Journal of Software Engineering and Applications, 2013, 6, 51-57
doi:10.4236/jsea.2013.63b012 Published Online March 2013 (http://www.scirp.org/journal/jsea)

Design and Implementation of Secure Subnet Inside of Data Sensitive Network

Haiwei Xue, Yunliang Zhang, Zhien Guo, Yiqi Dai

Department of Computer Science, Tsinghua University, Beijing China.
Email: heavyxue@gmail.com

Received 2013

ABSTRACT

Sensitive data leak can cause significant loss for some organizations, especially for technology intensive companies and country security departments. Traditional mandatory access control (MAC) can only control whether the user can access the sensitive data or not, and cannot prevent the user to leak or spread the data. So even designed impeccable access control policies, we still cannot prevent inside leak. A nature solution is using physical isolation to prevent sensitive data from being leaked outside network; however inside the physical isolated network, data still can be spread from one subnet to another. We present Secure Subnet System, a BLP model base security system that can provide more strong access control, which is called mandatory action control. In our system after a user read sensitive data, system will dynamically change security policies to prevent the user to leak these data or spread the data outside to another subnet. We use a state machine model to describe our system, and use secure transfer equations to dynamically calculate the system policies for each new state. Our model can be proved to be secure by formal methods. We implemented a demon of our system. In this paper we also show the design details of the demon and evaluate the demon both from security and performance. The evaluation results show that the output of the security tests case are under expected; and the performance test case show that, for the 64KB IO chunk size, IO read loss can be improved to 6.6%, IO write loss can be improved to 1.2% after optimization.

Keywords: Component; Privacy; Netwrok Security; Access Control; Inside Leak; Security Model; BLP

1. Introduction

Sensitive data is always one the most important resource for technology intensive companies and organizations, especially for the military and country's information security departments. For these departments any sensitive data leak could cause significant loss. There two common data privacy protection models in sensitive network: (1) using physical isolation between outside and inside network [1-3]; (2) using access control in inside network [4]. These designs can let the sensitive data be good controlled under the inside network and cannot goes outside.

While in some particular case, these protection models may not enough. For example inside network usually will be divided into several internal networks, which we call secure subnets. These secure subnets need to be isolated for they have different sensitive data and functions; while they also need to be connected for they have to cooperate with each other. For these secure subnets, it is hard for us to follow the common protection models: (1) if we use physical isolation among these subnets, it will be hard for information communication. In some extreme situations each computer could be a subnet, if we use physical isolation it means that there is no network. (2) if we use access control, we will face up to two problems; for one thing we need to define complex access policies, which can be a hard work when there are too many secure subnets cooperated with each other; for another even if we successfully defined these policies it cannot prevent inside leak. E.g. in the role based access control [4] if one user belong to subnet 1 and has a specified role in subnet 2; she can access some sensitive data in subnet 2; In this situation, the access control policy can only control whether the user can access the sensitive data or not, and cannot prevent the user to spread the sensitive data from subnet 2 to subnet 1.

In this paper we designed and implemented a security model for these subnets which cannot use physical isolation and still need to prevent information spread. In our model we labeled sensitive data in each subnet with a Security Level (SL); and each user has two security levels: Current Security Level (CSL) which means the largest data SL the user has accessed; Maximum Security Level (MSL) which means the largest data SL the user can access. In our system each user will start its computer remotely to make sure that they don't have any sensitive data at first. Users' CSL start at 0, and with the access of the different sensitive data, the CSL the user will dynamic changing. The basic rule in our model is "not reading forwards, not writing backwards" which means: (1) a user cannot read files whose SLs are larger than its MSL. (2) a user cannot write files whose SLs are smaller than its CSL. Our model is base from multilevel security models [5,6].

2. Design Overview

In our model the network is comprised of Subnet and Security Subnet Center (SSC), as it is shown in **Figure 1**.

