國立交通大學

網路工程研究所

碩 士 論 文

具隱私性的雲端偵測入侵系統

Intrusion Detection by Clouds with Privacy Preserving

研 究 生：鄭又銓

指導教授：曾文貴　教授

中 華 民 國 一 百 年 六 月

具隱私性的雲端偵測入侵系統

學生：鄭又銓　　指導教授：曾文貴博士

國立交通大學網路工程研究所碩士班

摘要

雲端運算在最近幾年變得越來越熱門。然而，當使用者使用這些雲端服務時，有一些安全上的議題是需要被考慮到的。隱私權就是一個主要的問題在雲端運算上。使用者可能想要使用這些雲端服務，但同時地他們不希望關於他們私密性的資訊因此而洩漏出去。在這篇論文中我們提出一個方法不但可以提供入侵偵測的能力且同時能夠保護使用者的隱私性在雲上。我們結合了隱藏關鍵字搜尋和 SNORT 的規則來建構出我們的系統。最後我們做一些實驗來驗證此方法的效能和偵測率。

Intrusion Detection by Clouds with Privacy Preserving

Student: Yu-Chuan Cheng Advisors: Dr. Weng-Guey Tzeng

Institute of Network Engineering College of Computer Science

National Chiao Tung University

ABSTRACT

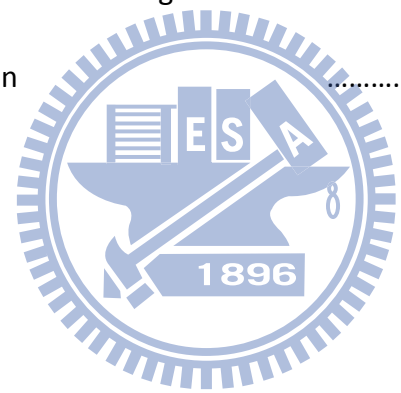
Cloud computing has become popular in recent years. However, when users use these services, there are some security issues to be considered. The privacy issue is one of the main problems in cloud computing. Users want to make use of these services and simultaneously make sure that their sensitive information will not be leaked. In this paper, we propose an approach that not only provides an intrusion detection system, but also protects user's privacy in the cloud. We combine hidden keyword search and SNORT rules to construct our system. Finally, we do experiments to evaluate the system performance and detection rates.

Acknowledgement

首先感謝我的指導教授曾文貴老師，在我碩士班兩年期間的學習過程中，帶領我深入密碼學的領域。老師積極認真的教學態度，使我受益良多。另外，我要感謝我的口試委員，清大孫宏民教授，交大謝續平教授以及交大蔡錫鈞教授，在我論文上給我許多指導和建議，讓我的論文更加地完善，除此之外，我還要感謝實驗室學姊林孝盈的許多幫助，以及實驗室的學弟妹和其他碩士班同學的幫忙。最後，我要感謝我的家人，不論在精神上以及物質上都給我極大的支持，讓我在無後顧之憂的情況下可以順利完成學業。在此，僅以此文獻給所有我想要感謝的人。想要謝的人太多了，那就謝天吧。

Table of Contents

List of Tables



List of Figures

1 Introduction

Advances in networking technology, computation resource and data storage become more important in our information society. Enterprises start to expand their storage and computing centers in order to speed up processing huge data. Since 2004, Google starts to publish some papers about cloud/distributing technologies. Cloud computing begin to become more popular and common. A lot of companies launch newly service about cloud in the internet.

Traditionally, security software always runs on local-side computer. Hence, the computer must own ability to execute the security software. However, cloud computing changes the security software model. Users may use cloud ability to execute security software to inspect their computers and then the cloud responses the result to the user. It increasingly reduces local-side computer ability and is suitable for mobile devices.

Although there is similar cloud-based security software now, it does not consider the privacy issues in the software. Users may leak some secret information to the service providers even the service provider is very famous and fairly. We observe this problem and then we want to solve this problem. Hence, we propose an approach to solve the privacy issues in cloud. Of course, the approach does not influence execution of cloud service. In other words, we propose an approach that not only provides security service but also protects privacy of users in the cloud.

In this paper we propose an application of cloud computing for intrusion detection. The user sends a suspected data to the cloud server for detecting whether the data contains a malicious signature. For security, we would like the user to keep privacy of the data. That is, we want the cloud server to detect whether a malicious signature is inside the data and the server does not know what the data is. Finally, we implement the system according to our approach and then do some experiments to explain our system.

Five Essential Characteristics	1. On-demand Self-service 2. Broad Network Access 3. Resource Pooling 4. Rapid Elasticity 5. Measured Service
Three Service Models	1. Cloud Software as a Service (SaaS) 2. Cloud Platform as a Service (PaaS) 3. Cloud Infrastructure as a Service (IaaS)
Four Deployment Models	1. Private Cloud 2. Community Cloud 3. Public Cloud 4. Hybrid Cloud

Table 1: The NIST Definition of Cloud Computing
Source: "The NIST Definition of Cloud Computing." in NIST, 2009.

1.1 Cloud Computing

Cloud computing [1] has become increasingly popular and common in recent years. More and more companies, such as Amazon, Microsoft and Google, allocate a lot of resources to research and develop cloud applications and services. These companies establish large-scale data centers to provide computing power and storage for users, for example, Amazon EC2, Microsoft Live Mesh, Google Gmail, etc. These convenient and useful services and software quickly began to be widely used in network.

Since 2004, Google start to publish some papers about cloud computing. Google File System (GFS) [6], MapReduce [4] and BigTable [3] are Google's three core technologies for distributed/cloud-computing systems. GFS is a scalable, distributed and fault tolerant file

system. MapReduce is a programming model for processing huge data on large cluster of distributed computers. BigTable is used for managing large distributed data storage system. It is a compressed, high-performance, high scalability database system. According to these three core technologies, cloud applications are to mushroom like bamboo shoots after a spring rain.

According to National Institute of Standards and Technology (NIST) [9], cloud computing is composed of five essential characteristics, three service models, and four deployment models. Generally speaking, Cloud computing is a model for providing on-demand services for users who do not need to care about how computers are managed and storage is located. It is treated by users as a utility such that users pay for the amount they use. Cloud computing is revolutionizing the IT industry.

One use of cloud computing is to let the cloud servers do massive computation for users. There are two benefits. The first is to release heavy computation and storage need from the client side so that the hardware demand in the client side is less. The second is to reduce bandwidth use on the networks since only the computation result is sent back to the client side. In other words, users can access these cloud services with any devices anytime and anywhere.

1.2 Intrusion Detection System

With the advances in network technology, more and more network applications and newly websites come with the tide of fashion. Web surfing has become the recreation in our life. However, our computers in the internet may be under threat of attacks such as virus, worm or Trojan. Hence, detecting malicious software or behaviors has become an increasingly challenging problem.

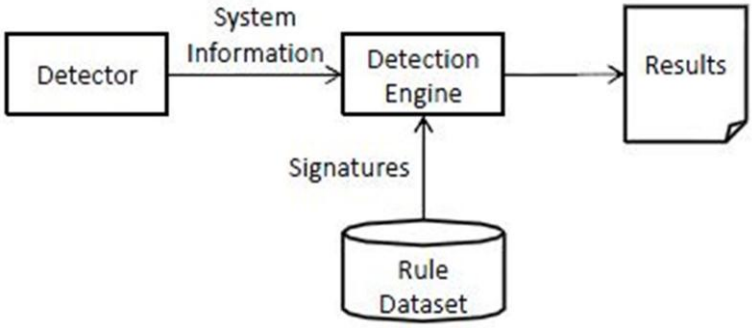


Figure 1: The Units of Intrusion Detection System

There are many tools that can help us to resist network attacks. An intrusion detection system (IDS) [8] plays an important role in network security. It is a useful tool to inspect network traffics, monitor system behavior and detect malicious attacks. An intrusion detection system is separated into three parts: the detector, the rule dataset and the detection engine. The detector is used to collect system information such as packet sniffer or log recorder. The rule dataset are signature of predefined known attacks. The detection engine determines whether the presence of attack according to detector collections and rule dataset.

In general, there are two types of intrusion detection approaches: anomaly-based IDS and signature-based IDS. Anomaly-based IDS defines a set of normal activities beforehand and assumes that the malicious attacks are different form normal activities. When the network traffics are different from threshold value of normal activities, the anomaly-based IDS will notify the system administrator. The advantage of anomaly-based IDS is that it can detect unknown attacks. However, it may have wrong judgments; it is called false-positive. The other type is signature-based IDS. It has to previously define known attacks, it is called signatures. The method is signatures matching which effectively detects computer against malicious attacks. It is a useful way to detect known attacks, but it cannot discover unknown attacks.

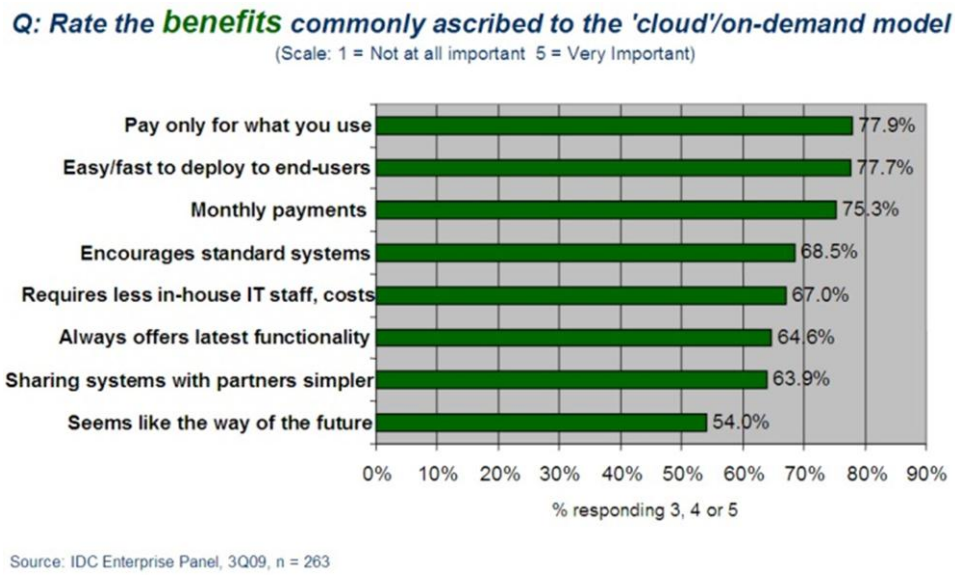


Figure 2: Rate the Benefits of the Cloud
Source: IDC Enterprise Panel

Although these two approaches can resist malicious attacks, many problems still remain. First, intrusion detection system must be often updated to withstand malicious attacks, thanks to the malicious attacks make rapid progress. Second, intrusion detection system must be having a good performance, so that it can be effectively detect and ward off malicious attacks. Finally, intrusion detection system must be quickly deployed to each computer, so that administrators can easily manage and monitor each computer.

