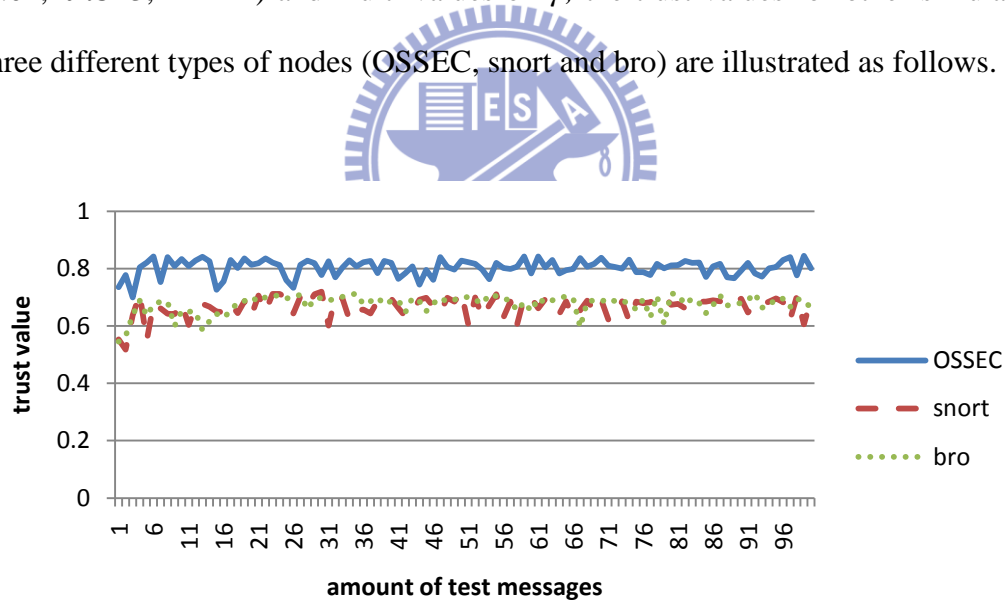


Figure 10. Average trust values for three types of simulators in OSSEC ( $\gamma=1$ )

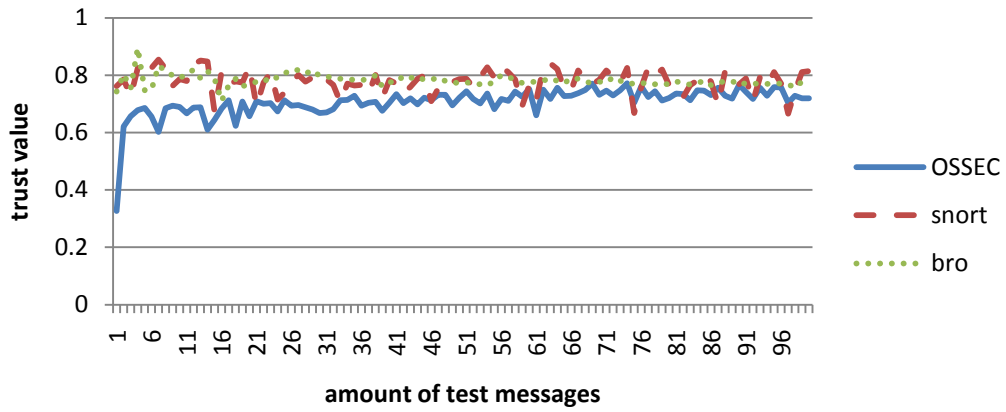


Figure 11. Average trust values for three types of simulators in snort ( $\gamma=1$ )

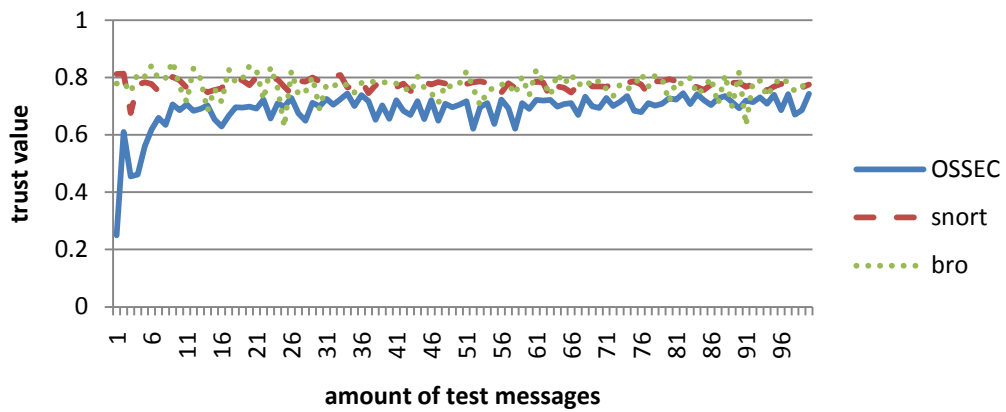


Figure 12. Average trust values for three types of simulators in bro ( $\gamma=1$ )

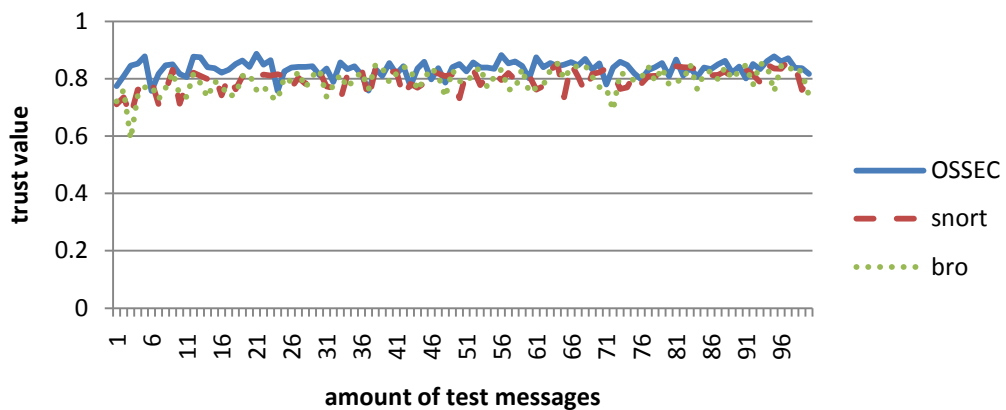


Figure 13. Average trust values for three types of simulators in OSSEC ( $\gamma=0.1$ )

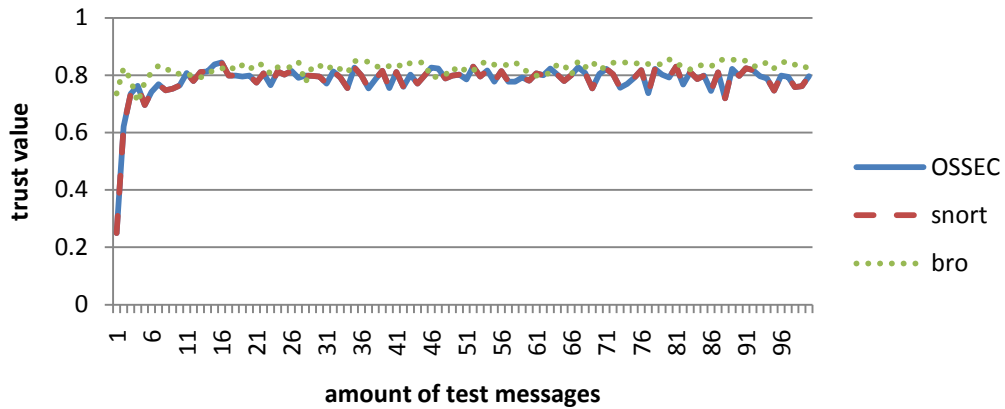


Figure 14. Average trust values for three types of simulators in snort ( $\gamma=0.1$ )

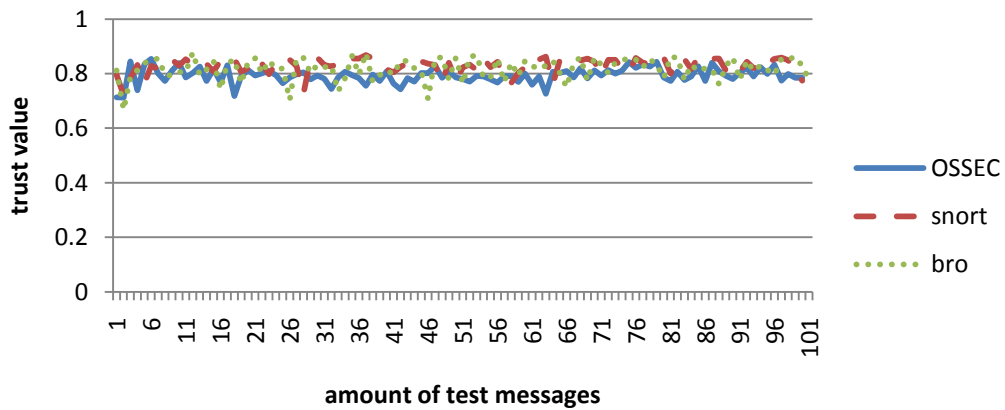


Figure 15. Average trust values for three types of simulators in bro ( $\gamma=0.1$ )

After observing the curve of the average trust values among 100 test messages, it shows that with the decreasing in  $\gamma$ , trust values for three types of simulators are gathering together at 0.8. But decreasing  $\gamma$  represents that it mitigates penalty when a node replying “do not know.” In addition, the same  $\gamma$  causes different penalties about the trust value in different mechanism (the proposed one and Fung et al.), shown in Figure 16 and Figure 17. With ensuring the same and enough severity of penalty about replying “do not know,” the values of  $\gamma$  are chosen to be 0.1 and 1 in this mechanism and Fung et al. separately. Both conditions show that the trust values are approaching to 0.5 from 1 linearly.