Subnet: An internal network where their sensitive data belongs to the same categories. In the real world, a subnet usually means a specific department network. In a security subnet system there is always more than one subnet. In each subnet there is a Secure File Server (SFS) which stores all the sensitive data belongs to its own, and there are also other common devices, such as PCs, which can be used by the internal users. Those PCs in subnet are all set to be started from the OS server in the SSC. In our system, when PCs are powered on they will download the OS image form OS server and started by the technology of Intel PXE [7].

Security Subnet Center (SSC): SSC is comprised of Operation System Servers and Policy Servers. OS Servers provide the remote starting service. Policy Servers are running the policy logic modules of the system which defined access rules of the system.

In our Secure Subject System, every Subject, as a user, is presented by <name, MAC Address, Subnet ID, CSL, MSL>; every Object, as a secure file, is presented by <Path Name, Subnet ID, SL>. When a Subject start up, the Subject do not have any sensitive data, so the Subject can do anything like a common PC, e.g. read/write U-Disk and local disk, using network and etc. When the Subject read a sensitive file from its SFS, the behavior of this PC will be constrained. E.g. it cannot write to U-Disk and local disk to prevent the Subject copy the file from SFS to it local storage.

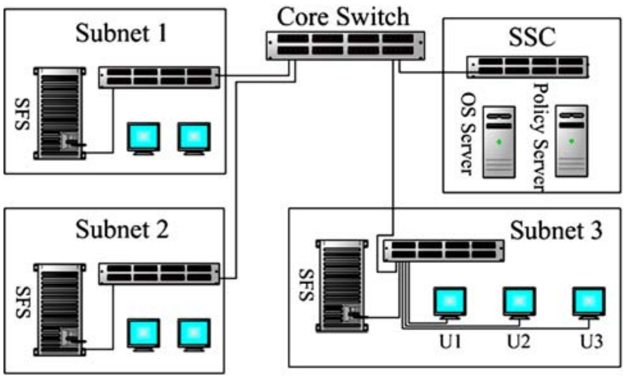

Figure 1. An inside network with three subnets.

So, the basic idea of our system is that: dynamic changing the system policies. We have a group of system policies for the current state of the system which can guarantee that there is no information leak under these policies; once the system state changed, these policies will be changed automatically. So if we using state machine to describe the subnet system state, and using state transfer equations to dynamic adjust the system policies, then we can guarantee the security as long as we carefully designed the security state transfer equation.

3. System Model and its Security Analysis

S: A Subject set, which is comprised of users in the subnets. Every Subject has a property n which denote its subnet ID.

O: A Object set, which is comprised of sensitive files in the SFS.

J: Security level functions, where $f^n(O)$ denotes the security level functions of object o in subnet n. By default, $f(O)$ is the SL of object o in its host subnet; $f_c^t(S)$ is the CSL of subject *s* in its host subnet at time *t*; $f_m(S)$ is the MSL of subject *s*. When the parameter of f is a subject, the value of f is the security level of the subject in its host subnet.

A: Access attributes set. $A = A_o \cup A_s \cup A_\varphi$, where A_o is the attributes set that subjects access objects, $A_o = \{r,a,w\}$, and r means read, a means write, w means read-write; A_s is the attributes set that subjects access subjects, $A_s = \{s\}$ and *s* means send; A_φ is the self attributes set, $A_\varphi = \{rst\}$ and rst means reset.

T: Access time set.

b: Current access set. $b = b_o \cup b_s \cup b_\varphi$ where b_o is the access set that subjects access objects, and b_s is the access set that subjects access subjects,

$$b_s = S_i \times S_j \times A_s \times T, i \neq j$$

and b_φ is the self access set.

M : Access matrix. $M_{ij} \subseteq A$.

3.1. Security Properties

Security properties define how a security state should be. Once a system state satisfies security properties, then the state is secure and there is no information leak in this state. We extend BLP [8,9] model's security properties to meet the network environment.