1.3 Privacy Issue in Cloud

Cloud brings a lot of benefits in our life. According to IDC market intelligence, the enterprise thinks that "Pay only for what you use" and "Fast to deploy" are the better advantages in the cloud. However, there are some problems for users according to IDC market intelligence. Security is the most important problem for users in the cloud. That is, most customers still cannot believe that the cloud has sufficient security environment. Therefore, most of the information which is stored in the cloud is not important.

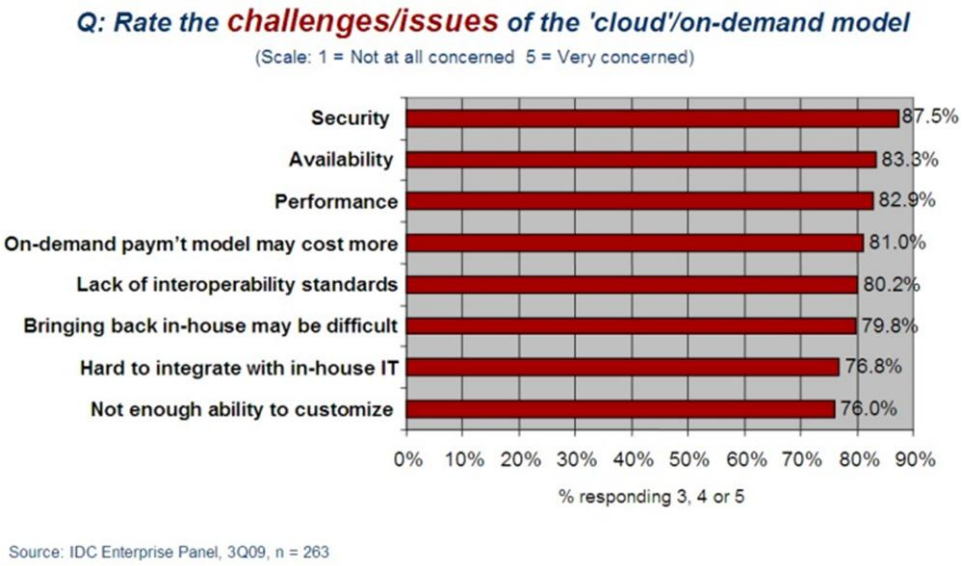


Figure 3: Rate the Challenges of the Cloud
Source: IDC Enterprise Panel

Privacy is a fundamental human right. The privacy issue for cloud service is a big challenge. It is hard to design a cloud service to decrease privacy risk. Processing and transferring sensitive information limit usage of cloud services. In fact, there are a lot of cloud storage services in recent years, such as Dropbox, ASUS Web Storage and Windows Live SkyDrive. These cloud service do not consider the privacy problem. The data in the storage servers may be leaked or misused by the service providers. For example, the recent event of recording user locations through iPhone use by Apple raises serious concern about location privacy of users.

1.4 Motivation

We discover that cloud computing has become more popular in recent years. A lot of large enterprises launch their newly cloud service and provide a charge mechanism to earn money. "Pay-as-you-use" is the biggest advantage for these enterprises. However, they do not notice that their cloud service may invade the privacy. Users take advantage of these cloud service,

but they must transfer some information to the cloud provider. The information may be misused or leaked by the cloud provider. Hence, we provide an approach to solve the privacy problem in cloud service.

Intrusion detection system is a security tool for users to protect their computers against the malicious attacks from the internet. However, most of intrusion detection system must download up-to-date signatures. It is a boring and tedious action. Furthermore, intrusion detection system can decrease system execution performance and it usually runs on a personal computer. That is, it is not suitable for cheaper and light devices.

Cloud intrusion detection system is a good solution to solve performance problems in user-side. However, intrusion detection system need to inspect system information, it may be leaked privacy of users. Hence, we adopt our approach to achieve privacy preserving in the cloud intrusion detection system. In this paper, it is called privacy-preservation cloud-based intrusion detection system.

1.5 Our Contribution

In this paper we propose an application of cloud computing for intrusion detection. The (virus) signatures are put into the cloud side. The user sends a suspected data *W* to the cloud server for detecting whether *W* contains a malicious signature. For security, we would like the user to keep privacy of *W*. That is, we want the cloud server to detect whether a malicious signature is inside *W* and the server does not know what *W* is. We call this privacy-preservation cloud-based intrusion detection system. This system is suitable for mobile users who use mobile platforms, such as smartphone, etc., where the communication bandwidth and storage size are limited. The user does not need to download up-to-date virus signatures. After receiving an encrypted suspected data *C* from a user, the cloud server compares it to the signature database and finds out a possible set *L* of matched signatures.

Then, *L* is sent to the user who filters *L* out to see whether *W* contains a signature in the server's signature database. Since the final confirmation of existence of a signature is done in the client side, the server does not know exactly what *W* is. Thus, the privacy of *W* is kept up to some extent.

We have implemented our design as follows. We use Hadoop as the cloud platform and a Linux operating system as the client. We convert Snort rules into signatures that are put into Hadoop. We design an encryption method of encrypting *W* into *C* by a hidden keyword search technique. Our result shows that the system design meets the requirements of privacy and efficiency.

2 Related Work

Formerly security services are host-based agent in the end-user computers, such as firewall, IDS, anti-virus, etc. So, users don't care that their private secret appears to have leaked out because all computations and data are in the user's computer. However, cloud computing has changed the design of the security service. Cloud anti-virus [10] [11] [12] is a new product in recent years. The virus signatures are put into the cloud side. The user sends a suspected data to the cloud server for detecting whether the data contains a malicious signature. It brings a benefit for users to release heavy computation and storage so that the hardware demand is less. Of course, users don't worry about they have whether up-to-date virus signatures.

However, these applications do not consider the security issues. For security, we would like the user to keep privacy of the data. That is, we want the cloud server to detect whether a malicious signature is inside the data and the server does not know what data is. In order to protect privacy, users may encrypt their information to against cloud provider. Hence, cloud provider must supply a mechanism to process the encrypted data. We concentrate on signature-base security service. In order to solve these security problems in cloud service, we need some approaches about searching in encrypted data in cloud.

Song, Wagner and Perrig proposed a cryptographic scheme for solving the problem of keyword search over encrypted data. Their cryptographic scheme is based on symmetric cryptography. Boneh, Crescenzo, Ostrovsky and Persiano also provide asymmetric cryptography with keyword search. In 2005, Freedman, Ishai, Pinkas, and Reingold use oblivious pseudorandom functions to accomplish Keyword search. So far, keyword search over encrypted data usually apply these related to technologies to accomplish.

In recent years, keyword search over encrypted data in cloud computing has become popular. Users encrypt their data and then outsource the data to the cloud in order to reduce

computation and storage space. Cloud provider must appropriately participate in the keyword search task. According to query of users, cloud provider has to search corresponding answer to reply to the users. Of course, in the processing, cloud provider does not know exact information of users.

3 Preliminary

3.1 Hadoop

Hadoop [15] [16], inspired by Google's core clouding-computing technologies, is a widely used software framework that combines GFS and MapReduce programming model. It supports data-intensive applications and provides reliable, scalable and efficient distributed computing. Hadoop's HDFS (Hadoop Distributed File System) is similar to GFS. Each HDFS is composed of a NameNode and many DataNodes. NameNode is responsible for managing Namespace, recording correspondence between files and blocks, and logging locations of blocks in DataNode. Hadoop's job scheduler has two functions: JobTrack and TaskTracker. JobTracker assigns tasks to TaskTracker. TaskTracker executes the MapReduce function and manages storage results.

Thanks to Hadoop's powerful ability, it can substantially reduce development time of cloud software and rapidly deploy a large number of machines. Furthermore, Hadoop's HDFS can achieve huge data storage and Hadoop's MapReduce API can accomplish large data processing on hundreds to thousands of machines. Of course, Hadoop's distributed computing mechanism let Hadoop become a reliable cloud computing system. Besides, Hadoop is also free and open source software. Hence, Hadoop is now the most common and practical use of large-scale commercial environment in the cloud computing platforms. Generally speaking, Hadoop is a platform that lets programmer can easily develop cloud software and process huge data. Any enterprises or institutions have requirements of processing huge data. They can use Hadoop to construct their cloud environments on their computers to analyze their plentiful data.

