SET Specific Assumptions

(A8) Client and merchant believe that they have order description and price, and all participants believe that they are internal parties.

Q believes Q has (OD, Price) Where *Q* stands for *C* and *M*.

R believes R'-is-internal-party Where *R* and *R'* stand for *A*, *C*, and *M*.

(A9) Every participant believes that client has sent *PReq* and received *PRes*, merchant has sent *AuthReq* and received *AuthRes*, and acquirer has received *AuthReq* and sent *AuthRes*.

P believes C says PReq *P believes M sees PReq*

P believes M says AuthReq *P believes A sees AuthReq*

P believes A says AuthRes *P believes M sees AuthRes*

P believes M says PRes *P believes C sees PRes*

Client Privacy

(A10) Client does not believe that merchant has payment information, and client and merchant do not believe that acquirer has order description.

$\neg$*C believes M has PI*

$\neg$*Q believes A has OD* Where *Q* stands for client and merchant.

General Assumptions for Proving Money Authorizations

(A11) Every participant believes that he can prove to verifier that if client has sent the message containing Merchant's ID, price, and the date of execution, then client has authorization on payment ordering.

P believes P CanProve [C says (M, Price, Date) $\rightarrow$ *C authorized payment(C, M, Price, Date)] to V*

General Assumptions for Proving Goods Authorizations

(A12) Every participant believes that he can prove to verifier that if client has sent the message containing Merchant's ID, order description, and the date of execution, then client has authorization on goods ordering.

P believes P CanProve [C says (M, OD, Date) $\rightarrow$ *C authorized goods-order(C, M, OD, Date)] to V*

Proving Money Accountability in SET

Proving accountability in SET, we focus on analyzing one of primitive transactions, which is *C authorized payment (C, M, Price, PReqDate)*. This means that client has authorization on making payment on the goods amount *Price* on the date of making the request *Date*.

The goal of proof:

M believes M CanProve (C authorized payment(C, M, Price, PReqDate)) to V

Where *V* stands for any external verifier.

In order to show that *M believes M CanProve (C authorized payment(C, M, Price, PReqDate)) to V*, it suffices to show that

M believes M CanProve (C says (M, Price, PReqDate)) to V (x)

It is easy to see that *M believes M CanProve (C says h(PI)) to V* follows mainly by axiom P3. Since *(M, Price, PReqDate)* is in *PI*, (x) may thus be shown by using axiom P4.

However, the proof for *M believes M CanProve (h(PI)-is-fingerprint-of-PI) to V* in axiom P4 would fail since it is not the case that *M believes M has* K_A^{-1} which is required to show that *M believes M has (h(PI), PI)*. Note that *PI* is encrypted with K_A. Note also that if it were the case that *M believes M has* K_A^{-1}, the proof would still fail since it would require *M believes (V has PI)*, due to A4, which contradicts to our assumption A7.

It is also easy to see that *M believes M CanProve (C says OI) to V* follows mainly by axioms P4 and P3. Since *h(OD, Price)* is in *OI*, the proof for *M believes M CanProve (C says Price) to V* may be shown by using axiom P4. However, such proof would require *M believes V has OD* due to A4, which contradicts to our assumption A7.

Proving Goods Accountability in SET and iKP

The goal of proof:

M believes M CanProve (C authorized goods-order(C, M, OD, PReqDate)) to V

We shall discuss the goods accountability in SET first. Similarly to the proof for *M believes M CanProve (C says Price) to V* discussed in section 4.2, the proof for *M believes M CanProve (C authorized goods-order(C, M, OD, PReqDate)) to V* would fail since it would require *M believes V has Price* which contradicts to our assumption A7.

IKP [BP et al 00] lacks of goods accountability due to similar reason to that in SET. The following shows a payment request from client *C* to merchant *M*.

Payment C → M: $PI, \{PI, h(OI)\}_{K_C}$

Where *PI* stands for payment information. *PI* contains *(Price, h(OI), Client's Credit-card information*$\}_{K_C}$. *OI* stands for order information. *OI* contains *(TID, Price, ClientID, MerchantID, Date, InvExpDate, h(OD)). InvExpDate* stands for invoice (offer) expiration date specified by merchant. *Date* stands for the date of invoice (offer) issued by merchant.

It is easy to see that *M believes M CanProve (C says h(OI)) to V* follows mainly by axiom P3. Since *(Price, h(OD)* is in *OI*, the proof for *M believes M CanProve (C says OI) to V* can be shown by using axiom P4. However, such proof would require *M believes V has Price* due to A4, which contradicts to our assumption A7.

Goals of SET and iKP as Accountability

In this section, we discuss the use of the accountability for specifying and analyzing the goals of SET and iKP protocols. For such kind of accountability, a verifier is an internal party, which involves in e-commerce protocols. Recall that the goal of e-commerce protocols is to ensure that all parties are convinced that they have authorized messages concerning primitive transactions relevant to them after the completion of the protocols.

After sending some messages to intended recipients, the originator must be able to prove the association of the originator with an intended action (or an intended message) to intended recipient (s). Such intended actions are just about primitive transactions. Such proof would ensure the originator that the intended recipients would recognize the originator's intention regarding to primitive transactions.

This intuition is formalized by axiom *P2'*. The axiom states explicitly the preconditions for the sending and receiving of messages. The rule also caters for the case where the intended recipient receives messages from an intermediate party.

Thus, the goals of e-commerce protocols can be expressed by the following:

a) *C believes C CanProve (C authorized payment(C, M, Price, Date)) to M*

b) *M believes M CanProve (M authorized payment(C, M, Price, Date)) to C*

c) *C believes C CanProve (C authorized value subtraction(A, C, Price, Date)) to A*

d) *A believes A CanProve (A authorized value subtraction(A, C, Price, Date)) to C*

e) *M believes M CanProve (M authorized value claim(A, M, Price, Date)) to A*

f) *A believes A CanProve (A authorized value claim(A, M, Price, Date)) to M*

g) *C believes C CanProve (C authorized goods-order(C, M, OD, Date)) to M*

h) *M believes M CanProve (M authorized goods-receipt(C, M, OD, Date)) to C*

It is not hard to see that in SET these goals cannot be shown due to similar reason to those for the accountability for dispute resolving discussed in sections 4.2 and 4.3. However, these goals can be shown for iKP.

With provable authorization in the present of private information, our logic can be used to specify and analyze goals of e-commerce protocols efficiently in that the originator can ensure that the recipient recognizes the intention about primitive transaction without revealing private information. This will be much benefit to protocol designers in that he can design a protocol with intended purposes.

What we demonstrate below is the analysis of client privacy using our logic.

Proving Client Privacy of SET

Client privacy can be understood as the accountability where a client is a prover, both a bank and a merchant are verifiers, and the proving statement is about the payment authorization. SET achieves client privacy if merchant cannot infer client's payment information *(PI)*, and acquirer cannot infer goods description *(OD)* from the protocol. We thus start proving client privacy by stating the goals of the proofs from *a)* and *c)*. In order to prove *a)*, it suffices to show that

C believes C CanProve (C says (M, Price, PReqDate)) to M

It is easy to see that *C believes C CanProve (C says (Price, PReqDate)) to M* follows mainly by axioms P4 and P3. Although the proof is not successful because of the lacking of merchant's ID, merchant cannot infer *PI* from the receiving message. We prove *c)* in the same way as proving *a)*. It is not hard to see that acquirer cannot infer *OD*, which is client's private information, from the receiving message.

As a result, our logic can analyze client privacy, which is an essential property of SET protocol, in that verifier can get only necessary information to prove without getting any private information.

Conclusion

In this research, we show that the existing logics for reasoning about accountability are inadequate to deal with real world e-commerce protocols. We then propose an extension of the existing logics to deal with such real-world protocols. Furthermore, we demonstrate the practicality of our logic by showing that the result obtained from Herreweghen's informal analysis [H99a] can be also obtained formally from our logic.

Our logic can be used not only for reasoning about dispute resolution amongst parties, but also for reasoning about the goals of e-commerce protocols, in particular, client privacy property. Indeed, both kinds of reasoning can be captured *uniformly* in our logic.

For reasoning about dispute resolution, we show *formally* that SET lacks of the money accountability whereas iKP does not. Moreover, we show that both SET and iKP lack of the goods accountability. The lack of the goods accountability in our sense means that the two protocols can still provide enough evidence tokens to resolve disputes on goods description, but in order to resolve such disputes, provers must reveal their private information to verifiers. In some situations, this is undesirable.

For reasoning about the goals of e-commerce protocols, we show that such kind of reasoning can be considered as a special case of the accountability. Thus, the accountability can be seen as a fundamental property for analyzing e-commerce protocols. Indeed, other kind of properties such real-world e-commerce protocols, for example each party's requirement [MS98], has also been studied by using our logic presented here.

บทที่ 7 ข้อวิจารณ์

In this chapter, we discuss experimental results obtained by applying our software prototypes to some case studies. We focus on the experimental results obtained from specification and verification for access control in Windows NT and network firewalls.

7.1) ผลการทดลองข้อกำหนดสำหรับระบบปฏิบัติการ [ก43]

We have developed a software prototype and applied it to an internet service providing system at computer center in King Mongkut's University of Technology Thonburi (KMUTT). The system is about configuring access control system for the following services.

1) Home Directory Services
   This service is to provide a place to keep files for users. Users include students, lecturers, etc. In this service, users can use telnet and ftp service to get to their home directories.
2) Electronic Mail Services
3) Web Services
4) Dial-up Connection Service
   This service is to allow users to connect into computer systems at the university from their homes.

According to the study, the existing system has the following problems.

1) The difficulty in checking the correctness of configurations. This is because there are a large quantity of users and objects. Moreover, the test is manual and thus it takes a lot of times.
2) Attacks. There are attacks coming from both outside and inside that cannot be detected.
3) The difficulty in extending the existing systems for new hardware or new groups of users. This is because of the lack of systematic policy. When there are new computer hardware obtained or new groups of users, the configurations have to be reconstructed. The extension of the system is unsystematic

Our method can solve the first problem effectively. This is because the checking for the correctness of configurations can be done by using software. This facilitates the correctness analysis.

The following shows some experimental result. We have developed a software prototype which can be used to test whether or not an access control configuration in Windows NT satisfies policy. In the following, the policy that we are interested in is that "an authorized user for highly sensitive information is allowed to access such information from the machine that hosted the information only". Such policy is selected by the following window.

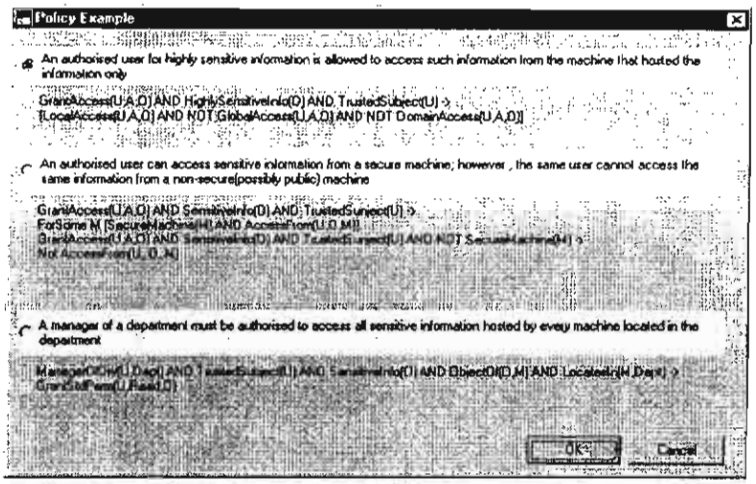

Figure 6

Highly sensitive information can be defined by the following.

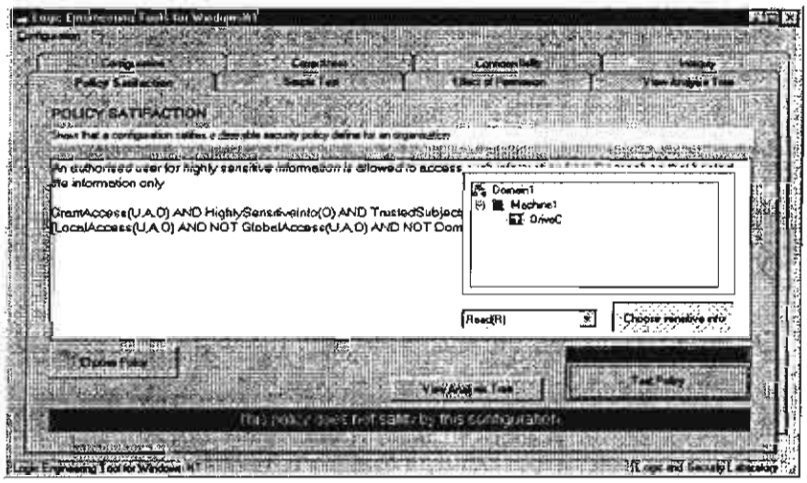

Figure 7

After the program has analyzed the configuration, it shows the following output.

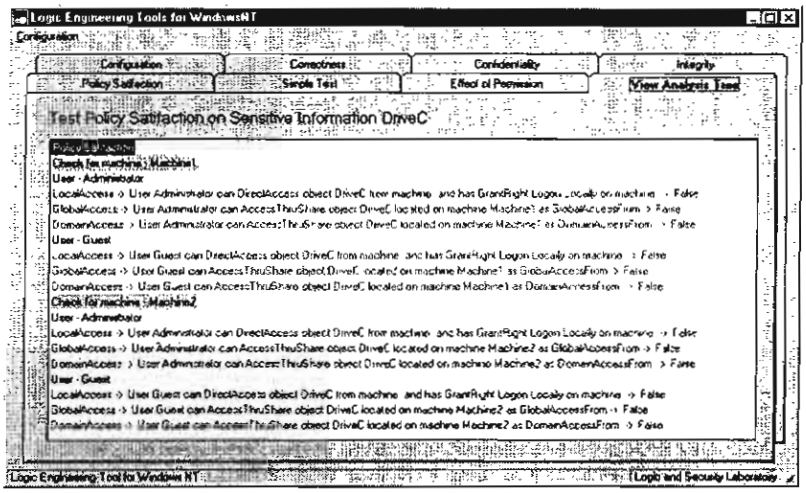

Figure 8

7.2) ผลการทดลองวิธีการแบบตรรกะศาสตร์สำหรับไฟร์วอลล์ [n45]

We have developed a software prototype for analyzing the effect of firewall configurations. The following shows the firewall topology.

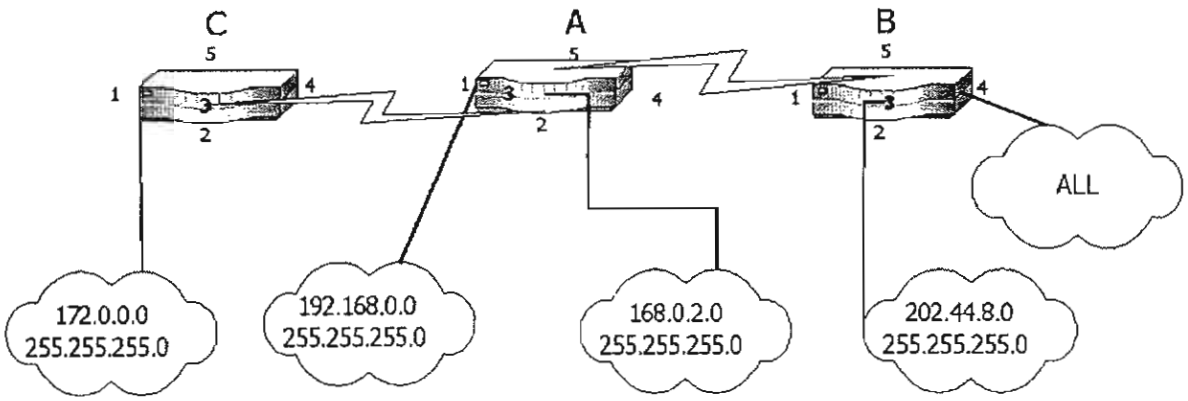

Figure 9

There are three firewall routers, namely A, B and C. The following shows configuration rules for firewall A.

Order	Source	Dest	Service	Direction	Action	Interface
1	168.0.2.7	192.168.0.7	TCP	IN	PERMIT	10.0.3.3
2	168.0.2.7	192.168.0.7	TCP	OUT	PERMIT	10.0.3.1
3	202.44.8.8	192.168.0.8	TCP	IN	PERMIT	10.0.3.5
4	202.44.8.8	192.168.0.8	TCP	OUT	PERMIT	10.0.3.1
5	172.0.0.1	202.44.8.0# 255.255.25 5.0	TCP	IN	PERMIT	10.0.3.2
6	172.0.0.1	202.44.8.0# 255.255.25 5.0	TCP	OUT	PERMIT	10.0.3.5

Table 1 : Rule for router A

The following shows configuration rules for firewall B.

Order	Source	Dest	Service	Direction	Action	Interface
1	202.44.8.8	192.168.0.8	TCP	IN	PERMIT	202.44.12.3
2	202.44.8.8	192.168.0.8	TCP	OUT	PERMIT	202.44.12.5
3	172.0.0.1	202.44.8.0# 255.255.25 5.0	TCP	IN	PERMIT	202.44.12.5
4	172.0.0.1	202.44.8.0# 255.255.25 5.0	TCP	OUT	PERMIT	202.44.12.3
5	Any	Any	TCP	IN	DENY	202.44.12.4

Table 2 : Rule for Router B

The following shows configuration rules for firewall C.

Order	Source	Dest	Service	Direction	Action	Interface
1	172.0.0.1	202.44.8.0# 255.255.25 5.0	TCP	IN	PERMIT	10.1.1.1
2	172.0.0.1	202.44.8.0# 255.255.25 5.0	TCP	OUT	PERMIT	10.1.1.3

Table 3 : Rule for router C

The following shows the effects of all firewall configuration rules.

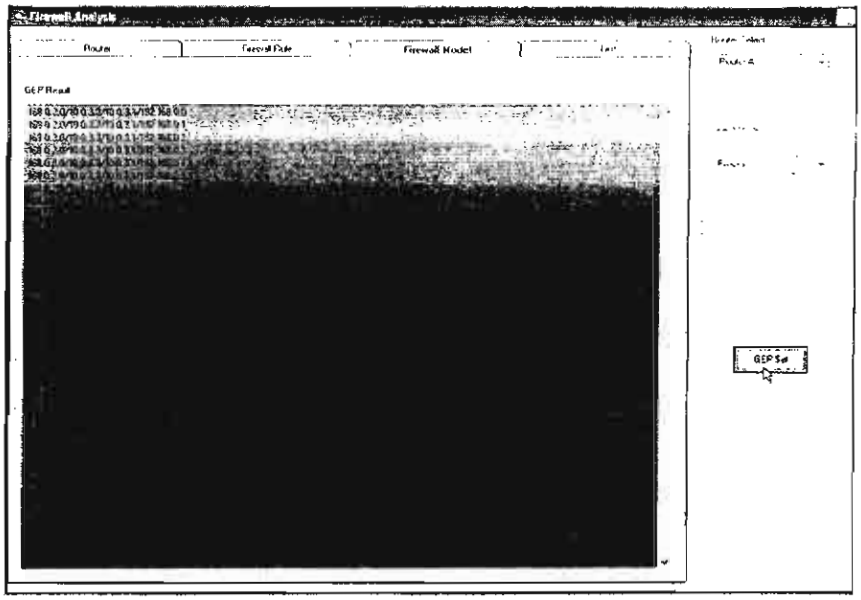

Figure 15

บทที่ 8 สรุป

8.1) Conclusion

We have developed formal specification and verification for information security systems which are in use in real world. In particular, we have developed formal specification for access control in Windows NT, network firewalls and electronic payment protocols.

The following concludes the novel contribution of our research.

1) Our formal specification for Windows NT can deal with distributed access control whereas existing specification can deal with centralized access control.

2) Our formal specification for network firewalls is the first model that predicts the effects of firewall configurations thoroughly and rigorously. Furthermore, our model can analyze the redundancy and inconsistency of firewall rules and can analyze the vulnerability of configurations for IP spoofing attacks.

3) Our formal specification for credit card-based electronic payment protocol is capable of analyzing SET whereas existing specification cannot. Many kinds of properties including client privacy can be captured.

8.2) Future work

1) To study formal methodology for internet security which focuses on the analysis of attacks occurring in the internet.
2) To study formal methodology for secure electronic government system.

บรรณานุกรม

[ก43] นายกรกนก ชุติพันธ์, นายกฤษดา บรรเจิดพิศาลกุล และนายเดชษกร ศรีณรงค์, การตรวจสอบการเข้าถึงข้อมูลบนระบบปฏิบัติการวินโดว์ เอนที โดยแบบจำลองทางตรรกะศาสตร์, วิทยานิพนธ์ระดับปริญญาตรี, ปีที่จบการศึกษา 2543

[ก45] นางสาวกุสุมา ชาครวิโรจน์ และนายเดกสิทธิ์ ทองหา, เครื่องมือสำหรับการตรวจสอบและการสร้างกฎของไฟร์วอล, วิทยานิพนธ์ระดับปริญญาตรี, ปีที่จบการศึกษา 2545

[ABLP93] M. Abadi, M. Burrows, B. Lampson and G. Plotkin, A Calculus for Access Control in Distributed Systems, *ACM Transactions on Programming Language and Systems*, Vol. 15 No. 3, 1993.

[AES] Advanced Encryption Standard, http://www.nist.gov/aes

[AHS98] N. Asokan, E. V. Herreweghen, and M. Steiner. Towards A Framework for Handling Disputes in Payment Systems. In the *Proceedings of the 3rd USENIX workshop on Electronic Commerce*, Boston, Massachusette, August31-September3,1998.

[B00] E.Bertino, S.Castano, E.Ferrari, M.Mesiti, "Specifying and Enforcing Access Control Policies for XML Document Sources", World Wide Web Journal (Baltzer Publ.),Vol.3(3), 2000.

[BAN90] M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. *ACM Transactions in Computer Systems*, February 1990.

[BBFR99] E. Bertino, F. Buccafurri, E. Ferrari and P. Rullo, A Logical Framework for Reasoning on Data Access Control Policies, *in Proceedings of IEEE Computer Security Foundation Workshop*, 1999.

[BBFR00] E. Bertino, F. Buccafurri, E. Ferrari, and P. Rullo, A Logic-Based Approach for Enforcing Access Control, *Journal of Computer Security*, 8(2&3), 2000.

[BG et al. 00] M. Bellare, J. A. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, E. V. Herreweghen, and M. Waidner. Design, Implementation, and Deployment of the iKP Secure Electronic Payment System. *IEEE Journal of Selected Areas in Communications* 2000.

[Bi96] P. Bieber, Formal Techniques for an ITSEC-E4 Secure Gateway, *In Proceedings of National Computer Security Conference*, USA, 1996.

[BJS96] E. Bertino, S. Jajodia and P. Samarati, "Supporting Multiple Access Control Policies in Database System" *Proceedings of IEEE Symposium on Research in Security and Privacy*, Oakland, California, May 1996. pp. 94-107.

[BMNW99] Y. Bartal, A. Mayer, K. Nissim and A. Wool, Firmato : A Novel Firewall Management Toolkit, *In proceedings of 20th IEEE Symposium on Security & Privacy*, Oakland, CA, 1999.

[BV97] Y. Bai and V. Varadharajan, A Logic for State Transformation in Authorization Policies, *in Proceedings of IEEE Computer Security Foundation Workshop*, 1997.

[C94] J. Cooper, 1984, "Computer-Security Technology", 2nd edition, 166 p.

[CB94] W.R. Cheswick and S.M. Bellovin, *Firewalls and Internet Security : Repelling the Wily Hacker*, Addison-Wesley, 1994.

[CFMS95] S. Castano, M.G. Fugini, G. Martella and P. Samarati, *Database Security*, Addison Wesley- ACM press, 1995.

[CZ95] D.B. Chapman and E.D. Zwicky, *Building Internet Firewall*, O'Reilly & Associates, 1995.

[DD98] D.E. Denning and P.J. Denning (editors), *Internet Besieged: Countering Cyberspace Scofflaws*, Addison-Wesley, ACM Press, 1998.

[FB97] D.F. Ferraiolo and J.F. Barkley, Specifying and Managing Role-Based Access Control within a Corporate Intranet, *In Proceedings of ACM Workshop on Role-Based Access Control*, ACM press, 1997.

[FCK95] D.F. Ferraiolo, J.A. Cugini, D.R. Kuhn, Role-Based Access Control (RBAC): Features and Motivations, *In Proceedings of IEEE Computer Security Applications*, IEEE Computer Society press, 1995.

[GB98] S.I. Gavrila and J.F. Barkley, Formal Specification for Role Based Access Control User/Role and Role/Role Relationship Management, *In Proceedings of ACM Workshop on Role-Based Access Control*, ACM press, 1998.

[GMP92] J. Glasgow, G. MacEwen and P. Panangaden, A Logic for Reasoning about Security, *ACM Transactions on Computer Systems*, Vol. 13, No. 3, 1992.

[GY98] J. Gross and J. Yellen, *Graph Theory and its Applications*, CRC Press LLC, 1998

[Gu97] J.D. Guttman, Filtering Postures : Local Enforcement for Global Policies, *In proceedings of 17th IEEE Symposium on Security & Privacy*, Oakland, CA. 1997.