SS-Property: State $v = (b, M, f)$, $x \in A_o$ satisfy SS-Property, if and only if $\forall(s,o,x,t) \in b$, *s* belong to subnet *n* there are:

$$x = r \Rightarrow f_m(s) \geq f^n(o)$$

$$x = a \Rightarrow f_m(s) \geq f^n(o)$$

$$x = w \Rightarrow f_m(s) \geq f^n(o)$$

*-Property: State $v = (b, M, f)$, $x \in A_o$ satisfy *-Property, if and only if $\forall(s,o,x,t) \in b, s \in S \setminus S^*$, *S* are trusted sub-

jects and s belong to subnet n there are:

$$x = r \Rightarrow f_c^t(s) \ge f^n(o)$$

$$x = a \Rightarrow f_c^t(s) \le f^n(o)$$

$$x = w \Rightarrow f_c^t(s) = f^n(o)$$

Network-Property: State $v = (b, M, f)$, $x \in A_s$ satisfy network-Property, if and only if $\forall (s_1, s_2, x, t) \in b, s_1, s_2 \in S \setminus S^*$, $s_1 \ne s_2$ and S are trusted subjects and s_1 belong to subnet n_1, s_2 belong to subnet n_2 there is:

$$x = s \Rightarrow f_c^t(s_1) \le f_c^t(s_2) \wedge n_1 = n_2$$

DS-Property: State $v = (b, M, f)$, $x \in A_o \cup A_s$ satisfy DS-Property, if and only if $\forall (s, o, x, t) \in b$, there is:

$$x \in M_{ij}$$

$\forall (s_i, s_j, s, t) \in b$, $s_i \ne s_j$, there is

$$\forall o_k \in O, M_{ik} \subseteq M_{jk}$$

Not Descending-Property: State $v = (b, M, f)$, $x \in A_o \cup A_s$ satisfy Not Descending-Property, if and only if $\forall s_1, v_1 = (s_1, o_1, x_1, t_1) \in b$, $v_2 = (s_1, o_2, x_2, t_2) \in b$, $t_1 \le t_2$ and $\{x = rst, t_1 \le t \le t_2 \mid b\} = \varphi$, s_1, s_2 belong to a same subnet, there is

$$f_c^{t_1}(s_1) \le f_c^{t_2}(s_1)$$

3.2. Security State Transfer Equations

We defined five state transfer equations to meet the operations in network. Suppose the last network operation occurs at time t and initially set $t = 0$, $f_c^0(S) = 0$, which means that the initial CSL of a subject is 0. Peculiarly if $f^n(O)$ is not defined, then $f^n(O) = \infty$.

Rule 1: Read only. Suppose current state $v = (b, M, f)$, $s \in S \setminus S^*$, S^* are trusted subjects, s belong to subnet n, for the request $R_k = (s_i, o_j, r)$:

```
IF $f_m(s_i) \ge f^n(o_j) \wedge r \in M_{ij}$ THEN
    SET $t_0 = Current\ Time$
    SET $f_c^{t_0}(s_i) = max(f_c^t(s_i), f^n(o_j))$
    SET $f^* = f \cup \{f_c^{t_0}(s_i)\}$
    SET $b^* = b \cup \{(s_i, o_j, r, t_0)\}$
    SET $t = t_0$, $v = (b^*, M, f^*)$
    PERMIT $R_k$
ELSE DENY $R_k$
```

Rule 2: Write only. Suppose current state $v = (b, M, f)$, $s \in S \setminus S^*$, S^* are trusted subjects, s belong to subnet n, for the request $R_k = (s_i, o_j, a)$:

```
IF $f_m(s_i) \ge f^n(o_j) \wedge f_c^t(s_i) \le f^n(o_j) \wedge a \in M_{ij}$ THEN
SET $t_0 = Current\ Time$
SET $b^* = b \cup \{(s_i, o_j, a, t_0)\}$
SET $t = t_0$, $v = (b^*, M, f)$
PERMIT $R_k$
ELSE DENY $R_k$
```