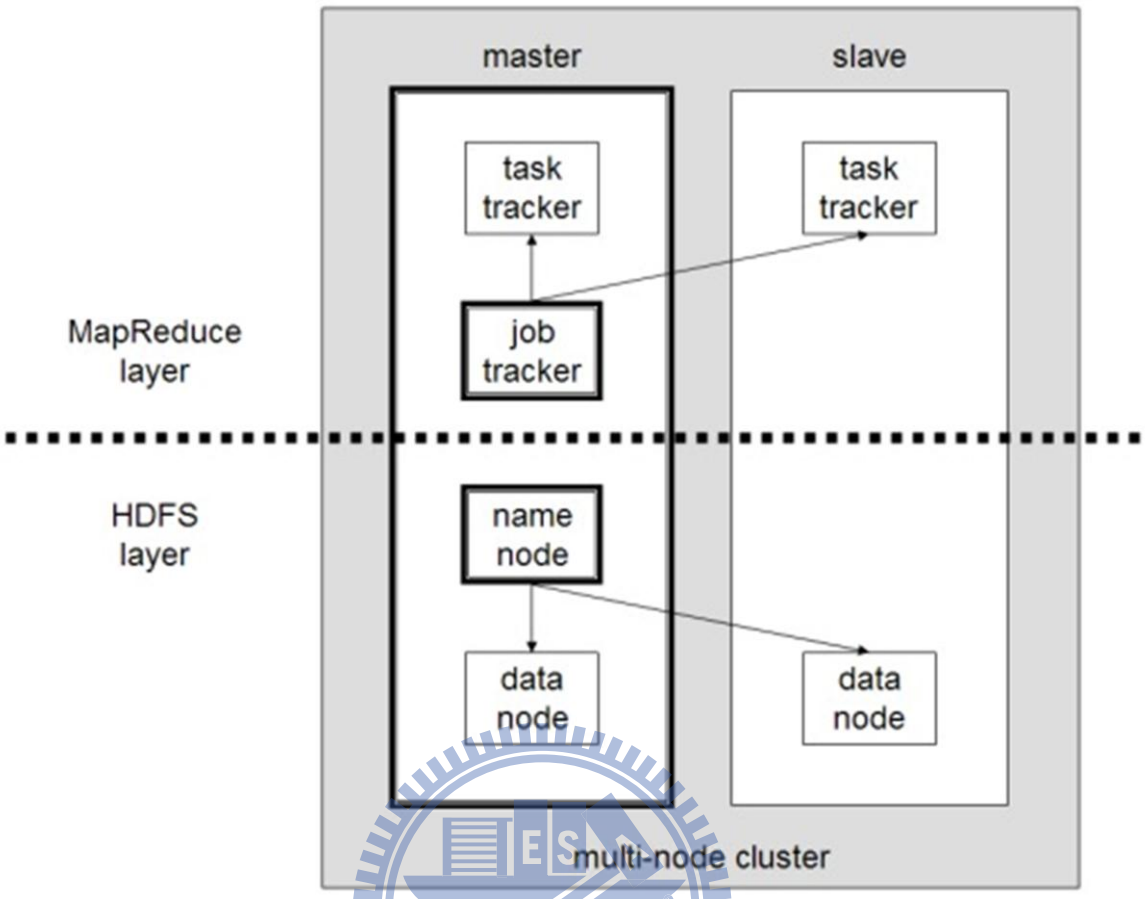


Figure 4: A multi-node Hadoop cluster
Source: http://en.wikipedia.org/wiki/Apache_Hadoop

3.2 SNORT

SNORT [13] is an open-source signature-based Intrusion Detection System (IDS) system. It inspects network traffics, monitor system behavior, and detect malicious attacks by finding whether malicious signatures appear in the activities. SNORT is lightweight, rule-based and cross-platform. It has three modes: sniffer, packet logger and network intrusion detection. In this paper, we use its capability of network intrusion detection.

SNORT detects malicious signatures by a set of rules. Each rule consists of a rule header and some rule options. The rule header contains rule action, protocol, address, port number

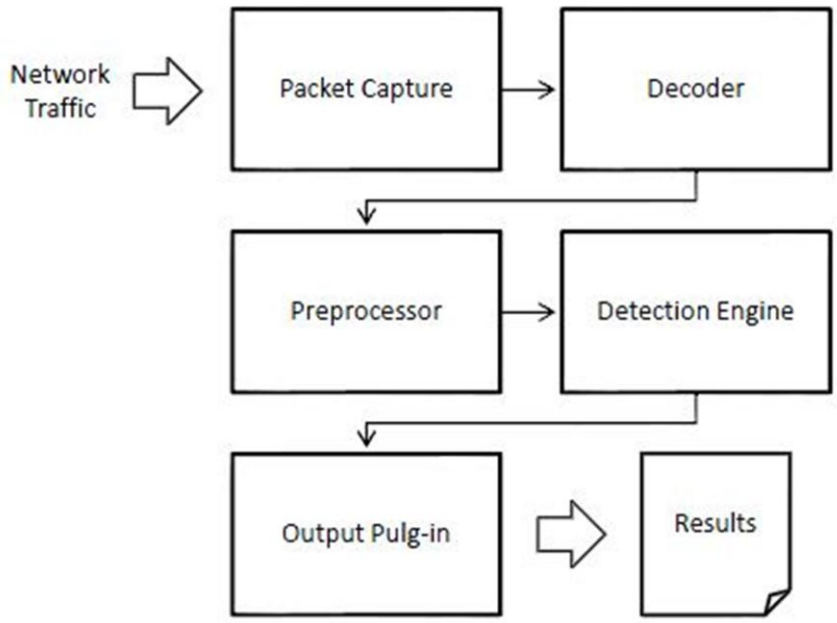


Figure 5: Snort Traffic flow

and the direction operator. The rule action is used to instruct what to do about a packet when the packet matches the criteria of the rule. For example, the action rules could be "alert", "log", "pass", etc. SNORT handles 4 types of protocols, IP, TCP, UDP and ICMP. The address defines the source and destination IP addresses. The rule options are the heart of the SNORT detection engine. It defines suspicious behaviors and patterns. Moreover, it is easy to expand SNORT rules. There are four major categories of rule options: general, payload, non-payload and post-detection. The general options give extra information about rules, but do not affect detection ability. The payload options inspect packet payload and correlation, such as content, length or case sensitive. The non-payload options examine packet headers, such as IP, TCP or ICMP header. The post-detection options trigger other actions.

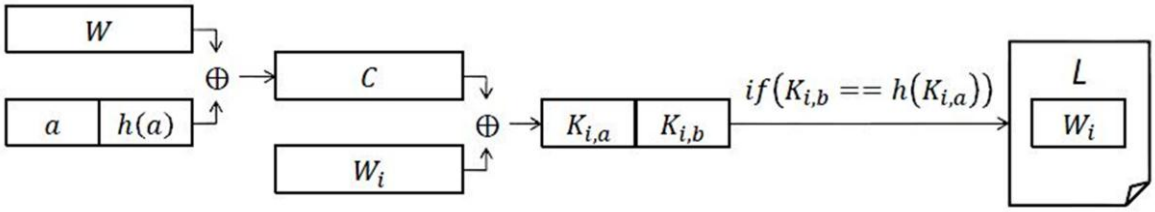


Figure 6: Hidden Keyword Search Flow

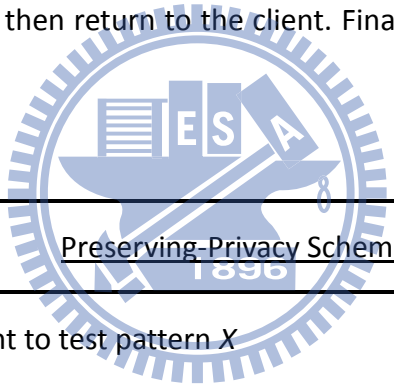
3.3 Hidden Keyword Search

Song, Wagner and Perrig proposed a cryptographic scheme for solving the problem of keyword search over encrypted data. In the setting, the client and the server agree on a hash function h. Suppose that the client wants to securely search a bit string W over the server's signature database that consists of signatures W_1, $W_2 \cdots$, W_m. First, the client generates a key $K=a||h(a)$ of length $|W|$, where a is randomly chosen bit string. Then, the ciphertext $C=W \oplus K$ is sent to the server. The server evaluates $K_i=W_i \oplus C=K_{i,a}||K_{i,b}$, where $|K_{i,a}|=|a|$, $|K_{i,b}|=|h(a)|$ and $i=1, 2,\ldots, n$. If $K_{i,a}=h(k_{i,b})$, the server puts W_i into the candidate list L. The server sends L back to the client. The client searches L to find out whether W appears in L. We can see easily that the privacy, with respect to the server, of W depends on the number of elements in L. The larger L is, the more secure W is. That is, the server can only know that W is some element in signature database, but does not know which one exactly. By adjusting the length of $h(a)$, we can pre-compute the average number of candidates in signature database.

4 Preserving-Privacy Scheme

The goal of our preserving-privacy scheme is to search the signature database of the server without revealing the pattern *W* of the client. The server is the cloud computing center that provides vast computing power. The server's powerful detection engine compares the query from the client with its signature database and returns a candidate list *L* of a possible match. In the end, the client searches *L* to confirm the existence of *W* in the signature database of the server.

In the preserving-privacy scheme, there are two roles, server and client. The client use ciphertext generation to generate a testing pattern *W*. The server runs search protocol to find the candidate list *L* and then return to the client. Finally, the client can gain final results according to candidate list *L*.



Preserving-Privacy Scheme

Suppose that the client want to test pattern *X*

1. The client uses conversion function to convert *X* to *W* and then save *W* in local-side.
2. The client uses ciphertext generation to generate a ciphertext *C* and then save *r* in local-side.
3. Then client sends ciphertext *C* to the server.
4. The server match signatures database according to search protocol.
5. The server responds the candidate list *L* to the client.
6. The Client compare candidate list *L* and random number *r*. If $K_{i,c}=r$, the *X* exists in signatures database.

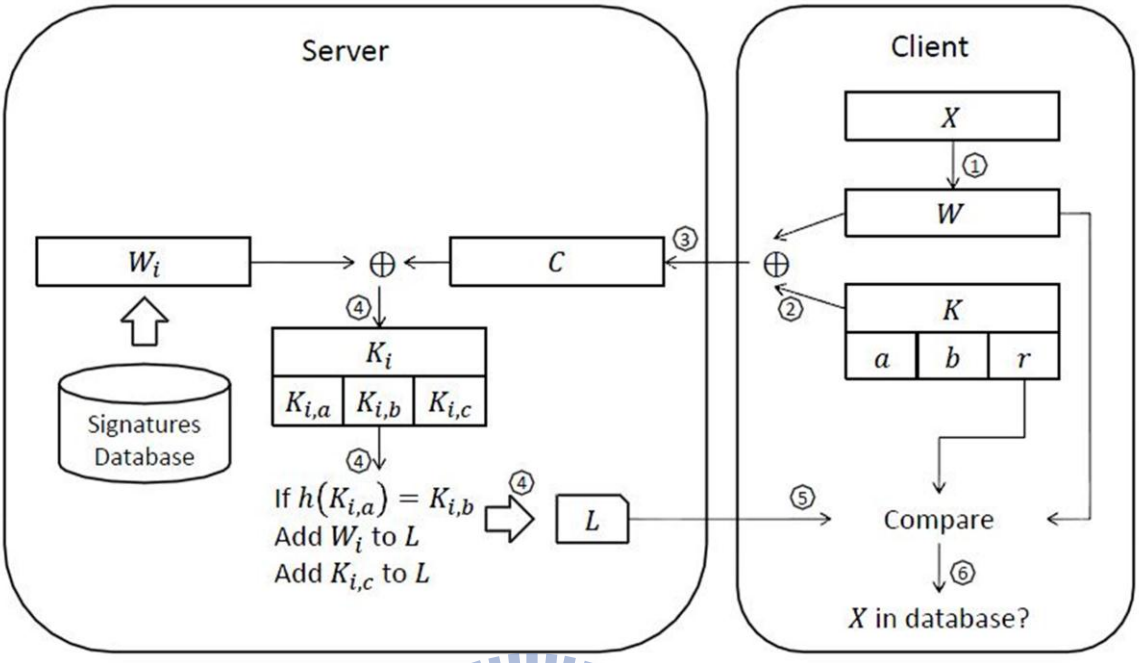


Figure 7: The Preserving-Privacy Scheme

4.1 Threat Model

As a matter of fact, internet is not a secure environment. There are a lot of malicious users such as eavesdroppers, fakers or malicious attacker in the internet. In order to security, users maybe encrypt their data, and then transfer these ciphertext via public communication channels. Of course, they also can use secure channels to deliver their information. Although these two approaches can defend against intermediate adversary, we also want to protect user-side against server-side/cloud provider. In other words, server-side cannot get extra information of client-side. Hence, server-side must have capability of processing encrypted data. Of course, they cannot decrypt user-side ciphertext. That is, server-side does not steal privacy of user-side or gain extra information.