Compared to the results in Fung et al. in the following, the trust values for different types of simulators are clustered into two groups. The group of same type with the observing node gains higher trust values than the different ones. On the contrary, the proposed mechanism eliminates the difference of trustworthiness between each node. Thus all simulators will not only consult the same types of nodes but also the different types. It can avoid forming some separate groups and make different types of simulators assist each other. In such way, collaboration can be applied among all simulators.

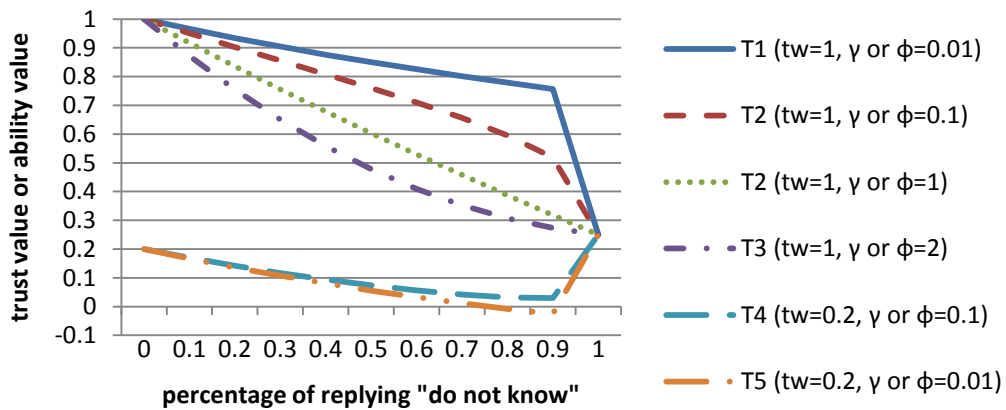


Figure 16. T and AW convergence curve with “do not know” responses

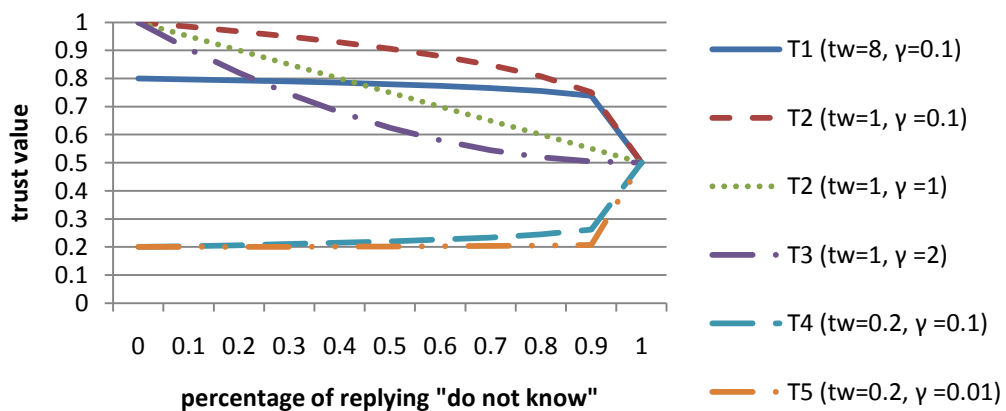


Figure 17. T convergence curve with “do not know” responses (Fung et al.)

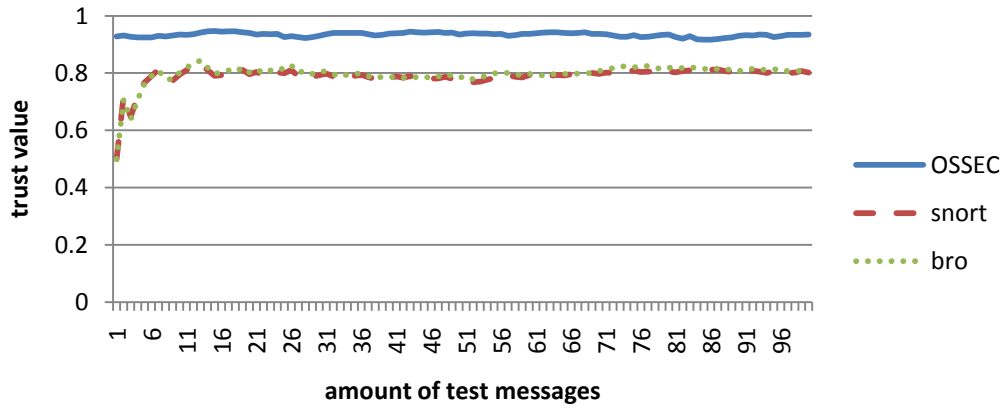


Figure 18. Trust values for others in OSSEC ( $\gamma=1$ , Fung et al.)

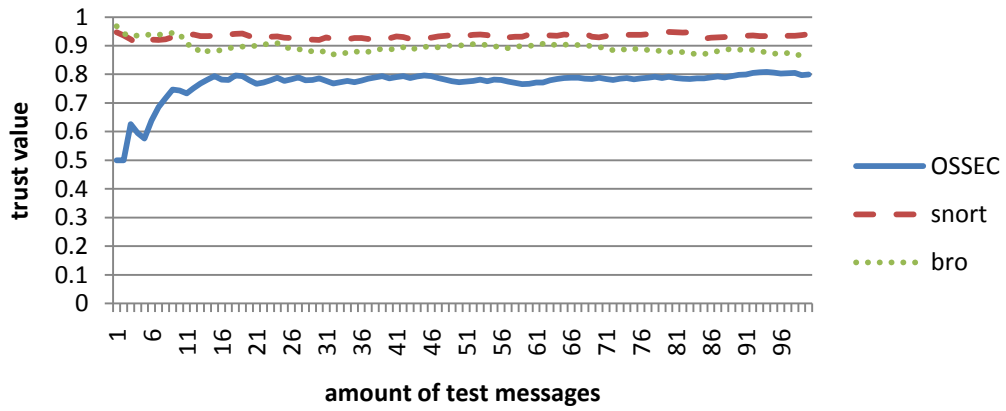


Figure 19. Trust values for others in snort ( $\gamma=1$ , Fung et al.)

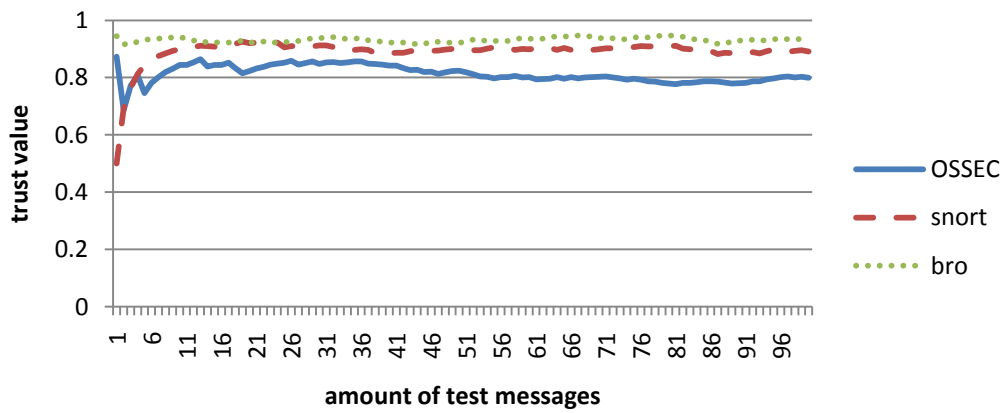


Figure 20. Trust values for others in bro ( $\gamma=1$ , Fung et al.)



If keep decreasing the value of  $\gamma$ , the trust values in both mechanisms will gather together. But it nearly relinquishes the adoption on the trust value when a node replying “do not know” responses. E.g. the trust value for a node will not be lower than 0.8 until the percentage of replying “do not know” is over 90% when  $\gamma=0.01$  in this mechanism or  $\gamma=0.1$  in Fung et al. Thus a node can maintain its high trust value without any contribution to others by keeping replying “do not know.” So it is not proposed to set the value of  $\gamma$  too low, even though this can help eliminating the difference of trustworthiness between each node.