[H91] G.J. Holzmann, *Design and Validation of Computer Protocols*, Prentice Hall Software Series, 1991.

[H99a] E. V. Herreweghen. Non-Repudiation in SET: Open Issues. In the *Proceedings of the Financial Cryptography 1999.*

[H99b] E. V. Herreweghen. Using Digital signatures as Evidence of Authorizations in Electronic Credit-Card Payments. *Research report 3156, IBM Research,* June 1999.

[HRU76] M.A. Harrison, W.L. Ruzzo and J.D. Ullman, Protection in Operating Systems, *Communication of ACM*, 19(8), 1976.

[JSSS01] S. Jajodia, P. Samarati, M.L. Sapino, and V.S. Subrahmanian, Flexible Support for Multiple Access Control Policies, in *ACM Transactions on Database Systems*, vol. 26, n. 2, June 2001, pp. 214-260.

[ITS91] ITSEC Working Group. *ITSEC: Information Technology Security Evaluation Criteria*, version 1.2, Sep 1991.

[JSS97] S. Jajodia, P. Samarati and V.S. Subrahmanian, A Logical Language for Expressing Authorizations, *In Proceedings of IEEE Symposium on Research in Security and Privacy*, 1997.

[K96] R. Kailar. Accountability in Electronic Commerce Protocols. *IEEE Transaction on Software Engineering 1996.*

[KG98] L.L Kassab and S. J. Greenwald, Towards formalizing the Java Security Architecture of JDK 1.2, *In proceedings of European Symposium on Research in Computer Security (ESORICS)*, Springer-Verlag, 1998.

[KN98] V. Kessler and H. Neumann. A Sound Logic for Analyzing Electronic Commerce Protocols. In the *Proceedings of ESORICS'98.*

[M97] F. Massacci, Reasoning about Security : a Logic and a decision method for Role-based Access Control, *in Proceedings of the International Joint Conference on Qualitative and Quantitative Practical Reasoning (ECSQARU/FAPR)*, Spinger-Verlag, 1997.

[MS98] C. Meadows and P. Syverson. A Formal Specification of Requirements for Payment Transactions in the SET Protocol. In the *Proceedings of Financial Cryptography*, February 1998.

[MWZ00] A. Mayer, A. Wool and E. Ziskind, Fang : A Firewall Analysis Engine, *In proceedings of 21st IEEE Symposium on Security & Privacy*, Oakland, CA, 2000.

[NCS85] National Computer Security Center, *Department of Defense Trusted Computer Security Evaluation Criteria*, DoD 5200.28-STD, 1985.

[R97] C. Rutstein, *Windows NT Security : A practical guide to securing Windows NT servers and workstations*, McGraw-Hill, 1997.

[R01] C. Rujimethabhas, *A graph-based methodology for Hardware-based Firewalls*, Master thesis, Department of Computer Engineering, King Mongkut's University of Technology Thonburi, Bangkok, Thailand, 2001.

[RS98] C. Ramaswamy and R. Sandhu, Role-Based Access Control Features in Commercial Database Management Systems, *In Proceedings of National Computer Security Conference*, 1998.

[S94] Bruce Schneier.*Applied Cryptography:* Protocols, Algorithms and Source Code *in C*. John Wiley & Sons, 1994

[S97] T. Sheldon, *Windows NT Security Handbook*, McGraw-Hill, 1997.

[S97] S. Sutton, *Windows NT Security Guide*, Addison-Wesley Press, 1997.

[S98] R.S. Sandhu, Role-Based Access Control, *Advances in Computers*, M. Zerkowitz (ed.), Vol. 48, Academic Press, 1998.

[SCFY96] R. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman, Role-Based Access Control Models, *IEEE Computer*, 29(2), Febuary 1996.

[SET97] Secure Electronic Transaction Specification, Version 1.0, May 1997. (available at http://www.visa.com/set).

[SS94] R.S. Sandhu and P. Samarati, Access Control: Principles and Practice, *IEEE Communications* 32(9), pp. 40-48, 1994.

[SSL] The SSL Protocol Version 3.0 < http://home.netscape.com/eng/ssl3/ssl-toc.html >

[St03] W. Stallings, *CRYPTOGRAPHY AND NETWORK SECURITY: PRINCIPLES AND PRACTICE*, Prentice Hall, 2003.

[T00] O. Tunsungwon, *A Formal Method for Dynamic Access Control in Windows NT*, Master Thesis, Department of Computer Engineering, King Mongkut's University of Technology Thonburi, Bangkok Thailand, 2000.

[WL93] T.Y.C. Woo and S.S. Lam, Authorization in Distributed Systems: A new approach, *Journal of Computer Security*, 2(2,3), 1993.

Output ที่ได้

1) ผลงานตีพิมพ์ในวารสารวิชาการในประเทศ

1) Yongyuth Permpoontanalarp, Reasoning about Access Control in Windows NT, *Engineering Transactions*, Mahanakorn University of Technology, Vol. 4, No.1(10), 2001.

2) ผลงานตีพิมพ์ในการประชุมวิชาการต่างประเทศ

2) Yongyuth Permpoontanalarp and Chaiwat Rujimethabhas, A Unified Methodology for Verification and Synthesis of Firewall Configurations, *In Proceeding of The Third International Conference on Information and Communications Security (ICICS),* China, Lecture Notes in Computer Science, Springer Verlag, 2001.
3) Supakorn Kungpisdan and Yongyuth Permpoontanalarp, Practical Reasoning about Accountability in Electronic Commerce Protocols, *In Proceedings of the 4th International Conference on Information Security and Cryptology (ICISC)* , Seoul, South Korea, Lecture Notes in Computer Science, Springer Verlag, 2001.
4) Yongyuth Permpoontanalarp and Chaiwat Rujimethabhas, A Graph Theoretic Model for Hardware-based Firewalls, *In Proceedings of the 9th IEEE International Conference on Networks (ICON),* Thailand, IEEE Computer Press, 2001.

3) ผลงานตีพิมพ์ในการประชุมวิชาการในประเทศ

5) Voravud Santiraveewan and Yongyuth Permpoontanalarp, A Verification Methodology for Analyzing IP Spoofing Attack, In Proceedings of the 7th *National Computer Science and Engineering Conference*, Chonburi, Thailand, 2003.

4) ผลงานที่กำลังเขียนเพื่อที่จะส่งไปวารสารเชิงวิชาการนานาชาติ

Yongyuth Permpoontanalarp, A Graph-based Methodology for Firewalls (In Preparation).

5) รายชื่อนักศึกษาที่ได้สำเร็จการศึกษาโดยการทำวิทยานิพนธ์ซึ่งเป็นส่วนหนึ่งของงานวิจัยในโครงการนี้

<u>ระดับปริญญาโท</u>

1. นางสาวอรวรรณ ตันสังวรณ์
   ชื่อหัวข้อวิทยานิพนธ์ A Formal Method for Dynamic Access Control in Windows NT
   ปีที่จบการศึกษา 2543

2. นายชัยวัฒน์ รุจิเมธาภาส
   ชื่อหัวข้อวิทยานิพนธ์ A Formal Methodology for Hardware-based Firewalls
   ปีที่จบการศึกษา 2543
3. นายคุภกร กังพิศดาร
   ชื่อหัวข้อวิทยานิพนธ์ Accountability as Fundamental Property for Electronic Commerce Protocols
   ปีที่จบการศึกษา 2544
4. นาย วรวุฒิ สันติลวีวรรณ
   ชื่อหัวข้อปริญญานิพนธ์ A Graph-based Methodology for Analyzing IP Spoofing Attacks
   ปีที่จบการศึกษา 2546

<u>ระดับปริญญาตรี</u>

1. นายกรกนก ชุติพันธ์, นายกฤษดา บรรเจิดพิศาลกุล และนายเดชษกร ศรีณรงค์
   ชื่อหัวข้อวิทยานิพนธ์ การตรวจสอบการเข้าถึงข้อมูลบนระบบปฏิบัติการวินโดว์ เอนที โดยแบบจำลองทางตรรกะศาสตร์
   ปีที่จบการศึกษา 2543
2. นางสาวกุสุมา ชาครวิโรจน์ และนายเศกสิทธิ์ ทองทา
   หัวข้อปริญญานิพนธ์ เครื่องมือสำหรับการตรวจสอบและการสร้างกฎของไฟร์วอล
   ปีที่จบการศึกษา 2545

ภาคผนวก

Reasoning about Access Control in Windows NT

Yongyuth Permpoontanalarp
Logic and Security Laboratory
Department of Computer Engineering
King Mongkut's University of Technology Thonburi
91 Suksawasd 48, Ratburana, Bangkok 10140 Thailand
yongyuth@cpe.eng.kmutt.ac.th , ypber@yahoo.com

Abstract

In this paper, we study a logical methodology for access control in a real-world application, namely Windows NT (NT). In particular, we extend existing logical specifications for access control in order to deal with *distributed* access control in NT. Then, we propose *practical* verification properties for analyzing access control configurations in NT. The results obtained show that our logical methodology has several benefits. In particular, our logical specification for NT helps clarifying the access control mechanism in NT. Moreover, our reasoning methodology helps system administrators not only to analyze access control configurations, but also to set up an access control configuration which has desirable properties.

Keywords: Access Control, Tools for analyzing the security of access control in Operating systems and Formal Methods for Computer Security

1) INTRODUCTION

During the past decades, several researchers have studied the development and the applying of formal methods to computer security. Those studies have focused on two major areas of computer security : access control and cryptographic protocols. In this paper, we study the use of formal methods for access control.

Most of the existing works (eg. [1 - 7]) on formal methods for access control do not study the use of formal methods in the context of real-world applications. Very few of them do study formal methods for real-world applications however, for example [8].

In addition, most of the existing works emphasizes either on the development of specifications for access control or on the reasoning about access control policies for their properties which are not concerned with the security aspect. In particular, [1, 3, 5] studied the use of formal methods mainly for specifications. Moreover, [2, 6, 7] studied the reasoning about conflicting access control policies and a method to resolve such conflicts. A few of them however studied the verification of access control policies concerning their security aspects, for example [9]. However, those works on the verification are of theoretical interest.

In this paper, we aim to study the use of formal methods for computer security in the context of a *real-world* application. In particular, we study a logical specification for access control in Windows NT [10, 11, 12] (NT). Moreover, we study *practical* verification properties in order to analyze access control configurations in NT.

Our specification is based on JSS's Authorization Specification Language (ASL) [4] which deals with access control in general setting, and logic-based Role-based Access Control (RBAC) specifications [13, 14]. In particular, we extend JSS's ASL and the RBAC specifications to deal with distributed access control in NT. In particular, our specification captures the two concepts: namely *access transparency* and *authority* to deal with the distributed access control in NT. *Access transparency* is concerned with the ability of a user to access an object stored at a machine from any machine in a distributed system. Moreover, *authority* is concerned with the scope of the power that each administrator in NT has in order to manage the system. In other words, authority is about distributed administration of a system.

Our specification offers a benefit in that it clarifies the access control mechanism in NT precisely. As a result, our specification provides a better understanding on the NT access control mechanism, than the non-formal approach. The non-formal approach would be to read many books on NT and to carry out some test data in order to understand the access control mechanism. Moreover, the natural language (English) that is used to describe the NT mechanism in those books is ambiguous and unclear.

Furthermore, we propose *practical* reasoning methodology to help system administrators to analyze access control configurations in NT. In particular, we propose *verification properties* of access control configurations and we propose a *synthesis methodology*

which can be used to generate new access control configurations. The verification properties proposed can be used to analyze the *correctness* and the *security* of access control configurations. Moreover, the verification properties include *policy satisfaction* which aims to verify that a given access control configuration satisfies desirable *security policies* designed for an organization.

The remainder of this paper is organized as follows. Section 2 provides some background for the rest of the paper. In section 3, we discuss our logical specification and we illustrate our reasoning methodology in section 4. We discuss a method to test that our specification corresponds to the NT system in section 5 and make a conclusion in section 6.

2) BACKGROUND

2.1) Windows NT Access Control

Windows NT (NT) is an operating system designed for distributed systems. In particular, NT provides services for users and machines both of which disperse geographically but coordinate to share resources and information. It provides security services, eg. access control and authentication. In this paper, we concern with access control only.

NT offers a systematic approach to manage a distributed system by grouping not only machines but also users. In particular, a domain is a collection of machines which are managed by an administrator. For example, machines in a department may be grouped into one domain. In a domain, there is a machine acting as a domain controller.

Similarly, a user group (or just group) is a collection of users who share similar privileges. There are two main kinds of groups : local and global groups. A local group is a group that is recognized by a particular machine only whereas a global group is a group which is recognized by all machines in a particular domain. The same concept is applied to user accounts also, ie. local users and global users.

It is useful to classify privileges into three kinds : *ACL permissions*, *rights* and *abilities*. ACL permissions are concerned with permitted actions that a user can perform on an object, eg. files and directories, in a system. Rights and abilities are concerned with permissions required for managing and maintaining a system. We shall discuss these three kinds of privileges in section 3.

Access control mechanism for resources in NT is based on Discretionary Access Control. In particular, an owner of an object decides who may have certain ACL permissions on the object.

NT provides a distributed approach for administering the system. For each domain, there are many groups of administrators which hold different responsibilities for administering the system. For example, tape backup operators are responsible for backing up files and directories into a tape, and restoring them later. Each group of administrators is granted certain permissions in order to perform their tasks. Groups of administrators who are in charge of domain controllers are different from those who are in charge of non-domain controllers.

As a matter of terminology, we use the term permissions to refer to ACL permissions, rights and abilities in general.

In this paper, we shall discuss details of NT which are sufficient for our purpose only. Further details on NT can be found in [10 - 12].

2.2) Authorization Specification Language (ASL)

JSS's ASL [4] is a general logical specification for modeling a discretionary access control system. Several kinds of access control policies can be captured by the specification. ASL consists of several kinds of rules. Conceptually, ASL is divided into layers and each layer is formulated by one kind of rules. There are five main kinds of rules, namely authorization rules, derivation rules, resolution rules, access control rules and integrity rules.

Authorization rules are at the bottom layer and they provide information on an access control configuration which is set up by a user. Such a configuration defines access permission that a subject has on an object. On top of the authorization rules, there are derivation rules which take permissions obtained from the authorization rules as an input and then propagate the permissions assigned for a group into the group members.

Resolution rules on the other hand are concerned with resolving conflicting permissions obtained from the derivation rules. Access control rules provide final decisions for an access. Integrity rules state conditions required in order to maintain the integrity of an access control system.

Even though ASL offers a general specification for dealing with access control, it deals with access control in a centralized stand-alone system. Thus, in order to apply ASL to model access control in NT, it requires an extension of ASL to deal with access control in distributed systems.

2.3) Role-based Access Control (RBAC)

RBAC (eg. [13, 15]) is a new kind of access control model which offers the controlling of accesses based on

roles that individual users take on in an organization. Roles stand for job titles in an organization. RBAC uses the concept of roles as a link between users and permissions on an object. A user is granted an access permission for an object if such permission is required in order for the user to perform his or her job functions. It has been employed in several applications, for example access control in Databases [16] and Intranet [17]. A logical formulation of RBAC was studies in [14].

We observe that NT approach for the administration of a system is closely related to RBAC in that each administrative group can be considered as a role, ie. administrative role.

3) THE LOGICAL SPECIFICATION FOR NT ACCESS CONTROL

Our logical specification represents two kinds of information : *an access control configuration* and *the access control mechanism*. An access control configuration is *mostly* represented by facts that are specified by atomic predicates whereas the access control mechanism is represented by rules that are specified by first-order formulas. Access control configurations that are specified here are *abstract* in that they represent only main and important parts of *actual* access control configurations that have many details. Moreover, our specification deals with *static* access control in that it considers an access control configuration given at a specific instance of times.

In general, our logical specification is divided into three parts and each part deals with each kind of permissions, ie. ACL permissions, rights and abilities. In particular, our specification employs JSS's ASL to deal with ACL permissions and rights, and it employs GB's RBAC to deal with abilities.

As a matter of notations, predicate, function and constant symbols begin with lower case letters. Variable symbols start with upper case letter. All variables in a sentence are assumed to be universally quantified with scope the entire sentence.

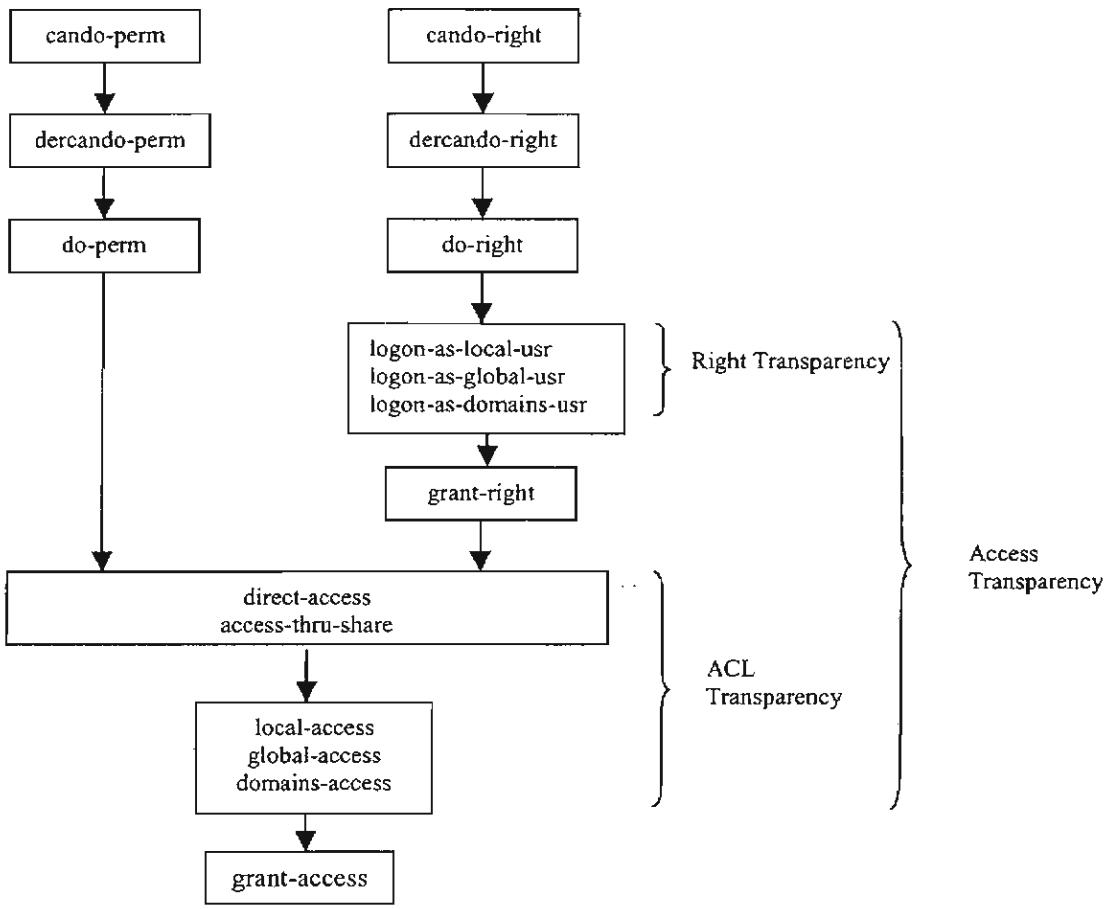

Fig. 1 Structure of Specification for ACL Permissions

The figure 1 illustrates the general structure of our specification for dealing with ACL permissions and rights.

In figure 1, predicates *cando-perm* and *cando-rights* are parts of an access control configuration and they are defined by objects' owners and administrators. Moreover, other predicates eg. *dercando-perm* are used to specify the access control mechanism in NT. Details of access control configurations are discussed in section 4.

We extend JSS's ASL by providing an additional level called *Access Transparency* on top of JSS's ASL. The Access Transparency will be discussed in section 3.3.

We shall discuss the detail of our specification sufficient for our purpose here. Further details of our specification can be found in [18].

3.1) ACL Permissions

For our purpose here, it is useful to classify ACL permissions into two kinds: *primitive* and *standard* permissions. Primitive permissions are basic permissions, for example *r* on an object stands for the permission to read data in the object. On the other hand, standard permissions are high-level permissions, for example, *change* on an object stands for the permission to modify and read data in the object. A standard permission is defined by primitive permissions.

The following is an authorization rule which stands for the assignment of ACL permissions on an object to a user (subject) by an owner of the object or by an administrator. The rule is specified by the *cando-perm* predicate.

cando-perm(S, P, O)

where *S* stands for a subject, *P* stands for an ACL permission and *O* means an object.

There are two kinds of derivation rules. The first kind is for the propagation of permissions into group members, and it is formulated as follows:

$$dercando\text{-}perm(S1, P, O) \leftarrow [\ cando\text{-}perm(S2, P, O) \wedge in^*(S1, S2)\ ]$$

where *S1, S2, P* and *O* stand for two subjects, an ACL permission and an object, respectively, and predicate *in** is the reflexive closure of predicate *dirin*, and predicate *dirin(S1,S2)* means that subject *S1* is a member of group *S2* and it is assigned by an administrator..

Intuitively, this rule means that if subject *S2* is assigned ACL permission *P* on object *O* and subject *S1* is in the group of *S2*, then *S1* is also assigned *P* on *O*.

The following is the other kind of derivation rules which is to derive primitive permissions from a standard permission.

$$dercando\text{-}perm(S, P, O) \leftarrow [\ dercando\text{-}perm(S, P1, O) \wedge std\text{-}perm\text{-}defn(P1, P2) \wedge P \in P2\ ]$$

where *std-perm-defn(P1,P2)* means that set *P2* of primitive permissions defines standard permission *P1*, eg. *std-perm-defn(change, {r,w,x,d})*.

The following rule is called the conflict resolution rule (*do-perm*) which aims to resolve conflicts amongst ACL permissions.

$$do\text{-}perm(S, P, O) \leftarrow [\ dercando\text{-}perm(S, P, O) \wedge \neg dercando\text{-}perm(S, none, O)\ ]$$

Intuitively, this conflict resolution rule states that the standard permission *none* assigned to subject *S* on object *O* cancels all other permissions given for subject *S* on object *O*. Note that *none* permission means no access, and it is to prevent any access to an object.

3.2) Rights

User rights are permissions that allow a user to use services provided by a system, for example, the right to log on a machine locally. Moreover, some of the user rights are permissions used for maintaining a system, for example the right to back up files and directories into a tape. Each administrator has some predefined rights for their administrative tasks. However, additional rights can be assigned to a user or an administrator, and they can also be revoked later. The assignment of rights to a user can be seen as a form of delegation.

The assignment of right *R* on object *O* to subject *S* by an administrator is formalized by the following predicate.

cando-right(S, R, O)

where *S, R* and *O* stand for a subject, a right and an object, respectively.

Similar to that for ACL permissions, there is the derivation rules for rights as follows.

$$dercando\text{-}right(S1, R, O) \leftarrow [\ cando\text{-}right(S2, R, O) \wedge in^*(S1, S2)\ ]$$

Before we discuss rules for conflict resolution, we need to discuss the concept of user properties. When a user account is created by an administrator, certain properties of the user account can be specified, eg. allowed logon hours and machines that the user can log on to.

The following is the conflict resolution rule for the right to logon locally (*ll*). Intuitively, it states that either

the absence of the user property *logon-to* or the presence of the user property *account-disabled* assigned to subject *S* on machine *M* cancels the right to log on locally for subject *S* on machine *M*.

$$do\text{-}right(S, R, M) \leftarrow [\ dercando\text{-}right(S, R, M) \wedge R = \text{"}ll\text{"} \wedge has\text{-}usr\text{-}prop(S, logon\text{-}to, M) \wedge \neg has\text{-}usr\text{-}prop(S, account\text{-}disabled, M)\ ]$$

where predicate *has-usr-prop(S, logon-to, M)* means that that user *S* has the user property *logon-to* to machine *M*.

For other rights, there is no conflict resolution rules and this is captured by the following.

$$do\text{-}right(S, R, M) \leftarrow [\ dercando\text{-}right(S, R, M) \wedge R \neq \text{"}ll\text{"}\ ]$$

3.3) Access Transparency

Access transparency is concerned with the ability of a user to make *an access* to an object stored at a machine from any machine in a distributed system. Access transparency involves two kinds of transparency : *ACL permission transparency* (or just ACL transparency) and *right transparency*.

ACL transparency is about whether or not an ACL permission is transparent (usable) to a user at a machine in a distributed system. In other words, ACL transparency means that a user can use an ACL permission to access an object from any machine in a distributed system. Right transparency however is concerned with whether or not a right is transparent to a user at a machine in a distributed system.

As illustrated in figure 1, *ACL transparency layer* is to determine whether a subject can access an object with an ACL permission *locally*, *globally* or *across domains*. The *right transparency layer* however is to determine whether a subject can use a right as a *local user*, a *global user* or a *domain user*.

By a *domain user*, we mean a kind of a *global user*. The difference between domain users and global users is discussed in section 3.3.2.

3.3.1) ACL Transparency

The following rule is for ACL transparency and it is called *distributed access control rule*.

$$grant\text{-}access(U, A, O) \leftrightarrow [\ local\text{-}access(U, A, O) \vee global\text{-}access(U, A, O) \vee domains\text{-}access(U, A, O)\ ]$$

where *U*, *A* and *O* stand for a subject, an ACL permission, and an object, respectively.