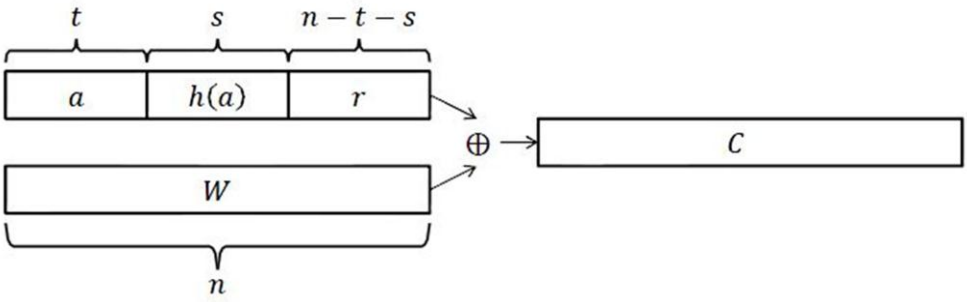


Figure 8: Ciphertext Generation

On the other hand, we do not discuss server-side security in this paper. That is to say, we believe all users are polite and legal. They do not do malicious damage or swindle in order to gain more information from the server-side.

4.2 Ciphertext Generation

When a client wants to test a pattern *W*, it uses ciphertext generator computes a ciphertext. The client makes use of ciphertext generation to generate a key and then the client uses the key to encrypt the pattern *W* which they want to test. Note that the key only uses once and the server does not know the key.

Ciphertext Generation

Input: *W*, *n*; (* *n* is the length of the ciphertext *)

1. Choose a random number *a* of length *t* bits.
2. Compute the hash value *h*(*a*) of *a*.
3. Choose a random bit string *r* of length *n*-*t*-*s* bits.
4. The ciphertext is in the form $C=W \oplus (a||b||r)$, where || is "concatenation of strings ".

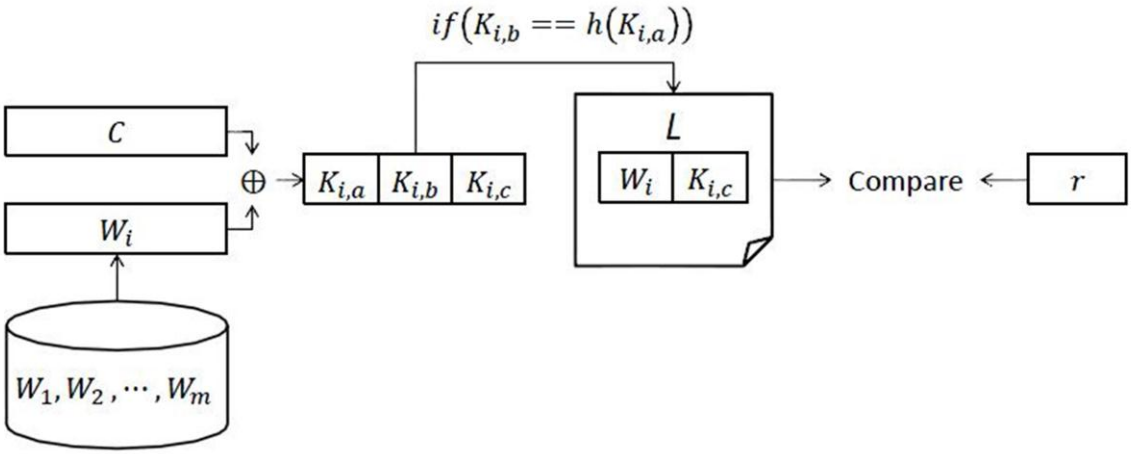


Figure 9: Search Protocol

4.3 Search Protocol

In the search protocol, after receiving a ciphertext query C from the client, the server computes $K_i=W_i \oplus C=K_{i,a}||K_{i,b}||K_{i,c}$, where $|K_{i,a}|=|a|=|t|$, $|K_{i,b}|=|h(a)|=s$, and $|K_{i,c}|=n\text{-}t\text{-}s$. The server checks whether $K_{i,b}=h(K_{i,a})$. If so, the server puts K_i into the candidate list L. Since the server has powerful computing powerful, the computation of K_i can be done very fast. Thus, even though the signature database is large, the system performance in the server side is reasonable.

Search Protocol

Input: C and $\{W_1, W_2,\ldots, W_m\}$.

1. For each i=1, 2,…, m
2. The server selects W_i in the signatures database and computes $K_i=C \oplus W_i$.
3. Let $K_i=K_{i,a}||K_{i,b}||K_{i,c}$, where $K_{i,a}$ is t-bit long, $K_{i,b}$ is s-bit long and $K_{i,c}$ is n-t-s-bit long.

If $K_{i,b} = h(K_{i,a})$, then server saves W_i and $K_{i,c}$ in the candidate list L.

4. The server sends L to the client.

5. For each K_i in L, the client check whether $K_{i,c}=r$, which is the random bit generated in the ciphertext generator. If so, reply the signature is found.

4.4 Analysis

The preserving privacy scheme can protect the client's data W because the server does not know what W is. The server receives the ciphertext from the client and does signatures matching. The server only gains the guest key $K=K_{i,a}||K_{i,b}||K_{i,c}$. The server test the $h(K_{i,a})=K_{i,b}$ to decide the whether the W_i is put into the candidate list L.

We use the number of the candidate list L as the privacy degree of the client's data W. That is, the server does not know what W is, but the server know that the candidate list L which may contain W or not. The number of the candidate list L is dependent on the length of the $h(a)$. The length of $h(a)$ may decide the types of the hash values. Hence, the number of candidate list L equals the all $|S|/2^s$, where the $|S|$ is denoted the number of all signatures and the s is denoted the length of the $h(a)$.

We use the preserving privacy scheme to gains the server's uncertainty of the client's data W. The server knows the exact W if the client does not do any processing. Now the server only knows the candidate list L via the preserving privacy scheme. For security, the preserving privacy scheme can protect the client's data W against the server, and the server's uncertainty about the data W increases.

5 Environment Construction

Before we do experiments, we have to construct our experimental environment. The construction is divided into three parts: cloud environment, signature database and preserving-privacy scheme. We use some tools to construct the environment as following.

Tool List	
1. Cloud environments	Ubuntu + Hadoop + Java + OpenSSH
2. Signature database	SNORT rules
3. Preserving-privacy scheme	jpcap

There are two roles in the environment such as a cloud provider which provides a service and a user which use a service. The cloud provider must supply intrusion detection system ability for user. In the other hand, the user must be able to access to cloud service. Generally speaking, our system is client-server architecture.

At first the client would collect packets and then analysis these packets. Next, the client would convert these packets according to predefined function and at the same time the client must store these original packets in order to the back steps for making a final decision. Subsequently, the client would generate a query using ciphertext generation algorithm and then sends the query to the server to do analysis and inspection. When the server receives the ciphertext, it compares the ciphertext and signatures database according to the signature matching algorithm. The program is a MapReduce model and it must run in cloud environment. Finally, the server would reply candidate list to the client.

Eventually, the client compares candidate list and storage of the original packets. If the client discovers that the query exists in candidate list, it displays dangerous message.

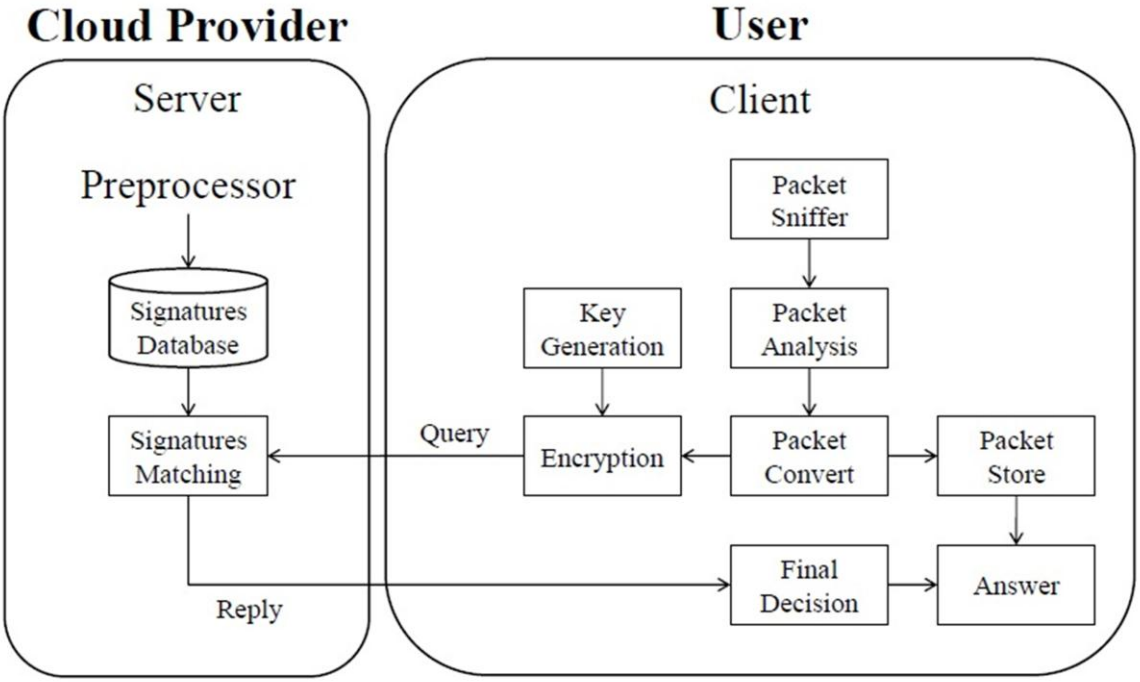


Figure 10: The System Flow Chart

5.1 Cloud Environment

In this paper, we use three machines to construct our cloud environment. We use Hadoop to implement our cloud environment and assign user "hadoop" as a dedicated Hadoop system user. Our software version is "Ubuntu 10.04 LTS" and "Hadoop 0.20.2". Hadoop requires running on Java 6 and uses SSH to access its nodes. Hence, we must install Java and OpenSSH before we install Hadoop.