- **Detection Accuracy**

In the proposed mechanism, with fixing the multiple parameters ( $\gamma=0.1$ ,  $\beta=0.1$ ,  $Err=0.2$ ) and the multi values of  $P_A$  and  $th3$ , detection accuracy is observed in the three different types of nodes, after the nodes detecting 5,000 events in each. In addition all nodes passed through learning phase, which represents that the nodes send 100 test messages to each other before detecting events. The results in both the proposed mechanism and Fung et al. are illustrated in the following figures.

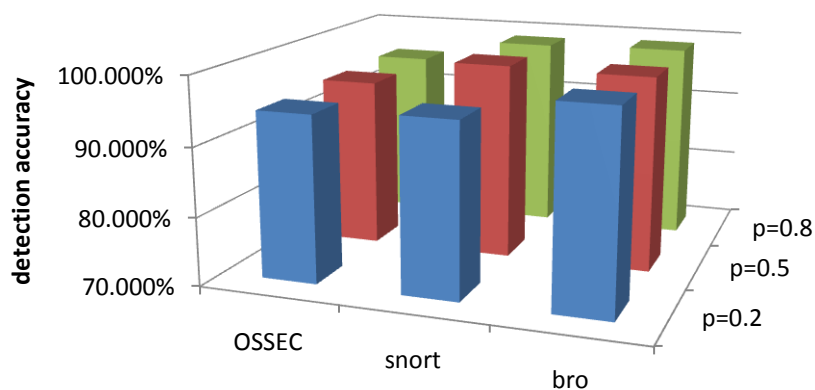


Figure 21. Detection accuracy under different  $P_A$  after learning ( $th3=6$ )

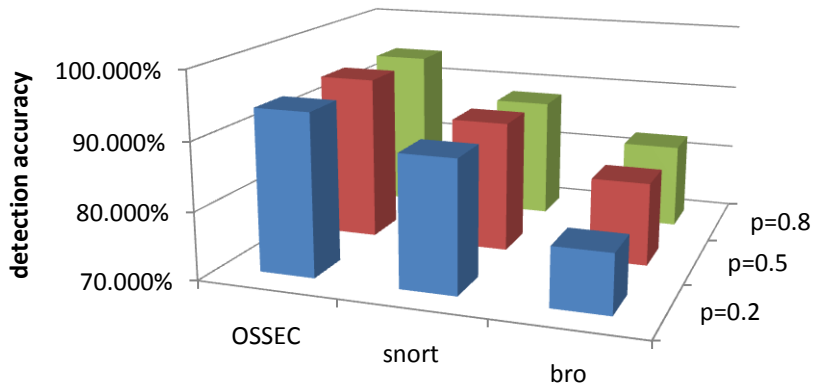


Figure 22. Detection accuracy under different  $P_A$  after learning ( $th3=6$ , Fung et al.)

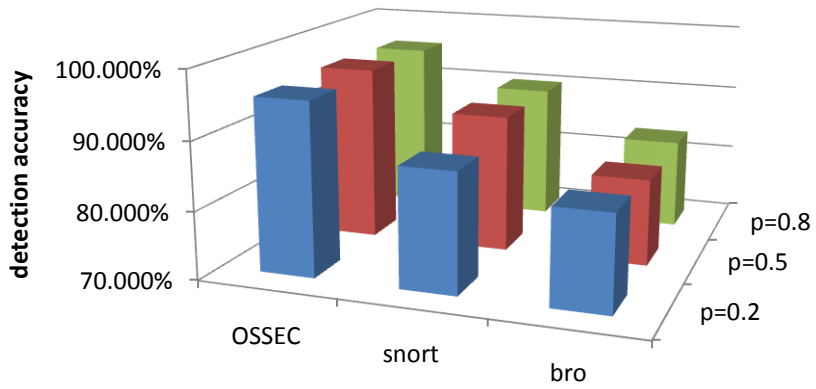


Figure 23. Detection accuracy under different  $P_A$  after learning ( $th3=3$ , Fung et al.)

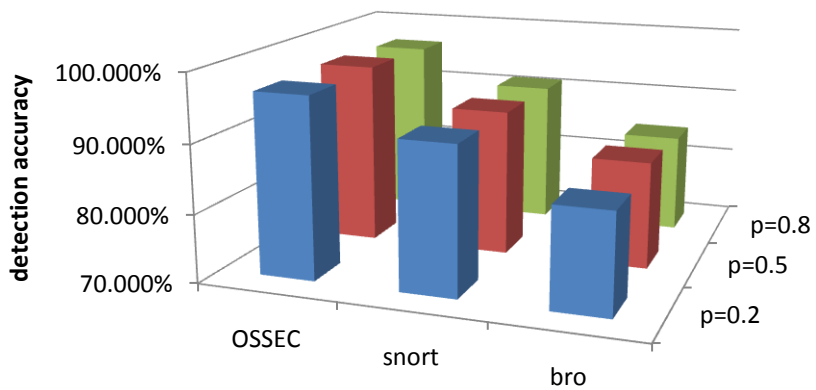


Figure 24. Detection accuracy under different  $P_A$  after learning ( $th3=1$ , Fung et al.)

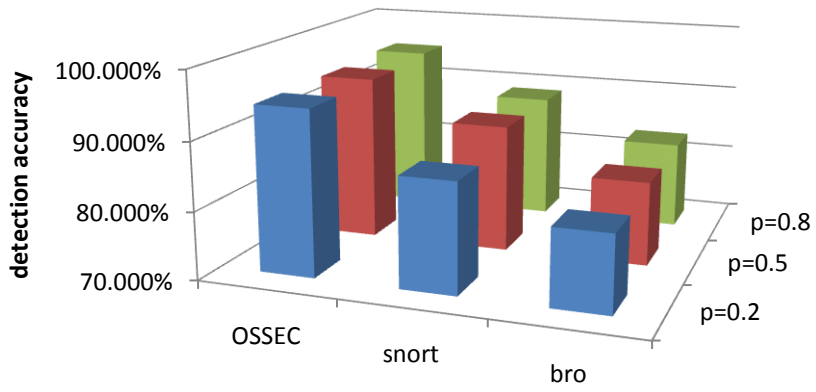


Figure 25. Detection accuracy under different  $P_A$  ( $th3=6$ , non-cooperation)

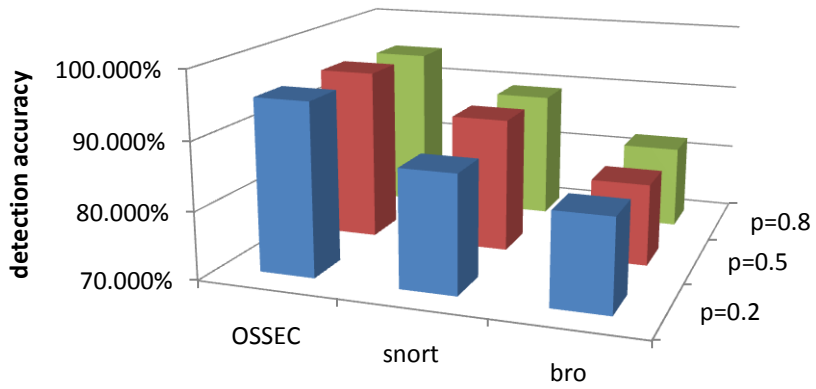


Figure 26. Detection accuracy under different  $P_A$  ( $th3=3$ , non-cooperation)

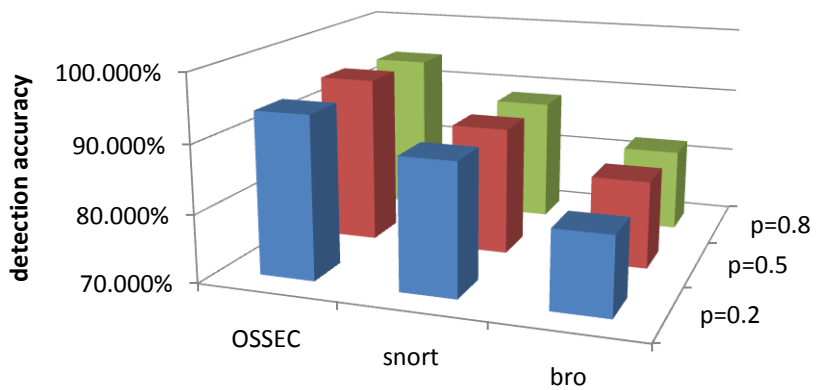


Figure 27. Detection accuracy under different  $P_A$  ( $th3=1$ , non-cooperation)

Generally malicious attacks occur very often, so the nodes with lower  $th3$  have higher probability to detect those attacks. Similarly the nodes detect more attacks in the environment with higher  $P_A$ .