Rule 3: Read-Write. Suppose current state $v = (b, M, f)$, $s \in S \setminus S^*$, S^* are trusted subjects, s belong to subnet n, for the request $R_k = (s_i, o_j, w)$:

```
IF $f_m(s_i) \ge f^n(o_j) \wedge f_c^t(s_i) \le f^n(o_j) \wedge w \in M_{ij}$ THEN
SET $t_0 = Current\ Time$
SET $f_c^{t_0}(s_i) = f^n(o_j)$
SET $f^* = f \cup \{f_c^{t_0}(s_i)\}$
SET $b^* = b \cup \{(s_i, o_j, w, t_0)\}$
SET $t = t_0$, $v = (b^*, M, f^*)$
PERMIT $R_k$
ELSE DENY $R_k$
```

Rule 4: Send. Suppose current state $v = (b, M, f)$, $s \in S \setminus S^*$, S^* are trusted subjects, s_1 belong to subnet n_1, s_2 belong to subnet n_2, for the request $R_k = (s_i, s_j, s)$, $i \ne j$:

```
IF $f_c^t(s_i) \le f_c^t(s_j) \wedge \left[\forall o_k \in O, M_{ik} \subseteq M_{jk}\right] \wedge n_1 = n_2$ THEN
SET $t_0 = Current\ Time$
SET $b^* = b \cup \{(s_i, s_j, s, t_0)\}$
SET $t = t_0$, $v = (b^*, M, f)$
PERMIT $R_k$
ELSE DENY $R_k$
```

Rule 5: Read-Write. Suppose current state $v = (b, M, f)$, for the request $R_k = (s_i, \varphi, rst)$:

```
SET $t_0 = Current\ Time$
SET $f_c^{t_0}(s_i) = f_c^0(s_i)$
SET $f^* = f \cup \{f_c^{t_0}(s_i)\}$
SET $b^* = b \cup \{(s_i, \varphi, rst, t_0)\}$
SET $t = t_0$, $v = (b^*, M, f^*)$
PERMIT $R_k$
```

3.3. Security Proofs of Transfer Equations

For a secure transfer equation, given a secure input state the output of the equation is also secure. All the five equations in this paper can be proved to be secure. In considerations of space we only give the proof of "Rule 1 keep SS-Property" in this paper.

Proofs of 'Rule 1 Keep SS-Property': Suppose input state $v=(b,M,f)$ meets SS-Property, for the request $R_k=(s_i,o_j,r)$; after executed Rule 1, we get the new state v^*; from rule 1 we know $v^*=v$ or $v^*=(b^*,M,f^*)$; if $v^*=v$, then the output state meets SS-Property; else $v^*=(b^*,M,f^*)$, then from Rule 1 we get $v^*-v=((s_i,o_j,r,t_0),M,f^*)$, and $f^*=f\cup\{f_c^{t_0}(s_i)\}$, and we also have $f_m(s_i)\geq f^n(o_j)$, so we get v^*-v meets SS-Property, finally we get v^* meets SS-Property.

From this session we can find that given a secure state as an input the output of these transfer equations are always secure. So if we can make sure that the initial input state is secure, and follow those transfer equations, then the system is always secure. The rest of the paper will focus on how to implement those transfer equations.

4. Implementation

The model of secure subnet system is base from state machine model. Each of the system states has specific system policies. Whenever the system state changed, we can calculate the system policies in the new system state from the transfer equations. So the implementation of the system can be dived into three parts: state changed signal Capture, calculate system policies and enforce the policies.

4.1. State Changed Signal Capture

From the transfer equations we can find that the factors that can affect transfer equations' execution results include: the SL of a Object: $f^n(O)$; the CSL of a Subject: $f_c^t(S)$; the MSL of a Subject: $f_m(S)$; the subnet id n and access matrix M. There are some other parameters in system state v and although the system state changes if they change, they don't affect transfer equations' execution results; so we don't capture those parameters change.

Among those factors that can affect transfer equations' execution results, $f^n(O)$, $f_m(S)$, n, M can be configured in advance, and the system's security administrator or director can define SL, MSL, subnet ID and access matrix. So we need only capture CSL. As we know CSL changes only when Subject access sensitive files. Finally we get that we need only capture the operations that Subject access sensitive Object.