5.1.1 Networking Configuration

We have to edit "/etc/host/" to assign the IP address to each node. The IP address is not a special address. It is only point out that both machines must be able to reach each other

```
hadoop@ubuntu:~$ netstat -tulpn
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
-
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
-
tcp6       0      0 :::22                   :::*                    LISTEN
```

Figure 11: The Execution of SSH Service

over the network.

192.168.0.162 node1

192.168.0.161 node2

192.168.0.164 node3

5.1.2 SSH Installation and Configuration

We install OpenSSH which is a free version of the SSH connectivity tools. Now we type the following command to install OpenSSH and make sure that OpenSSH is running.

$ sudo apt-get install openssh-server

$ netstat –tulpn

Due to Hadoop require SSH access to manage its nodes, we need to configure SSH access to each machines for user "hadoop". Hence, we may configure these machines to allow SSH public key authentication. We have to add the public SSH key for the user "hadoop" to each nodes as the following commands.

$ ssh-keygen -t rsa -f ~/.ssh/id_rsa -P ""

$ cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys

$ scp -r ~/.ssh node2:~/

$ scp -r ~/.ssh node3:~/

After the configuration of SSH, we may test each node access to other nodes without password via SSH.

$ ssh node1

$ ssh node2

$ ssh node3

5.1.3 Sun Java 6 Installation

We install Sun's Java Runtime Environment (JRE) via apt-get. For Ubuntu 10.04 LTS, the sun-java6 packages have been dropped from the Multiverse section of the Ubuntu archive. Now we install sun java packages using the following commands.

1. Add partner repository

   $ sudo add-apt-repository "deb http://archive.canonical.com/ lucid partner"

2. Update the source list

   $ sudo apt-get update

3. Install sun java packages

   $ sudo apt-get install sun-java6-jre sun-java6-plugin sun-java6-fonts

4. Check that the JRE is properly installed

   $ java -version

After installation, we discover that the Java is placed in "/usr/lib/jvm/java-6-sun" on Ubuntu. This is an important location path because of subsequent Hadoop configuration.

```
hadoop@ubuntu:~$ java -version
java version "1.6.0_24"
Java(TM) SE Runtime Environment (build 1.6.0_24-b07)
Java HotSpot(TM) Client VM (build 19.1-b02, mixed mode, sharing)
```

Figure 12: The Java Version

5.1.4 Hadoop Installation and Execution

There are five steps to finish installation of Hadoop. In this paper, we only use three nodes. One is master, and the other two are slaves.

1. Installation

   $ cd /opt

   $ sudo wget http://ftp.twaren.net/Unix/Web/apache/hadoop/core/hadoop-0.20.2/hadoop-0.20.2.tar.gz

   $ sudo tar zxvf hadoop-0.20.2.tar.gz

   $ sudo chown -R hadoop:hadoop hadoop-0.20.2

   According to above commands, we install Hadoop in /opt/hadoop-0.20.2 and only user "hadoop" and group "hadoop" can execute Hadoop.

2. Configuration

   We have to configure Hadoop's some file after we finish installation of Hadoop. This configuration not only specify Hadoop execution path but also determine our cloud environments. Of course, this configuration is not the only. We configure these parameters in order to satisfy with our condition.

   2.1 hadoop-env.sh

   export JAVA_HOME=/usr/lib/jvm/java-6-sun

   export HADOOP_HOME=/opt/hadoop-0.20.2

export HADOOP_CONF_DIR=/opt/hadoop-0.20.2/conf

2.2 core-site.xml

```
<property>
    <name>fs.default.name</name>
    <value>hdfs://node1:9000</value>
</property>
<property>
    <name>hadoop.tmp.dir</name>
    <value>/opt/hadoop-0.20.2/HDFS-${user.name}</value>
</property>
```

2.3 hdfs-site.xml

```
<property>
    <name>dfs.replication</name>
    <value>2</value>
</property>
```

2.4 mapred-site.xml

```
<property>
    <name>mapred.job.tracker</name>
    <value>node1:9001</value>
</property>
```

2.5 master

node1

2.6 slaver

node2

node3

3. Formatting the name node

   $ cd /opt/hadoop-0.20.2/

   $ bin/hadoop namenode -format

4. Starting the multi-node cluster

   $ cd /opt/hadoop-0.20.2/

   $ bin/start-all.sh

5. Checking the execution state

   http://node1:50030/

   http://node1:50070/

   http://node2:50060/

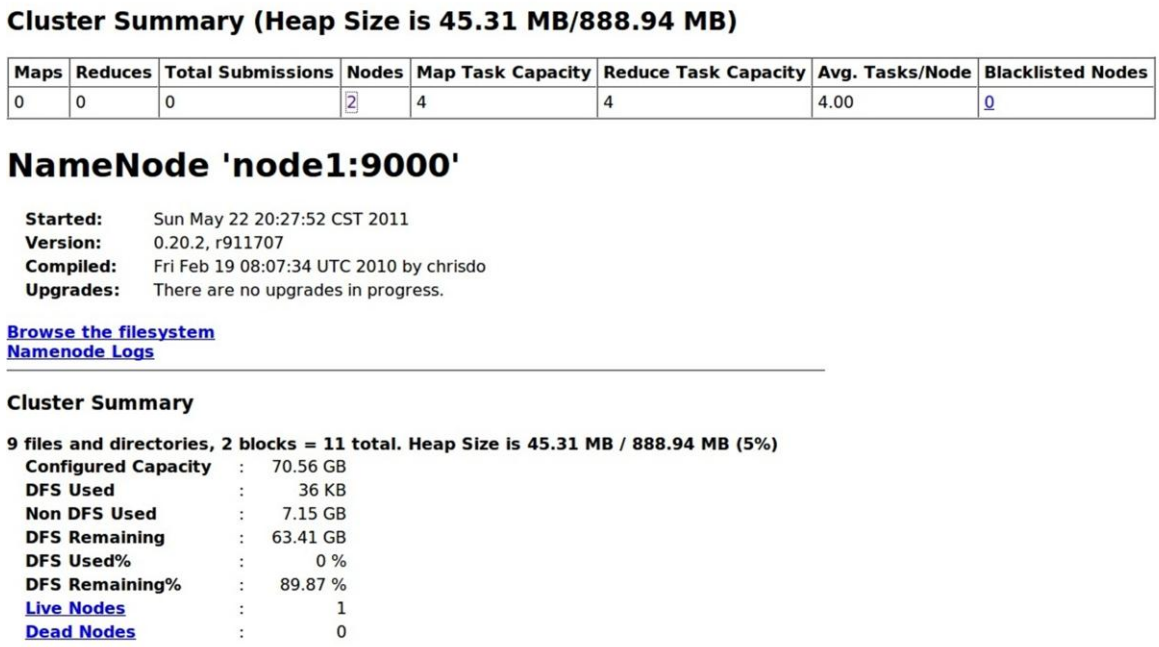
   http://node3:50060/

When we finish above mentioned five steps, we can use Hadoop API to implement our cloud service.

5.2 Signature Database

The signature database is predefined known attack signatures. It comes from SNORT rule, but it has a little different from SNORT rules. Our signature database has to be converted according to our convert function. Our convert function is SHA-1 hash function in. That is, the client's query and key and the server's signatures are fixed in 160-bits.

The SNORT rule is divided into two parts: header and options. The option part is the core of the detection of the SNORT. We focus on payload detection rule and non-payload detection rule in the option part. In these two detection rule, the option has a lot of keyword,



Cluster Summary (Heap Size is 45.31 MB/888.94 MB)

Maps	Reduces	Total Submissions	Nodes	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes
0	0	0	2	4	4	4.00	0

NameNode 'node1:9000'

Started: Sun May 22 20:27:52 CST 2011
Version: 0.20.2, r911707
Compiled: Fri Feb 19 08:07:34 UTC 2010 by chrisdo
Upgrades: There are no upgrades in progress.

Browse the filesystem
Namenode Logs

Cluster Summary

9 files and directories, 2 blocks = 11 total. Heap Size is 45.31 MB / 888.94 MB (5%)

Configured Capacity	:	70.56 GB
DFS Used	:	36 KB
Non DFS Used	:	7.15 GB
DFS Remaining	:	63.41 GB
DFS Used%	:	0 %
DFS Remaining%	:	89.87 %
Live Nodes	:	1
Dead Nodes	:	0

Figure 13: The State of Hadoop

such as "content", "flow", "icode", etc. We use SHA-1 hash function to convert these rules as following. Finally, we put the signature database in the cloud after we finish the rule conversion.

Rule Conversion

Input: {$rule_1$, $rule_2$, …, $rule_m$} and keyword list *KL*.

1. The $rule_i$ is divided into $header_i$ and $option_i$.
2. Let the $option_i$=($op_{i,1}||s_{i,1}$)($op_{i,2}||s_{i,2}$)…($op_{i,n}||s_{i,n}$).
3. To search the $option_i$ according to keyword list *KL*.
4. If $op_{i,j} \in KL$. To compute $hs_{i,j}$=SHA-1($s_{i,j}$).
5. To save $signature_i$=($op_{1,j}||hs_{1,j}$)($op_{2,j}||hs_{2,j}$)…($op_{n,j}||hs_{n,j}$).
6. The signature database saves $rule_i$ || $signature_i$, where i=1,2, …, m.