Compared to Fung et al. after learning phase, the proposed mechanism performs much better than it. Even though the value of  $th3$  is set to be 6 in the proposed mechanism, it outperforms Fung et al. when the value of  $th3$  is set to be 1. The reason for such phenomenon is that each node has sufficient time to construct the symptom table in the proposed mechanism when passing through learning phase. A node has ability on detecting an attack without contacting it before. In addition, the nodes in both collaborative mechanisms outperform the non-cooperation conditions.

And the following figures show that the detection accuracy in the proposed mechanism without passing through learning phase. Although the detection accuracy in the nodes without learning is not as high as the one after learning phase, it still outperforms others (Fung et al. and non-cooperation) mentioned above.

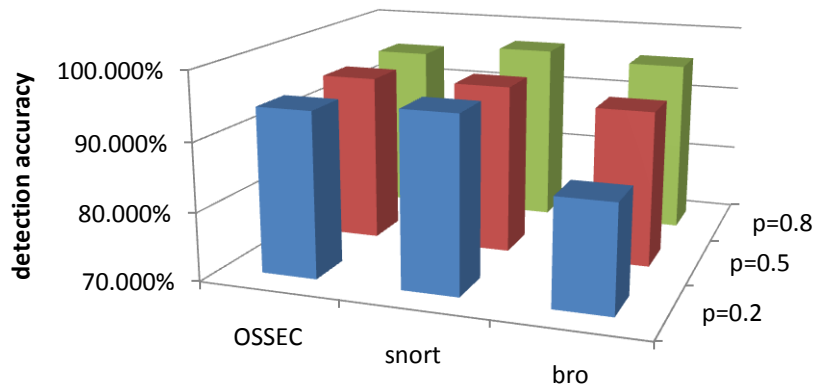


Figure 28. Detection accuracy under different  $P_A$  without learning ( $th3=6$ )

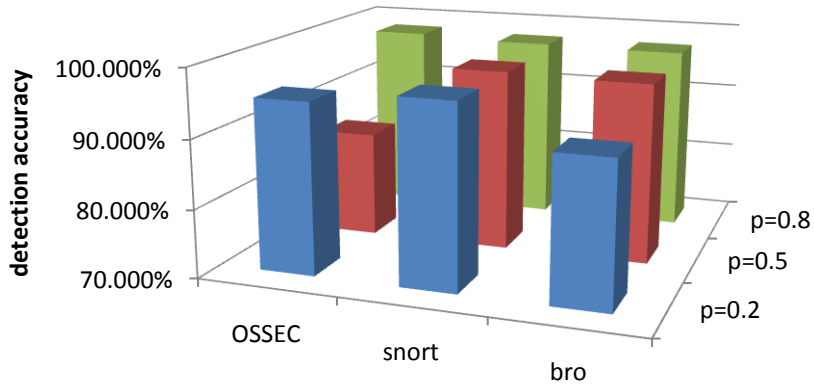


Figure 29. Detection accuracy under different  $P_A$  without learning ( $th3=3$ )

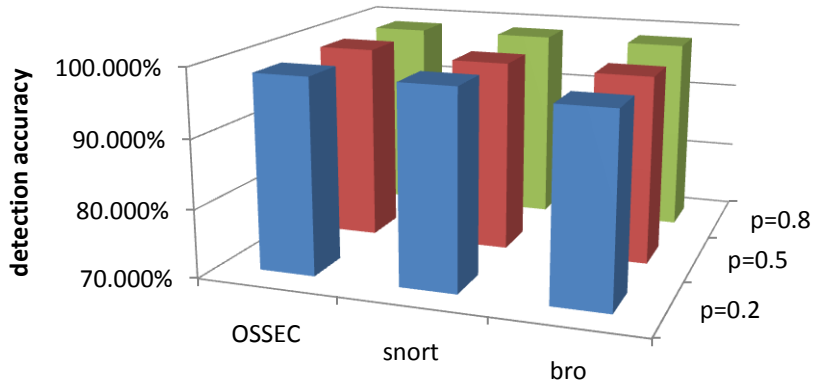


Figure 30. Detection accuracy under different  $P_A$  without learning ( $th3=1$ )

**(2) Dishonest Environment (Case 2)**

Similarly there are two performance metrics will be analyzed in such case. They are trust value and detection accuracy and the results of them are in the following.

- **Trust Value**

In this proposed mechanism, with fixing the multiple parameters ( $P_A=0.5$ ,  $th3=3$ ,  $\gamma=0.1$ ), and the multi values of  $Err$  and  $\beta$ , the trust values for three different types of dishonest nodes in the other three different types of honest nodes are

illustrated in the following figures. All dishonest nodes passed through 100 times test messages while being honest. Moreover, the trust value when the dishonest nodes replies fake response in second times is presented in particular. Thus it is simple to perceive the efficiency and effectiveness on the penalties about a node replying fake responses.

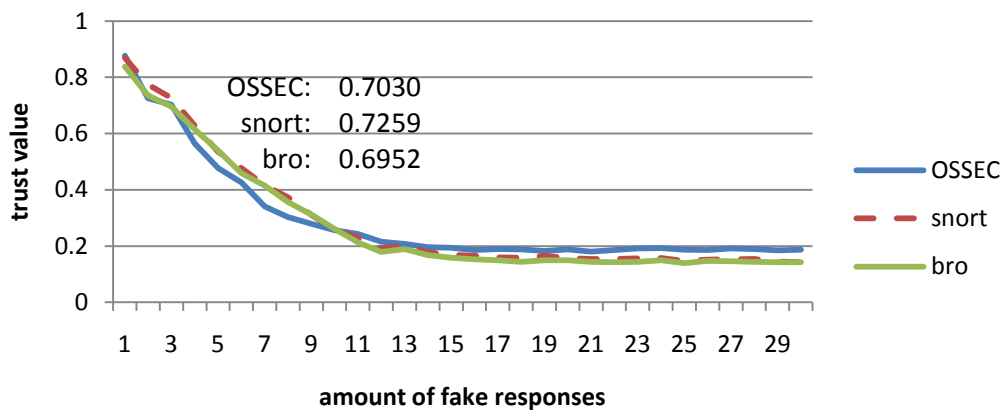


Figure 31. Trust values for three dishonest nodes in OSSEC ( $Err=1$ ,  $\beta=0.01$ )

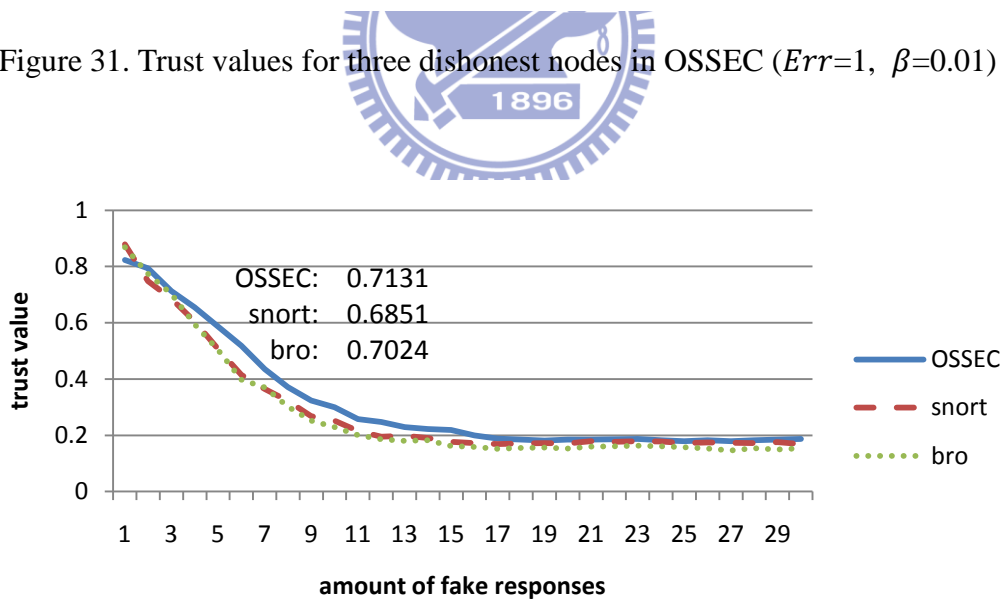


Figure 32. Trust values for three dishonest nodes in snort ( $Err=1$ ,  $\beta=0.01$ )