In our paper, the terminals that Subject using are all common PCs, and the OS are Window XP or Windows 7. We add Windows files system filter driver in the Windows Kernel to capture all the files access operations. As it is shown in **Figure 2**, when a Subject open a file in user space by some particular program, the program will call a system routine *CreateFile, CreateFile* will send a system IO Request Packet (IRP) to kernel. This IRP will be captured by our driver whet it passes through system device stack. Our driver will send this IRP to an Agent in user mode, and then by this Agent, the IRP will be send to Policy Server by network. The Policy Server will analyze the IRP and decides whether to permit it or deny it and send the result back to Agent, and finally back to the Driver. In this procedure when the Policy Server got the IRP and finally made a decision, it already know whether CSL is changed or not, then it can update CSL.

State changed signal is capture by Driver, and actually the Driver is also simultaneously doing another work: enforces the policies, which will be introduced in the section C.

4.2. Calculate System Policies

Policies in Secure Subnet System are comprised of Network Policies (NPs) and Host Policies (HPs). NPs are defined to control Subjects' network operations, and prevent Subjects leak information by network; HPs are defined to control Subjects' file operations, and prevent Subjects leak information by local disk, etc.

In our system NPs and HPs are implemented in a different way. For NPs we need set Access Control List (ACL) in the switches. So we need calculate the ACLs according to system's current state and secure transfer equations. For HPs we can take the advantage of transfer equations and avoid complex algorithms to calculate host ACLs.

HPs are mainly implemented in Policy Server. As we know the system policies change only when system state changes; from the secure transfer equations we can know that system state change means one of those equations is execute the TRUE part of IF statement, which mean that the operation is permitted. So we can take the transfer equations as the system policy, although those equations never change, they change system state and system state will react on the equations. So the same IRP in different system state may get a different execution result, which

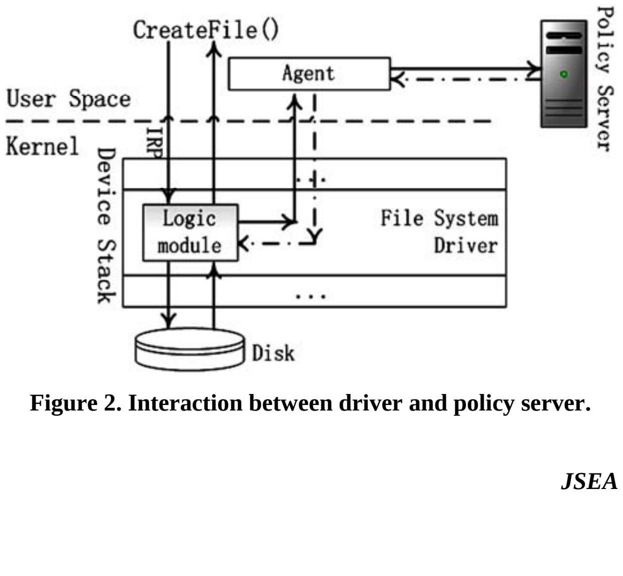

Figure 2. Interaction between driver and policy server.

means the system policies are dynamic changing. So we don't need to calculate the ACLs when system state changed, we only need simply execute the corresponding secure equations when Policy Server received an IRP, and the result of equations is the same as the ACLs.

NPs are mainly implemented in Policy Server and Switch Controller. In Software Defined Network [10] (SDN), the flow table can be control by controller. But in consideration of expense we did not use SDN supported switch, instead we implemented a CLI [11] based Switch Controller, which can connect to switch (in our implementation, we use H3C S3600) and change its ACLs by CLI.

Only Rule 4 is the network operation in secure equations. From Rule 4 we can know that the operation can be permitted only when the CLS $f_c^t(s_i)$ is not lager than receiver's CSL $f_c^t(s_j)$; and the sender and the receiver are in the same subnet. So we can group Subjects by its subnet ID and sort them in a ascending order by its CLS, as it is shown if Figure 3, when there is new Subject join to the list, firstly we sequentially find the insert point and then from the insert point:

- The current Subject can only read Subjects behind the insert point in the list. We need create real-only ACLs
- The current Subject can read and write Subjects whose CSL is equal to the current Subject. We need create read-write ACLs
- The current Subject can only write other subjects in the list. We need create write-only ACLs.