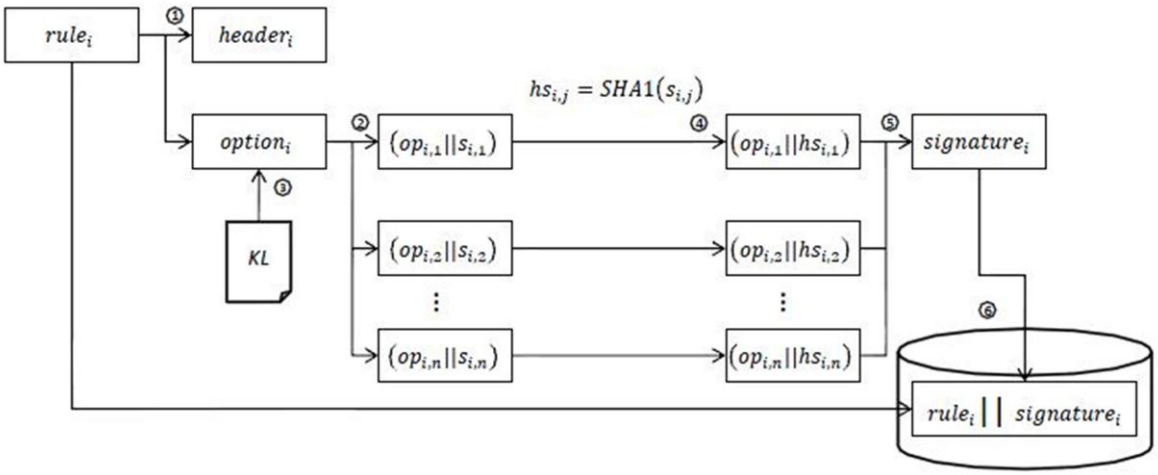


Figure 14: Rule Conversion

Contents of directory /user/hadoop/signatures

Goto : /user/hadoop/signatures go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
Sig_attack-responses.rules	file	2.95 KB	2	64 MB	2011-05-27 16:58	rw-r--r--	hadoop	supergroup
Sig_backdoor.rules	file	205.18 KB	2	64 MB	2011-05-27 16:58	rw-r--r--	hadoop	supergroup
Sig_bad-traffic.rules	file	0.52 KB	2	64 MB	2011-05-27 16:58	rw-r--r--	hadoop	supergroup
Sig_blacklist.rules	file	24.72 KB	2	64 MB	2011-05-27 16:59	rw-r--r--	hadoop	supergroup
Sig_botnet-cnc.rules	file	15.15 KB	2	64 MB	2011-05-27 16:59	rw-r--r--	hadoop	supergroup
Sig_chat.rules	file	14.41 KB	2	64 MB	2011-05-27 16:59	rw-r--r--	hadoop	supergroup
Sig_content-replace.rules	file	6.38 KB	2	64 MB	2011-05-27 16:58	rw-r--r--	hadoop	supergroup
Sig_ddos.rules	file	6.23 KB	2	64 MB	2011-05-27 16:58	rw-r--r--	hadoop	supergroup
Sig_dns.rules	file	6.13 KB	2	64 MB	2011-05-27 16:58	rw-r--r--	hadoop	supergroup
Sig_dos.rules	file	10.33 KB	2	64 MB	2011-05-27 16:59	rw-r--r--	hadoop	supergroup
Sig_exploit.rules	file	73.68 KB	2	64 MB	2011-05-27 16:59	rw-r--r--	hadoop	supergroup
Sig_finger.rules	file	1.87 KB	2	64 MB	2011-05-27 16:59	rw-r--r--	hadoop	supergroup
Sig_ftp.rules	file	13.95 KB	2	64 MB	2011-05-27 16:59	rw-r--r--	hadoop	supergroup
Sig_icmp-info.rules	file	34.11 KB	2	64 MB	2011-05-27 16:58	rw-r--r--	hadoop	supergroup
Sig_icmp.rules	file	4.88 KB	2	64 MB	2011-05-27 16:58	rw-r--r--	hadoop	supergroup
Sig_imap.rules	file	12.51 KB	2	64 MB	2011-05-27 16:59	rw-r--r--	hadoop	supergroup
Sig_misc.rules	file	18.86 KB	2	64 MB	2011-05-27 16:59	rw-r--r--	hadoop	supergroup
Sig_multimedia.rules	file	3.63 KB	2	64 MB	2011-05-27 16:58	rw-r--r--	hadoop	supergroup
Sig_mysql.rules	file	6.27 KB	2	64 MB	2011-05-27 16:58	rw-r--r--	hadoop	supergroup
Sig_netbios.rules	file	127.76 KB	2	64 MB	2011-05-27 16:59	rw-r--r--	hadoop	supergroup

Figure 15: Signature in the Cloud

5.3 Preserving-Privacy

In Preserving-privacy scheme, there are two roles: client and server. The client has to collect packets and do ciphertext generation. When the client receives the candidate list, the client must compare the query and the candidate list wherever the query is in the signature database. The server has to receive the client's query, do signature matching and response the candidate list to the client.

5.3.1 Client

The client is a program run on the client side (user's computer or PDA). It connects to the (cloud) server or procedure which is installed in the user's computer. It has the following functions.

Functions
1. Packet collection: it collects packets *X* from traffics.
2. Packet conversion: the extracted packet *X* is converted into a predefined pattern *W*.
3. Ciphertext generation: the predefined pattern *W* is converted into a ciphertext *C*.
4. Uploading: it uploads *C* into the cloud server of signature search.
5. Decision: After receiving a candidate list from the server, it finds out whether the pattern *W* is in the candidate list by inspecting the last part of each string in *L*.

As mentioned above functions, we implement the client program using java. We use "jpcap" library to implement packet collection. The library "jpcap" can collect network traffics

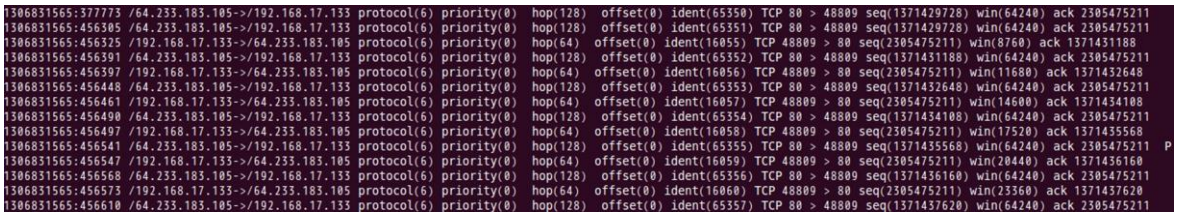


Figure 16: Function of Packet Collection

Figure 17: Function of Ciphertext Generation

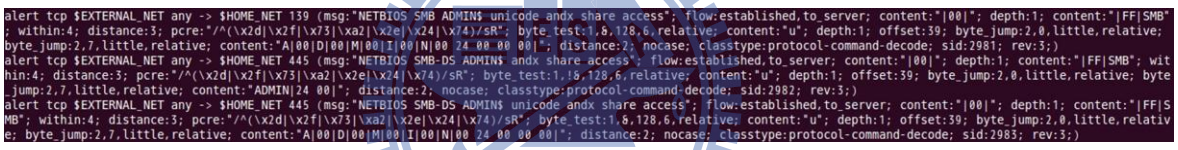


Figure 18: Function of Decision

using java. After that, we choose SHA-1 hash function as convert function to achieve packet conversion function. According to preserving-privacy scheme and socket programming, we can complete ciphertext generation and uploading functions. In order to finish decision function, we must store the extracted packets in user-side.

5.3.2 Server

The server is a program run on the cloud server side. It provided the basic ability for users to detect malicious attack and inspect system information of users. Additionally, it cannot gain extra information which is related to users. It has the following functions.

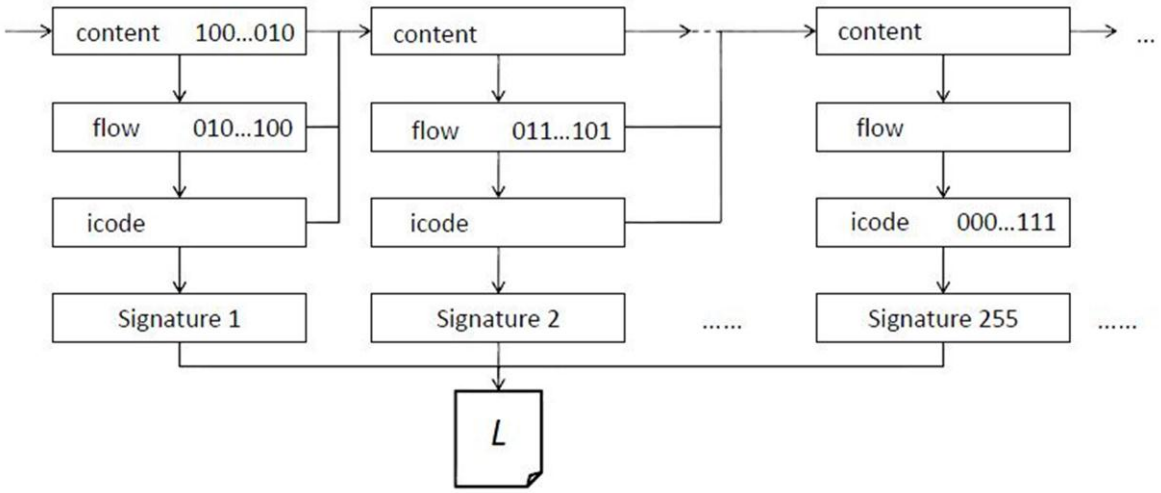


Figure 19: The Approach of Signature Matching

Functions
1. Signature database maintenance: signature-based IDS need a large signature data for malicious checking. The client shall maintain such a database from many sources. We initially put the signatures converted from the SNORT rules into the database.
2. Signatures matching: it finds candidate signatures from the signature database to match the ciphertext query from the server. The main technique is to use the MapReduce function in Hadoop in order to handle massive computation.
3. Reply: it sends back the candidate list *L* to the client.

As mentioned above, we start to implement our server program using java. We construct signature database and put it in the cloud. Furthermore, we use MapReduce programming model to finish Signatures matching function and generate candidate list. Finally, we reply candidate list according to socket programming.

6 Evaluation

We construct our experimental environment according above mentioned section. Now we do experiments to evaluate some criteria, such as privacy, detection rate and performance.