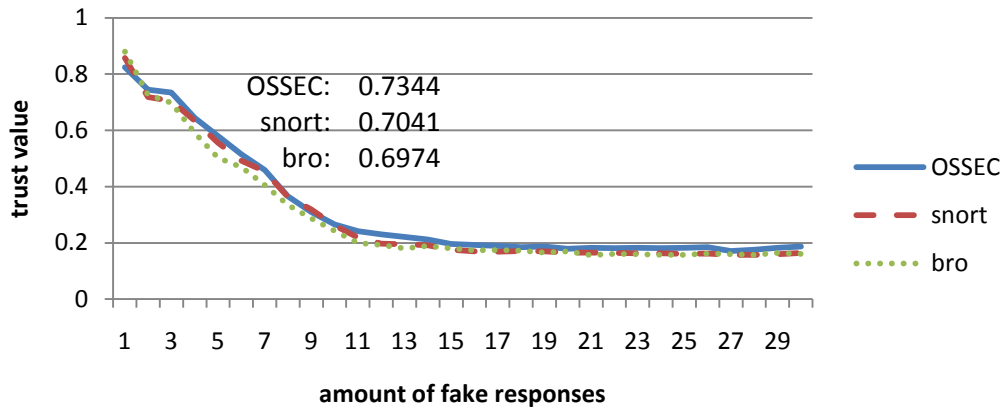


Figure 33. Trust values for three dishonest nodes in bro ( $Err=1$ ,  $\beta=0.01$ )

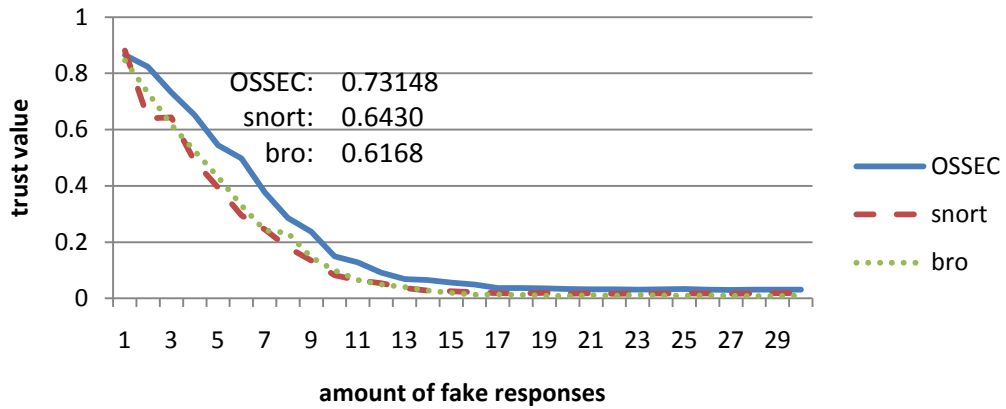


Figure 34. Trust values for three dishonest nodes in OSSEC ( $Err=0.2$ ,  $\beta=0.01$ )

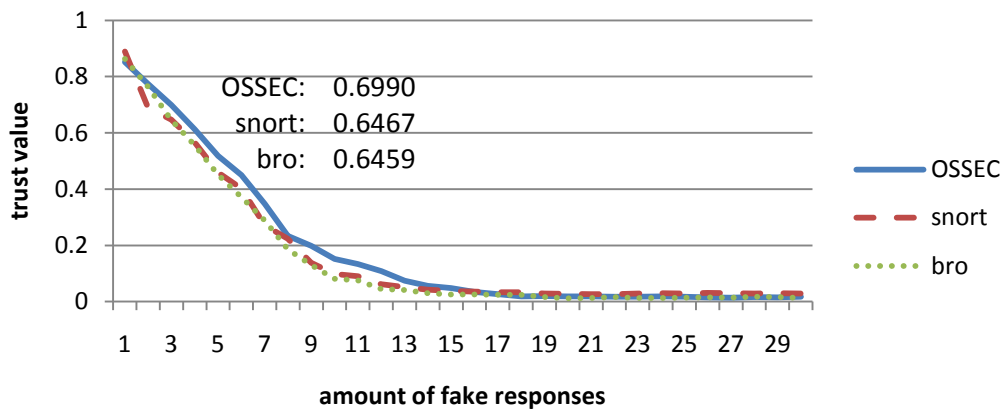


Figure 35. Trust values for three dishonest nodes in snort ( $Err=0.2$ ,  $\beta=0.01$ )

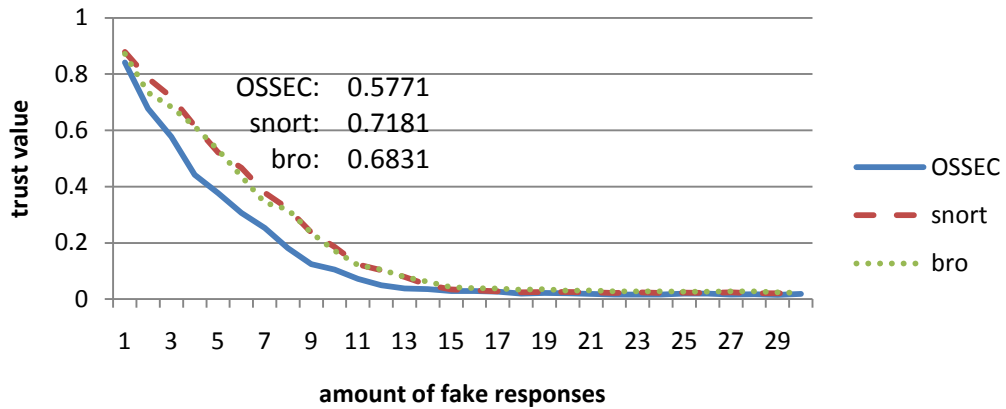


Figure 36. Trust values for three dishonest nodes in bro ( $Err=0.2$ ,  $\beta=0.01$ )

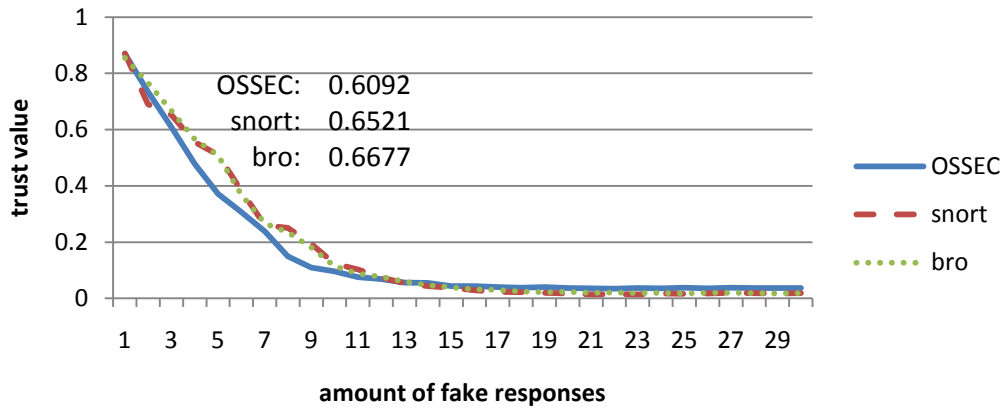


Figure 37. Trust values for three dishonest nodes in OSSEC ( $Err=0.2$ ,  $\beta=0.1$ )

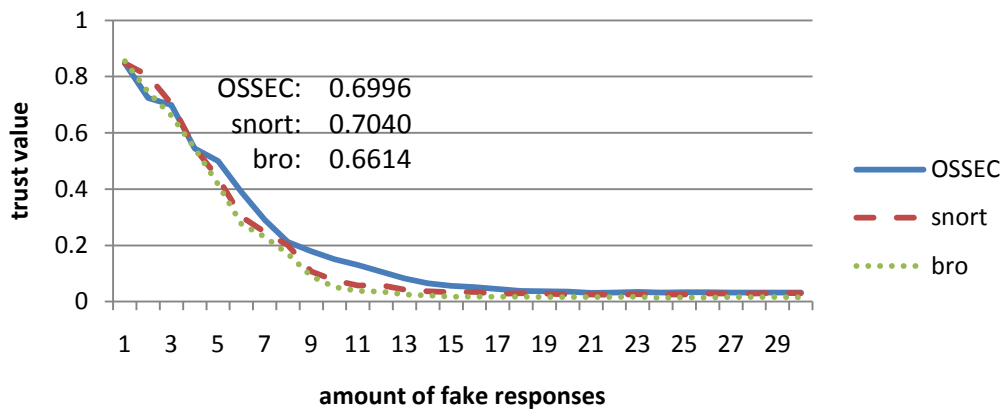


Figure 38. Trust values for three dishonest nodes in snort ( $Err=0.2$ ,  $\beta=0.1$ )



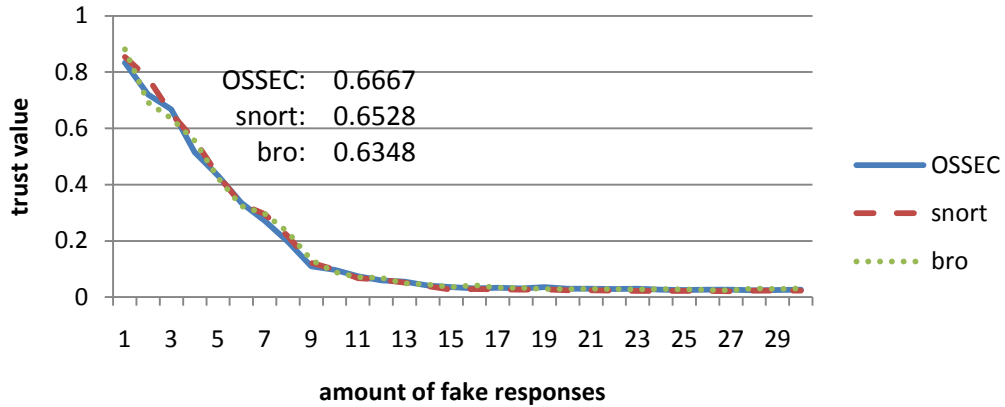


Figure 39. Trust values for three dishonest nodes in bro ( $Err=0.2$ ,  $\beta=0.1$ )

After observing the curve of trust values for three dishonest nodes among 30 fake responses, it shows that with the decreasing in value of  $Err$  from 1 to 0.2, the lowest trust values decrease from 0.2 to 0. And with the increasing in  $\beta$  from 0.01 to 0.1, it shows that the trust values decrease little rapider. Totally the trust values for three dishonest nodes decrease to the values lower than 0.5 in 5 times.

