セキュアマルチパーティ計算に基づくプライバシー保護リスク分析手法

劉　瑜 †　　後藤　成聡 †　　金岡　晃 ‡　　岡本　栄司 †

† 筑波大学
305-8573 茨城県つくば市天王台 1-1-1
liuyu@cipher.risk.tsukuba.ac.jp
goto@cipher.risk.tsukuba.ac.jp
okamoto@risk.tsukuba.ac.jp

‡ 東邦大学
274-8510 千葉県船橋市三山 2-2-1
akira.kanaoka@is.sci.toho-u.ac.jp

あらまし ネットワークシステムへのリスク分析は、近年の攻撃動向によりますます重要性が増している。しかし、ネットワークシステムの構成も複雑化してきており、そのリスク解析は容易ではない。そういったリスク分析を専門家や業者に依頼することは 1 つの解決策である。しかし、リスク分析時に渡されるネットワーク構成や脆弱性情報を含む各ホストの情報は気密性が高く、保護されるべきである。本論文では、ルールベースで行われるリスク分析に対しプライバシ保護技術の適用を提案する。試作システムを、秘密計算を行う FariplayMP を用いて実現し、小規模ネットワーク環境で評価した。

Secure Multi-party Computation Based Privacy Preserved Risk Analysis

Yu Liu†　　Nasato Goto†　　Akira Kanaoka‡　　Eiji Okamoto†

†University of Tsukuba.
1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, JAPAN
‡Toho University
2-2-1, Miyama, Funabashi-shi, Chiba 274-8510, JAPAN

Abstract Risk analysis for networked system is becoming increasingly required by recent attack trends. Since networked system is also becoming increasingly complex by functions, achieving risk analysis for networked system is not an easy task. One reasonable solution is delegating risk analysis to professional third party. Highly confidential Requirement for risk analysis of networked systems is increasing because of recent network attack trends. However recent networked systems are very complex, so their risk analyses are not easy. One reasonable solution for this problem is delegating risk analysis job to professional third parties. In this case highly confidential data like network configurations and vulnerabilities in the network or hosts, should be protected. In this paper, risk analysis in privacy preserving way is proposed based on rule-based method. Prototype system is implemented experimentally using FairplayMP for secure multi-party computation and is evaluated in a small network environment.

1 Introduction

1.1 Background

Network system is becoming the core component of information technical infrastructures. Protection of these networks from malicious attacks is an urgent priority to our society.

However, dealing with vulnerabilities in the network has brought enormous challenges to the network management. Information sys-

tem also faces complex attackers who combine multiple vulnerabilities, multiple hosts, multi-stage to penetrate the network that may result in devastating impact. To more accurately evaluate the security of enterprise systems, understanding how potential attacks will be staged by using multiple vulnerabilities is significant. However, considering all of the security threats for system administrators is very complicated, easily-missed and error-prone, so one reasonable solution for risk analysis is delegating analysis of networked system to third parties who have more professional risk analysis knowledge. However, highly confidential data like network configuration and vulnerabilities in the network and each hosts, is needed when delegating risk analysis to third party. Without protection of system information, it will become a fatal threat to network system, because it will cause information leakage, other organizations may abuse system information.

1.2 Purpose

Our goal is to check and monitor network system safety automatically with the premise that confidential information is not known by risk analysis system. In this paper, we proposed a risk analysis system that receives the classified confidential information instead receiving the unprotected data, and analyzes classified value, finally sends the result to the system information provider. Thus, automatic risk analysis will be realized under the condition that confidential network system information is protected.

2 Related Works

2.1 Automatic Attack Graph Generation Tool

In this research, risk analysis is considered to use Attack Graph to find the potential threats, and make clear how those potential threats would be staged in a network system based on its configuration and topology.

Attack graph [1] is first proposed by Laura Painton Swiler and Cynthia Phillips in 1998, it represents system states using a collection of basic network conditions, such as vulnerability, network service, the connectivity between different hosts, etc. Vulnerability exploitation is modeled as system states transitions.

One of the logic-based Automatic Attack Graph Generation Tool is called MulVAL [2]. In MulVAL, algorithm that has $O(N^2)$ complexity, and the experimental result shows that running time for worst case is between $O(N^2)$ and $O(N^3)$ [3], where N is number of hosts.