To minimize the number of ACLs, we use DENY as default ACL, which means that when a Subject's CLS is larger than 0, we acquiescently DENY all the operations of this Subjects, and then add ACLs that the Subject can access SFS, and add ACLs that the Subject can access OS Server and Policy Server, Finally we calculate the ACLs using the method as it is shown in **Figure 3**.

4.3. Policies Enforcement

We use a different way to enforce NPs and HPs. For NPs we use the ACL module in switch devices. Firstly we

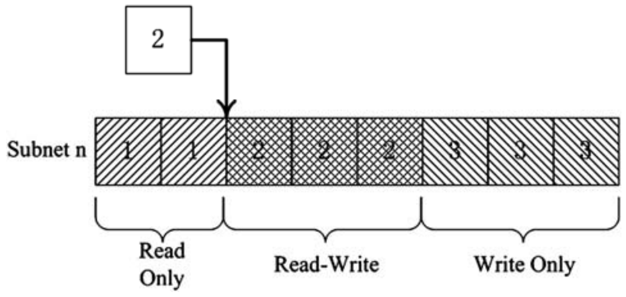

Figure 3. Algorithm for calculating ACLs of NPs.

calculate ACLs by Algorithm 1, and then send these ACLs to switches by CIL.CIL is introduced in most of the switch instructions, we did not talk in this paper. For HPs we capture the file operation requests by Driver, and then send these requests to Policy Server. Finally the Driver will decide whether to permit or deny this operation according to Policy Server's response.

The Subnet Driver is running in the file system device stack. When a program in user space want to access a file, it will create an IRP to the windows kernel; The IRP will pass through the device stack and processed by each driver in the stack; when the IRP is processed by Subnet Driver, the Driver will forward the request to Policy Server; since it is hard for the Driver to communicate with Policy Server by socket in the kernel, we implement a Agent in user space; the Agent read Subnet Driver request then forward it to Policy Server by socket; when Agent get the response it write the response to Subnet Driver. By this way, we can implement the policy logic in Policy Server and enforce the policies in Subnet Driver.

4.4. System Optimization

The Subnet Driver is a file system filter for windows, and it will affect system's IO throughput. In this section we will optimize our system to maximize systems IO throughput.

The Subnet Driver will forward the IRP to a remote Policy Server after captured a system IO request, which will cause a long latency. We can use the following method the decrease the latency:

- Local Cache: From the transfer equations we can find that Rule 2 did not affect to $f_c^t(S)$, and will not change system state; Rule 1 and Rule 3 are not always change the $f_c^t(S)$, only when $f_c^t(s_i) < f^n(o_j)$, it will affect $f_c^t(S)$, so we can cache the current Subject's $f_c^t(s_i)$ and all the Objects' SL; when the Driver captured a request it will forward the request to Policy Server to refresh the system's state only when $f_c^t(s_i) < f^n(o_j)$, otherwise it will directly process the request in window kernel.
- Security Label: Local Cache can decrease the forwarding of IRP, but the price is that it will use a lot of memory to support cache. In the worst case, the Diver needs to cache all the system's secure files' path names and SLs, which could be very large. It will cause two problems: (1) It will use a lot of physical memory.
(2) It need complex algorithm to index these path names. In this paper we use a common prefix to label all the sensitive files. E.g. for a sensitive file

whose SL is 2, the path name could be '\??\secret\c2\file.doc' where '\??\' is kernel prefix added by OS and 'secret\c2\' is our prefix; 'secret' means a sensitive file; 'c2' means the SL is 2; 'file.doc' is the file name. So by using the method we only need a four bytes integer to store the Subject's $f_c^t(S)$, the other information can be calculated by the path name.