6.1 Privacy Analysis

We want to use the preserving-privacy scheme to keep privacy of the query of the client. That is, we want the cloud server to detect whether a malicious signature is inside the query and the server does not know what the query is.

6.1.1 The Key on Privacy

We want to test the relationship of the key $K=a||h(a)||r$ of the preserving-privacy scheme and the candidate list L. We predict that the privacy is indeed dependent on the length of $h(a)$. That is, the longer length of $h(a)$, the fewer of the privacy.

1. Experiment illustration

   We use rule conversion to convert SNORT rule to the signatures. In these signatures, we focus on keyword "content". We collect all signatures which contain the keyword "content", it is called $S_{content}$. There are 9755 types in $S_{content}$, it is denoted $|S_{content}|=9813$. The length of $h(a)$ is s-bits. We use preserving-privacy scheme and above mentioned section to do 100 times the experiments. In the experiment, we test number of candidate list according to the length $h(a)$. Finally, we compute the average number of candidate list about the keyword "content".

2. Experiment result

   The "length" is denoted the length of $h(a)$. The "estimate" is denoted our predicted

engthh	s = 1	s = 2	s = 3	s = 4	s = 5	s = 6	s = 7	s = 8
estimate	4877.5	2438.8	1219.4	609.69	304.84	152.42	76.21	38.11
number	4847.3	2424.1	1211.2	605.41	303.09	151.56	75.84	37.85
length	s = 9	s = 10	s = 11	s = 12	s = 13	s = 14	s = 15	s = 16
estimate	19.05	9.53	4.76	2.38	1.19	0.60	0.30	0.15
number	18.99	9.55	4.80	2.37	1.21	0.60	0.29	0.15

...

Table 2: The Result of the Key on Privacy

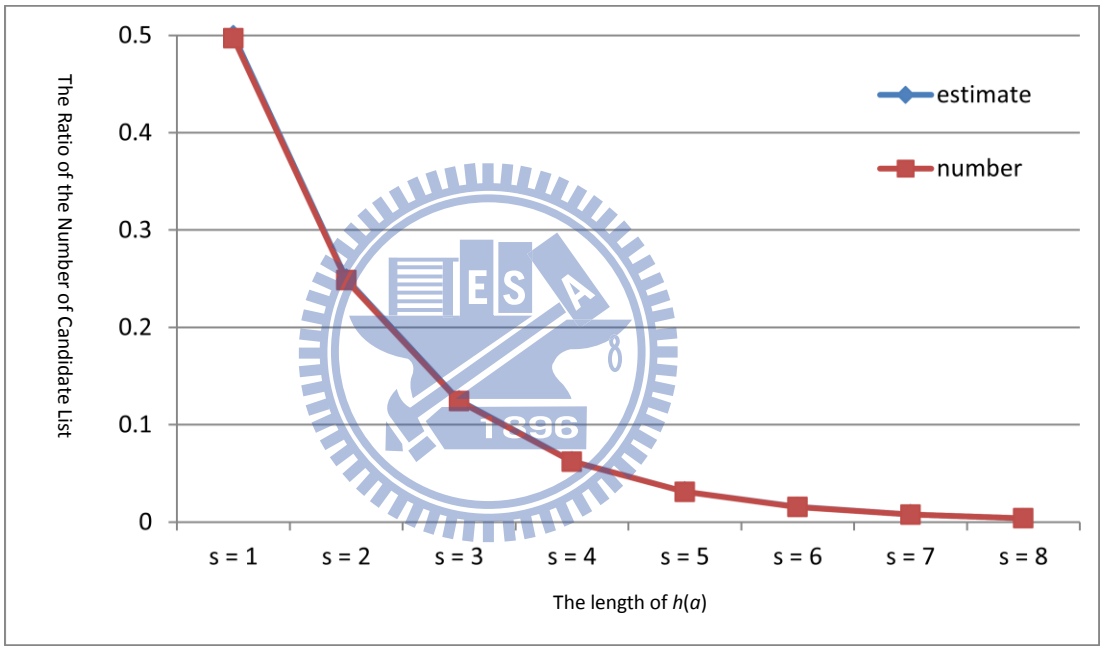


Figure 20: The Figure of the Table 2

number of candidate list about keyword "content". The "number" is denoted our experiment result about number of candidate list.

We send a query that contains keyword "content". We configure the random number *a* is 40-bits and *h*(*a*) is 1 to 16 bits. The number of candidate list is decreased approximated by 50% when the *s* increases. The results show that the privacy is indeed dependent on the length *s*, as predicted by our scheme.

6.1.2 The Privacy of the Signatures

In the SNORT rule, there are a lot of options in a one rule. We want to test the relationship between the number of the keywords and the privacy.

1. Experiment illustration

   We choose "content", "flow" and "icode" as the keywords. There are 8862 rules in SNORT. Let S_1 be the set of rules that contains "content", which specify the signature to search for triggering actions. $|S_1|$=8683. That is, only 179 rules do not specify "content". Among the 179 rules that do not contain "content", 118 rules do not contain "flow", which specify direction of packets. Let S_2 be such a set, where $|S_2|$=61. Among the 118 rules, 39 rules do not contain "icode", which denotes a specific ICMP value. Let S_3 be such a set, where $|S_3|$=79. The experimental flow is as following.

The Experimental Flow

---

1. The query is tested by all signatures and is filter by the keyword "content". It generates candidate list L_1.
2. The query is tested by candidate list L_1 and is filter by the keyword "flow". It generates candidate list L_2.
3. The query is tested by candidate list L_2 and is filter by the keyword "flow". It generates candidate list L_3.

---

2. Experiment result

   The "length" is denoted the length of $h(a)$. The "L_1" is denoted our experiment result about number of candidate list which is filter by the keyword "content". The "L_2" is denoted our experiment result about number of candidate list which is filter by the

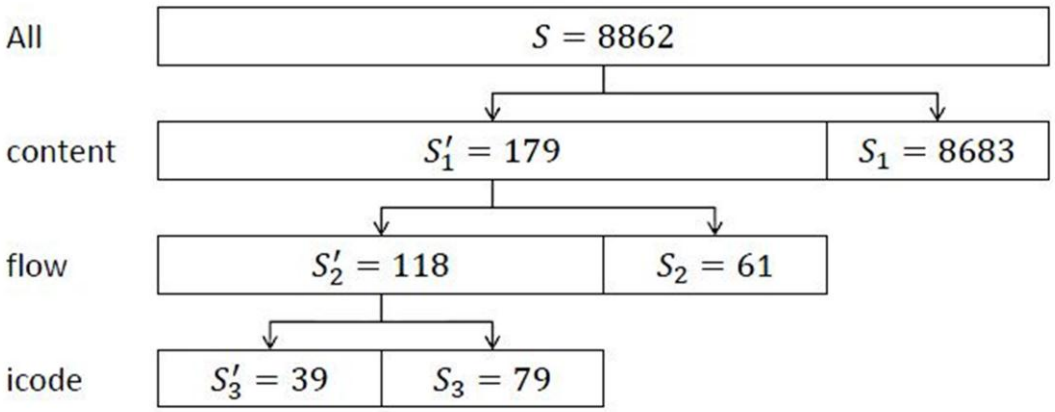


Figure 21: The Number of Filter

length	L_1	L_2	L_3
1	5351.4	2846.0	2812.1
2	3146.4	974.0	934.6
3	1803.5	580.2	530.9
4	1051.9	238.4	184.8
5	620.2	163.7	112.9
6	387.5	126.5	71.7
7	317.3	120.5	65.5
8	228.7	119.9	65.0

Table 3: The Result of the Privacy of the Signatures

keyword "flow". The "L_3" is denoted our experiment result about number of candidate list which is filter by the keyword "icode".

The results also show that the privacy is indeed dependent on the length *s*. Furthermore, we can choose some filter to determine the number of candidate list. The table shows that the L_1 and L_2 decrease obliviously because the number of "content" and "flow" is a large proportion of signatures. In the other hand, the L_3 decreases a little.

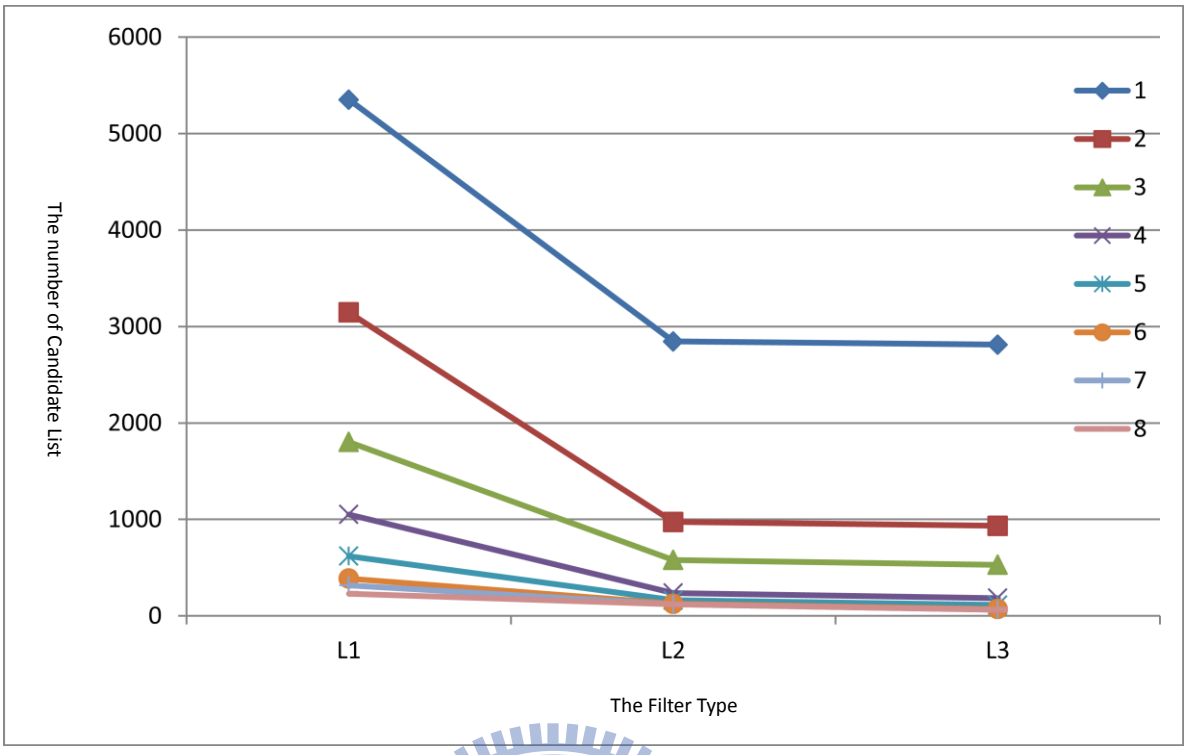


Figure 22: The Figure of Table 3

6.2 Detection Rate Analysis

The detection rate represents that the system can detect malicious attack ability. The stronger intrusion detection system usually has higher detection rate to against malicious attacks. In rule-based intrusion detection system, the more known attacks rules usually has stronger detection rate. In the experiment, we want to test the preserving-privacy scheme which is using in the SNORT rule whether affect the detection rate.