Predicate *grant-access(U, A, O)* means that user *U* is granted an ACL permission *A* on object *O* if and only if such a distributed access is granted *locally*, *globally* or *across domains*.

Local access means that a user can access an object from a machine that hosts the object. It is formalized as follows:

$$local\text{-}access(U, A, O) \leftrightarrow \exists M\ [\ direct\text{-}access(U, A, O) \wedge object\text{-}of(O, M) \wedge grant\text{-}right(U, ll, M)\ ]$$

where *direct-access(U, A, O)* stands for the grant of a direct access to subject *U* on object *O* with ACL permission *A*, *object-of(O, M)* means that object *O* is located in machine *M*, and *grant-right(U, ll, M)* means that user *U* is granted right *ll* at machine *M*.

A *direct access* is an access made directly to the desired object whereas an *indirect access* is an access made to the desired object via another object (ie. *share*). Such indirect access is formalized by predicate *access-thru-share* and the concept of indirect access will be discussed below.

On the other hand, global access means that a user can access an object from a machine which does not host the object, but it is in the same domain as the machine that hosts the object.

$$global\text{-}access(U, A, O) \leftrightarrow \exists O', M\ [\ access\text{-}thru\text{-}share(U, A, O, O') \wedge glob\text{-}access\text{-}from(U, O, M)\ ]$$

where predicate *access-thru-share(U, A, O, O')* means that subject *U* is allowed to access object *O* via share *O'* with ACL permission *A*, and predicate *glob-access-from (U, O, M)* means that machine *M* allows subject *U* to make a global access to object *O*.

A *share* is an object which acts as a link (or an entry) to an *actual* object that locates in another machine. Any access to a machine that hosts the object from another machine must be made through a share. A share can be assigned ACL permissions. Thus, it can be used as an additional filter to control an access from other machines.

Domain access means that a user can access an object from a machine which locates in a different domain than that of the machine that hosts the object.

$$domains\text{-}access(U, A, O) \leftrightarrow \exists O', M\ [\ access\text{-}thru\text{-}share(U, A, O, O') \wedge dom\text{-}access\text{-}from(U, O, M)\ ]$$

where predicate *dom-access-from(U, O, M)* means that machine *M* allows subject *U* to make a domain access to object *O*.

The following two rules show definitions of global and domain accesses from certain machines, respectively.

glob-access-from(U, O, M) $\leftrightarrow \exists D,M'$ [
object-of(O, M') $\wedge$ *machine-of(M', D)* $\wedge$
grant-right(U, acn, M') $\wedge$
global-user-of(U,D) $\wedge$ *machine-of(M, D)* $\wedge$
$M \neq M'$ $\wedge$ *grant-right(U, ll, M)*]

where predicate *machine-of(M, D)* means that M is a machine in domain D, predicate *global-user-of(U, D)* means that subject U is a member in domain D, *acn* shorts for the right to access a machine across the network.

Intuitively, this rule states that subject U can access object O from machine M globally if and only if U can logon at machine M which is in the same domain as machine M' that hosts O, and U has the right to access M' across the network.

dom-access-from(U, O, M) $\leftrightarrow \exists D1,D2,D3,M'$ [
object-of(O, M') $\wedge$ *machine-of(M', D1)* $\wedge$
global-user-of(U, D3) $\wedge$ $D1 \neq D3$ $\wedge$
trust(D1,D3) $\wedge$ *grant-right(U, acn, M')* $\wedge$
machine-of(M, D2) $\wedge$ *grant-right(U, ll, M)*
$\wedge$ *trust*(D2, D3)*]

where predicate *trust** is the reflexive closure of predicate *trust*, and predicate *trust(D1,D2)* means that domain $D1$ trusts domain $D2$ in that domain $D1$ allows user accounts from domain $D2$ to make an access to $D1$.

This rule captures two cases of the domain access. The first case is that user U in a domain which is trusted by the domain hosting object O accesses from a machine in U's domain. The second case is that user U in a domain trusted by O's domain accesses object O from a machine which locates in yet *another domain* that trusts U's domain. Note that in the second case, there are three domains involved.

3.3.2) Right Transparency

For right transparency, we deal with the right to logon locally (*ll*) and the right to access a computer from network (*acn*) separately from other rights.

The following rule deals with the right *acn*.

grant-right(U, acn, M) $\leftarrow$ *do-right(U, acn, M)*

where predicate *grant-right(U, R, M)* means that subject U is granted right R that can be used at machine M.

This rule captures the most straightforward kind of right transparency.

The following rule deals with the right *ll*. It should be noted that it shares some similarity with *grant-access*.

grant-right(U, ll, M) $\leftarrow$ [*logon-as-local-usr(U, M)* $\vee$
logon-as-global-usr(U, M) $\vee$
logon-as-domains-usr(U, M)]

where *logon-as-local-usr(U, M)*, *logon-as-global-usr(U, M)* and *logon-as-domain-usr(U, M)* mean that user U can logon at machine M as a local user, a global user or a domain user, respectively.

Since the right transparency is concerned with whether or not a user can use a right at a machine, the concept of local users, global users and domain users has to be discussed with respect to machines.

A *local user* at a machine is simply a *local* user at the machine. A *global user* at machine M is a global user of the same domain as M whereas a *domain user* at machine M' is a global user of other domain different from the domain in which M' is.

The following two rules show definition of *logon-as-local-usr* and *logon-as-global-usr*, respectively. We omit the definition of *logon-as-domains-usr* here.

logon-as-local-usr(U, M) $\leftrightarrow$ [*do-right(U, ll, M)* $\wedge$
local-user-of(U, M)]

where *local-user-of(U, M)* means that U is a local user at machine M.

logon-as-global-usr(U, M) $\leftrightarrow \exists D$ [*do-right(U, ll, M)* $\wedge$
machine-of(M, D) $\wedge$ *global-user-of(U, D)*]

The following rule deals with all the other rights, ie. those except the right *ll* and *acn*.

grant-right(U, R, M) $\leftarrow$[*do-right(U, R, M)* $\wedge$ $R \neq ll$ $\wedge$
$R \neq$ *acn* $\wedge$ *grant-right(U, ll, M)*]

This rule means that apart from the rights *ll* and *acn*, the granting of any other rights requires the right *ll*. In other words, a user requires the right *ll* at a machine in order to obtain all the other rights at the machine.

3.4) Abilities

Abilities [11] can be considered as a special kind of rights. However, the difference between rights and abilities is that rights are permissions used for maintaining a system whereas abilities are permissions used for significant administration of a system. Examples of abilities are the ability to create and remove local user accounts, and the ability to assign user rights to users.

Abilities are assigned to each type of administrators according their administrative functions. For example, account operators are assigned the ability to create and manage user accounts. Moreover, abilities are *predefined* for each type of administrator and they are unrevokable, unlike user rights which are revokable.

Thus, abilities can be considered as *authority* that each type of administrators has in order to manage the NT system.

Each type of administrators can be thought of as a role. Therefore, they can be dealt with naturally in the model of RBAC.

The figure 2 shows the relationship between four entities, ie. roles, users, abilities (administrative permissions) and supervisees (objects), that is employed in our logical specification. The relationship establishes that a user due to its role has an administrative permission (ability) on an object (supervisee).

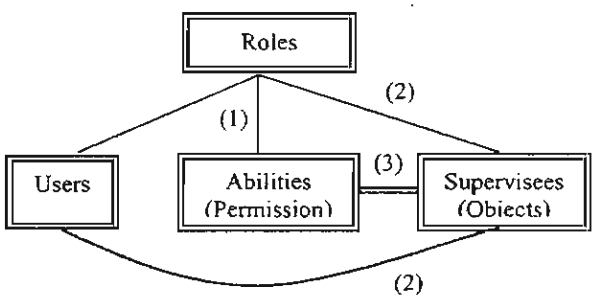

***Fig.2** Relation between roles, users, abilities and supervisees*

Intuitively, the figure 2 shows that there are two kinds of relations between the four entities. In particular, the first kind of relation, denoted by a single line, is based on the inheritance of roles whereas the second kind of relation denoted by a double line is just an ordinary type matching which relates the types of abilities to the types of their supervisees.

Our specification extends existing logic-based specifications [13, 14] for RBAC to deal with the relationships :

a) between roles/users (supervisors) and objects (supervisees) (2),
b) between roles and permissions (1), and
c) between permissions and objects (3).

The figure 3 illustrates the general structure of our specification for dealing with abilities.

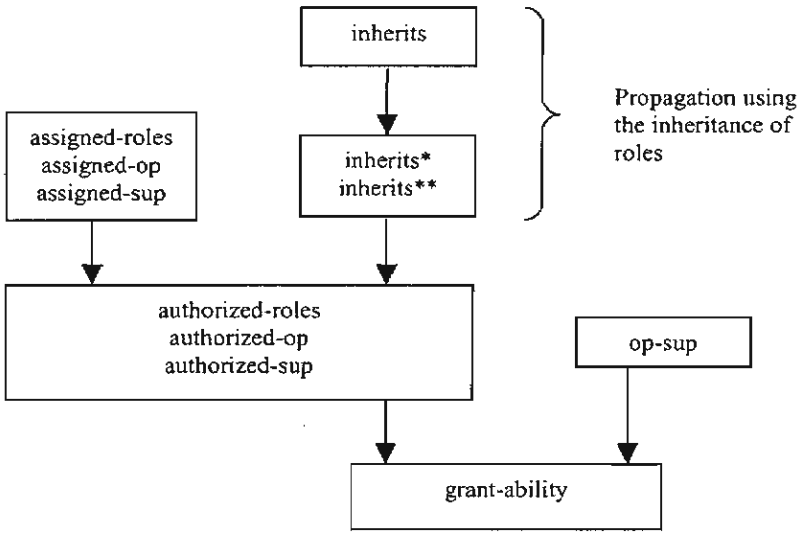

***Fig. 3** Structure of Our Specification for Abilities*

Referring to figure 3, predicates *inherits(Role, Role)*, *assigned-roles(User, Role)*, *assigned-op(Role, Ability)*, *assigned-sup(Supervisor, Supervisee)* and *op-sup (Ability, Supervisee)* are parts of an access control configuration whereas other predicates, eg. *authorized-roles*, represent the access control mechanism.

There are two kinds of access control configurations for dealing with abilities : *fixed* and *varied*. Fixed configurations are those which do not change but are predefined whereas varied configurations are those that can be defined and changed later by system administrators. Predicates *inherits*, *assigned-op*, *assigned-sup* and *op-sup* express fixed configurations

but predicate *assigned-roles(User, Role)* specifies varied configurations.

The following is a top-level rule for dealing with abilities.

$$grant\text{-}ability(U, Ab, S) \leftrightarrow \exists R [authorized\text{-}roles(U, R) \wedge authorized\text{-}op(R, Ab) \wedge (authorized\text{-}sup(U, S) \vee authorized\text{-}sup(R, S)) \wedge op\text{-}sup(Ab, S)]$$

where *U, Ab, S, R* stand for a subject (user), an ability, a supervisee (object), a role, respectively, and predicate *grant-ability(U, Ab, S)* means that subject *U* is granted ability *Ab* over supervisee *S*.

Intuitively, this rule means that such ability over *S* is *granted* to subject *U* if and only if *U* is *authorized* for role *R* for which the ability is *authorized*, and either *R* or *U* is *authorized* to manage supervisee (object) *S*.

Following [14], the rule for the authorization of users and roles is formulated as follows:

$$authorized\text{-}roles(U,R) \leftrightarrow \exists R' [assigned\text{-}roles(U,R') \wedge inherits^*(R',R)]$$

where *authorized-roles(U,R)* means that subject *U* is authorized for role *R*, *assigned-roles(U,R')* means that role *R'* is assigned to subject *U*, and *inherit** is the reflexive and transitive closure of predicate *inherits*, and *inherits(R1,R2)* means that role *R1* inherits abilities from role *R2* in the sense that *R1* is *superior* to *R2*.

Intuitively, this rule means that user *U* is authorized for not only an assigned role but also any role which is *inferior* to the assigned role. Predicate *assigned-roles* is definable by a system administrator.

Predicates *authorized-op* and *authorized-sup* are defined similarly to predicate *authorized-roles*, and their definitions are given as follows:

$$authorized\text{-}op(R,Ab) \leftrightarrow \exists R' [assigned\text{-}op(R',Ab) \wedge inherits^*(R,R')]$$

where *authorized-op(R,Ab)* means that ability *Ab* is authorized for role *R*, and *assigned-op(R', Ab)* means that role *R'* is assigned to ability *Ab*.

$$authorized\text{-}sup(S1,S2) \leftrightarrow \exists S [assigned\text{-}sup(S,S2) \wedge inherits^{**}(S1,S)]$$

where *authorized-sup(S1,S2)* means that supervisor *S1* is authorized to manage supervisee *S2*, *assigned-sup(S, S2)* means that supervisor *S* is assigned to manage supervisee *S2*, *inherits*** is the reflexive and transitive closure over two predicates *inherits* and *ind-inherits*. Predicate *ind-inherits* expresses a kind of inheritance and its detail is omitted here.

Predicate *op-sup(Op,S)* is used to capture the relation between abilities and supervisees. It matches type of abilities to type of supervisees. The following shows an example.

$$op\text{-}sup(Op, S) \leftarrow [local\text{-}account\text{-}ability(Op) \wedge is\text{-}local\text{-}account(S)]$$

This rule is used to match abilities concerning local user account to local user accounts, for example the ability to create and manage local user accounts is paired up with local user accounts.

The following are examples of three predicates : *inherits, assigned-op* and *assigned-sup*.

$$inherits(domain\text{-}admin(D), local\text{-}admin\text{-}ctrl(C)) \leftarrow controller\text{-}of(C, D)$$

where *C* and *D* stand for a machine acting as a domain controller and a domain name, respectively, and *domain-admin(D)* and *local-admin-ctrl(C)* stand for domain administrator role and local administrator role, respectively.

$$assigned\text{-}op(power\text{-}user(M), mla) \leftarrow machine(M)$$

where *mla* stands for the ability to create and manage local user accounts, and *power-user(M)* stands for power user role at machine *M*.

$$assigned\text{-}sup(power\text{-}user(M), G) \leftarrow [(G = guest) \vee (G = users) \vee (G = power\text{-}user(M))]$$

where *G* and *M* stand for a user group and a machine, respectively.

4) REASONING ABOUT ACCESS CONTROL

We discuss two kinds of reasoning about access control configurations. The first kind is to verify properties of a given access control configuration. The second kind is to generate a new access control configuration.

4.1) Verification of Access Control Configurations

We propose *novel* and *practical* verification properties for helping *object owners and system administrators* to analyze access control configurations in NT. Moreover, our methodology is practical in that it deals with the *use* of an access control system.

Formally, the verification of properties of access control configurations can be understood as follows:

$$M \cup \Delta \models P$$

where *M* stands for our logical specification of NT discussed in the previous section, Δ stands for an access control configuration, and *P* stands for properties that we are interested in verifying.

Intuitively, *an access control configuration* is considered as a set of *assumptions* and our *specification* discussed in the previous section is thought of as a set of *axioms*. Thus, any *logical consequence* derived (deductively) from the set of axioms and such a set of assumptions is considered as a *property* provable from the given access control configuration and the access control specification.

An access control configuration is defined by predicates *cando-perm, cando-right, dir-in, has-user-prop, global-user-of, local-user-of, trust, assigned-roles, machine-of, object-of, local-group-of, global-group-of, inherits, controller-of, assigned-roles, assigned-op, assigned-sup* and *op-sup*. Some of these predicates are defined by objects' owners but others are defined only by system administrators.

4.1.1) Correctness

This property aims to analyze whether an NT configuration defined by an administrator or an object owner is *correct* with respect to the administrator's or the object owner's intention, respectively. The intentions represent high-level requirement of object owners and administrators on access control in the system. We argue that in general the intention of owners of objects is to allow only trusted users to access some of their information. For convenience here, we shall use the term "*users*" to refer to both object owners and administrators.

An access control configuration is *correct* if it captures users' intention. Users' intention is defined by using predicate *intended-access(S, P, O)* where *S* stands for *trusted* subject, *O* stands for information whose accessibility the owner is concerned about, and *P* stands for access permission : ACL permissions and rights. The concepts of *trust subjects* and *owner-concerned information* are *application-dependent* and are up to users to define who should be trusted and what objects whose accessibility are concerned.

We argue that it is *intuitive* to consider *owner-concerned* information to represent users' intention, since users are normally concerned with the control of access to some of their information (objects), not all of objects.

The correctness property of a configuration can be defined by the following formulae.

$$intended\text{-}access(U, P, O) \rightarrow grant\text{-}access(U, P, O)$$

Intuitively, these formulae ensure that permissions intended by users are granted by an access control configuration.

4.1.2) Security : Confidentiality and Integrity

The security property consists of two components : *confidentiality* and *integrity*. The confidentiality property is expressed as follows:

$$[\, grant\text{-}access(U, P, O) \wedge sensitive\text{-}info(O) \,] \rightarrow trusted\text{-}subject(U)$$

Intuitively, this means that *only* trusted subjects can access sensitive information. In other words, if sensitive object can be accessed by a subject, then the subject must be a trusted (authorized) one. Note that as in the correctness property, it is up to users to define which objects are sensitive.

On the other hand, the integrity property is expressed as follows:

$$[\, grant\text{-}access(U, P, O) \wedge sensitive\text{-}info(O) \wedge trusted\text{-}subject(U) \,] \rightarrow intended\text{-}access(U, P, O)$$

Intuitively, this ensures that an authorized (trusted) subject is allowed to access to sensitive information in an *authorized* manner only. In other words, it guarantees that permissions granted by an access control configuration are those intended by users. Note that this property can be seen as a *converse* of the correctness property.

4.1.3) Policy Satisfaction

This kind of property aims to show that an access control configuration satisfies a desirable security policy defined by a security manager for an organization. Such security policy represents high-level security requirements of a system. Security policy for NT can be defined systematically, but it is beyond the scope of our paper here. In the following, we shall consider examples of security policy, instead.

1) The policy "*an authorized user for highly sensitive information is allowed to access such information from the machine that hosts the information only*" can be expressed as follows:

$$[\, grant\text{-}access(U, P, O) \wedge highly\text{-}sensitive\text{-}info(O) \wedge trusted\text{-}subject(U) \,] \rightarrow [\, local\text{-}access(U, P, O) \wedge \neg global\text{-}access(U, P, O) \wedge \neg domain\text{-}access(U, P, O) \,]$$

2) The policy "*an authorized user can access sensitive information from a secure machine; however, the same user cannot access the same information from a non-secure (possibly public) machine*" can be captured by the following two statements.

[grant-access(U, P, O) ∧
sensitive-info(O) ∧ trusted-subject(U)] →
∃M [secure-machine(M) ∧ access-from(U, O, M)]

[grant-access(U, P, O) ∧ sensitive-info(O) ∧
trusted-subject(U) ∧ ¬secure-machine(M)] →
¬access-from(U, O, M)

where *access-from(U, O, M)* means that subject *U* can access object *O* from machine *M*.

4.1.4) Finding an Explanation of Granted Permissions

This kind of property aims to find an explanation of certain permissions granted by a given access control configuration. Such explanation would be a part of a given configuration responsible for the grant of permissions. In other words, such explanation would clarify the cause of the grant of such permission in the configuration. This property is classified as verification of access control configurations since it is about an analysis of an existing configuration.

Let consider an example of this property. An administrator would like to find out why John, a user in the system, has ACL permission *r* to file *grade*. One explanation might be that John is granted the permission by the owner of the file directly, or John is a member of a local group granted the permission.

Formally, this property can be understood as follows.

Definition 1 Explanation
Given an access control configuration Δ' and a permission Q such that M ∪ Δ' ⊨ Q where M stands for our logical specification for NT, an explanation for the grant of permission Q is Δ that satisfies the following : M ∪ Δ ⊨ Q and Δ ⊆ Δ'.

There may be several explanations for the grant of some permission. However, the kind of explanations that are useful is the minimal one, which is defined by the following definition.

Definition 2 Minimal Explanation
An explanation *Δ* for the grant of permission *Q* is *minimal* if and only if there is no other explanation *D* which satisfies *M ∪ D ⊨ Q and D ⊂ Δ*.

Note that if a minimal explanation of the grant of a permission is an empty one, then this means that the permission is pre-defined, eg. the ability of administrators.

Example 1 Finding an explanation
Suppose that *Δ* = { *object-of('c:\data\salary.db', tiger), has-usr-prop(john, logon-to, tiger), cando-right(john, ll, tiger), local-user-of(john, tiger), cando-perm(john, r, 'c:\data\salary.db'), cando-perm(manager, r, 'c:\data\salary.db'), dirin(john, manager)* }. Suppose further that *Q* is *grant-access(john, r, 'c:\data\salary.db')*. There are two minimal explanations for the grant of *Q*, which are *Δ1* = { *object-of ('c:\data\salary.db', tiger), has-usr-prop(john, logon-to, tiger), cando-right(john, ll, tiger), local-user-of(john, tiger), cando-perm(john, r, 'c:\data\salary.db')* } and *Δ2* = { *object-of('c:\data\salary.db', tiger), has-usr-prop (john, logon-to, tiger), cando-right(john, ll, tiger), local-user-of(john, tiger), cando-perm(manager, r, 'c:\data\salary.db'), dirin(john, manager)* }.

4.2) Synthesis of Access Control Configurations

This property aims to generate a new access control configuration that grants a set of desirable permissions. This property would help a novice administrator to configure an access control system.

It is useful to consider situations where administrators are given a *partial* configuration and are asked to generate the rest of the configuration. Such a situation may be that an administrator must design a configuration with the presence of some constraints. For example, in some system, it requires that users are divided into two groups : students and staffs.

Moreover, another example of such situation is that there is a change of the system environment, eg. new machines, new printers or new users. In that case, a partial configuration represents an existing and unchanged part of environment.

Definition 3 Synthesis of an access control configuration
The partial synthesis of an access control configuration is the generation of Δ in the presence of a given partial configuration Δ' such that M ∪ Δ ∪ Δ' ⊨ G and Δ ∩ Δ' = ∅, where G represents a set of desirable permissions.

Example 2 Synthesis of Access Control Configurations
Suppose that a given partial configuration *Δ'* = { *object-of('c:\data\salary.db', tiger), machine-of(tiger, jungle), global-user-of(john, jungle)* }. This partial configuration means that the file *salary.db* locates in machine *tiger* which is the domain *jungle*, and *jim* is a global user in domain *jungle*. Suppose that *G* is *grant-access(jim, r, 'c:\data\salary.db')*. One possibility of the rest of the configuration *Δ* is that { *has-usr-prop(jim, logon-to, tiger), cando-right(jim, ll, tiger), cando-perm(jim, r, 'c:\data\salary.db')* }

The new configuration (Δ) obtained from the synthesis must satisfy the following two properties:

Property 1 Desirable Synthesized Configurations

1) Δ is *minimal* with respect to set inclusion, ie. there is no other configuration D which satisfies $M \cup D \cup \Delta' \models G$ and $D \subset \Delta$.
2) Δ satisfies some set of *integrity constraints* (IC), ie. $M \cup \Delta \cup \Delta' \models IC$.

Intuitively, the first condition means that the configuration generated is minimal in that there is no smaller configuration that grants the same permissions. Note that if a minimal configuration that is synthesized is an empty one, then this means that an existing partial configuration Δ' can already produce desirable permissions G.

There are many kinds of integrity constraints. Some is for the purpose of maintaining the consistency of the system. For example, the following constraint means that a local user which is assigned to be a member of a local group must be the local user at the same machine as the local group.

$$[\ dirin(S, G) \wedge is\text{-}local\text{-}user(S) \wedge is\text{-}local\text{-}group(G)\ ] \rightarrow \exists M\ [\ local\text{-}user\text{-}of(S, M) \wedge local\text{-}group\text{-}of(G, M)\ ]$$

Other kind of integrity constraints may express the policy on the synthesis of the new configuration. For example, such policy may be that ACL permissions should be assigned to a group of users rather than to users directly. This policy aims to obtain a configuration which is easy to maintain. Such policy can be expressed as the following integrity constraint:

$$grant\text{-}access(U, P, O) \rightarrow \exists G\ [\ dir\text{-}in(U, G) \wedge is\text{-}group(G) \wedge cando\text{-}perm(G, P, O) \wedge \neg cando\text{-}perm(U, P, O)\ ]$$

5) Conformance Testing

The conformance testing [19] aims to ensure that a specification corresponds exactly to an actual implementation. In particular, the conformance testing would guarantee that a specification is both *correct* (sound) and *complete* with respect to an actual implementation.

A specification is *correct* with respect to an implementation if and only if what the specification describes is an actual behavior of the implementation. Moreover, a specification is *complete* with respect to an implementation if and only if the specification is capable of describing all aspects of the behavior of the implementation.

In the conformance testing [19], there are two kinds of tests : functional test and structural test. We discuss how the conformance testing can be applied to our specification here. However, we argue that the conformance testing employed here ensures that our specification is correct with respect to the NT system, but it does not guarantee that our specification is complete with respect to the NT system.

5.1) Functional Tests

Functional tests aim to ensure that a specification corresponds to an implementation in terms of their *functionality*. Since our specification is about access control, we argue that the following is the main functionality of our specification.

The specification (and the implementation) must grant a permission to access an object to a subject if and only if the permission is authorized by the owner of the object.