Compared with results in Fung et al. in the following, they give different penalties to different types of nodes. For the dishonest node of the same type, it obtains server penalty than the different type.

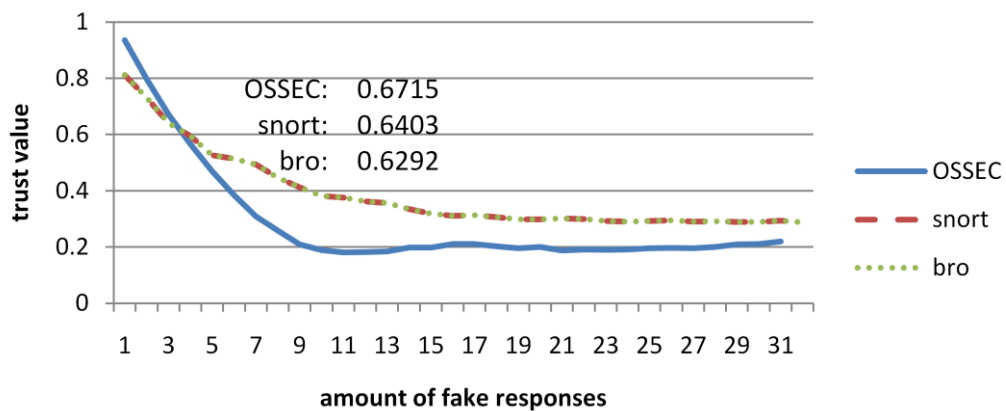


Figure 40. Trust values for three dishonest nodes in OSSEC (Fung et al.)

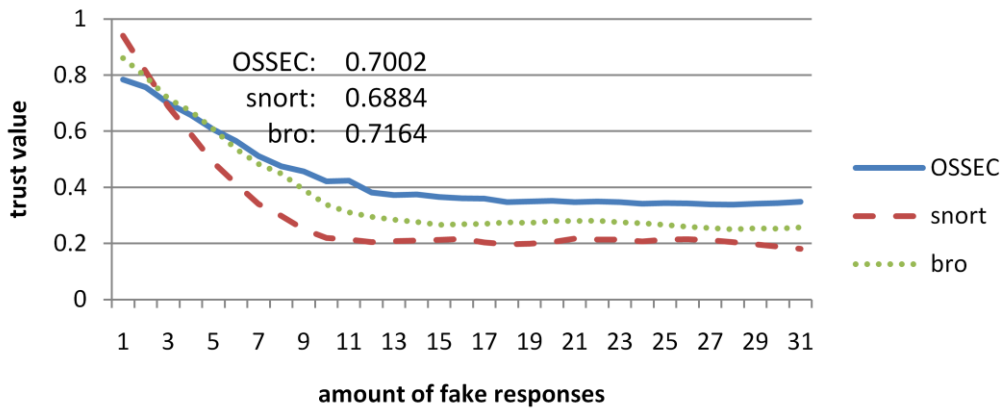


Figure 41. Trust values for three dishonest nodes in snort (Fung et al.)

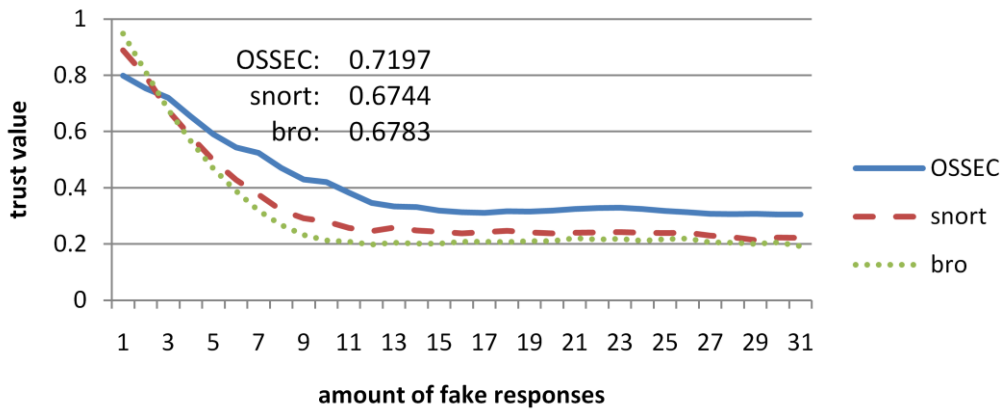


Figure 42. Trust values for three dishonest nodes in bro (Fung et al.)

The reason for the trust values for dishonest nodes in both mechanisms do not decrease to the lowest value immediately is that they do not only consider one satisfaction value each time. Other honest feedback before can comprise the dishonest feedback in the beginning. Moreover, in the proposed mechanism a node considers the feedback of other nodes about the dishonest nodes. Other nodes probably do not receive the fake responses yet, so they provide high trust values about the dishonest nodes. Thus the trust values for dishonest nodes decrease more slowly than Fung et al., but it can be improved by increasing  $\beta$ .

- **Detection Accuracy**

In the proposed mechanism, with fixing multiple parameters ( $Err=0.2$ ,  $\gamma=0.1$ ,  $\beta=0.1$ ) and multi values of  $P_A$ ,  $th3$  and amount of dishonest nodes, detection accuracy in the three different types of honest nodes is observed after each node receiving 1,000 events. Moreover, all the nodes do not pass through learning phase in such case. The results are illustrated in the following.

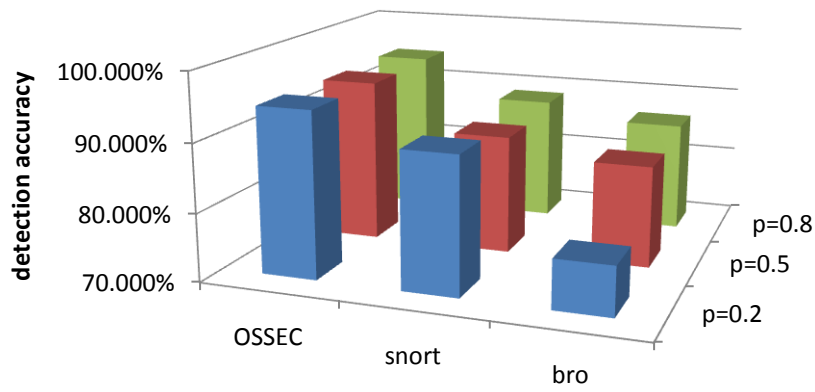


Figure 43. Detection accuracy under different  $P_A$  with 3 dishonest nodes ( $th3=6$ )

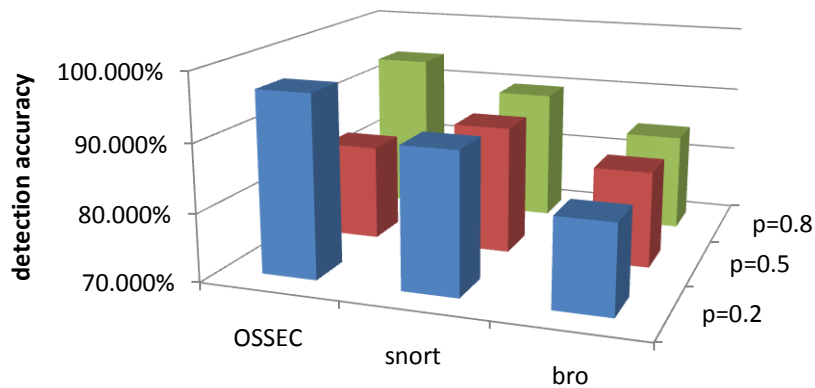


Figure 44. Detection accuracy under different  $P_A$  with 15 dishonest nodes ( $th3=6$ )