5. System Measurement

5.1. System's Security Test

Test environment: Sensitive files are separately grouped in each subnet in three categories: L1, L2 and L3 (SL equals to 1, 2, 3). There are three users in subnet 3: U1, whose MSL is 1; U2, whose MSL is 2; U3, whose MSL is 3. There is a sensitive file in subnet 2, named '2_File_2.doc' whose SL is 2; this file wants to be shared with subnet 3, and only users in subnet 3 whose MSL is not less than 3 can access it. The administrator sends a request to Policy Server to share file '2_File_2.doc' in subnet 2 as a L3 file in subnet 3. We use the following test case to test our demon:

- U2 try to read file2 whose SL equals to 2 in subnet 3. Because U2 belongs to subnet 3, and its MSL is 2, so the expected result is success.
- U2 try to read file3 whose SL equals to 3 in subnet 3. Because the SL of the file is larger than U2's MSL, so the expected result is fail.
- After U2 successfully read file2, she try to read file1 whose SL equals to 1, the expected result is success; and then try to write file1, the expected result is fail; finally try to send data to U1, the expected result is fail.
- U2 try to read-write shared file 2_File_2.doc, the expected result is fail, because the SL of 2_File_2.doc in subnet 3 is 3.
- U3 try to read shared file 2_File_2.doc, the expected result is success. U3 try to write file 2_File_2.doc, the expected result is fail, because 2_File_2.doc is in the SFS of subnet 2, but U3 is in subnet 3 and have no permission to write subnet 2's SFS. U3 try to write file2, the expected result is fail, because after U3 read 2_File_2.doc its CSL goes to 3. U3 try to send data to a user in subnet 1, the expected result is fail.

All of these test case achieved it expected result in our test.

5.2. Performance Test

We implemented and tested the performance of the demon system. We use IO Meter to measure the IO throughput of the disk. The test object is a common PC running Windows XP system; the processor is Intel Pentium Dual E2140; and the disk is WD 1600 AAJS, 160GB SATA, 7200 rpm. We tested the sequential IO read and write access with size from 1KB to 64 KB. **Figure 4** show the IO throughput of original OS, with Subnet Driver before optimization and with Subnet Driver after optimization; we also given the IO throughput loss.