Our methodology to carry out the functional test is first to create test data : one for each kind of authorization which could possibly be given by an object's owner. Then, such test data are taken as inputs to both our specification and the NT system.

Our approach to classify types of authorization is to classify types of subjects, permissions and objects those of which can be allowed by an owner of the objects. For example, subjects are divided into two main kinds : individual users and groups.

The functional test reveals the *main aspect* of the correspondence between our specification and NT since it addresses the functionality of the two. However, this kind of test is *subjective* to test data.

5.2) Structural Tests

Structural tests aim to ensure that a specification corresponds to an implementation in terms of their *behaviors*. In other words, the structural tests should deal with how a specification and an implementation achieve the desired access control. Since our specification is divided into layers and each layer performs certain tasks, it is intuitive to test our specification for each of its layers with respect to the NT system.

For example, the structural tests should include tests on the conflict resolution rule, the derivation from standard ACL permissions, and etc. Similarly to the functional test, a group of test data is created which cover each possibility of the condition of each rule.

Even though the structural test is also subjective to test data, it reveals the correspondence between our specification and NT more *systematically* than the functional test does.

5.3) Discussion

The result obtained [20] for both the functional test and the structural test shows that our specification presented here produces the same output as the NT system as far as the test data are concerned. This shows that our specification is *correct* with respect to the NT system. However, since we cannot enumerate all possible functionalities and behaviors of the NT system and we cannot create test data which cover all such possibilities, we cannot guarantee for the completeness of our specification. It is commonly recognized that the enumeration of all possible behaviors of a black-box implementation is impossible [19]. As a result, the guarantee for the incompleteness of such kind of implementation, including NT, is not possible also.

However, we argue that it is *sufficient* to consider just a correct specification in order to perform the reasoning about access control discussed in section 4. In particular, if it can be verified that a configuration satisfies the verification properties (eg. correctness, security and policy satisfaction), then the result is correct due to the correctness of our specification. However, if it cannot be verified that a configuration satisfies the properties, then it may be due to either an error in the configuration or the incompleteness of our specification. In such situation, human assistance to examine the actual cause is required.

6) CONCLUSION AND FUTURE WORK

In this paper, we have carried out a study on a logical methodology for access control in NT. The contribution of this paper is a logical specification for distributed access control in NT and a reasoning methodology for analyzing access control configurations in NT. The logical specification developed here is new in that it deals with the distributed access control system whereas existing specifications deal with access control in a centralized system. The benefit of our specification is that it provides a precise and clear understanding in the access control mechanism in NT.

Moreover, our reasoning methodology helps system administrators to analyze access control configurations. In particular, the reasoning methodology helps verifying whether access control configurations have desirable properties, such as correctness and security. Moreover, it helps generating new access control configurations that have desirable properties. We argue that our reasoning methodology is not only practical but also new.

Our specification is static in that it deals with an access control configuration given at an instance of times. Therefore, several verification properties, in particular the security, are restricted to the static notion. However, a dynamic version of our specification has been developed in [20] and several properties for dynamic access control have been studied there.

Furthermore, a prototype of our methodology has been developed also, and our methodology has been applied to a case study, namely, an internet service providing system at KMUTT university. The results obtained show that our methodology has several benefits. In particular, it provides a systematic and automatic approach to deal with the analysis of access control configurations. Moreover, it can avoid errors that might occur in manual analysis of access control configurations. The details of the prototype and the case study can be found in [21].

As a future direction, we plan to apply our logical methodology for other kinds of access control applications, namely Database systems. Moreover, it would be interesting to study this logical methodology for other areas in computer security.

Acknowledgement

I would like to thank Professor Phan Minh Dung, Professor Bob Kowalski and Dr. Mark Halls for their helpful and encouraging discussions. Moreover, I would like to thank Chukiat Yangyuenbangchan and Orawan Tunsungwon for the discussion on access control in NT. The research reported in this paper is supported by Thailand Research Fund, National Research Council of Thailand and King Mongkut's University of Technology Thonburi.

Reference

[1] J. Glasgow, G. MacEwen and P. Panangaden, A Logic for Reasoning about Security, *ACM Transactions on Computer Systems*, Vol. 13, No. 3, 1992.

[2] T.Y.C. Woo and S.S. Lam, Authorization in Distributed Systems: A new approach, *Journal of Computer Security*, 2(2,3), 1993.

[3] M. Abadi, M. Burrows, B. Lampson and G. Plotkin, A Calculus for Access Control in Distributed Systems, *ACM Transactions on Programming Language and Systems*, Vol. 15 No. 3, 1993.

[4] S. Jajodia, P. Samarati and V.S. Subrahmanian, A Logical Language for Expressing Authorizations, *In Proceedings of IEEE Symposium on Research in Security and Privacy*, 1997.

[5] F. Massacci, Reasoning about Security : a Logic and a decision method for Role-based Access Control, *in Proceedings of the International Joint Conference on Qualitative and Quantitative*

Practical Reasoning (ECSQARU/FAPR), Spinger-Verlag, 1997.

[6] Y. Bai and V. Varadharajan, A Logic for State Transformation in Authorization Policies, *in Proceedings of IEEE Computer Security Foundation Workshop*, 1997.

[7] E. Bertino, F. Buccafurri, E. Ferrari and P. Rullo, A Logical Framework for Reasoning on Data Access Control Policies, *in Proceedings of IEEE Computer Security Foundation Workshop*, 1999.

[8] L.L Kassab and S. J. Greenwald, Towards formalizing the Java Security Architecture of JDK 1.2, *In proceedings of European Symposium on Research in Computer Security (ESORICS)*, Springer-Verlag, 1998.

[9] M.A. Harrison, W.L. Ruzzo and J.D. Ullman, Protection in Operating Systems, *Communication of ACM*, 19(8), 1976.

[10] C. Rutstein, *Windows NT Security : A practical guide to securing Windows NT servers and workstations*, McGraw-Hill, 1997.

[11] T. Sheldon, *Windows NT Security Handbook*, McGraw-Hill, 1997.

[12] S. Sutton, *Windows NT Security Guide*, Addison-Wesley Press, 1997.

[13] D.F. Ferraiolo, J.A. Cugini, D.R. Kuhn, Role-Based Access Control (RBAC): Features and Motivations, *In Proceedings of IEEE Computer Security Applications*, IEEE Computer Society press, 1995.

[14] S.I. Gavrila and J.F. Barkley, Formal Specification for Role Based Access Control User/Role and Role/Role Relationship Management, *In Proceedings of ACM Workshop on Role-Based Access Control*, ACM press, 1998.

[15] R.S. Sandhu, Role-Based Access Control, *Advances in Computers*, M. Zerkowitz (ed.), Vol. 48, Academic Press, 1998.

[16] C. Ramaswamy and R. Sandhu, Role-Based Access Control Features in Commercial Database Management Systems, *In Proceedings of National Computer Security Conference*, 1998.

[17] D.F. Ferraiolo and J.F. Barkley, Specifying and Managing Role-Based Access Control within a Corporate Intranet, *In Proceedings of ACM Workshop on Role-Based Access Control*, ACM press, 1997.

[18] Y. Permpoontanalarp, *A Logical Methodology for Access Control in Windows NT*, Technical Report, Department of Computer Engineering, King Mongkut's University of Technology Thonburi, 2000.

[19] G.J. Holzmann, *Design and Validation of Computer Protocols*, Prentice Hall Software Series, 1991.

[20] O. Tunsungwon, *A Formal Method for Dynamic Access Control in Windows NT*, Master Thesis, Department of Computer Engineering, King Mongkut's University of Technology Thonburi, Bangkok Thailand, 2000.

[21] K. Chutipan, K. Banjerdpisarnkul and D. Srinarong, *A software tool for analyzing access control in Windows NT and its application to an KMUTT internet service providing system*, Bachelor Thesis, Department of Computer Engineering, King Mongkut's University of Technology Thonburi, Bangkok Thailand, 2001.

A Graph Theoretic Model for Hardware-based Firewalls

Yongyuth Permpoontanalarp and Chaiwat Rujimethabhas
Logic and Security Laboratory
Department of Computer Engineering
King Mongkut's University of Technology Thonburi
91 Suksawasd 48, Ratburana, Bangkok 10140 Thailand
yongyuth@cpe.eng.kmutt.ac.th, s1410008@cc.kmutt.ac.th

Abstract

Firewalls offer a protection for private networks against external attacks. However, configuring firewalls is a difficult task. The reason is that the effects of a firewall configuration cannot be easily seen during the configuration time. As a result, errors and loopholes in firewall configurations, if exist, are discovered only after they actually happen at the execution time. In this paper, we propose a preliminary yet novel model and its methodology for hardware-based firewalls. Our model offers precise and simple understanding of effects of firewall configurations. Moreover, our methodology offers an analysis of effects of firewall configurations. In particular, it provides reasoning about the correctness of firewall configurations. Also, the redundancy and inconsistency of firewall rules can be reasoned about. As a result, many kinds of errors and loopholes of firewall configurations can be detected during the configuration time.

1. Introduction

Nowadays, firewalls (eg. [1,2]) become a widely used mechanism to achieve Internet security. Most, if not all, organizations whose computers have an Internet access are currently using firewalls. Firewalls locate between an internal network and an external network. Firewalls offer a protection for private (and internal) networks against external threats. In particular, firewalls ensure that only authorized information flows between internal networks and the external network are allowed.

Firewalls can be classified into two kinds : hardware-based and software-based. Hardware-based firewalls are routers that are also capable of filtering packets passing through the routers. Software-based firewalls are computers installed software for filtering packets. Such computers may be a gateway or a server. Normally, hardware-based firewalls are more complicated than software-based firewalls since the former is primitive and is not easily extensible. In this paper, we focus on hardware-based firewalls.

Even though firewalls could provide protections against external attacks, configuring firewalls is a difficult task. In general, a configuration for firewalls consists of a set of filtering rules and a set of activation rules. Filtering rules are rules that determine which packets will be allowed (or disallowed) to pass through a firewall. On the other hand, activation rules are rules stating the activation of an *ordered* set of filtering rules at a particular *direction* of a firewall.

The cause of the difficulty in configuring firewalls is that the effects of a firewall configuration cannot be easily seen during the configuration time (or before the execution time).

Usually, there are many filtering rules activated at *a specific direction of a firewall*. Those filtering rules are ordered since a firewall processes those rules from top to bottom in order to decide whether or not a packet should pass through the firewall. In other words, an upper rule takes precedence over a lower one. Therefore, in order to understand the effects of a set of those filtering rules, every rule in the set must be taken into account and a mechanism to calculate the total effects of those rules is not straightforward.

Furthermore, in order to see the effects of all filtering rules at *all directions of every firewall* in a system, the effects of filtering rules at each direction of each firewall have to be calculated first. Then, the combined effects of those filtering rules are computed, which is not simple at all.

Since the effects of a firewall configuration cannot be seen at the configuration time, many firewall configurations often have errors and loopholes. Most often, such errors and loopholes are discovered only after they actually happen at the execution time. This causes severe damage to the system.

Furthermore, it is difficult to maintain existing configurations written by someone else. Firewall

administrators cannot be sure of the actual effects of a filtering rule.

We argue that these problems occur because of the lack of firewall model and its methodology to understand and to analyze the effects of firewall configurations. In this paper, we propose a graph theoretic model and methodology for reasoning about hardware-based firewall configurations. Even though our model proposed here is preliminary, it is novel is that it is formal and it can analyze effects of firewall configurations in details.

Our model offers precise and simple understanding of effects of firewall configurations. Moreover, our methodology provides reasoning about the correctness of firewall configurations. Also, the redundancy and inconsistency of firewall rules can be reasoned about.

We discuss the background in section 2, and our model in section 3. Our methodology is discussed in section 4 and the justification that our model corresponds to actual firewalls is given in section 5. Related works and conclusion are discussed in section 6 and 7, respectively.

2. Background

2.1. Hardware-based firewalls

Hardware-based firewalls are routers which are capable of filtering packets. Such hardware-based firewalls may have many interfaces attached, and each interface applies a set of filtering rules in order to filter packets. Examples of hardware-based firewalls are Cisco firewalls and Nokia firewalls.

The model that we shall present is general for all hardware-based firewalls. However, for the ease of the presentation here, we shall present our model for Cisco routers. In the following, we shall discuss a simplified version of Cisco firewall configurations.

Cisco Firewall Rules are represented by (*SFR, SAR*) where *SFR* stands for a set of filtering rules and *SAR* stands for a set of activation rules.

In definition 1, words in boldface mean keywords and words in italic mean variables. *{a | b}* means a choice between *a* and *b*. Moreover, *[x]* means that *x* is optional.

Definition 1 *Filtering and Activation Rules*
Filtering rules are rules that define either the permission or prohibition of the flow of packets from one place to another whereas activation rules are rules which apply filtering rules to particular direction of certain router interfaces.

a) Filtering rules : Each filtering rule is assigned a number, called access-list-number. Many filtering rules can be grouped together by assigning the same access-list-number. A filtering rule is defined by the following:

access-list *access-list-number* {**deny** | **permit** }
{ *source-address* |
(*protocol* { *source-address* | **any** } [*Op Source-port*]
{ *dest-address* | **any** } [*Op Dest-port*]) }

where

- *source-address* and *dest-address* stand for IP address which originates a packet, and IP address to which a packet is sent, respectively, and they can be specified by either a specific IP address (ie. **host** *IP-address*) or a range of IP addresses (ie. *IP-address Netmask*).
- *protocol* stands for a session-layer protocol, eg. *tcp*.
- *Op* stands for a binary operator such as = and ≥.
- *Source-port* and *Dest-port* mean a port at *source-address* and a port at *dest-address*, respectively.

b) Activation rules : One activation rule is defined for an interface, and it can activate only one access-list number for each direction (ie. **in** or **out**). An activation rule is defined by the following:

interface *Name*
ip address *interface-address*
ip access-group *access-list-number* { **in** | **out** }

where *interface-address* stands for a specific IP address that is assigned to the interface *Name*.

Example 1 *Filtering and Activation rules*
The following shows an example of activation rules.

```
interface Ethernet 0/0
description Router A
ip address 202.44.8.1
ip access-group 100 in
ip access-group 101 out
```

The following shows an example of filtering rules.

```
access-list 100 deny tcp host 34.1.2.2
                         host 202.44.9.2 eq 80
access-list 100 permit tcp 34.1.0.0 255.255.0.0
                           host 202.44.9.2 eq 80
access-list 100 deny ip host 203.155.33.1 any
access-list 100 permit ip any any
access-list 101 deny ip any host 203.155.33.1
access-list 101 deny ip any host 202.144.255.1
access-list 101 permit ip any any
```

3. Model

Our model is based on graph theory (eg. [3]). In particular, since network topology can be represented by a graph, we argue that a firewall configuration rule can be understood as a set of paths in the graph. By treating firewall configuration rules as paths, reasoning about effects of those rules can be achieved easily.

We propose the following generalized firewall rule which can be considered as a general representation of firewall configurations. Such generalized firewall rules would facilitate our task here.

Definition 2 *Generalized Firewall Rules*

A generalized firewall rule for Cisco consists of the following:
(Source, Destination, Service, Direction, Action, FW Interfaces)
where

- *Source* and *Destination* stand for sender's IP address and receiver's IP address, respectively. Note that *any* means any IP address.
- *Service* consists of a protocol and ports.
- *Direction* stands for the direction of flow from *Source* to *Destination* via *FW Interfaces*, and it can be either *inbound* or *outbound.*
- *Action* stands for whether flow is allowed or not, ie.(*permit* or *deny*).
- *FW Interfaces* stand for firewall interfaces which perform packet filtering.

For each activation rule *AR* in *SAR*, we build a set of generalized firewall rules in the same order as the filtering rules activated by the rule *AR. Thus, only one set of generalized firewall rules is defined for an activating firewall object.* Moreover, the order of filtering rules in simplified Cisco rules is preserved in the set of generalized firewall rules.

Definition 3 *Converting from Cisco rules to generalized firewall rules*
For each activation rule $AR \in SAR$ and for each filtering rule $FR \in SFR$ which is activated by *AR*, we can obtain the corresponding generalized firewall rule as follows:
(Source, Destination, Service, Direction, Action, FW Interfaces)
where *Source*, *Destination*, *Service* and *Action* can be obtained from corresponding items in *FR*, and *Direction and FW Interfaces* can be obtained from corresponding items in *AR*, which activates *FR.*

Example 2 *Converting from Cisco rules to generalized firewall rules*
The first four filtering rules in example 1 can be converted to the following generalized firewall rules.
#a: (34.1.2.2, 202.44.9.2, http, inbound, deny, 202.44.8.1)
#b: (34.1.0.0 255.255.0.0, 202.44.9.2, http, inbound, permit, 202.44.8.1)
#c: (203.155.33.1, any, any, inbound, deny, 202.44.8.1)
#d: (any, any, any, inbound, permit, 202.44.8.1)

The following shows the definition of network topology.

Definition 4 *The Network Topology for Cisco*
The network topology for Cisco is a *labeled* and *undirected* graph where a vertex (node) in the network topology is labeled with a set of IP addresses, and an edge (arc) between two vertices represents a communication link between two sets of IP addresses.

For an internal network, a vertex is labeled with a set of a *single* IP address. Such a vertex intuitively stands for an interface to either a computer or a network device. For the external network, there is a special vertex called *all* representing a set of all valid IP addresses. The *all* vertex is labeled with set $\{x \mid x$ is a valid IP address$\}$. From now on, we use *all* to denote the set.

Intuitively, a vertex labeled with a set of IP addresses means that the vertex *can represent* any IP addresses in the set.

The figure 1 illustrates an example of the network topology. In figure 1, there is a symbol next to each IP address. For simplicity of our discussion here, such symbol is used to represent such IP addresses, and a set consisting of a single IP address is represented by the single IP address. For example, *x200* means IP *203.155.33.1*. There is a requirement for the network topology. All firewall interfaces appearing in generalized firewall rules must be present as vertices in the network topology.

The following shows that Cisco firewall rule can be considered as a set of *directed* and *non-cyclic* paths defined at a particular interface in a network topology. Since such a set of directed paths is defined at a particular interface, it is considered as a *local* set.

Definition 5 *A Cisco firewall rule as a local set of directed and non-cyclic paths*
Given a generalized firewall rule *GR* (for Cisco) and a network topology *T*, a *local* set of directed paths which corresponds to *GR* is a set of all *directed* and *non-cyclic* paths in *T* that

a) begin with *Source* vertex,
b) end at *Destination* vertex,
c) pass through *FW Interfaces* vertex in specified *Direction*, and
d) are labeled with *Service*,

where *Source, Destination, Service, Direction, FW Interfaces* are those stated in *GR.*

Definition 6
A path that begins with *Source* vertex in rule *GR* is the path that satisfies the following:

a) if *Source* is a specific IP address (*ip*), then $SIP = \{ip\}$, where *SIP* is a set of IP addresses represented by the initial vertex of the path, or
b) if *Source* is a set (*gip*) of IP addresses, then $SIP = \{ip'\}$, where $ip' \in gip$.
c) if *Source* is "*any*", then *SIP* can be of any set.

Note that *SIP* can be understood as an *instantiated* label of the vertex for the path by generalized firewall rule *GR*. Note also that the definition for a path that ends at *Destination* vertex can be given similarly to this definition.

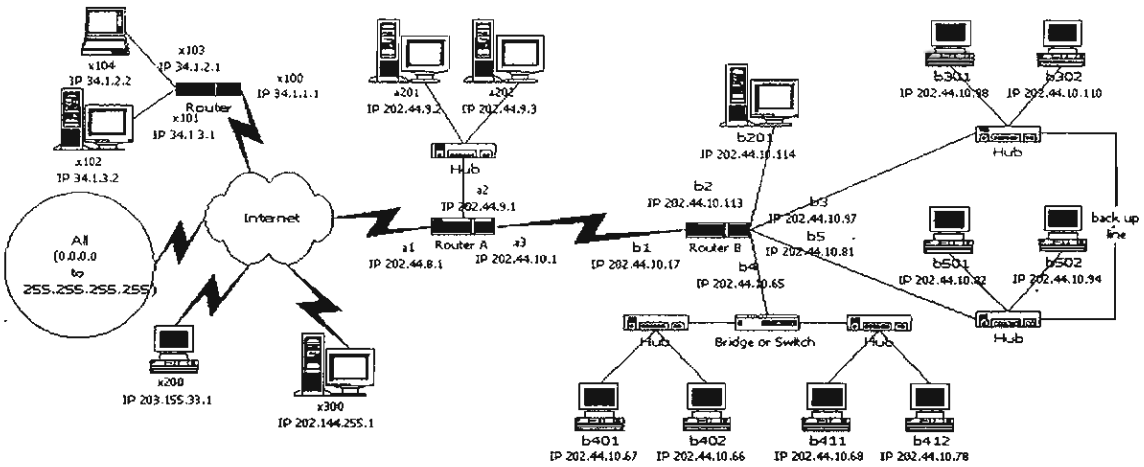

Figure 1. The network topology

Example 3 *A Cisco firewall rule as a local set of directed and non-cyclic paths*
The local set of paths that corresponds to the generalized firewall rule *#a* in example 2, ie. *(x104, a201, http, inbound, deny, a1)*, is *{<x104, x103, x100, a1, a2, a201>}*, whereas the local set of paths that corresponds to rule *#b* in example 2, ie. *({x100..x104}, a201, http, inbound, permit, a1)*, is *{<x100, a1, a2, a201>, <x101, x100, a1, a2, a201>, <x102, x101, x100, a1, a2, a201>, <x103, x100, a1, a2, a201>, <x104, x103, x100, a1, a2, a201>}*.

Note that *<n1, n2, n3>* represents a path beginning at vertex representing *n1* and ending at vertex representing *n3*, and such path visits vertex that represents *n2*.

It should be noted here that the transformation of a firewall rule to a set of paths requires the presence of firewall objects in the network topology. Such requirement ensures that such directed paths obtained correspond to actual paths that packets can travel.

Since there may be many rules activating at an interface, we need to compute effects of those rules at such an interface. This is dealt with by the concept of *effective* paths. Note that we shall focus on the effective paths which are permitted at an interface, rather than those which are prohibited at an interface. However, similar definition for prohibited paths can be given also.

The following shows a definition to compute locally a set of *effective* and *permitted* paths from a set of generalized firewall rules defined at an interface.

Definition 7 *A local set of effective and permitted paths*
Given an ordered set *SGR* of generalized firewall rules at an interface, a local set of *effective* and *permitted* paths at the interface is P_n where

- n is the number of rules in *SGR*,
- GR_j for j such that $1 \le j$ and $j \le n$ is the j-th generalized firewall rule in the ordered set *SGR*,
- LP_j is a local set of directed and labeled paths generated from the generalized firewall rule GR_j,
- P_0 is simply $\varnothing$, and
- For each i such that $0 \le i$ and $i < n$,
  if GR_{n-i} is a "*deny*" rule, then $P_{i+1} = P_i - LP_{n-i}$
  if GR_{n-i} is a "*permit*" rule, $P_{i+1} = P_i \cup LP_{n-i}$

We assume that the resultant set P_n is minimal, since the set may result from the set subtraction that could remove some paths out.

Example 4 *A local set of effective and permitted paths*
Suppose that at interface *a1*, there are only rules *#a* and *#b* shown in example 2. Thus, the local set of effective and permitted paths at interface *a1* is *{<x100, a1, a2, a201>, <x101, x100, a1, a2, a201>, <x102, x101, x100, a1, a2, a201>, <x103, x100, a1, a2, a201>}*.

We argue that the process of calculating effective paths is similar to the process of evaluating nested expressions. Note that nested expressions can be seen as ordered expressions.

Intuitively, this definition can be seen as a transformation from an ordered set of firewall rules to nested expressions of set operations on paths. In particular, the top most firewall rule would become some sub-expression at the out most level. Moreover, the bottom most firewall rule would become some sub-expression at the inner most level.

Example 5 *Order of rules as order of expressions*
Consider the firewall rules in example 2. Suppose that there are only those rules. The definition of finding effective paths would transform the ordered set *SGR* of those rules into nested expression $(((\varnothing \cup LP_4) - LP_3) \cup$

LP_3) – LP_1 where LP_i is the local set of paths that corresponds to the i-th rule in SGR.

The concept that an upper rule takes precedence over a lower rule is still preserved in nested expressions in that an outer sub-expression could cancel the effect of an inner sub-expression.

Moreover, the set operations on paths can be simplified to corresponding set operations on vertices in those paths.

Example 6
Suppose $P1 = \{ <all, a1, a3, b1, b2, b201> \}$, $P2 = \{<x100, a1, a3, b1, b2, b201>\}$ and $P3 = \{ <x300, a1, a3, b1, b2, b201> \}$. Thus $(P1 \cup P2) - P3 = \{ <all - \{x300\}, a1, a3, b1, b2, b201> \}$.

The following shows the definition to compute a global set of effective and permitted paths from every local set of effective and permitted paths. Intuitively, a global set of effective and permitted paths stands for directed paths that are permitted by all Cisco firewalls.

Definition 8 *A global set of effective and permitted paths*
For every local set LEP of effective and permitted paths and for every path $P \in LEP$, if

$$P \in \bigcap_{i \in fw\text{-}interface(p)} LEP_i$$

then P is in the *global* set of effective and permitted paths, where LEP_i is a local set of effective and permitted paths at interface i, and *fw-interface(P)* indicates a set of vertices in path P, which are interfaces to firewalls.