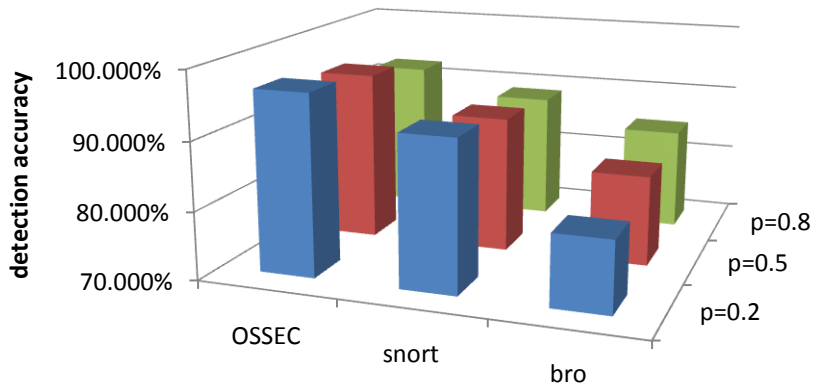


Figure 45. Detection accuracy under different  $P_A$  with 27 dishonest nodes ( $th3=6$ )

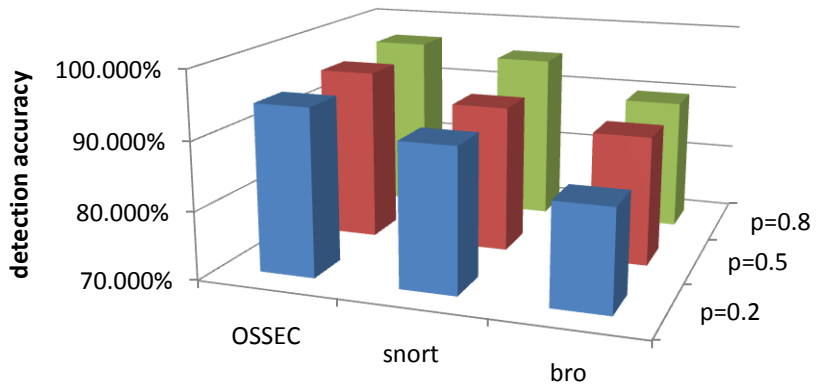


Figure 46. Detection accuracy under different  $P_A$  with 3 dishonest nodes ( $th3=3$ )

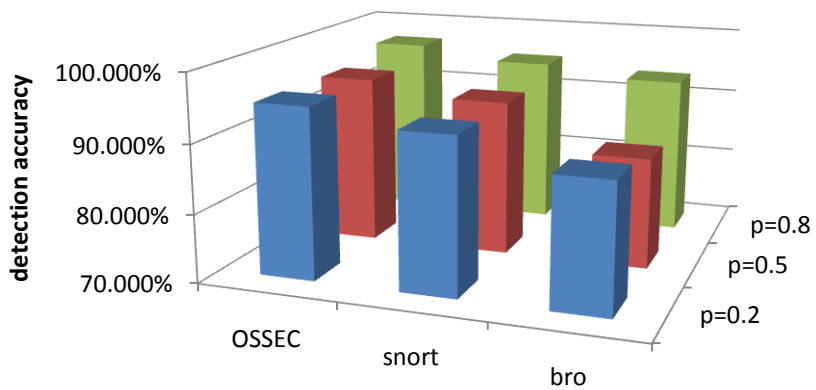


Figure 47. Detection accuracy under different  $P_A$  with 15 dishonest nodes ( $th3=3$ )

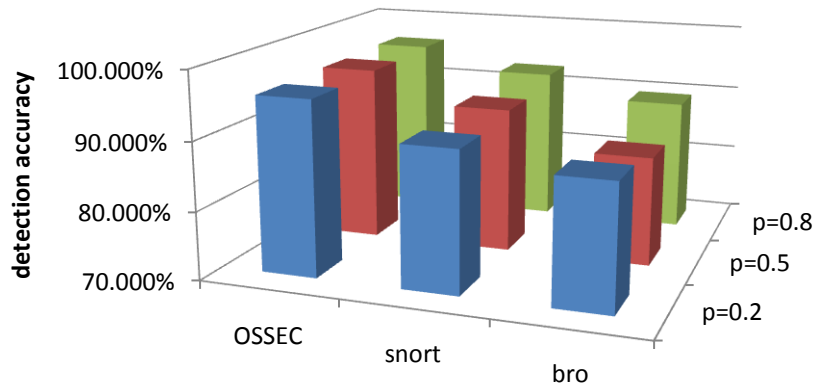


Figure 48. Detection accuracy under different  $P_A$  with 27 dishonest nodes ( $th3=3$ )

Although with the dishonest nodes the detection accuracy is decreased, totally the proposed mechanism in the dishonest environment still outperforms Fung et al. and non-cooperation in the honest environment even 90% of nodes are dishonest. These results prove that it is a robust mechanism. But this performance is based on the rest of honest nodes are different types, since only different types of nodes can share different information with each other. If the rest of honest nodes are in the same type, it is helpless to increase the performance. Because of that all the nodes have the same little information.

## 4.2 Security Analyses

The two-phase collaborative mechanism provides an effective collaboration between the IDSs. However the trust management and the collaborative properties inside may become the target of attacks and be compromised. Thus, in the following, there are the possible attacks and how the proposed mechanism defends against them.

First one are Sybil attack occurring when an IDS creates a large amount of fake identities. These are used to provide false alert ranking to cause unfavorable influence. The design of authentication mechanism is applied to defend such attack. An IDS registers to CA is difficult since one virtual host only has one HIDS and the number of

NIDS is controlled by administrators instead of increasing automatically. On the other hand, all the fake identities need to build up their trust before they affect the decisions of others because of the trust management in each IDS.

Second one is identity cloning attack occurring when an IDS steals the identity of another one and tries to communicate with others. This is defended by the asymmetric encryption between each IDS which has a pair of public key and private key. The CA certifies the ownership of each key and the authenticity of identities are protected in such way.

Third one is new comer attack occurring when a malicious IDS can registers to CA as a new comer easily in order to erase its bad history of others. Registering to CA is difficult as mentioned above and a default trust value of a new comer is low. Such authenticity and trust management can defend this attack.

Forth one is betrayal attack occurring when a trustworthy IDS suddenly turns into a malicious one and starts providing false feedback even malware. This is defended by the trust management model adopting  $\alpha$ ,  $Err$  and  $\beta$ . The trust value for a node decreases dramatically and immediately when it changes its behavior from honest to dishonest.

Fifth one is collusion attack occurring when a group of malicious IDSs cooperating in sending false feedback to others. In the proposed mechanism, it has a chance being affected by such attack. The reason for this is that there is no similar test message in testing the trust value or ability value feedback from others. A group of malicious nodes can compromise the trust values for the members by providing false trust values when other nodes consult about them.

### **4.3 Discussion**

This section summarizes the performance of the proposed mechanism in

different cases mention above. In the honest environment, the proposed mechanism eliminates the difference of trust worthiness between different types of simulators. The trust values are gathering at about 0.8 instead of clustering into two values (0.9 and 0.8) in Fung et al. Moreover, the average detection accuracy with different  $P_A$  after learning phase in multi cases are listed in Table 11.

It is 94%, 97% and 98% in three different types (OSSEC, snort, bro) of simulators with  $th3$  is 6. And without learning phase, it is 94%, 95% and 91% with  $th3$  is 6, 92%, 97% and 94% with  $th3$  is 3 as well as 98%, 98% and 97% with  $th3$  is 1. In the same conditions (after learning), the detection accuracy in Fung et al. and non-cooperation with  $th3$  is set to be 1 (represents that it has higher chance to detect attacks) is 96%, 91% and 84% as well as 94%, 89%, and 82%. The analysis shows that Fung et al. is only 2% better than non-cooperation, but the improvement rate in this mechanism is 4%, 9% and 15% in each type of simulator. It can increase the detection accuracy of the most powerless simulator to 98%.

Table 11. Detection accuracy in the honest environment

	$th3=6$	$th3=6$ <i>no learning</i>	$th3=3$ <i>no learning</i>	$th3=1$ <i>no learning</i>	<i>Fung et al.</i> $th3=1$	<i>No Cooperation</i> $th3=1$
<i>OSSEC</i>	94%	94%	92%	98%	96%	94%
<i>snort</i>	97%	95%	97%	98%	91%	89%
<i>bro</i>	98%	91%	94%	97%	84%	82%

In the dishonest environment, the proposed mechanism provides more impartial and stricter penalties about a node changing its behavior from honest to dishonest than Fung et al. The average trust values in the second times for the nodes replying fake responses in this mechanism and Fung et al. are 0.6609 and 0.6798. And if the nodes

keep lying, the trust value is closing to 0 in this mechanism instead of being 0.2 or 0.3 in Fung et al. Moreover, the average detection accuracy with different  $P_A$  and  $th3$  is 6, without learning in three different types (OSSEC, snort, bro) of simulators is listed in .  $Dis$  is used to instead the amount of dishonest nodes in the table.