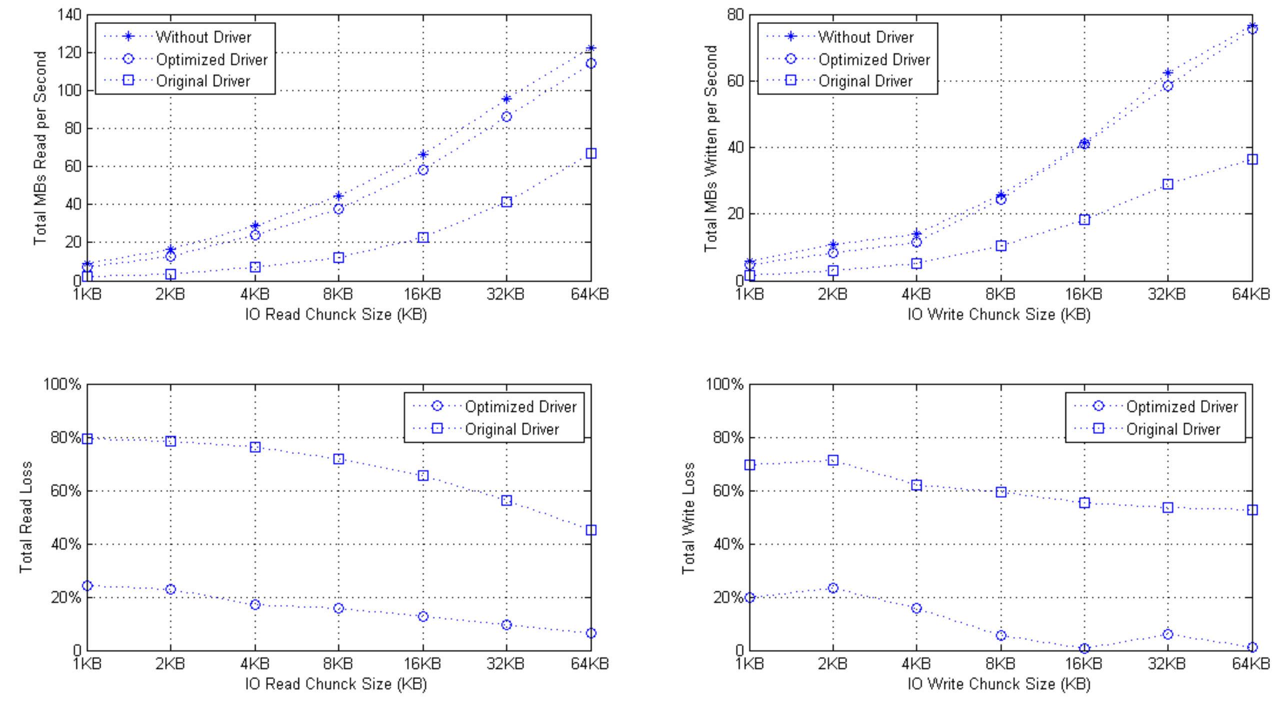

Figure 4. IO throughput and IO throughput loss.

From **Figure 4** we can find that the IO throughput improved a lot after Subnet Driver optimization; and with the IO chunk size increase the IO throughput loss decrease. When the IO size is increased to 64KB, the optimized system IO read loss is improved to 6.6%, the optimized system IO write loss is improved to 1.2%; this is because with the increase of IO chuck size, time that data transferring used is growing, so the ratio of time that Subnet Driver used to total time get less and less; so the system IO throughput loss get down.

6. Conclusion

Secure Subnet System can provide mandatory action control which is a more strong control method. In this system even for users already read sensitive data, their operations still under control and system will dynamically change security policies to prevent the user leak these data or spread the data outside to another subnet. We use state machine model to describe our model, and use secure transfer equations to dynamically calculate the system policies for each new state. We proved the security of our model by formal methods.

We implemented a demon of Secure Subnet System, and evaluate the demon both from security and performance. The evaluation results show that the system is secure; and the performance test case show that the IO throughput improved a lot after Subnet Driver optimization; and with the IO chunk size increase the IO throughput loss decrease; for the 64KB IO chunk size, IO read can be improved to 6.6%, IO write loss can be improved to 1.2% after optimization.

7. Acknowledgment

We are grateful to Prof. Ren and Prof. Wang for their valuable comments and advice, and Xiaoping Feng, Sheng Chen who have given us a lot of help in our implementation.

REFERENCES

[1] Jack Brassil, "Physical Layer Network Isolation in Multi-tenant Clouds", IEEE 30th International Conference on Distributed Computing Systems Workshops 2010.

[2] Charles J. Trantanella, "A novel power divider with enhanced physical and electrical port isolation", Microwave Symposium Digest (MTT), 2010 IEEE MTT-S International;

[3] Wan Guoping, "Network isolation and NetGap", China machine press.

[4] RS Sandhu, EJ Coyne, HL Feinstein, CE Youman; Role-based access control models; IEEE Comput., Volume (29), Page(s): 38 - 47, Feb 1996.

[5] Haiwei Xue, Yiqi Dai; A privacy protection model for transparent computing system; International Journal of Cloud Computing, Volume 1, Pages 367-384.

[6] Haiwei Xue, Xiong Liu, Yiqi Dai, "A privacy protection model on internal networks", 13th IEEE Joint International Computer Science and Information Technology Conference 2011.

[7] Intel Corporation, "Preboot Execution Environment (PXE) Specification", technical document of Intel Corporation 1999.

[8] Bell D E, LaPadula L J., "Secure Computer System: Unified Exposition and MULTICS Interpretation[R]", Bedford, MA: The MITRE Corporation, 1976.

[9] Sitian Ge, Raoxue Zhang, YiqiDai, "L-BLP Security Model in Local Area Network", Chinese of Journal Electrics, vol. 35. Pp. 1005-1008.

[10] http://en.wikipedia.org/wiki/Software-defined_networking

[11] http://en.wikipedia.org/wiki/Command-line_interface