Intuitively, this definition states that a path in a local set of effective paths would be in the global set of effective paths if the path is permitted at every interface through which the path visits.

Example 7 *A global set of effective and permitted paths*
Suppose that the local set of effective and permitted paths at interface $a2$ is $\{<x100, a1, a2, a201>, <x101, x100, a1, a2, a201>, <x102, x101, x100, a1, a2, a201>, <x103, x100, a1, a2, a201>, <b201, b2, b1, a3, a2, a201>, <b301, b3, b1, a3, a2, a201>, <b301, b5, b1, a3, a2, a201>, <b501, b5, b1, a3, a2, a201>, <b501, b3, b1, a3, a2, a201>, <b401, b4, b1, a3, a2, a201>, <b402, b4, b1, a3, a2, a201>, <b411, b4, b1, a3, a2, a201>, <b412, b4, b1, a3, a2, a201>\}$. Moreover, suppose that there are only two local sets of effective and permitted paths (at interfaces $a1$ and $a2$). Thus, the global set of effective and permitted paths is $\{<x100, a1, a2, a201>, <x101, x100, a1, a2, a201>, <x102, x101, x100, a1, a2, a201>, <x103, x100, a1, a2, a201>\}$.

4. Reasoning methodology

Our methodology provides an analysis of effects of firewall configurations in such a way that given a network topology and Cisco firewall rules, all the effects of those rules are produced in terms of paths. Once those paths which are effects of firewall rules are obtained, many kinds of analysis can be further performed, which will be discussed below.

An information flow can be represented by either a path or a triple (source, destination, service). Such a triple represents a kind of information flow that is independent of paths. For simplicity here, we shall focus on the information flow represented by paths.

One simple kind of reasoning offered in our model is to allow administrators to test whether an information flow is permitted by a set of generalized firewall rules. This can be done by testing the membership of such information flow in the global set of effective and permitted paths.

4.1. Correctness of firewall rules

This kind of reasoning aims to test that effects of a set of firewall rules are those that are intended by a firewall administrator. Initially, a firewall administrator has to define a set of intended information flow, and then calculated effects of firewall rules will be compared with this set of intended information flows.

Definition 9 *Correctness*
Given a set IF of paths that are intended as *permitted* information flow, a set SGR of generalized firewall rules is *correct* iff $IF = GEP$ where GEP is the global set of effective and permitted paths generated from SGR.

Note that this definition of correctness is defined for the context of closed policy, since the closed policy expresses only what is permitted and assumes that what is not permitted must be denied.

4.2. Redundancy and inconsistency of rules

This kind of reasoning allows firewall administrators to examine whether a set of firewall rules contain some rules that produce no effects. If there is such rule, it can be eliminated.

Definition 10 *Redundancy (Inconsistency)*
Firewall rules $R1$ and $R2$ are *redundant* (*inconsistent*) iff
a) $R1$ and $R2$ are defined for the same kind of actions, services and directions ($R1$ and $R2$ are defined for the same kinds of *services* and *directions* but they are of different kinds of *action*, respectively) and
b) either $LP1 \subseteq LP2$ or $LP2 \subseteq LP1$
where $LP1$ and $LP2$ stand for the local sets of paths generated from rules $R1$ and $R2$, respectively.

Example 8 *Redundancy and Inconsistency*
Consider rules in example 2. Suppose that those rules are reordered into *#b, #a, #c* and *#d*. Thus, rule *#b* and rule *#d* are redundant, and rule *#b* and *#a* are inconsistent.

Definition 11 *Redundantly(inconsistently)ineffective rules*
Given a set *SGR* of generalized firewall rules at an interface, rule *GR* in *SGR* is *redundantly (inconsistently) ineffective* if and only if there is *GR'* in *SGR* such that
a) *GR* and *GR'* are *redundant (inconsistent, resp.)* and
b) GEP = GEP'
where *GEP* and *GEP'* are the global sets of *effective* and permitted paths generated from *SGR*, and (*SGR – GR*), respectively.

Example 9 *Ineffective rules*
Consider the rules discussed in example 8. Rule *#a* is inconsistently ineffective, but rule *#b* and rule *#d* are not redundantly ineffective. Thus, rule *#a* can be removed.

4.3. Explanation of flows in terms of firewall rules

Our model allows us to reason about firewall rules that are the causes for either allowing or disallowing particular flows. For simplicity, we shall consider the kinds of explanations for allowing particular flows only.

Definition 12 *An explanation of a flow in terms of firewall rules*
Given a set *SGR* of generalized firewall rules and a path $P \in GEP$, an explanation for allowing path *P* is the set
$\{GR \mid GR \in SGR,\ GR \text{ is a permit-type rule},\ GR \text{ is not ineffective and } P \in LP_{GR}\}$
where LP_{GR} is a local set of paths generated from rule *GR*, and *GEP* stands for the global set of effective and permitted paths.

Intuitively, rule *GR* is an explanation for path *P* if the effect of rule *GR* includes path *P*.

5. Conformance testing

The conformance testing [4] aims to ensure that a model is both *correct* (*sound*) and *complete* with respect to an actual implementation. The correctness means that a model correctly captures an implementation whereas the completeness means that a model captures all aspects of an implementation.

We have carried out conformance testing by creating test data which cover main functionality and behavior of firewall, and comparing the outputs obtained from our model with those from Cisco firewall.

The results obtained show that our model is *correct* with respect to Cisco firewalls. However, we *cannot* ensure the completeness. This is because it is impossible to construct test data which cover all possibilities of the functionality and the behavior of firewalls.

6. Related works

The most related work to ours is Fang [5]. Fang is a software tool which aims to analyze firewall configurations. Fang's approach is to simulate the final effects of a firewall configuration in terms of a set of packets allowed to pass through firewalls. A user must submit a query on a particular set of packets to Fang, and based on the query, Fang will generate the output consisting of packets in the query that are allowed to pass through firewalls.

Fang's approach is *ad hoc* in that it can only produce the final effects of a firewall configuration. It does not provide an explanation of the processing of ordered rules activated at an interface. Moreover, it does not offer an explanation of the processing of filtering rules at all interfaces of every firewall. In addition, it cannot reason about the redundancy and inconsistency of filtering rules. Fang cannot reason about an explanation of a flow in terms of filtering rules.

7. Conclusion

We have presented a preliminary but novel model for firewalls and its methodology. The model is based on graph theory. Our model provides simple and precise understanding of firewall configurations. Also our methodology offers a thorough analysis of firewall configurations

We have applied our methodology to case studies [6]. Currently, we are implementing a software prototype of our model. As a future work, we aim to apply our model to other kinds of firewalls, ie. software-based firewalls.

Acknowledgement

The first author would like to acknowledge supports from The Thailand Research Fund and The National Research Council of Thailand.

References

[1] W. Cheswick and S.Bellovin, *Firewalls and Internet Security : Repelling the Wily Hacker*, Addison-Wesley, 1994.
[2] D.B. Chapman and E.D. Zwicky, *Building Internet Firewall*, O'Reilly & Associates, 1995.
[3] J. Gross and J. Yellen, *Graph Theory and its Applications*, CRC Press LLC, 1998.
[4] G.J. Holzmann, *Design and Validation of Computer Protocols*, Prentice Hall Software Series, 1991.

[5] A. Mayer, A. Wool and E. Ziskind, Fang : A Firewall Analysis Engine, *In proceedings of 21st IEEE Symposium on Security & Privacy*, Oakland, CA, 2000.

[6] C. Rujimethabhas. *A graph-based methodology for Hardware-based Firewalls*, Master thesis, Department of Computer Engineering, King Mongkut's University of Technology Thonburi, Bangkok, Thailand, 2001.

A Unified Methodology for Verification and Synthesis of Firewall Configurations

Yongyuth Permpoontanalarp and Chaiwat Rujimethabhas

Logic and Security Laboratory
Department of Computer Engineering
King Mongkut's University of Technology Thonburi
91 Suksawasd 48, Ratburana, Bangkok 10140 Thailand
yongyuth@cpe.eng.kmutt.ac.th and s1410008@cc.kmutt.ac.th

Abstract. Firewalls offer a protection for private networks against external attacks. However, configuring firewalls correctly is a difficult task. There are two main reasons. One is that the effects of a firewall configuration cannot be easily seen during the configuration time. Another one is the lack of guidance to help configuring firewalls. In this paper, we propose a general and unified methodology for the verification and the synthesis of firewall configurations. Our verification methodology offers a way to foresee and analyze effects of firewall configurations during the configuration time. Furthermore, our synthesis methodology can generate firewall configurations that satisfies users' requirements. As a result, firewall configurations that are free of many kinds of errors and loopholes can be obtained easily.

1 Introduction

Nowadays, firewalls (e.g.[1, 2]) become a widely used mechanism to achieve Internet security. Most, if not all, organizations whose computers have an Internet access are currently using firewalls. Firewalls locate between an internal network and an external network. Firewalls offer a protection for private (and internal) networks against external threats. In particular, firewalls ensure that only authorized information flows between internal networks and the external network are allowed.

Even though firewalls could provide protections against external attacks, configuring firewalls correctly is a difficult task. There are two main reasons. One is that the effects of a firewall configuration cannot be easily seen during the configuration time. Another one is the lack of guidance to help configuring firewalls.

Since the effects of a firewall configuration cannot be seen at the configuration time, many firewall configurations often have errors and loopholes. Most often, such errors and loopholes are discovered only after they actually happen at the execution time. This causes great damage to the system.

Due to the lack of guidance to help configuring firewalls, to configure them requires a great deal of experience which is certainly not available to novice

administrators. Moreover, configuring firewalls can be a complex and time- consuming task due to the large number of host computers, required services and firewalls. Furthermore, the networks of computers in any organizations are always changed due to the change of the structure of organizations themselves and the replacement for new equipment. Most often a change of such networks requires a new firewall configuration. Thus, this worsens the situation.

We argue that all these problems occur because of the lack of firewall methodology to analyze the effects of firewall configurations, and to help configuring firewalls. In this paper, we propose a general and unified methodology for verifying and synthesizing firewall configurations.

In [3], we proposed a graph-based model and its methodology to analyze effects of Cisco firewall configurations. In this paper, we extend the model and the methodology there in several aspects. Firstly, we extend the model to be able to deal with both the verification and the synthesis within the same framework. Secondly, we define the notion of correctness of firewall configurations in the context of several kinds of policies whereas [3] deals with one kind of policy only. Those polices are useful not only for the verification but also for the synthesis. Furthermore, we show here that our model is general in that it can be used to analyze effects of Firewall-1 configurations also, not just Cisco firewalls.

Our approach is novel in that it is formal and it combines both verification and synthesis within the same framework. We show that our approach is more general than existing related approaches. Furthermore, we obtain the correctness justification and proof for the verification and the synthesis, respectively.

We discuss the background in section 2, and present our model in section 3. Our verification and synthesis methodology is discussed in section 4 and 5, respectively. The correctness of our methodology is discussed in section 6, related works are discussed in section 7 and conclusion is given in section 8.

2 Background

2.1 Firewall-1 Firewalls

Firewall-1 firewall is a software-based firewall since it is a computer installed software for filtering packets.

Definition 1. *Firewall-1 rules are represented by tuple (SFR, SPR) where SFR stands for a set of filtering rules and SPR stands for a set of firewall property rules.*

Definition 2. *A firewall-1 filtering rule consists of the following : (Source, Destination, Service, Action, Activating FW objects) where*
- Source and Destination stand for senders' IP addresses and receivers' IP addresses, respectively,
- Service consists of a protocol and a port,
- Action stands for whether flow from Source to Destination is allowed or not, i.e.(permit or drop), and

- Activating FW objects stand for names representing firewall objects which perform the filtering of the flow of packets.

Intuitively, a filtering rule defines the permission or prohibition of flows from *Source* to *Destination* via *Activating FW Objects* for *Service*. Note that *any* can be used to specify sources and/or destinations and it means any IP address.

A Firewall-1 property rule defines not only a set of interfaces of a firewall object, but also the direction of the filtering of the flow of packets at all interfaces of the firewall object. Note that the direction is defined for *all interfaces* of a firewall. Moreover, *all interfaces* of a Firewall-1 firewall object enforce the same set of filtering rules.

Definition 3. *A firewall property rule for firewall-1 is defined by (FW object, Interfaces, Direction), where*
- FW object stands for a name representing a firewall object,
- Interfaces stands for a set of all interfaces of FW object, and
- Direction stands for the direction of packet filtering, i.e. (in or out).

3 Our Model

Our model is based on graph theory (e.g. [4]). In particular, network topology can be represented by a graph. Then, we argue that a firewall configuration rule can be understood as a set of paths in the graph. By treating firewall configuration rules as paths, we can reason about the verification and the synthesis of firewall rules intuitively and easily.

First, we propose a general form of firewall rules, called generalized firewall rules as a representation of firewall rules which will be used for the verification and the synthesis. We shall show that Firewall-1 rules can be converted into our generalized firewall rules. The following shows the definition of generalized firewall rules.

Definition 4. *A generalized firewall rule consists of the following: (Source, Destination, Service, Direction, Action, FW Interfaces) where*
- Source, Destination, Service, Action are identical to those defined for Firewall-1 rules,
- Direction means similarly to that in Firewall-1 rule, but it can be inbound, outbound or both (bound), and
- FW Interfaces stand for a set of firewall interfaces which perform (or activate) the filtering of the flow.

Note that the direction *both* means that packet filtering is performed in both directions at specified firewall interfaces. In particular, *both* is defined by both *inbound* and *outbound*. Such a rule with *both* directions is useful for the synthesis of firewall rules in the context of either closed or open policies which will be discussed later.

The conversion from Firewall-1 rules to our generalized firewall rules is straightforward, and is shown by the following definition. The resultant generalized firewall rules preserve the order of filtering rules in firewall-1 rules.

Definition 5. *Given tuple (SFR, SPR) of firewall-1 rules, for each filtering rule FR ∈ SFR and for each firewall property rule PR ∈ SPR of which its FW object appears in Activating FW objects of FR, we can obtain the corresponding generalized firewall rule GR in that*
- Source, Destination, Service and Action in GR are Source, Destination, Service and Action in FR, respectively, and
- Direction and FW Interfaces in GR are Direction and Interfaces in PR, respectively.

The following shows the definition of logical network topology. Our logical network topology can capture the ability of sending packets between two parties in the physical network topology.

Definition 6. *The network topology is a labeled and undirected graph (V,E) where a vertex in V stands for a set of IP addresses, and an edge between two vertices stands for a communication link between two sets of IP addresses.*

Indeed, the set V of vertices is defined by the power set of the set of all valid IP addresses. Hence, a vertex is represented by the set of IP addresses that the vertex stands for. An edge between two vertices means that an IP address in the former vertex can send a packet (or information) to another IP address in the latter vertex.

For the internal network, a vertex is represented by a set of a *single* IP address. Such a vertex intuitively stands for an interface to either a computer or a network device. For the external network, there is a special vertex called *all* standing for a set of all valid IP addresses. In particular, $all = \{ x \mid x \text{ is a valid IP address} \}$.

It is required that all firewall interfaces appearing in generalized firewall rules must be present as vertices in the network topology.

Example 1. Physical Network Topology

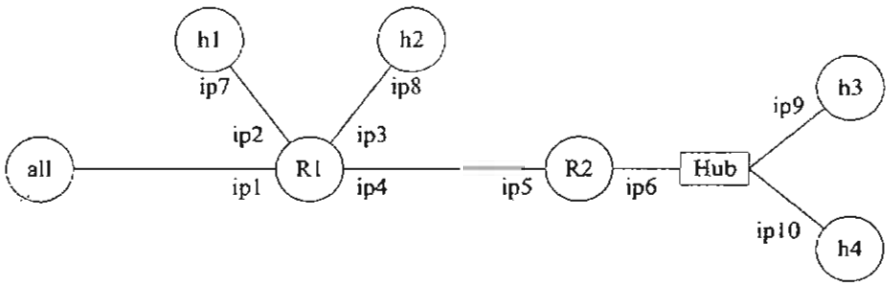

Fig. 1. Network Topology

In figure 1, *R1* and *R2* represent two firewall objects. *h1*, *h2*, *h3* and *h4* stand for host computers. *ip1*, *ip2*, ..., *ip10* denote IP addresses. In particular, *ip1*, *ip2*,

ip3, ip4 are IP addresses of four firewall interfaces of firewall object *R1*. *ip7, ip8, ip9 and ip10* are IP addresses of interfaces of host computers *h1, h2, h3* and *h4*, respectively. It is easy to represent this topology using our definition of logical network topology. This network topology will be used as examples throughout this paper.

It should be noted that the connections between all firewall interfaces (e.g. *ip1, ip2, ip3 and ip4*) of firewall object *R1* are determined by a routing table. For convenience here, we assume that such information is present. Any paths passing through those firewall interfaces can be determined, and they visit only necessary firewall interfaces that perform the actual packet filtering.

Since an edge between two vertices represents an ability of sending a packet, the edge is closed under the membership of its vertices. This is expressed by the following definition.

Definition 7. *The logical network topology has the following properties:*
1) $\forall u,v,v1,v2 \in V\ [\ v1 \subset v \wedge v2 \subset v \wedge v1 \cap v2 = \emptyset \wedge v1 \neq \emptyset \wedge v2 \neq \emptyset \rightarrow$
$\exists e \in E\ (f(e) = \{u, v\}) \leftrightarrow \exists e1,e2 \in E\ (\ f(e1) = \{u, v1\} \wedge f(e2) = \{u, v2\}\ )\ ]$
where f is a function mapping from an edge to its vertices.
2) $\neg\exists e \in E\ [\ f(e) = \{u, v\}$ *and* $u = v\ ]$.

Definition 8. *We use a special name* ω *to represent all possible non-cyclic paths labeled with all possible services in the network topology.* ω *is treated as a path itself.*

The following shows properties of ω paths.

Definition 9. *The following are properties of* ω.
1) $P \cup \{P1\} = P \cup \{\omega\}$ *iff* $[initial(P1) = initial(\omega) \wedge terminal(P1) = terminal(\omega)]$
where P and P1 stands for a set of paths and a path, respectively, and initial(P1) and terminal(P1) denote initial and terminal vertex of P1, respectively.
2) $P \cup P' = P \cup \{P1\}$ *if*
2.1) $[\ initial(P1) = initial(\omega) \wedge terminal(P1) = v \wedge$
for all paths P2 in T ($(terminal(P2) = v) \leftrightarrow (P2 \in P')$ *)* $]$
where T, v and P' stand for a network topology, a vertex, and a set of paths, respectively.
2.2) $[\ initial(P1) = v \wedge terminal(P1) = terminal(\omega) \wedge$
for all paths P2 in T ($(initial(P2) = v) \leftrightarrow (P2 \in P')$ *)* $]$
2.3) $[\ P1 = \omega \wedge$ *for all paths P2 (* $P2$ *in* $T \leftrightarrow P2 \in P'$ *)* $]$

Note that property *1)* defines the initial and terminal vertices of the ω path. Property *2)* however defines the meaning of the ω path. For example, *2.1)* states that a path of which its initial is $initial(\omega)$, but its terminal is an ordinary vertex v is equivalent to all paths in the network topology ending at v.

We argue that a generalized firewall rule can be considered as a set of (either permitted or prohibited) paths in a network topology, and such paths are defined at a particular interface. Since such a set of paths is defined at a particular interface, it is called a *local* set of paths. The following definition shows the equivalence between a generalized firewall rule and a local set of paths.

Definition 10. *Given a network topology T, a local set of paths which corresponds to a generalized firewall rule $GR_{FW\,Interfaces}$ defined for interface FW Interfaces is a set of all non-cyclic paths in T which*
a) begin with Source vertex,
b) end at Destination vertex,
c) pass through FW Interfaces vertices in the specified Direction, and
d) are labeled with Service,
where Source, Destination, Service, Direction, FW Interfaces are those stated in $GR_{FW\,Interfaces}$.

The following defines precisely paths that begin with *Source* vertex, end at *Destination* vertex, and pass through *FW Interfaces* vertices in the specified *Direction*.

Definition 11. *A path that begins with Source (S) vertex, ends at Destination (D) vertex, and passes through FW Interfaces vertices in the specified Direction is the path P that satisfies the following:*
a) S (D) is a specific IP address (ip) iff initial(P) = ip (terminal(P) = ip, resp.).
b) S (D) is a set (gip) of IP addresses iff
$initial(P) = sip$ *(terminal(P) = sip, resp.), where* $sip \subseteq gip$.
c) S (D) is "any" iff initial(P) = initial(ω) (terminal(P) = terminal(ω), resp.).
d) Direction = inbound iff $\exists F \in FW\_Objects\ \exists i.\ 1 < i \leq n$
$[\ vertex_i(P) \subseteq FW\ Interfaces \wedge FW\ Interfaces \subseteq Interfaces\text{-}of(F) \wedge$
$vertex_{(i-1)}(P) \not\subseteq Interfaces\text{-}of(F)\ ]$
where n is the length of path P, and $vertex_i(P1)$ *means i-th vertex in path P1.*
e) Direction = both iff [initial(P) = initial(ω) ∧ terminal(P) = terminal(ω) ∧
$\exists F \in FW\_Objects\ (\ FW\ Interfaces \subseteq Interfaces\text{-}of(F)\ )\ ]$

Intuitively, a path that satisfies *d)* must be a path that travels to the designated *FW Interfaces* of a firewall from a vertex which is not in *FW Interfaces* of the same firewall. The definition of the *outbound* direction is omitted here due to space limit, but it is similar to that for the *inbound* direction.

As a matter of notations, we use a single IP address to represent the set of the single IP address.

Example 2. The local set of paths that corresponds to rule *(ip15, ip9, ftp, inbound, permit, ip1)* is {<*ip15, ip1, ip4, ip5, ip6, ip9*>}. Note that this path exists due to definition 7. Moreover, the local set of paths that corresponds to rule *({ip9, ip10}, any, http, inbound, permit, ip6)* is {<*ip9, ip6, terminal(ω)*>, <*ip10, ip6, terminal(ω)*>, <{*ip9, ip10*}, *ip6, terminal(ω)*> }. In addition, the local set of paths that corresponds to rule *(any, any, http, inbound, permit, ip6)* is {<*initial(ω), ip6, terminal(ω)*>}.

Definition 12. *Paths that correspond to a generalized firewall rule obtained from Firewall-1 firewall rules are undirected.*

Note that the *undirected* paths of Firewall-1 mean two-way communications between *Source* and *Destination* parties, initiated by *Source* and responded by *Destination*.

The set operations on paths can be simplified to corresponding set operations on vertices in those paths. Such simplification is useful for the verification of firewall rules.

Definition 13. *Path expressions can be reduced to vertex expressions as follows:*
1) P1 op P2 = P3 if
a) paths P1, P2 and P3 are of the same length (i.e. n).
b) there is at most one vertex i such that $vertex_i(P1) \neq vertex_i(P2)$ *and* $\forall j \neq i.\ 1 \leq j \leq n\ [\ vertex_j(P1) = vertex_j(P2)\ ]$
c) path P3 is exactly like path P1 (or P2), except that $vertex_i(P3) = vertex_i(P1)$ ***op*** $vertex_i(P2)$*, where* ***op*** *is a set operation (e.g.* $\cup$ *or -).*
2) $P \cup \{P1\} = P$ *if* $\exists i\ [vertex_i(P1) = \emptyset\ ]$*, where P is a set of paths, and P1 is a path.*

Example 3. Suppose that path $P1 = \{<all, ip1, ip4, ip5, ip6, ip9>\}$, path $P2 = \{<ip17, ip1, ip4, ip5, ip6, ip9>\}$ and path $P3 = \{<ip13, ip1, ip4, ip5, ip6, ip9>\}$. Thus $(P1 \cup P2) - P3 = \{<all - \{ip13\}, ip1, ip4, ip5, ip6, ip9>\ \}$.

Since there may be many rules activating at an interface, we need to process those rules in order to obtain actual effects of those rules at the interface. A set of effective paths at an interface is used to represent the actual effects at the interface, and it is defined as follows.

Definition 14. *A local set of effective and permitted (prohibited) paths at interface fw-i for which an ordered set* SGR_{fw-i} *of generalized firewall rules is specified, is defined by* P_n *that satisfies the following two conditions:*
a) $\forall i.\ 0 \leq i \leq n$ *[rule* GR^{n-i}_{fw-i} *has* LP_{n-i} *as the local set of paths*
$\rightarrow$ *(* $P_{i+1} = P_i - LP_{n-i} \leftrightarrow GR^{n-i}_{fw-i}$ *is a drop (permit, resp.) rule)* $\vee$
($P_{i+1} = P_i \cup LP_{n-i} \leftrightarrow GR^{n-i}_{fw-i}$ *is a permit (drop, resp.) rule)]*
b) P_n *is minimal in that there is no* P'_n *that satisfies a) and* $P'_n \subset P_n$*.*
where
- n *is the number of rules in* SGR_{fw-i}*,*
- GR^j_{fw-i} *is the* j*-th generalized firewall rule in* SGR_{fw-i}*, defined at fw-i,*
- P_0 *is simply* $\emptyset$*.*

Note that in condition *b)*, P'_n that satisfies condition *a)* means that such P'_n is obtained from the same set expressions as P_n, according to condition *a)*.