DARPA 98 is a network traffic dataset. It contains sample network traffic for the 1998 DARPA Intrusion Detection Evaluation. The dataset use command "tcpdump" to collect network traffic. Tcpdump was started with the command as following, where <datafile> is the name of the dump file.

tcpdump -s 66000 -F options -w <datafile>

In this experiment, we use the datafile that is called "sample_data01".

We use the dataset as an input and use SNORT to inspect the dataset. The "sample_data01" is a ".tcpdump file". So we can use SNORT command as following.

```
snort –r sample_data01
```

We can check the "alert" file to find that how many rules does the SNORT can detect malicious attacks about the dataset - "sample_data01".

In the other hand, we have to use "sample_data01" to test our preserving-privacy scheme. First, we must fetch "sample_data01" according to the keyword, such as "content", "flow" and "icode". In the "sample_data01", there are 14523 network traffics and 15 protocols. We catch the hexadecimal expression of the network traffic and convert to our input format according to the standard of the network protocol.

The fetched network traffics are the format of "00 0c 04 41 bc ……".

The Conversion Approach

1. Check the network traffics belong to which protocol.
2. Convert the network traffics according to network protocol format.
3. Follow the above mentioned section to generate ciphertext and finish the preserving privacy scheme.

We use the number of the finding malicious attacks as the detection rate. The following table shows that the SNORT and the preserving scheme have the same detection rate.

	Number of malicious Attacks
SNORT	31
Preserving Privacy Scheme	31

Table 4: The Result of the Detection Rate

6.3 Performance Analysis

We estimate the time about packet collection, signatures matching and decision. In this case, we suppose network is high speed and no delay. The experiment environments follow above mentioned sections.

1. The time of packet collection

We collect packets using Wireshark, our client program without doing ciphertext generation and our client program with doing ciphertext generation. The network traffics come from using browser to connect Google and using command ping to send ICMP packets to Google. We connect Google using browser which has a lot of network traffics. In the other hand, we use command ping to decide number of ICMP packets. However, using ping to send ICMP packet is slow according to system initial configuration. The following table specifics the estimated results.

Tool	The Time of using browser to connect Google	Time of using command ping to send ICMP packet to Google
Wireshark	37 ms	0.5 sec
Without Ciphertext Generation	52 ms	0.5 sec
With Ciphertext Generation	86 ms	0.5 sec

Table 5: The Time of Packet Collection

Wireshark is famous software of packet sniffer. We use it to collect packets about connecting to "Google". The time of Wireshark compares our client program is soon. Clearly, the packets with doing ciphertext generation are slow because it may be computed by SHA-1 hash function and analysis by keywords. In the other hand, all of the tools which collect ICMP packets have the same time because of system configuration of

command ping. That is, the client program can finish packet collection and ciphertext generation with the reasonable time according to the network traffics speed.

2. The time of signatures matching

We use MapReduce model to do signatures matching and rules matching on Hadoop. The signatures matching have been finished preserving privacy scheme and the signatures also have been converted. The rule matching is like signatures matching, but it is without doing preserving privacy scheme. Furthermore, we also use Hadoop to run some programs to test the execution time. The word count is a program that computes the number of word of the text. The char count is a program that computes the number of chart of word of text. The pi estimator is a program that computes pi's value.

	size	time
Signatures Matching	8862 signatures	24 sec
	100 signatures	24 sec
	1 signature	24 sec
Rules Matching	8862 rules	23 sec
	100 rules	23 sec
	1 rule	23 sec
Word Count	2 MB	24 sec
	1 KB	22 sec
Chart Count	2 MB	25 sec
	1 KB	23 sec
Pi Estimator	X	23 sec

Table 6: The Time of Signatures Matching

According to the Table 6, we find that the execution time of any program on Hadoop is more than 20 seconds. That is, we compare the preserving privacy scheme and other

programs on Hadoop. The preserving privacy scheme does not seem to be wasted too much time. However, the results show that the preserving privacy scheme running on Hadoop is not practical.

3. The time of decision

We estimate the time of matching the candidate list when the client receives the candidate list. We can choose the size of candidate list according to the length of $h(a)$. The "length" is denoted the length of $h(a)$. The "size" is denoted the size of candidate list. The "time" is denoted the matching time. We estimate matching time according to the size of candidate list as following.

Length	Size	Time
4	179 rules	50 ms
8	68 rules	17 ms
16	40 rules	11 ms

Table 7: The Time of Decision

The more rules, the more time is. Clearly, the number of candidate size is large, the decision time is more.

Finally, we conclude the time of performance. The ciphertext generation time is relation to the network traffic and the time is less than 100ms. The decision time is relation to the size of candidate list and the time is less than 50ms. However, the signatures matching time on Hadoop wastes a lot of time. It at least use more than 20 sec. Hence, it is not practical.

6.4 Discussion

Our approach's detection rate relates to the SNORT rules. In fact, the SNORT rules actually have the weaker detection rate in practice. However, we focus on the privacy preserving issue. That is, we do no increase the detection rate. If we want to have the better detection rate, we have to select powerful rule set as our signatures.

We use three machines and Java to conduct our experiment. In this architecture, the signature matching spends a lot of time. The experiments only provide service to one user in the same time. We may use service load balance mechanism and the more machines to replace Hadoop. Of course, we can select C to replace Java to write the program.

We inspect every packet that is collected by client-side in the cloud server. It is not practical in the real world because of the huge flow. We must to add "first checking mechanism" in the client-side. It decreases the number of the packets to e inspected by the cloud server. Finally, we may select the better filter to inspect the packets. In this way, we can make more efficient signature matching methods to inspect useful information.

7 Conclusion

We use the preserving privacy scheme to protect user's privacy in the cloud service. The privacy is achieved by finding out a set of possible candidates in the signature database so that the server does not know which one is real. Our technique is based on hidden keyword search developed in the cryptography field.

We use Hadoop to construct experimental cloud experiment. We also convert SNORT rule to the signatures database. The signatures are converted according to SHA-1 hash function. We use jpcap library and java to implement the preserving privacy scheme that is client-server architecture. Finally, we use these tools to do experiments and evaluate the privacy, detection rate and performance.

The experiment results show that the privacy is dependent on the length of the $h(a)$. We can choose three filters which contain "content", "flow" and "icode", as the signatures matching keywords. However, there are 39 rules which do not do signatures matching. That is, the 39 rules are always in candidate list. In the other hand, the preserving privacy scheme does not affect the detection rate of the SNORT. Finally, we find that the signatures matching which runs on Hadoop waste a lot of time. It is not practical in intrusion detection system.

In the further, we maybe improve the performance and the stability in our experimental environments. Of course, we may test the preserving privacy scheme with the different conditions. Finally, we hope the approach can be suitable in the real intrusion detection system in our life.

Reference

[1] M. Armbrust, A. Fox, R Griffith, A. Joseph, R Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “A view of cloud computing.” in Communications of the ACM, Vol. 53, No. 4. April 2010, pp. 50-58.

[2] D. Boneh, G. D. Crescenzo, R. Ostrovsky and G. Persiano, “Public Key Encryption with Keyword Search.” in proceedings of Eurocrypt 2004, LNCS 3027, 2004, pp. 506-522.

[3] F. Chang, J. Dean, S. Ghemawat, W. Hsieh, D. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. Gruber, “Bigtable: A distributed storage system for structured data.” in Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation, 2006.

[4] J. Dean, and S. Ghemawat, “MapReduce: Simplified data processing on large clusters.” in Proceedings of Operating Systems Design and Implementation, 2004, pp.137-150.

[5] M. J. Freedman, Y. Ishai, B. Pinkas, O. Reingold, “Keyword search and oblivious pseudorandom functions.” in 2nd Theory of Cryptography Conference, volume 3378 of LNCS, 2005, pp 303-324.

[6] S. GHEMAWAT, H. GOBIOFF, and S. -T. LEUNG, "The google file system." in Proceedings of the nineteenth ACM symposium on Operating systems principles, 2003, pp. 29–43.

[7] C. Hazay and Y. Lindell, “Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries.” in 5th Theory of Cryptography Conference, volume 4948 of LNCS, 2008, pp. 155-175.

[8] R. A. Kemmerer, G. Vigna, "Intrusion detection: a brief history and overview." Computer Volume: 35 Issue: 4, 2002, pp.27 - 30.

[9] P. Mell and T. Grance, "The NIST Definition of Cloud Computing." in NIST, 2009.

[10] J. Oberheide, E. Cooke, and F. Jahanian, “CloudAV: N-Version Antivirus in the Network Cloud.” in Proc. of the 17th USENIX Security Symposium, July 2008.

[11] J. Oberheide, E. Cooke, and F. Jahanian, “Rethinking Antivirus: Executable Analysis in the Network Cloud.” in USENIX Workshop on Hot Topics in Security, August 2007.

[12] J. Oberheide, K. Veeraraghavan, E. Cooke, J. Flinn, and F. Jahanian, “Virtualized In-Cloud Security Services for Mobile Devices.” in Workshop on Virtualization in Mobile Computing, June 2008.

[13] M. Roesch, “Snort - Lightweight Intrusion Detection for Networks.” in USENIX 13th Systems Administration Conference, 1999.

[14] D. X. Song, D. Wagner, and A. Perrig, “Practical Techniques for Searches on Encrypted Data.” in IEEE Symposium on Security and Privacy, 2000.

[15] J. Venner, Pro Hadoop. Apress, June 22, 2009.

[16] T. White. 2009. Hadoop: The Definitive Guide. O’Reilly Media, Inc. June 2009.