It is 94%, 88% and 82% with the amount of dishonest nodes is 3, 91%, 89% and 83% with 15 dishonest nodes as well as 94%, 90% and 82% with 27 dishonest nodes. While  $th3$  is 3, the different detection accuracy is as follows. It is 95%, 92% and 87% with 3 dishonest nodes, 95%, 93% and 89% with 15 dishonest nodes as well as 95%, 91% and 87% with 27 dishonest nodes. Although with the dishonest nodes the detection accuracy is decreased, it still performs as well as Fung et al. and non-cooperation in the honest environment. And there is no obvious decreasing in detection accuracy with the increasing in amount of dishonest nodes. But this is based on the rest of honest nodes are all different types. If the rest of honest nodes are in the same type, it is helpless to increase the performance.

Table 12. Detection accuracy in the dishonest environment

	<b><i>th3=6</i></b> <b><i>Dis=3</i></b>	<b><i>th3=6</i></b> <b><i>Dis=15</i></b>	<b><i>th3=6</i></b> <b><i>Dis=27</i></b>	<b><i>th3=3</i></b> <b><i>Dis=3</i></b>	<b><i>th3=3</i></b> <b><i>Dis=15</i></b>	<b><i>th3=3</i></b> <b><i>Dis=27</i></b>
<b><i>OSSEC</i></b>	94%	91%	94%	95%	95%	95%
<b><i>snort</i></b>	88%	89%	90%	92%	93%	91%
<b><i>bro</i></b>	82%	83%	82%	87%	89%	87%

According to security analysis, the proposed mechanism can defend several common threats such as Sybil attack, identity cloning attack, new comer attack and betrayal attack. But it cannot defend collusion attacks, since there is no appropriate way to constrain IDSs providing honest feedback about the trust value or ability value for others.



## Chapter 5 Conclusion and Future Works

In this chapter, it concludes this proposed mechanism with the motivation, overview of methodology and the performance. It provides some works could be studied in the future.

### 5.1. Conclusion

With the advent of the cloud computing, a number of issues are discussed and among them, security is an important one. This thesis concentrates on intrusion detection. It studies how to apply the intrusion detection systems (IDS) in cloud and makes them cooperate with each other to provide a more secure solution.

A two-phase collaborative mechanism is proposed to enhance the security in cloud. The first phase is constructing the trust management model. Such model is designed to establish the trustworthiness relationships between each IDS. It is contributed by three steps, sending test messages, encouraging replying, and considering the transitivity of trust. With such steps, an IDS can evaluate the trust values for others and it is encouraged to provide benign feedback when communicating. The second phase is collaborating. The trustworthiness between each system, derived at first phase, is used to strengthen the quality of collaboration. There are two ways to collaborate, alert correlation and symptoms sharing. With alert correlation an IDS can derive more precise ranking of suspicious symptoms. And with symptoms sharing an IDS can detect some attacks which has not contacted before. Both are used to increase the performance of IDSs by making them share information with each other.

Eventually, a simulation environment is designed with IDS simulators (OSSEC, snort and bro) and the event (malicious attacks or normal activities) generator. The

proposed mechanism eliminates the difference of trust value for different types of simulators in both honest and dishonest environment. And with observing the average detection accuracy in the honest environment, it is 4%, 9% and 15% better in OSSEC, snort and bro separately than non-cooperation condition while it is 2%, 2% and 2% in Fung et al. The detection accuracy can be increased to 98%. Although with the dishonest nodes the detection accuracy is decreased in the dishonest environment, it still performs as well as Fung et al. and non-cooperation in the honest environment. Even 90% of nodes are dishonest. But this is based on the rest of honest nodes are all different types.

## **5.2. Future works**

In the future, there are a number of issues can be discussed. For instance, the mechanism can be modified to handle the problems that how a node detects another node merely sends fake responses to it. Or how the mechanism handles a node with normal behaviors but sends out fake symptoms to confuse others. And the fault alarm rate should be considered in simulation. Moreover, how the mechanism works with actual IDSs is also an interesting issue.

## References

- [1] A. Abdui-Rahman, S. Hailes, "A Distributed Trust Model," Proceedings of the 1997 workshop on New security paradigms, 1997
- [2] U. Aickelin, P. Bentley, S. Cayzer, J. Kim, J. McLeod, "Danger Theory: The Link between AIS and IDS," Lecture Notes in Computer Science, Vol. 2787, pp. 147-155, 2003
- [3] M. Armbrust, et al. "Above the Clouds: A Berkeley View of Cloud computing," Technical Report No. UCB/EECS-2009-28, University of California at Berkley, USA, 2009
- [4] C. R. Attanasio, "Virtual machines and data security," Proceedings of the workshop on virtual computer systems, New York, USA, ACM, pp. 206-209, 1973
- [5] G. Casella, R. L. Berger, "Statistical Inference," New Delhi: Wadsworth, 2<sup>nd</sup> edition, 2002
- [6] T. Crothers, "Implementing Intrusion detection Systems: A Hands-On Guide for Securing the Network," Wiley Publishing Inc., Indiana, 2003
- [7] H. Debar, D. Curry, and B. Feinstein, "Rfc 4765 - the intrusion detection message exchange format (idmef)," Internet draft, IETF, 2007
- [8] H. T. Elshoush and I. M. Osman, "Alert correlation in collaborative intelligent intrusion detection systems-A survey," Applied Soft Computing, Article in press, 2010
- [9] C. J. Fung, O. Baysal, J. Zhang, I. Aib, and R. Boutaba, "Trust Management for

Host-based Collaborative Intrusion Detection,” Lecture Notes in Computer Science, Vol. 5273, pp. 109-122, 2008

[10] J. Koziol, L. Bump, J. Waston, “Intusion Dtection with Snort,” Sams Publishing, USA, 2003

[11] R. L. Krutz, and R.D. Vines, “Cloud Security: A Comprehensive Guide to Secure Cloud Computing,” Wiley Publishing, Inc., Indiana, 2010

[12] M. Laureano, C. Maziero, and E. Jamhour, “Protecting host-based intrusion detectors through virtual machines,” Computer Networks, Vol. 51, No. 5, pp. 1275-1283, 2007

[13] P. Mell, and T. Grance, “The NIST Definition of Cloud Computing,” National Institute of Standards and Technology (NIST), Vol. 53, Issue 6, pp. 50, 2009

[14] B. Morin, et al. “A logic-based model to support alert correlation in intrusion detection,” Information Fusion, Vol. 10, Issue 4, pp. 285-299, 2009

[15] S. Subashini, V.Kavitha, “A survey on security issues in service delivery models of cloud computing,” Journal of Network and Computer Applications, Vol. 34, Issues 1, pp. 1-11, 2011

[16] A. N. Toosi, and M. Kahani, “A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers,” Computer Communications, Vol. 30, Issue 10, pp. 2201-2212, 2007

[17] S. X. Wu, W. Banzhaf, “The use of computational intelligence in intrusion detection systems: A review ,” Applied Soft Computing, Vol. 10, Issue 1, pp. 1-35, 2010

[18] D. Xu, and P. Ning, “Correlation Analysis of Intrusion Alerts,” Intrusion

Detection Systems, Series on Advances in Information Security, Vol. 38, pp. 65-92, 2008

[19] J. Zhang, and R. Cohen, "Trusting advice from other buyers in e-marketplaces: the problem of unfair ratings," ICEC '06 Proceedings of the 8th international conference on Electronic commerce: The new e-commerce: innovations for conquering current barriers, obstacles and limitations to conducting successful business on the internet, New York, USA, 2006

[20] C. V. Zhou, C. Leckie, and S. Karunasekera, "A survey of coordinated attacks and collaborative intrusion detection," Computers & Security, Vol. 29, Issue 1, pp124-140, 2010

[21] C. V. Zhou, C. Leckie, and S. Karunasekera, "Decentralized multi-dimensional alert correlation for collaborative intrusion detection," Journal of Network and Computer Applications, Vol. 38, Issue 5, pp. 1106-1123, 2009

[22] C. V. Zhou, S. Karunasekera and C. Leckie, "Evaluation of a Decentralized Architecture for Large Scale Collaborative Intrusion Detection," Proceedings of IFIP / IEEE International Symposium on Integrated Network Management, Munich, Germany, 2007

[23] U. Zurutuza, and R. Uribeetxeberria, "Intrusion detection alarm correlation: a survey," Proceedings of the IADAT International Conference on Telecommunications and Computer Networks, Donostia, Spain, 2004

[24] Home of OSSEC, <http://www.ossec.net/>

[25] Home of snort, <http://www.snort.org/>

[26] Home of bro, <http://www.bro-ids.org/>