There might be *several possible sets* of paths that satisfy *a)* in this definition. By requiring the minimality of the resultant set of effective paths, we can ensure that a set of effective paths must be the one that consists of paths which have been simplified from path expressions to vertex expressions by definition 13 as much as possible. It should be noted that if the subtraction on sets is evaluated correctly, the resultant set would be smaller than that which would have been obtained incorrectly.

Example 4. The followings are examples of local sets of effective paths:
a) Suppose (ordered set) SGR_{ip6} = {*(ip9, any, http, inbound, drop, ip6), ({ip9, ip10}, any, http, inbound, permit, ip6)*}. The local set of effective and permitted paths at interface *ip6* is {<*ip10, ip6, terminal*(ω)>}. Note that this set satisfies the minimal requirement in *b)*.
b) Suppose SGR_{ip6} = {*(ip9, any, http, inbound, drop, ip6), (any, any, http, inbound, permit, ip6)*}. The local set of effective and permitted paths is {<*initial*(ω)-*ip9, ip6, terminal*(ω)>}.

Intuitively, definition 14 can be seen as a transformation from an *ordered* set of firewall rules to *nested* expressions of set operations on paths. The concept that an upper rule takes precedence over a lower rule is still preserved in nested expressions. Further discussion on definition 14 can be found in [3].

The following shows the definition to compute a global set of effective and permitted paths from every local set of effective and permitted paths. Intuitively, a global set of effective paths stands for paths that are effective at all firewall interfaces through which the paths visit.

Definition 15. *The global set of effective and permitted (prohibited) paths is GEP that satisfies the following:*

$$\forall P[P \in GEP \leftrightarrow \bigcap_{i \in fw-interface(P)} LEP_i]$$

where
- LEP_i is a local set of effective and permitted (prohibited, resp.) paths at interface i, and
- fw-interface(P) denotes a set of vertices in path P, which are interfaces to some firewalls.

4 Verification of Firewall Configurations

Before we discuss the verification of firewall configurations, we need to discuss the concept of information flows, first.

Definition 16. *An information flow can be represented by either a path or a triple (source, destination, service).*

The triple represents *end-to-end* information flow which is regardless of paths between the two ends. For simplicity here, we shall focus on the information flow represented by paths, called *path-based* information flows.

4.1 Correctness of Firewall rules

This kind of reasoning aims to test that effects of a set of firewall rules are those that are intended by a firewall administrator. Initially, a firewall administrator must define a set of intended information flow, and then calculated effects of firewall rules will be compared with the set of intended information flows.

Definition 17. *An intended information flow is tuple (PIF, NIF) where PIF and NIF are two finite sets which represent positive and negative information flows, respectively, for each service.*

Intuitively, positive and negative information flows represent permitted and prohibited information flows, respectively. Moreover, an intended information flow can also be defined for a particular service.

Definition 18. *The intended information flow has the following properties:*
a) $PIF \cup NIF \neq \emptyset$
b) $PIF \cap NIF = \emptyset$

We argue that our definition for intended information flows is adequate for its purpose since it can easily capture the correctness of firewall configurations in the context of many kinds of policies. Intuitively, those kinds of policy offer different ways to characterize the global set of effective and *permitted* paths.

The following shows the definition of the correctness in the context of closed, open, openly neutral and closely neutral policies.

Definition 19. *Given an intended information flow (PIF, NIF), a set of generalized firewall rules is correct in the context of*
a) closed policy iff $PIF = GEP$
b) open policy iff $\neg\exists F \in NIF\ [\ F \in GEP\ ]$, *and*
$$\forall F \in AllFlows(T)\ [\ F \notin NIF \rightarrow F \in GEP\ ]$$
c) openly neutral policy iff $(PIF \subseteq GEP)$, *and* $\neg\exists F \in NIF\ [\ F \in GEP\ ]$
d) closely neutral policy iff $(GEP \subseteq PIF)$, *and* $\neg\exists F \in NIF\ [\ F \in GEP\ ]$

where GEP is the global set of effective and permitted paths generated from the generalized firewall rules, and AllFlows(T) denotes the set of all possible flows in network topology T.

Intuitively, the closed policy states that *PIF* is exactly the only set of flows globally permitted. On the other hand, the open policy states that what is not in *NIF* is globally permitted. The neutral policies however do not state what globally permitted flows (*GEP*) *exactly* consist of, but it requires that *NIF* must not be globally permitted. In particular, the openly neutral policy states that *GEP* can be any superset of *PIF*. On the other hand, the closely neutral policy states that *GEP* can be just any subset of *PIF*.

4.2 Ineffective Firewall Rules

Our model can reason about rules that produce no effects. We call those rules ineffective.

Definition 20. *Given a set SGR of generalized firewall rules at an interface, rule GR in SGR is ineffective if and only if* $GEP = GEP'$ *where GEP and GEP' are the global sets of effective and permitted paths generated from SGR, and (SGR - GR), respectively.*

Example 5. Rule *(ip9, ip7, http, outbound, permit, ip4)* is ineffective since all the paths from *ip9* to *ip7* visit *ip4* in the *inbound* direction according to the network topology.

5 Synthesis of Firewall rules

The synthesis methodology takes intended information flows as input, and produces an *ordered* set of generalized firewall rules that satisfies the intended information flows. In other words, such set of generalized firewall rules obtained is correct with respect to those intended information flows.

Before we discuss the synthesis methodology, we need to understand some notations.

Definition 21. *We use LEP_i^+ and LEP_i^- to represent local sets of effective and permitted (and prohibited, respectively) paths at interface i.*

Definition 22. *The local sets of effective and permitted (and prohibited) paths have the following properties:*
a) For any interface i, $LEP_i^+ \cup LEP_i^- = \{\omega\}$.
b) For any interface i, $LEP_i^+ \cap LEP_i^- = \emptyset$.

Proposition 1. *For any interface i, $LEP_i^+ = (\{\omega\}$ - $LEP_i^-)$*

It should be noted that an intended information flow (*PIF*, *NIF*) can be considered as two *global* sets of effective and permitted (and prohibited, respectively) paths. The following shows the definition for the synthesis of firewall configurations in the context of many policies.

Definition 23. *Given an intended information flow (PIF, NIF) defined for a particular service, the synthesis of a set of generalized firewall rules consists of the following two steps:*
1) Decompose two global sets of effective paths (PIF, NIF) into local sets of effective paths (PIF_i, NIF_i) at firewall interface i, by using definition 15.
2) Generate an ordered set SGR_j of generalized firewall rules defined for firewall interface j from local set (LS) of effective paths, obtained from 1), at the interface j by using definition 14, and the following:

2.1) For the closed policy, LS is $LEP_j^- = (\{\omega\}$ - $PIF_j)$.

2.2) For the open policy, LS is $LEP_j^+ = (\{\omega\}$ - $NIF_j)$.

2.3) For the openly neutral policy,

If $PIF_j \neq \emptyset$,

then LS is $LEP_j^+ = (PIF_j \cup PIF_j^{'})$ - NIF_j, where $PIF_j^{'} \subseteq \{\omega\}$.

else LS is $LEP_j^- = \{\omega\}$ - $PIF_j^{''}$, where $PIF_j^{''} \subseteq \{\omega\}$ - NIF_j.

2.4) For the closely neutral policy,

If $PIF_j \neq \emptyset$,

then LS is $LEP_j^+ = PIF_j^{'}$ - NIF_j where $PIF_j^{'} \subseteq PIF_j$ and $PIF_j^{'} \neq \emptyset$

else LS is $LEP_j^- = \{\omega\}$

Example 6. Suppose that $PIF = \{<ip7, ip2, ip4, ip5, ip6, ip9>\}$, and its service label is *http*. Suppose we want to generate a set of firewall rules that implement the closed policy. Thus, it follows from step 1) that $PIF = PIF_{ip2} = PIF_{ip4} = PIF_{ip5} = PIF_{ip6}$. Let consider only PIF_{ip6}. By 2.1), $LEP_{ip6} = \{\omega\} \cdot PIF_{ip6}$. Since this local set of effective paths is for prohibited paths, it follows from definition 14 that a rule defined for *ip6* that corresponds to PIF_{ip6} is *(ip7, ip9, http, outbound, permit, ip6)*. As a result, $SGR_{ip6} = \{$ *(ip7, ip9, http, outbound, permit, ip6), (any, any, http, both, drop, ip6)* $\}$.

Definition 24. *The following are required desirable properties of a set SGR of generalized firewall rules that is obtained from our synthesis methodology.*
1) Finiteness : SGR must be finite.
2) Effectiveness : SGR must be free of any ineffective rules.
3) Minimal : SGR must be minimal in that there is no other set SGR' of generalized firewall rules such that GEP = GEP' and $|SGR'| < |SGR|$ where GEP and GEP' are global sets of effective paths generated from SGR and SGR', resp.

6 The Correctness of the verification and synthesis

The conformance testing technique [5] is employed here to ensure that the model presented in this paper for verification is correct with respect to actual firewall products. Conformance testing is a general technique to ensure that a specification corresponds to an actual implementation.

Similar to the conformance testing done in [3] for Cisco routers, the conformance testing here is carried out by constructing test data and comparing the outputs obtained from our model with those from the actual Firewall-1 firewall. The result obtained shows that the model for verification presented here is correct with respect to the Firewall-1 firewall.

We prove the correctness of the synthesis in the following. Due to space limit, we omit the proof details here.

Theorem 1. *A set SGR of generalized firewall rules that is generated by our synthesis methodology produces the global set GEP of effective paths that satisfies the correctness property for each kind of policy.*

7 Related Works

Firmato [6] offers a synthesis methodology for firewall rules. It offers the use of role-based policy for specifying intended information flows. Such role-based policy is a high-level policy. However, Firmato deals only with the synthesis in the context of *closed policy*. Furthermore, it does not analyze any desirable properties of synthesized firewall rules at all. It would be interesting to incorporate the use of role-based policy to specify intended information flows into our framework.

Filtering postures [7] offers both verification and synthesis of firewall rules. However, the kind of firewall rules that are verified or synthesized is order-

insensitive, and thus those rules are very different from actual firewall rules. As a result, it does not offer any understanding on the effect of rule ordering, unlike our approach. Furthermore, the only verification offered by filtering postures is identical to the correctness in the context of *closely neutral policy* in our approach. It does not analyze about properties of synthesized firewall rules.

Fang [8] is a software tool which aims to verify firewall rules. The main verification that Fang offers is the generation of final effects of firewall rules. However, Fang's approach is *ad hoc* in that it simply simulate the final effects of firewall rules without giving any explanation of the effects of ordered rules.

8 Conclusion

We have presented a general and unified methodology for verifying and synthesizing firewall configurations. Our methodology has several benefits in that it can analyze the correctness of firewall configurations and also generate configurations in the context of many kinds of policies.

We have applied our verification methodology to case studies in [9]. We are currently implementing a software prototype of our model. Also, we are applying our synthesis methodology to case studies.

Acknowledgement

The first author would like to acknowledge supports from the Thailand Research Fund, and the National Research Council of Thailand.

References

1. Cheswick W.R. and Bellovin S.M., Firewalls and Internet Security : Repelling the Wily Hacker, Addison- Wesley, 1994.
2. Chapman D.B. and Zwicky E.D., Building Internet Firewall, O' Reilly & Associates, 1995.
3. Permpoontanalarp Y. and Rujimethabhas C., A Graph Theoretic Model for Hardware-based Firewalls, In proceedings of 9th IEEE International Conference on Networks (ICON), Thailand, 2001.
4. Gross J. and Yellen J., Graph Theory and its Applications, CRC Press LLC, 1998
5. Holzmann G.J., Design and Validation of Computer Protocols, Prentice Hall Software Series, 1991.
6. Bartal Y., Mayer A., Nissim K. and Wool A., Firmato : A Novel Firewall Management Toolkit, In proceedings of 20th IEEE Symposium on Security & Privacy, Oakland, CA, 1999.
7. Guttman J.D., Filtering Postures : Local Enforcement for Global Policies, In proceedings of 17th IEEE Symposium on Security & Privacy, Oakland, CA, 1997.
8. Mayer A., Wool A. and Ziskind E., Fang : A Firewall Analysis Engine, In proceedings of 21st IEEE Symposium on Security & Privacy, Oakland, CA, 2000.
9. Rujimethabhas C., A Graph-based Methodology for Hardware-based Firewalls, Master Thesis, Department of Computer Engineering, King Mongkut's University of Technology Thonburi, Bangkok, Thailand, 2001.

Practical Reasoning about Accountability in Electronic Commerce Protocols

Supakorn Kungpisdan[1] and Yongyuth Permpoontanalarp[2]

Logic and Security Laboratory
Department of Computer Engineering
Faculty of Engineering
King Mongkut's University of Technology Thonburi
91 Pracha-utit Rd. Bangmod, Thoongkru, Bangkok, 10140, Thailand
[1]hotkeng@hotmail.com
[2]yongyuth@cpe.eng.kmutt.ac.th

Abstract. Accountability in electronic commerce (e-commerce) protocols is concerned with the ability to show that particular parties who engage in the protocols are responsible for some transactions. Traditionally, it is used only for resolving disputes amongst parties. Many logics were proposed for reasoning about the accountability. However, these logics lack of the application to real-world e-commerce protocols. In this paper, we show that these logics are inadequate to analyze the accountability property of such real-world protocols. We then propose a modification of the existing logics to deal with such real-world protocols. Furthermore, we propose a novel use of the accountability for specifying and analyzing the goals of e-commerce protocols, in particular client privacy property.

Keywords: Formal methods for security protocols, analysis of electronic commerce protocols

1 Introduction

In electronic commerce (e-commerce), every party requires a guarantee that any transaction that occurs is carried out in a secure fashion. In fact, it is unavoidable to prevent malicious behaviors of some parties. These malicious behaviors cause "*Disputes*". Although the system cannot prevent the dishonest party from modifying transactions, it should allow honest parties to prove their rightful behavior when a dispute amongst parties occurs. This property should be included in well-designed protocols since it gives assurances to users in that each user can prove what has actually happened in a transaction to a verifier, who may be a court or someone acting as a dispute solver. This property is called accountability.

Accountability in e-commerce protocols [1] is concerned with the ability to show that particular parties who engage in such protocols are responsible for some transactions. In particular, the accountability [1] involves the ability of a party, called a prover to convince to another party, called a verifier, that a statement is true *without revealing any secret information to the verifier*. Traditionally, the accountability is

used only to resolve disputes amongst parties. In such cases, a judge would act as the verifier, and a defendant would act as a prover.

On one hand, many logics [1,3] were proposed for reasoning about the accountability in e-commerce protocols. However, these logics lack of the application to real-world e-commerce protocols. On the other hand, Herreweghen in [2] proposed an analysis of two real world protocols: SET [8] and iKP [7] on the accountability property. The analysis shows that SET lacks of the accountability whereas iKP has the property. The analysis is carried not informally since it does not use any formal logic.

In this paper, we argue that the existing logics for analyzing the accountability are *inadequate* to analyze real-world e-commerce protocols. In particular, we argue that the existing logics are not able to reason about *multiply* encrypted and/or hashed messages which are signed. Such messages are obtained from applying encryption, digital signature, hash function, or a combination of them to a plain message. Indeed, this kind of messages is that which is employed in most, if not all, e-commerce protocols.

Moreover, in order to reason about the accountability in general, it requires reasoning about verifiers whether a verifier would be convinced of some statement, after a prover sends some *non-secret* or *non-private* information to the verifier.

We argue that Kailar's logic [1] does not provide reasoning about verifiers at all. Even though Kessler and Neumann's (KN) logic [3] offers reasoning about verifier's belief as a result of the receipt of some information, it does not address the issue about proving without revealing secret or private information to verifiers. Note that this issue is a part of the definition of the accountability [1].

In this paper, we propose a modification of KN's logic to reason about the accountability of real-world e-commerce protocols. Our logic can reason not only about the accountability of complex cryptographic messages which are signed, but also about verifiers' beliefs as a result of the receipt of some information. Moreover, our logic captures the provability without revealing any secret or private information to verifiers naturally by using provers' beliefs about verifiers' possession of information. We demonstrate the practicality of our logic by showing that the result obtained from the informal analysis [2] can also be obtained from our logic.

The practical aspect of the accountability that was discussed so far in the literature [2,5] is about payment or money *only*. In particular, the money accountability is about the authorized transfer of money from customer's account to merchant's account. In this paper, we introduce another kind of accountability, called *goods* accountability, which is about the authorized order of goods by a client. The goods accountability can be used to resolve disputes on the mismatch between the goods which is ordered and that which is delivered. By using our logic, we show that both SET and iKP lack of the goods accountability.

Furthermore, we propose a novel use of the accountability for specifying and analyzing the goals of e-commerce protocols, in particular *client privacy* property. The main goal of e-commerce protocols is to ensure that after the completion of the protocols, all parties who engage in the protocols are convinced that they have authorized messages concerning transactions relevant to them. We argue that this goal can be understood as *a special case* of the accountability where the prover is the

originator of a message, the verifier is the intended recipient and the statement to be proved is about some transactions.

The *client privacy* is concerned with clients' provability of their authorized payment for a transaction without revealing goods description and payment information (e.g. credit card numbers) in the transaction to banks and merchants, respectively. Thus, the client privacy can be understood as the accountability where a client is a prover, both a bank and a merchant are verifiers, and the proving statement is about the payment authorization. Moreover, both goods description and payment information are considered as client's private information, which must not be revealed to the bank and the merchant, respectively.

This paper is organized as follows. Section 2 provides the background for the accountability and the existing logics. Section 3 presents our logic. In section 4, we applied our logic to analyze SET and iKP on the two kinds of the accountability, and to specify and analyze SET on its goals. Section 5 presents the conclusion of our work.

2 Background

2.1 Formal Approach to Accountability

Kailar's Logic

Kailar [1] is probably the first who propose a modal logic to reason about accountability. Kailar's definition of accountability is concerned with the ability to prove the association of an originator (of a message) with some action to a third party without revealing any private information to the third party. The party who can prove such a statement is called a prover whereas the third party who is convinced of the proof is called a verifier. Kailar employs the modal operator '*CanProve*' to formalize the concept of accountability, i.e. *P CanProve* ϕ *to V* where *P* and *V* stand for prover and verifier, respectively, and ϕ stands for a general statement about some action.

However, Kailar's logic [1] is not suitable for analyze the real-world e-commerce protocols because of the following two reasons: firstly, Kailar's logic can analyze the signed plain message only. However, messages in real-world e-commerce protocols are not just signed plain messages, but they often are multiply encrypted and/or hashed messages which are signed, i.e. $\{h(X)\}_K$ *or* $\{\{X\}_{K'}\}_{K^{-1}}$, where K' is public key of a party and K^{-1} is private key of another party. Secondly, Kailar's logic does not reason about verifiers at all. Recently, it was pointed out in [2] that reasoning about verifiers is essential for analyzing real-world e-commerce protocols. We shall discuss this issue in detail in a later section.

It should be noted also that Kailar's definition for accountability is *general* in that the actions which are associated with an originator can be of any kinds.

Kessler and Neumann's Logic

Following Kailar, Kessler and Neumann (KN) [3] employs a modal logic to reason about the accountability. However, KN provides an alternative definition of the modal operator '*CanProve*' by means of sending messages. KN's goal to show the accountability is to show '*P believes P CanProve* ϕ *to V*' where *P* and *V* stands for a prover and a verifier, respectively. One way to show that *P believes P CanProve* ϕ *to V* holds is for *P* to believe that *P* can convince *V* to believe ϕ by sending some messages that *P* has to *V*. Thus, this logic offers reasoning about both prover's beliefs and verifier's beliefs, and in particular, prover's beliefs about verifier's beliefs.

Even though KN's logic offers reasoning about the accountability of hashed messages which are signed, its definition for dealing with such messages is *too strong* in that verifiers are able to infer the input *m* of any hash *h(m)*, which may be *private information*. Consider the following rule for dealing with such kind of message in KN's logic:

$$P \text{ CanProve } (Q \text{ said } h(X)) \text{ to } V \wedge V \text{ believes } \neg Q \text{ sees } h(X) \rightarrow P \text{ CanProve } (Q \text{ said } X) \text{ to } V$$

This rule states that if *P* can prove that *Q* said the hash value of *X, i.e. h(X)*, and *V* believes that *Q* does not receive *h(X)* from anyone, then *P* can prove to *V* that *Q* said *X*. We argue that this is *not intuitive*.

Generally, to prove that *Q* said the input *X* of a hash function, both prover and verifier need to know:

a) The hashed message *h(X)* and the input (*X*), and
b) That the hashed message is really obtained from applying hash function *h* to *X*.

Obviously, in KN's approach, the verifier does not need to know *a)* and *b)* but simply that the verifier must believe that the initiator does not receive the hash of the message from anyone in order to be convinced that the initiator originates the hashed message and thus its input. The verifier is convinced even though to show *b)* requires the knowledge of some private information, which should not be revealed to any third party (e.g. goods description).

Moreover, their logic does not deal with the accountability of encrypted messages which are signed, i.e. $\{\{X\}_{K'}\}_{K^{-1}}$ where K' is public key of a party and K^{-1} is private key of another party.

2.2 Herreweghen's Approach

Herreweghen [2,4] proposed the analysis of the accountability of the real-world e-commerce protocols: SET [8] and iKP [7]. The analysis shows that SET lacks of the accountability whereas iKP does not. The analysis is not formal since it is done without using any formal logic. However, the analysis is presented partly in rule-based styles.

Unlike the accountability in Kailar's approach and KN's approach, the kind of the accountability that is analyzed in [2] is *specific* in that it focuses on primitive transactions [5], which are the core of e-commerce protocols. It was proposed in [5] that a payment transaction in any e-commerce protocol consists of three types of primitive transactions:

- ***Payment:*** client makes the payment to merchant.
- ***Value subtraction:*** client allows acquirer to deduct money from his account.
- ***Value claim:*** merchant requests acquirer to deposit money to merchant's account.

Thus, in order to show that an e-commerce protocol has the accountability for resolving disputes, it suffices to show that at the completion of the protocol, each party in the protocol must be able to prove the authorizations of primitive transactions concerning the party to an external verifier. The following shows such kind of statements:

- *M can prove "C **authorized** payment(C, M, Amount, Date, Ref)"*
  Merchant can prove that client has sent authorized message on making the payment to him.
- *C can prove "M **authorized** payment(C, M, Amount, Date, Ref)"*
  Client can prove that merchant has sent authorized message on acknowledgement of payment to him.
- *A can prove "C **authorized** value subtraction(A, C, Amount, Date, Ref)"*
  Acquirer can prove that client has sent authorized message on requesting him to deduct money from client's account.
- *C can prove "A **authorized** value subtraction(A, C, Amount, Date, Ref)"*
  Client can prove that acquirer has sent authorized message on acknowledgement of requesting to deduct money from client's account.
- *A* can prove "*M **authorized** value claim(A, M, Amount, Date, Ref)*"
  Acquirer can prove that merchant has sent authorized message on requesting him to transfer money to merchant's account.
- *M can prove "A **authorized** value claim(A, M, Amount, Date, Ref)"*
  Merchant can prove that acquirer has sent authorized message on acknowledgement of transferring money to merchant's account.

The analysis in [2] shows that SET lacks of the accountability because of the following two problems:

Firstly, prover does not have key for decrypting the encrypted message, which contains the necessary information. For example, prover wants to derive *PI* from $\{PI\}_{K_A}$, but he does not have acquirer's private key.

Secondly, prover has to reveal the private information to verifier in order to prove the authorization. The source of this problem is that the required information is hashed together with the information which is considered to be private information to verifier e.g. *h(Price,OD)* where *Price* is the required information and *OD* is private information. To derive *Price*, prover is required to send both *Price* and *OD* to verifier to convince him. Thus the verifier knows the private information.

This problem does not occur in iKP. The message format in iKP is $h(Price, h(OD))$. Thus, the prover can convince the verifier about *Price* without revealing *OD* because it is hashed.

We argue that Herreweghen's analysis [2] is sensible because in order to solve disputes, prover wants to send only the necessary evidence to judge. The unnecessary private information is not what he wants to reveal for proving some statements. It is intuitive in proving something to other parties.

The disadvantages of this approach are as the following:

1) It is unsystematic since the analysis is not formal.
2) It is specific to only SET and iKP. Those rules cannot be applied to other protocols.

3 Our Logic

Our logic is based on KN's logic [3]. In particular, our logic can be seen as an extension and a simplification of KN's logic. It employs the concept of *provable authorization in the present of private information*. In order to solve disputes, a prover wants to send only the necessary information to prove some statements to a judge who acts as a verifier without revealing the unnecessary private information. With this concept, prover can prove the statement without revealing any private information to verifier. We extend KN's logic in two main aspects. Firstly, we provide axioms for the accountability of multiply encrypted and/or hashed messages which are signed in order to resolve disputes. Secondly, we propose axioms for dealing with the use accountability to specify and analyze the goals of e-commerce protocols.

Syntax

Terms

- $\{P, Q, V\}$: A set of principles that communicate with each other in the protocol.
- $\{X, Y\}$: Messages or message components sent in the protocol.
- $\{\phi, \psi\}$: The statements derived from messages.
- $\{K_P, K_Q\}$: A set of the public key of the principal P and Q, respectively.
- $\{K_P^{-1}, K_Q^{-1}\}$: A set of private key of the principal P and Q, respectively.
- $\{X\}_K$: The message X encrypted with key K.
- $h(X)$: The hash function of a message X.
- $\xrightarrow{K} Q$: Key K is the public key of Q.
- $P \overset{K}{\leftrightarrow} Q$: K is the shared key between P and Q.
- *X-is-fingerprint-of-Y* : X can be used as the representative (fingerprint) of Y (in other words, X may be the hashed form of Y).
- *K-is-decrypting-key-for-*$\{X\}_K$: Key K can be used to decrypt the encrypted message $\{X\}_K$.

Formulae

- *P believes* ϕ : A principle P believes that the statement ϕ is true by doing some actions.
- *P sees* X : Someone has sent a message X to P and P is able to read X.
- *P has* X : A principle P possesses a message X. He can send X to other principles or use it for further computations.
- *P says* X : A principle P has sent a message X.
- *P CanProve* ϕ *to* Q : A principal P can prove to Q that the statement ϕ is true by sending a message X to Q. After Q receives X, he believes that the statement ϕ is true.
- *P authorized payment(P, Q, Price, Date)* : A principle P has authorization on making the payment amount *Price* to Q on date *Date*.
- *P authorized goods-order(P, Q, OD, Date)* : A principle P has authorization on ordering goods as specified in order description *OD* to Q on date *Date*.

Axioms

Modalities: KD45-logic

K: $P \text{ believes } \phi \wedge P \text{ believes } (\phi \rightarrow \psi) \rightarrow P \text{ believes } \psi$

D: $P \text{ believes } \phi \rightarrow \neg P \text{ believes } \neg\phi$

4: $P \text{ believes } \phi \rightarrow P \text{ believes } P \text{ believes } \phi$

5: $\neg P \text{ believes } \phi \rightarrow P \text{ believes } \neg P \text{ believes } \phi$

Possession

H1: $P \text{ sees } X \rightarrow P \text{ has } X$

H2: $(P \text{ has } X_1 \wedge \ldots \wedge P \text{ has } X_n) \leftrightarrow P \text{ has } (X_1, \ldots, X_n)$
Where $(X_1, \ldots, X_n)$ stands for a list of message $X_1, X_2, \ldots, X_n$, respectively.

H3: $P \text{ has } X \rightarrow P \text{ has } h(X)$

The following axioms (H4, H5, and H6) are based on BAN logic [6] in order to deal with cryptographic messages.

H4: If P has an encrypted message $\{X\}_K$, P believes that K is shared between P' and Q, and P has key K, then P has X.

$$(P \text{ has } \{X\}_K \wedge P \text{ believes } P' \overset{K}{\leftrightarrow} Q \wedge P \text{ has } K) \rightarrow P \text{ has } X$$

H5: If P has an encrypted message $\{X\}_K$, P believes that K is the public key of P', and P has its private key K^{-1}, then P has X.

$$(P \text{ has } \{X\}_K \wedge P \text{ believes } \overset{K}{\rightarrow} P' \wedge P \text{ has } K^{-1}) \rightarrow P \text{ has } X$$

H6: If P has a signed message $\{X\}_{K^{-1}}$, P believes that K is the public key of P', and P has its verification key K, then P has X.

$(P \text{ has } \{X\}_{K^{-1}} \wedge P \text{ believes } \xrightarrow{K} P' \wedge P \text{ has } K) \rightarrow P \text{ has } X$

Comprehension

C1: If P has received X, P then believes that he receives X.

$P \text{ sees } X \rightarrow P \text{ believes } P \text{ sees } X$

C2: If P has sent X, he then believes that he has sent X.

$P \text{ says } X \rightarrow P \text{ believes } P \text{ says } X$

Seeing

SE1: $P \text{ sees } (X_1, \ldots, X_n) \leftrightarrow (P \text{ sees } X_1 \wedge P \text{ sees } X_2 \wedge \ldots \wedge P \text{ sees } X_n)$

Saying

SA1: $P \text{ says } (X_1, \ldots, X_n) \leftrightarrow (P \text{ says } X_1 \wedge P \text{ says } X_2 \wedge \ldots \wedge P \text{ says } X_n)$

SA2: $P \text{ says } X \rightarrow P \text{ has } X$

Provability

P1: $P \text{ CanProve } (\phi \rightarrow \psi) \text{ to } V$
$\rightarrow (P \text{ CanProve } \phi \text{ to } V \rightarrow P \text{ CanProve } \psi \text{ to } V)$

P2: $(V\text{-is-external-party}) \wedge (P \text{ has } X) \wedge (V \text{ sees } X \rightarrow V \text{ believes } \phi)$
$\rightarrow (P \text{ CanProve } \phi \text{ to } V)$

It can be seen that axiom P2 can be used to deal with the provability to an external verifier. Thus, the axiom states the accountability for resolving disputes.

P2': If V is an internal party, P has sent a message X', V receives X, and if V receives X, then he believes that ϕ is true and he has X', then P can prove to V that ϕ is true.

$(V\text{-is-internal-party}) \wedge (P \text{ says } X') \wedge (V \text{ sees } X) \wedge$
$[\, V \text{ sees } X \rightarrow (V \text{ believes } \phi \wedge V \text{ has } X')\,]$
$\rightarrow (P \text{ CanProve } \phi \text{ to } V)$

It should be noted that the verifier V in this axiom is an internal party who engages in the protocol. This axiom is intended to deal with the accountability for specifying and analyzing the goals of e-commerce protocols. We shall discuss this issue and the intuition of axiom **P2'** in section 4.4.

P3: $P \text{ has } \{X\}_{K^{-1}} \wedge P \text{ has } (K, X) \wedge P \text{ CanProve } (\xrightarrow{K} Q) \text{ to } V$
$\rightarrow P \text{ CanProve } (Q \text{ says } X) \text{ to } V$

The following shows our axiom to deal with the provability about hashed messages. Intuitively, it states that in order to prove a statement about a hashed message, it requires both prover and verifier to know the input of the hash in order to compare the input with such hashed message.

P4: If P can prove to V that Q has sent $h(X)$ and he can prove to V that $h(X)$ is represented as fingerprint of X, P then can prove to V that Q has sent X.

P CanProve (Q says h(X)) to V ∧ P CanProve (h(X)-is-fingerprint-of-X) to V
→ P CanProve (Q says X) to V

The following shows our axiom to deal with the provability about encrypted messages. In order to prove a statement about an encrypted message, it requires both prover and verifier to know the decrypting key for such encrypted message.

P5: If P can prove to V that Q has sent the encrypted message $\{X\}_K$ and he can prove to V that the key K' can be used to decrypt this encrypted message, he can prove to V that Q has sent message X.

P CanProve (Q says $\{X\}_K$) to V ∧
P CanProve (K'-is-decrypting-key-for-$\{X\}_K$) to V
→ P CanProve {Q says X} to V

P6: *P CanProve (Q says (X_1, ..., X_n) to V*
↔ [P CanProve (Q says X_1) to V ∧ ... ∧ P CanProve (Q says X_n) to V]

Inference Rules
MP: If ϕ and $\phi \rightarrow \psi$ then ψ
M: If ϕ is a theorem then P believes ϕ is a theorem
Where theorem is a formula, which can be derived from axioms alone.

The money accountability property can be specified by the following formula:

M believes M CanProve [C says (M, Price, Date)
→ C authorized payment(C, M, Price, Date)] to V

If merchant believes that he can prove that client has sent merchant's ID, price, and date of execution, then the authorization on *Payment* in [2] can be proved. Proving money accountability mainly concerns that the price can be proved to verifier. Goods description (*OD*) is considered to be client's private information. Similarly, the goods accountability property can be specified by the following formula:

M believes M CanProve [C says (M, OD, Date)
→ C authorized goods-order(C, M, OD, Date)] to V

Proving goods accountability mainly concerns that the goods description can be proved to verifier. *Price* is considered to be merchant's private information.

We argue that if money accountability in [2] is sensible, goods accountability is sensible as well since goods accountability is focused on solving disputes on the mismatch between the goods which is ordered and that which is delivered, which can be occurred in practical transactions.

Apart from the axioms and rules of inference presented here, our logic contains a set of assumptions. Some of these assumptions are general whereas others are specific to protocols which are to be analyzed. For the ease of the presentation here, we discuss both kinds of assumptions in the following section.

4 Practical Reasoning about Accountability

We demonstrate how our logic works by analyzing SET [8] and iKP [7], which are the real-world and widely used protocols. In section 4.1, we describe the overview of SET protocol and the set of assumptions used in the analysis. Section 4.2 and 4.3 illustrate the analysis of proving money and goods accountability, respectively. Particularly, section 4.2 shows that the results obtained from Herreweghen's analysis [2] can also be obtained in our logic. Section 4.4 shows the analysis of accountability for specifying goals of SET.

4.1 SET Protocol Overview

SET is the most widely used credit-card based protocol using public key cryptography scheme. In SET, every participant has his own certificate. The basic SET protocol model is shown as follows:

PinitReq:	C→M:	*Initial Request*
PinitRes:	M→C:	$\{TID\}_{K_M^{-1}}$
PReq:	C→M:	$TID, OI, h(PI), \{h(OI), h(PI)\}_{K_C^{-1}}, \{h(OI), PI\}_{K_A}$
AuthReq:	M→A:	$\{\{TID, Price, AuthReqDate, h(OI), OI, \{h(OI), h(PI)\}_{K_C^{-1}}, \{h(OI), PI\}_{K_A}\}_{K_M^{-1}}\}_{K_A}$
AuthRes:	A→M:	$\{\{TID, Price, AuthDate\}_{K_A^{-1}}\}_{K_M}$
PRes:	M→C:	$\{TID, AuthDate\}_{K_M^{-1}}$

Where

- $\{A, C, M\}$ stands for a set of acquirer (or bank), client, and merchant, respectively.
- $\{K_A, K_C, K_M\}$ stands for the set of public key of acquirer, client, and merchant, respectively.
- $\{K_A^{-1}, K_C^{-1}, K_M^{-1}\}$ stands for the set of private key of acquirer, client, and merchant, respectively.

- *Price* stands for amount and currency of goods.
- *OI* stands for order information. *OI* contains *{TID, h(OD, Price)}*.
- *PI* stands for payment information. *PI* contains *{TID, h(OD, Price), MerchantID, Price, Client's Credit-card Information}*.
- *TID* stands for transaction ID. *TID* contains purchase request date (*PReqDate*), which is the date of issuing purchase request.
- *AuthReqDate* stands for the date of making request by merchant for transferring money.
- *AuthDate* stands for the date of authorization made by bank.

Set of Assumptions

Notations
- *P* stands for any participant.
- *{A, C, M}* stands for a set of acquirer (or bank), client, and merchant, respectively.
- *V* stands for a verifier.
- *X* stands for *A*, *C*, or *M*.
- $Cert_X$ stands for the certificate of principle *X*. $Cert_X$ contains (X, K_X), where *X* is the identity of a party *X* and K_X stands for the public key of *X*.

Assumptions

(A1) Every participant believes that he has the certificates of all participants.

$$P \text{ believes } P \text{ has } (Cert_A, Cert_C, Cert_M)$$

(A2) *P* believes that if *V* has *X*'s certificate, he will believe that K_X is the public key of *X*.

$$P \text{ believes } (V \text{ has } Cert_X \rightarrow V \text{ believes } (\xrightarrow{K_X} X))$$

(A3) Every participant believes that he has only his private key, and nobody believes that he has the private keys of other participants.

$$P \text{ believes } P \text{ has } K_P^{-1} \rightarrow P \text{ believes } P \text{ has } (K_{Q_1}^{-1}, K_{Q_2}^{-1})$$

Where Q_1 and Q_2 are the parties, which are different from *P*.

(A4) Every participant believes that if *V* receives messages *X* and *Y* and the message *X* is *h(Y)*, *V* will believe that *X* is fingerprint of *Y*.

$$P \text{ believes } [\, (V \text{ has } (X, Y) \wedge X = h(Y)) \rightarrow V \text{ believes } X\text{-is-fingerprint-of-}Y \,]$$

(A5) Every participant believes that if *V* receives a message encrypted with key *K* ($\{X\}_K$) and the key *K'*, and *K'* can be used to decrypt $\{X\}_K$, *V* will believe that *K'* is the decrypting key for $\{X\}_K$.

P believes (V has $(\{X\}_K, K') \wedge X = \{\{X\}_K\}_{K'}$
→ V believes K'-is-decrypting-key-for-$\{X\}_K$)

The assumptions A4 and A5 are general in that they can be used for analyzing any protocol. Indeed, together with P4 and P6, the two assumptions are used to reason about the provability of hashed messages and encrypted messages.

(A6) Every participant believes that if V has the signed message ($\{X\}_{K^{-1}}$), the key K, and the message X, and V believes that K is the public key of Q, then V will believe that Q has sent X.

P believes [(V has $\{X\}_{K^{-1}} \wedge$ *V has (K, X)* $\wedge$ *V believe* ($\xrightarrow{K}$ *Q))*
→ V believes (Q says X)]

This assumption together with axiom P2 or P2' can be used to derive similar conclusion as the axiom P3. However, this assumption would derive the conclusion, based on verifier's possession of some information whereas the axiom P3 would derive the conclusion, based on prover's possession of some information.

No Disclosure of Private Information to Verifier

(A7) Every participant does not believe that verifier has payment information and the private keys of all participants, order description in case of proving money authorization, and price in case of proving goods authorization.

$\neg$*P believes V has PI*	where V is not acquirer.
$\neg$*P believes V has* K_P^{-1}	where V is not the same party as P.

In case of Money Authorization,	$\neg$*P believes V has OD*
In case of Goods Authorization,	$\neg$*P believes V has Price*

By specifying the requirement "No disclosure of private information to verifier" as a set of assumptions specific to protocols, our logic formalizes intuitively the concept of the unrevealing private information to verifier when a prover proves a certain statement. This concept is essential in the accountability property for reasoning about accountability in the present of party privacy.

SET Specific Assumptions

(A8) Client and merchant believe that they have order description and price, and all participants believe that they are internal parties.

Q believes Q has (OD, Price)	Where Q stands for C and M.
R believes R'-is-internal-party	Where R and R' stand for A, C, and M.

(A9) Every participant believes that client has sent *PReq* and received *PRes*, merchant has sent *AuthReq* and received *AuthRes*, and acquirer has received *AuthReq* and sent *AuthRes*.

P believes C says PReq	*P believes M sees PReq*
P believes M says AuthReq	*P believes A sees AuthReq*
P believes A says AuthRes	*P believes M sees AuthRes*
P believes M says PRes	*P believes C sees PRes*

Client Privacy

(A10) Client does not believe that merchant has payment information, and client and merchant do not believe that acquirer has order description.

¬C believes M has PI
¬Q believes A has OD Where *Q* stands for client and merchant.

General Assumptions for Proving Money Authorizations

(A11) Every participant believes that he can prove to verifier that if client has sent the message containing Merchant's ID, price, and the date of execution, then client has authorization on payment ordering.

P believes P CanProve [C says (M, Price, Date)
→ C authorized payment(C, M, Price, Date)] to V

Assumptions for proving money authorizations for other primitive transactions are shown in Appendix.

General Assumptions for Proving Goods Authorizations

(A12) Every participant believes that he can prove to verifier that if client has sent the message containing Merchant's ID, order description, and the date of execution, then client has authorization on goods ordering.

P believes P CanProve [C says (M, OD, Date)
→ C authorized goods-order(C, M, OD, Date)] to V

Assumptions for proving goods authorizations for other primitive transactions are shown in Appendix.

4.2 Proving Money Accountability in SET

Proving accountability in SET, we focus on analyzing one of primitive transactions, which is *C authorized payment (C, M, Price, PReqDate)*. This means that client has authorization on making payment on the goods amount *Price* on the date of making the request *Date*.

The goal of proof:
M believes M CanProve (C authorized payment(C, M, Price, PReqDate)) to V

Where *V* stands for any external verifier.

In order to show that *M believes M CanProve (C authorized payment(C, M, Price, PReqDate)) to V*, it suffices to show that

M believes M CanProve (C says (M, Price, PReqDate)) to V (x)

It is easy to see that *M believes M CanProve (C says h(PI)) to V* follows mainly by axiom P3. Since *(M, Price, PReqDate)* is in *PI*, (x) may thus be shown by using axiom P4.

However, the proof for *M believes M CanProve (h(PI)-is-fingerprint-of-PI) to V* in axiom P4 would fail since it is not the case that *M believes M has* K_A^{-1} which is required to show that *M believes M has (h(PI), PI)*. Note that *PI* is encrypted with K_A. Note also that if it were the case that *M believes M has* K_A^{-1}, the proof would still fail since it would require *M believes (V has PI)*, due to A4, which contradicts to our assumption A7.

It is also easy to see that *M believes M CanProve (C says OI) to V* follows mainly by axioms P4 and P3. Since *h(OD, Price)* is in *OI*, the proof for *M believes M CanProve (C says Price) to V* may be shown by using axiom P4. However, such proof would require *M believes V has OD* due to A4, which contradicts to our assumption A7.

4.3 Proving Goods Accountability in SET and iKP

The goal of proof:
M believes M CanProve (C authorized goods-order(C, M, OD, PReqDate)) to V

We shall discuss the goods accountability in SET first. Similarly to the proof for *M believes M CanProve (C says Price) to V* discussed in section 4.2, the proof for *M believes M CanProve (C authorized goods-order(C, M, OD, PReqDate)) to V* would fail since it would require *M believes V has Price* which contradicts to our assumption A7.

IKP [7] lacks of goods accountability due to similar reason to that in SET. The following shows a payment request from client *C* to merchant *M*.

Payment $C \rightarrow M$: $PI, \{PI, h(OI)\}_{K_C^{-1}}$

Where *PI* stands for payment information. *PI* contains *{Price, h(OI), Client's Credit-card Information}*$_{K_A}$. *OI* stands for order information. *OI* contains *(TID, Price, ClientID, MerchantID, Date, InvExpDate, h(OD))*. *InvExpDate* stands for invoice (offer) expiration date specified by merchant. *Date* stands for the date of invoice (offer) issued by merchant.

It is easy to see that *M believes M CanProve (C says h(OI)) to V* follows mainly by axiom P3. Since *(Price, h(OD)* is in *OI*, the proof for *M believes M CanProve (C says OI) to V* can be shown by using axiom P4. However, such proof would require *M believes V has Price* due to A4, which contradicts to our assumption A7.

4.4. Goals of SET and iKP as Accountability

In this section, we discuss the use of the accountability for specifying and analyzing the goals of SET and iKP protocols. For such kind of accountability, a verifier is an internal party, which involves in e-commerce protocols. Recall that the goal of e-commerce protocols is to ensure that all parties are convinced that they have authorized messages concerning primitive transactions relevant to them after the completion of the protocols.

After sending some messages to intended recipients, the originator must be able to prove the association of the originator with an intended action (or an intended message) to intended recipient(s). Such intended actions are just about primitive transactions. Such proof would ensure the originator that the intended recipients would recognize the originator's intention regarding to primitive transactions.

This intuition is formalized by axiom *P2'*. The axiom states explicitly the preconditions for the sending and receiving of messages. The rule also caters for the case where the intended recipient receives messages from an intermediate party.

Thus, the goals of e-commerce protocols can be expressed by the following:

a) C believes C CanProve (C authorized payment(C, M, Price, Date)) to M
b) M believes M CanProve (M authorized payment(C, M, Price, Date)) to C
c) C believes C CanProve (C authorized value subtraction(A, C, Price, Date)) to A
d) A believes A CanProve (A authorized value subtraction(A, C, Price, Date)) to C
e) M believes M CanProve (M authorized value claim(A, M, Price, Date)) to A
f) A believes A CanProve (A authorized value claim(A, M, Price, Date)) to M
g) C believes C CanProve (C authorized goods-order(C, M, OD, Date)) to M
h) M believes M CanProve (M authorized goods-receipt(C, M, OD, Date)) to C

It is not hard to see that in SET these goals cannot be shown due to similar reason to those for the accountability for dispute resolving discussed in sections 4.2 and 4.3. However, these goals can be shown for iKP.

With provable authorization in the present of private information, our logic can be used to specify and analyze goals of e-commerce protocols efficiently in that the originator can ensure that the recipient recognizes the intention about primitive transaction without revealing private information. This will be much benefit to protocol designers in that he can design a protocol with intended purposes.

What we demonstrate below is the analysis of client privacy using our logic.

Proving Client Privacy of SET

Client privacy can be understood as the accountability where a client is a prover, both a bank and a merchant are verifiers, and the proving statement is about the payment

authorization. SET achieves client privacy if merchant cannot infer client's payment information *(PI)*, and acquirer cannot infer goods description *(OD)* from the protocol. We thus start proving client privacy by stating the goals of the proofs from *a)* and *c)*. In order to prove *a)*, it suffices to show that

C believes C CanProve (C says (M, Price, PReqDate)) to M

It is easy to see that *C believes C CanProve (C says (Price, PReqDate)) to M* follows mainly by axioms P4 and P3. Although the proof is not successful because of the lacking of merchant's ID, merchant cannot infer *PI* from the receiving message. We prove *c)* in the same way as proving *a)*. It is not hard to see that acquirer cannot infer *OD*, which is client's private information, from the receiving message.

As a result, our logic can analyze client privacy, which is an essential property of SET protocol, in that verifier can get only necessary information to prove without getting any private information.

5 Conclusion

In this paper, we show that the existing logics for reasoning about accountability are inadequate to deal with real world e-commerce protocols. We then propose an extension of the existing logics to deal with such real-world protocols. Furthermore, we demonstrate the practicality of our logic by showing that the result obtained from Herreweghen's informal analysis [2] can be also obtained formally from our logic.

Our logic can be used not only for reasoning about dispute resolution amongst parties, but also for reasoning about the goals of e-commerce protocols, in particular, client privacy property. Indeed, both kinds of reasoning can be captured *uniformly* in our logic.

For reasoning about dispute resolution, we show *formally* that SET lacks of the money accountability whereas iKP does not. Moreover, we show that both SET and iKP lack of the goods accountability. The lack of the goods accountability in our sense means that the two protocols can still provide enough evidence tokens to resolve disputes on goods description, but in order to resolve such disputes, provers must reveal their private information to verifiers. In some situations, this is undesirable.

For reasoning about the goals of e-commerce protocols, we show that such kind of reasoning can be considered as a special case of the accountability. Thus, the accountability can be seen as a fundamental property for analyzing e-commerce protocols. Indeed, other kind of properties such real-world e-commerce protocols, for example each party's requirement [9], has also been studied in [10] by using our logic presented here.

As a future work, we aim to apply our logic to analyze other kinds of e-commerce protocols, e.g. micropayment. Also, we aim to study an automated tool for our logic.

Acknowledgement

The second author would like to acknowledge supports from Thailand Research Fund.

References

[1] R. Kailar. Accountability in Electronic Commerce Protocols. *IEEE Transaction on Software Engineering 1996.*

[2] E. V. Herreweghen. Non-Repudiation in SET: Open Issues. In the *Proceedings of the Financial Cryptography 1999.*

[3] V. Kessler and H. Neumann. A Sound Logic for Analyzing Electronic Commerce Protocols. In the *Proceedings of ESORICS'98.*

[4] E. V. Herreweghen. Using Digital signatures as Evidence of Authorizations in Electronic Credit-Card Payments. *Research report 3156, IBM Research,* June 1999.

[5] N. Asokan, E. V. Herreweghen, and M. Steiner. Towards A Framework for Handling Disputes in Payment Systems. In the *Proceedings of the 3rd USENIX workshop on Electronic Commerce*, Boston, Massachusette, August31-September3,1998.

[6] M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. *ACM Transactions in Computer Systems*, February 1990.

[7] M. Bellare, J. A. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, E. V. Herreweghen, and M. Waidner. Design, Implementation, and Deployment of the *i*KP Secure Electronic Payment System. *IEEE Journal of Selected Areas in Communications* 2000.

[8] Mastercard and Visa. SET Protocol Specifications. `http://www.setco.org/set_specifications.html`

[9] C. Meadows and P. Syverson. A Formal Specification of Requirements for Payment Transactions in the SET Protocol. In the *Proceedings of Financial Cryptography*, February 1998.

[10] S. Kungpisdan. Accountability as Fundamental Property for Electronic Commerce Protocols. *Master Thesis, Department of Computer Engineering, King Mongkut's University of Technology Thonburi, Thailand, 2001. (In preparation).*

Appendix

General Assumptions for Proving Money Authorizations

Every participant believes that he can prove to verifier that if merchant has sent the message containing Client's ID, price, and the date of execution, then merchant has authorization on payment receipt.

P believes P CanProve [M says (C, Price, Date)

→ M authorized payment(C, M, Price, Date)] to V

Every participant believes that he can prove to verifier that if acquirer has sent the message containing Client's ID, price, and the date of execution, then acquirer has authorization on value subtraction.

P believes P CanProve [A says (C, Price, Date)
→ A authorized value subtraction(A, C, Price, Date)] to V

Every participant believes that he can prove to verifier that if client has sent the message containing Acquirer's ID, price, and the date of execution, then client has authorization on value subtraction.

P believes P CanProve [C says (A, Price, Date)
→ C authorized value subtraction(A, C, Price, Date)] to V

Every participant believes that he can prove to verifier that if acquirer has sent the message containing Merchant's ID, price, and the date of execution, then acquirer has authorization on value claim.

P believes P CanProve [A says (M, Price, Date)
→ A authorized value claim(A, M, Price, Date)] to V

Every participant believes that he can prove to verifier that if merchant has sent the message containing Acquirer's ID, price, and the date of execution, then merchant has authorization on value claim.

P believes P CanProve [M says (A, Price, Date)
→ M authorized value claim(A, M, Price, Date)] to V

General Assumptions for Proving Goods Authorizations

Every participant believes that he can prove to verifier that if merchant has sent the message containing Merchant's ID, order description, and the date of execution, then merchant has authorization on goods receipt.

P believes P CanProve [M says (C, OD, Date)
→ M authorized goods-receipt(C, M, OD, Date)] to V

A Verification Methodology for Analyzing IP Spoofing Attack

Voravud Santiraveewan and Yongyuth Permpoontanalarp
Logic and Security Laboratory
Department of Computer Engineering
King Mongkut's University of Technology Thonburi
91 Suksawasd 48, Ratburana, Bangkok 10140, Thailand
E-mail: voravuds@hotmail.com, yongyuth@cpe.eng.kmutt.ac.th

Abstract

Firewalls offer a protection for private networks against both internal and external attacks. However, configuring firewalls to ensure the protections is a difficult task. The main reason is the lack of methodology to analyze the security of firewall configurations. IP spoofing attack is an attack in network in which an attacker can impersonate another person towards a victim. In this paper, we propose a new methodology for verifying the vulnerability of firewall configurations for IP spoofing attacks. In particular, our methodology is based on graph theory which provides a simple and intuitive approach to the vulnerability analysis of the attack. Furthermore, we propose a class of configurations that are free of spoofing attacks. We show that any configuration which is in the class is also free of spoofing attacks.

Key-Words: Network Security, Firewalls, IP spoofing and Formal Method for Network Security

1. Introduction

Nowadays, firewalls (eg. [10, 11]) become a widely used mechanism to achieve Internet security. Most, if not all, organizations whose computers have an internet access are currently using firewalls. Firewalls locate between an internal network and an external network. Firewalls offer a protection for private (and internal) networks against both internal and external threats. In particular, firewalls ensure that only authorized information flows between internal networks and the external network are allowed.

Even though firewalls could provide protections against both internal and external attacks, configuring firewalls to ensure the protections is a difficult task. There are two main reasons. One is that it is difficult to foresee effects of firewall configurations during the configuration time. It requires administrators' experience to figure out such effects of configurations. Another reason is the lack of methodology to analyze the security of firewall configurations. We argue that the lack of such methodology is caused by the lack of the understanding on the relationship between firewall configurations and attacks.

Several solutions eg. [1, 2, 3, 4] have been proposed to offer a way to foresee effects of firewall configurations. However, a systematic methodology for analyzing the security of firewall configurations remains unexplored. In this paper, we propose a verification methodology of the analysis on the security of firewall configurations. In particular, our methodology offers the vulnerability analysis of firewall configurations for IP spoofing attack.

IP spoofing attack is an attack in which an attacker can impersonate another person towards a victim. In other words, an attacker can impersonate another person and then send a message to a victim but after the victim receives the message, the victim thinks that the message is sent from the impersonated person. It is considered as a fundamental attack because it can be combined with other kinds of attack to create devastating effects to users.

In this paper, we analyze IP spoofing attack by focusing on IP spoofing attacks which may occur in private networks of an organization. We classify IP spoofing attacks into three kinds: incoming, outgoing and internal attacks. Incoming attack deals with the attack by an external attacker to an internal victim whereas outgoing attack deals with the attack by an internal attacker to an external victim. Internal attack deals with the attack by an internal attacker to an internal victim.

Then, we propose a novel methodology for verifying the vulnerability of firewall configurations for IP spoofing attacks. Our methodology offers the vulnerability analysis of firewall configurations for

the three kinds of the attack. Our result shows not only the characteristics (or signatures) of those attack but also the causes of those attacks in terms of firewall configurations. To the best of our knowledge, these kinds of analysis have not been studied before.

Furthermore, we propose a class of configurations that are free of spoofing attacks. Then we show that such class of configurations can actually prevent spoofing attacks. Finally, to verify that a configuration is free of spoofing attacks is to show that the configuration is in the class.

Our methodology is based on Graph-based model of firewall configurations [1, 2]. Since our approach employs graph theory, it provides a simple and intuitive approach to the vulnerability analysis of the attack. Furthermore, our approach offers a number of advantages over existing related works in that our approach can verify the vulnerability of firewall configurations for the attack for all cases. Furthermore, the proposed class of configurations can be used as a guidance to configure firewalls securely.

2. Background

Graph-based model [1, 2] employs graph theory to describe effects of firewall configurations. In particular, effects of firewall configurations are expressed in terms of a set of paths permitted to pass through firewalls.

Graph-based model proposed *generalized firewall rules (GR)* which consist of *(Source, Destination, Service, Direction, Action, FW Interfaces)* to represent firewall rules. In addition we represent actual network as *labeled and connected* graph called *abstract network topology (NT)*.

In figure 1, *R1* and R2 represent two firewall objects. *h1, h2, h3* and *h4* stand for host computers. *ip1, ip2, ..., ip10* denote IP addresses. In particular, *ip1, ip2, ip3, ip4* are IP addresses of four firewall interfaces of firewall object *R1*. *ip7, ip8, ip9* and *ip10* are IP addresses of interfaces of host computers *h1, h2, h3* and *h4*, respectively.

Moreover, *A local set of path(LP)* represent the effect of each firewall rule as a path in network topology. For example, The local set of paths that corresponds to rule *(ip15, ip9, ftp, inbound, permit, ip1)* is *{<ip15, ip1, ip4, ip5, ip6, ip9>}*. *A local set of effective and permitted (prohibited) paths (LEP)* represents effect of all firewall rules in specified firewall interface. In the other words, it represents path that specified firewall interface permit or prohibit. For example, Suppose (ordered set) SGR_{ip6} = *{(ip9, ip7, http, inbound, permit, ip6), (any, any, http, inbound, drop, ip6)}*. The local set of effective and permitted paths at interface *ip6* is *{<ip9, ip6,ip5, ip4, ip2, ip7>}*.

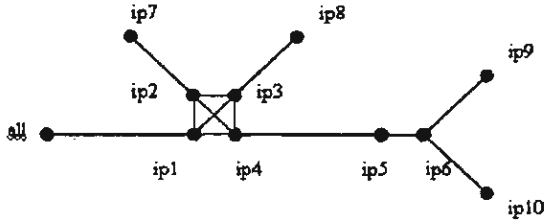

Figure 1. Abstract network topology

A global set of effective and permitted (prohibited) paths (GEP) represents all paths that can occur in the entire network. In the other word, it represents all paths that was allowed by firewall configuration.

3. Modified Graph-based model

Our concept of Graph-based provides the methodology to verify the effect of firewall configuration. However, those can not deal with IP spoofing attack since they depend on network topology that represent only legitimate path while IP spoofing path is illegal path.

In order to dealing with IP spoofing attack, we propose modified graph-based model that can capture all kind of path in the actual network.

3.1 Abstract Configuration

An abstract configuration consists of a set of firewall rules and networking topologies representing network connections within an organization. A set of firewall configuration was represented by the generalized firewall rules (GR) in our former model.

There are two kinds of networking topologies in our approach: namely abstract network topology and abstract firewall topology. While the abstract network topology represents a topology of computer networks in an organization which include hosts, servers, firewalls, networking devices and the internet, the abstract firewall topology represents the topology of only firewalls. Note that abstract network topology was described in [1, 2].

As a matter of terminology, network or firewall topologies mean abstract network or abstract firewall topologies, respectively. In our approach, all graphs are undirected.

Firewall Topology expresses the connections between firewalls in a network topology. Indeed, *FT* can be obtained from *NT*.

Definition 1 *Abstract Firewall Topology (FT)*

The abstract firewall topology is a *labeled* and *undirected* graph *(V, E)* where a vertex in *V* represents an IP address of a firewall interface, and an edge between two vertices represents a communication link between two firewall interfaces.

In order to obtain a firewall topology from a network topology, we need to eliminate a network (LAN) which does not have firewalls and locates behind a firewall and to eliminate a network which does not have firewalls and locates between any two firewalls. An example of network between two firewalls is a DMZ.

Definition 2 *Relationship between Network Topology and Firewall Topology*

Given network topology *NT* and firewall topology *FT*, *NT* contains a subgraph which is *homeomorphic* to *FT*, and both *NT* and *FT* have the same set of firewall vertices.

It is important to note that we use *homeomorphic* (in particular elementary division) to remove networks between two firewalls and we use sub-graph to remove networks behind firewalls.

Figure 2 shows an abstract firewall topology. It is easy to see that graph in figure 1 contains a sub-graph which is homeomorphic to graph in figure 2. The rest of examples in this paper use network topology in figure 1 and firewall topology in figure 2.

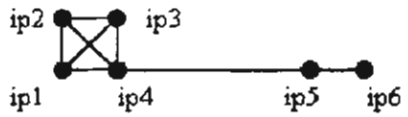

Figure 2. Abstract firewall topology

3.2 Graph-based Interpretation of Configurations

From modified graph-based model, we use firewall topology (*FT*) to generate actual effects of firewall configurations. In particular, the effect of a firewall rule is described as a set of paths called a local set of paths (*LP*). Furthermore, *a local set of effective paths (LEP)* and *a global set of effective paths (GEP)* still the same.

Those paths generated from *FT* express *actual* effects of firewall configurations but those paths may not exist in network topology. Even though those paths do not exist in *NT*, they must be normal form of path [14].

The following is definition of new *LP* that based on firewall topology

Definition 3 *A generalized firewall rule as a local set of paths (LP)*

Given a firewall topology *FT*, a *local* set of paths, called *LP*, which corresponds to a generalized firewall rule GR_{FWI} *=(Source, Destination, Service, Direction, Action, FWI)* defined for *FWI* interface is a set of all paths in FT which

a) begin with *Source* vertex,

b) end at *Destination* vertex,

c) pass through *FWI* vertices in the specified *Direction*, and

d) are labeled with *Service*

Example 1 *Local Set of Path*

The *local set of paths* that corresponds to rule *(ip9, ip7, FTP, outbound, permit, ip2)* is*{<ip9, ip6, ip5, ip4, ip2, ip7>, <ip9, ip5, ip4, ip2, ip7>,<ip9, ip3, ip2, ip7> and<ip9, ip1, ip2, ip7>}*. Note that the *local set of path* show all possible paths that were permitted by that rule that is *<ip9, ip6, ip5, ip4, ip2, ip7>* which is the normal packet flows from *ip9* to *ip7* and *<ip9, ip5, ip4, ip2, ip7>,<ip9, ip3, ip2, ip7>, <ip9, ip1, ip2, ip7>* is the spoofed path.

4. IP Spoofing Attack

We classify IP spoofing attack into three main kinds based on the location of attacker and victim.

1) Incoming attack is an attack from external network to internal network. So attacker is external address and victim is internal address.
2) Outgoing attack is an attack from internal address to internal network. So attacker is internal address and victim is external address.
3) Internal attack is an attack within internal network. So attacker and victim are internal addresses.

Example 2 *IP Spoofing*

ip11 (external host) impersonate *ip7* (host h1 – internal host) toward *ip9* (host h3 – internal host) is consider as *incoming attack*. While *ip8* (host h2 – internal host) impersonate *ip12* (external host) toward *ip11* (external host) is consider as *outgoing attack*. Moreover, *ip10* (host h4 – internal host) impersonate *ip8* (host h2 – internal host) toward *ip7* (host h1 – internal host) is consider as *internal attack*.

The following shows our definition to verify the vulnerability of firewall configurations for IP spoofing attacks. The definition can be used to detect the three kinds of attacks. To detect whether an attacker can carry out IP spoofing attack to a victim, our approach is to generate a path from the attacker to the victim. Then, we analyze on how many firewalls along the path allow the attacker to

impersonate another entity and to send a packet to the victim. If there exist some firewall(s) allowing that, local IP spoofing occurs. However, if all firewalls along the path allow that, global IP spoofing occurs. Thus, our approach uses *paths* as a main concept to detect IP spoofing attack.

Definition 4 *Graph-based IP spoofing Vulnerability*

Given an abstract configuration, an active entity A can impersonate (local and global) an entity (B) towards another entity (C) where $A \neq B$ (or A and B do not belong to the same object) and $A \neq C$ (or A and C do not belong to the same object) iff

1) There is a non-empty set of paths, called SP, from A to C in network topology NT, ie. $SP \subseteq AP_{NT}$, and
2) Consider the following kinds of spoofing:
   2.1) For *local* IP spoofing,
   $\forall P \in SP \ \exists FI \in FW\_IP$ (*P visits firewall interface FI* $\wedge$ *there is a path from B to C in LEP at FI*)
   2.2) For *global* IP spoofing,
   $\forall P \in SP \ \forall FI \in FW\_IP$ (*P visits firewall interface FI* $\rightarrow$ *there is a path from B to C in LEP at FI*)

where LEP is generated from the abstract configuration, and AP_{NT} stands for the set of all ordinary paths in a network topology.

Intuitively, this definition means that A can impersonate B towards C if and only if A can send a packet to C according to the network topology, and firewall(s) along the path from A to C allow(s) B to send the packet to C. Thus, A can send a packet to C by impersonating B.

Global and *local* IP spoofing show different degrees of vulnerability of configurations. *Global* IP spoofing represents the real vulnerability in that a firewall configuration for the whole firewall system allows IP spoofing attack, and *local* IP spoofing represents the potential vulnerability in that at least a firewall configuration at a firewall interface allows IP spoofing attack.

Example 3 *Graph-based IP spoofing Vulnerability*

Suppose that a configuration generates LEP_{ip2} and LEP_{ip3} both of which contain path <*ip9, ip3, ip2, ip7*>. The configuration is vulnerable to *global IP spoofing* because host *h2* (*ip8*) can impersonate host h3 (*ip9*) toward host h1 (*ip7*). The reason for this is that there is a set of path P from h2 (*ip8*) to h1 (*ip7*) that is {<*ip8, ip3, ip2, ip7*>} and all firewalls in the path (i.e. *ip2* and *ip3*) allow *ip8* to impersonate *ip9* to *ip7* due to the existence of path from *ip9* to *ip7* in LEP_{ip2} and LEP_{ip3}.

Suppose that in another configuration, only LEP_{ip3} (not LEP_{ip2}) contains path <*ip9, ip3, ip2, ip7*>. The configuration is vulnerable to *local IP spoofing*. Note that only firewall *ip3* allows *ip8* to impersonate *ip9* to victim *ip7*, but firewall *ip2* does not.

5. A Verification Methodology of Configurations for IP spoofing Attack

We propose a verification methodology of configurations for IP spoofing attacks. In particular, we propose a class of configurations and show that such classes are free of IP spoofing attack. Thus, to verify that a configuration is free of IP spoofing is to show that the configuration is in the classes.

5.1 Properties of IP Spoofing free Configurations

Before we discuss classes of IP spoofing free configurations, we discuss desirable properties of IP spoofing free configurations.

Definition 5 *Properties of IP spoofing free configuration*

1) An abstract configuration can prevent an incoming attack if and only if it prohibits any incoming packets with *internal* source address from the external network to the internal network.
2) An abstract configuration can prevent an outgoing attack if and only if it prohibits any internal host to send packets with source address that does not belong to itself to the external network.

It follows that a configuration that can prevent an outgoing attack can also prevent internal attack since it prohibits any internal host to attack any victim including both external host and internal host.

5.2 Classes of IP Spoofing free Configurations

We propose two kinds of configurations that are free of IP spoofing attacks, namely *entry-point filer* and *successive filter*.

Entry-point filter means the filtering of permitted flows passing through an organization network *at the entry-point* of the organization network. The entry-point of the network for a flow stands for a firewall interface which is *connected without any firewall* to the entity initiating the flow. Such filtering at an entry point must allow only entities which are *connected without any firewall* to the entry point to send packets through the entry point to anywhere. Note that entry-point filter concerns with the filtering at an entry point only, not at non-entry point nodes.

Definition 6 *entry-point filter*

Given an abstract configuration, a set of *SGR* is called *entry-point filter* if and only if
for every *entry point firewall interface i*,

$\forall(ip_0, ip_1, \ldots, ip_n) \in LEP_i, ip_0 \subseteq$ *non-firewall-connected-to(i)*

where LEP_i generated from SGR_i and *i* is an entry point firewall interface iff $ip \in$ *non-firewall-connected-to(i)* and *ip* is an IP address of an active entity.

It should be noted that entry-point filter does not necessarily allow all entities which are *connected without any firewall* to the entry point to send packets through the entry point. Instead, it prohibits entities which are not *connected without any firewall* to the entry point to send packets through the entry point.

Since an *Entry-point filter* prohibits an active to send a packet with source address that does not belong to the active entity itself, it is not possible that the *active entity* can send any spoofed packets.

Example 4 *entry-point filter*

Suppose that *SGR1* = {*(ip10, ip8, ftp, inbound, permit, ip6)*}. *SGR1* is entry-point filter since LEP_{ip6} = *{(ip10, ip6, ip5, ip4, ip3, ip8)}* and *ip10* ∈ *non-firewall-connected-to(ip6)*.

Successive filter means the filtering at successive firewalls from the entry point to all firewall interfaces connected to the entry point. It ensures that if a flow is allowed to enter to the firewall topology, it reaches its destination. This is because successive filter ensures that any flows allowed to enter their entry point are allowed to enter also all firewall interfaces connected to the entry point.

Definition 7 *successive filter*

Given an abstract configuration, a set of SGR is called *successive filter* iff

1) $\forall F \in FW\_Object\ \forall j \in interfaces\text{-}of(F)$

$$[\forall i \underset{i \neq j \wedge i \in interfaces-of(F)}{\cup} LEP_i^{out} \supseteq LEP_j^{in}]$$

where LEP_i^{out} = *{p | p* ∈ LEP_i *and p is a path passing through interface i in outbound direction }*

LEP_i^{in} = *{p | p* ∈ LEP_i *and p is a path passing through interface i in inbound direction }, and*

2) $\forall F1, F2 \in FW\_Object\ \forall i \in interfaces\text{-}of(F1)\ \forall j \in interfaces\text{-}of(F2)$

[$F1 \neq F2$ *and j* ∈ *non-firewall-connected-to(i) and* $LEP_i^{in} \supseteq LEP_j^{out}$ *]*

(1) in this definition deals with a flow passing through a firewall. Intuitively, it states that any incoming packet entering to a firewall interface of a firewall is allowed to leave the firewall at another firewall interface. (2) deals with a packet flow from a firewall to another firewall. Intuitively, it states that any packet leaving a firewall from a firewall interface *f* is allowed to enter into another firewall at firewall interface which is *connected without any firewall* to *f*.

Example5 *successive filter*

Suppose that *SGR1* = *{(ip9, ip7, ftp, inbound, permit, ip4), (ip9, ip7, ftp, outbound, permit, ip2)}*. Clearly, *SGR1* is successive since $LEP_{ip2}^{out} \supseteq LEP_{ip4}^{in}$. Suppose that *SGR2* = *{(ip9, ip7, ftp, outbound, permit, ip5), (ip9, ip7, ftp, inbound, deny, ip4)}*. *SGR2* is not successive since *ip5* ∈ *non-firewall-connected-to(ip4)* but $LEP_{ip5}^{out} \not\supseteq LEP_{ip4}^{in}$.

The two propositions, *entry-point filter* and *successive filter*, lead to the following definition.

Definition 8 *A Verification Methodology of IP spoofing*

An abstract configuration is free of any kind of IP spoofing attacks if and only if the set of firewall rules in the configuration is both *entry-point* and *successive filters*.

It is important to note that in order to verify whether a configuration is free of any IP spoofing attack, we do not need to enumerate all possible attackers and victims. To verify an abstract configuration, our methodology simply checks whether any source and destination in LEP at entry-point firewall interfaces are in an allowed set constructed by *non-firewall-connected-to*. Furthermore, to verify successive filter it involves only firewall interfaces used in an organization. Those firewall interfaces are known and small.

Finally, both entry-point and successive filters can be used as a guidance to configure firewalls to avoid IP spoofing attacks.

6. Related Works

NetSTAT [5] is an intrusion detection system that can deal with internal IP spoofing attack. The difference between NetSTAT and our approach is that NetSTAT detects attacks only when they occur whereas our approach can detect attacks before they occur. More importantly, to detect spoofing attack, NetStat needs to enumerate each possible attacker and victim. In other words, NetStat may work only if the number of attackers and victims are small. This assumption is too restrictive for the Internet and attack situations nowadays.

Moreover, NetSTAT neither analyze the causes of the attack nor offer any guidance on how to avoid the attack whereas our approach provides both

explanation on causes of the attack and guidance on how to avoid the attack. Thus, our approach offers a better understanding on the attack Finally, our approach provides an assurance that network configuration (firewalls in particular) is free of all cases of IP spoofing attack whereas NetSTAT cannot.

Fang [3] is a software tool which aims to analyze firewall rules. Fang simulate attack scenarios cases by cases. In particular, it offers attack scenarios for specific attacker A, victim C and B, the entity that is impersonated. Similar to NetSTAT, in Fang it also assumes that the number of attackers and victims are known and small, which is too restrictive. Thus, it cannot provide assurance that any configuration is free of the attack for all cases. Furthermore, it offers the vulnerability analysis of configuration for global IP spoofing attack only. Moreover, it neither discusses the causes of configurations for the attacks nor provides any guidance to prevent the attack.

Ingress [12] and Egress filter [13] proposed filtering configurations that can prevent incoming and outgoing IP spoofing attacks, respectively. Both kinds of configurations are concerned with boundary routers only, and they use the classification *inside* and *outside* to detect spoofing attack. Ingress filter does not permit incoming packets with *inside* source address whereas Egress filter does not permit outgoing packets with *non-inside* source address. However, *inside* and *outside* cannot deal with internal attack for some network topologies, such as ring, since the distinction between such inside and outside is not clear. While their configurations deal only with incoming and outgoing attacks, our filters can deal with incoming, outgoing and internal attacks at once.

7. Conclusion

In this paper, we propose a verification methodology for analyzing the vulnerability of firewall configurations for IP spoofing attacks. Our methodology can be used not only to verify existing firewall configurations. Several kinds of IP spoofing attacks are discussed. The relationship between the attacks and firewall configurations is explained. Two degrees of the vulnerability for the attack are discussed and formalized. Our approach offers a proof-based assurance on the vulnerability of a firewall configuration for the attack. Furthermore, we propose a class of configurations which are free of the attacks. The class can be used as a guidance to configure firewalls securely.

Currently, we are implementing a software prototype for our method. As a future work, we aim to extend our method for analyzing other kinds of attacks in network.

Acknowledgement

The second author would like to acknowledge financial support from Thailand Research Fund and National Research Council of Thailand.

Reference

[1] Y. Permpoontanalarp and C. Rujimethabhas, A Graph Theoretic Model for Hardware-based Firewalls, *In proceedings of 9th IEEE International Conference on Networks (ICON)*, Thailand, IEEE Press, 2001.

[2] Y. Permpoontanalarp and C. Rujimethabhas, A Unified Methodology for Verification and Synthesis of Firewall Configurations, *In Proceedings of The Third International Conference on Information and Communications Security (ICICS)*, China, Lecture Notes in Computer Science, Springer Verlag, 2001.

[3] Alain Mayer, Avishai Wool and Elisha Ziskind, Fang A Firewall Analysis Engine, 21st IEEE Symposium on Security & Privacy, Oakland, CA, May 2000.

[4] J.D. Guttman, Filtering Postures : Local Enforcement for Global Policies, *In proceedings of 17th IEEE Symposium on Security & Privacy*, Oakland, CA, 1997.

[5] G. Vigna and R. Kemmerer, "NetSTAT: A Network-based Intrusion Detection System", Journal of Computer Security, 7(1), IOS Press, 1999.

[6] CERT® Advisory CA-1995-01 IP Spoofing Attacks and Hijacked Terminal Connections, CERT® Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks

[7] John D. Howard. "An Analysis of Security Incidents On The Internet 1989 – 1995", Carnegie Mellon University, 1997. http://www.cert.org/research/JHThesis/Start.html.

[8] P. Ferguson et al. RFC 2827. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. Technical report, IETF, February 2000. Available at http://sunsite.cnlab-witch.ch/ftp/doc/standard/rfc/28xx/2827.

[9] SANS Institute. Egress filtering v 0.2, 2000. Available at http://www.sans.org/y2k/egress.htm.

[10] W.R. Cheswick and S.M. Bellovin, *Firewalls and Internet Security : Repelling the Wily Hacker*, Addison-Wesley, 1994.

[11] D.B. Chapman and E.D. Zwicky, *Building Internet Firewall*, O'Reilly & Associates, 1995.

[12] P. Ferguson et al. RFC 2827. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. Technical report, IETF, February 2000. Available at http://sunsite.cnlab-switch.ch/ftp/doc/standard/rfc/28xx/2827.

[13] SANS Institute. Egress filtering v 0.2, 2000. Available at http://www.sans.org/y2k/egress.htm.

[14] Voravud Santiraveewan, 2003, A Graph-based Methodology for Analyzing IP Spoofing Attacks, Master Degree Project, Computer Engineering, Faculty of Engineering, King Mongkut's University of Technology Thonburi.