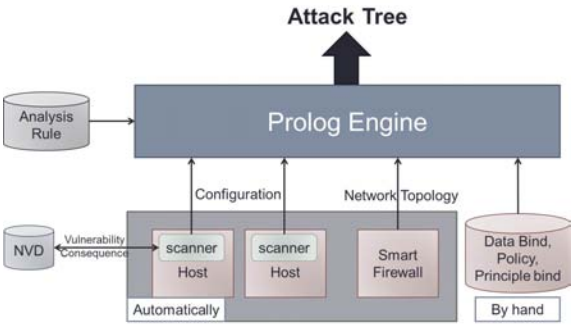

Figure 1: MulVAL Framework

The framework is shown in Figure 1 . An OVAL scanner [4] installed on each host machine, outputs OVAL report about vulnerability and relevant host configuration with the collaboration of NVD [5]; Smart firewall is able to analysis network topology automatically; Data binding information that maps a data to a path on a machine, and principle binding information that maps a principal to its user accounts on hosts and security policy that specifies which principal can access what data are input by network system administrator. Prolog engine analyzes input facts with analysis rules, finally outputs attack tree.

2.1.1 Input of MulVAL

The input is like:
A) Vulnerabilities in a machine,
B) Software and service in a host,
C) Configuration of routers and firewall,
D) Authority of principle,
E) Policy determined by administrator,
F) Data binding by administrator.

Among of above, A, B and C could be automatically collected by scanner and smart fire-

wall report. The rest of them are input by system administrator. Inputs are input as the format of fact, by which we can make some statements in Prolog language. For example, $networkServiceInfo(webServer, httpd, TCP, 80, apache)$ means that program *httpd* runs on machine *webServer* as user *apache*, and port is 80 by TCP protocol.

2.1.2 Automatic Attack Graph Generation Tool

The reasoning rules in MulVAL are declared as Prolog clauses. A Horn clause [6] in MulVAL: $l_0 : -l_1, l_2, ..., l_n$ which means if $l_1, l_2, ..., l_n$ are true, then l_0 is true.

Multi-stage, multi-host attacks are encoded as Horn clauses, where the first line is possible attack state, and the remaining lines are the conditions. If all of the conditions are hold, the state of the first line is probable to be achieved. For example:

$$
\begin{aligned}
&execCode(Host, Privilege) :- \\
&\quad -vulExists(Host, VulID, Software, \\
&\qquad remoteExploit, privEscalation), \\
&\quad -networkServiceInfo(Host, Software, \\
&\qquad Protocol, Port, Privilege), \\
&\quad -netAccess(Host, Protocol, Port)),
\end{aligned}
$$

This interaction rule means that if a program is running in a host with vulnerability whose impact is *remoteexploitable* range and *privilegeescalation* and this program is listening on *protocol* and *port*. Then execute arbitrary code attack would be staged in an accessible network with the privilege. All of the parameters in the same color must be equal to each other to make the rule become true. There are 24 kinds of rules [7] in the MulVAL in order to simulate various attack types.

2.1.3 Evaluation Strategies

Logic programing language will do query automatically if it is provided with facts. In another word, thanks to Prolog, MulVAL will analyze network risk automatically. MulVAL has two basic evaluation strategies: $bottom - up$ and $top - down$.

In the $bottom - up$ evaluation, the interaction rule is tried if the input facts match the rules, if so, more new facts would be derived by rules, until no new fact would be derived. In the $top - down$ evaluation, an attack goal is given, and sub-goals are found by using backward rules, until the all of sub-goals hit the input facts.

Output from MulVAL is attack tree, which represents the attack steps and states of potential attacks. The tree leaves are the primitive facts, that is, system input information. The internal nodes, which are tree nodes except leaves are derived by attack rules with primitive facts. Each internal node shows the state that attacker would reach by launching multistage, multi-host attack. Node of attack tree is attacker's attack goal, each path in the tree stands for a multistage attack path.

2.2 Privacy Preservation

The confidential data in this research is the input data in section 2.1.1. None input, intermediate, or output should leak out the input data.

2.2.1 Secure Multi-party Computation

The goal of secure multi-party computation (SMC) is to compute a function with multiparties and got the final result without revealing any input of the computing parties. For example, using SMC, two millionaires can compute which one is richer, but without revealing their net worth. This example was initially suggested by Andrew C. Yao [8], named as *millionaireproblem*.

2.2.2 Adversary Model in SMC

The SMC literature gives two basic adversarial models: Semi-honest model and malicious model.

$Semi - honest$ Model: all computing parties follow the protocol; but dishonest parties could learn everything while following the protocol.

$Malicious$ Model: dishonest parties can do anything they want to violate the privacy.

Obviously, malicious model is harder to handle. Almost of current SMC protocol is oriented for $semi-honest$ model.

2.2.3 FairplayMP

FairplayMP [9] is a kind of system for secure multi-party computation. The current version of FairplayMP handles only the $semi-honest$ model. It provides a C-liked programming language called SFDL[10], and a configuration file describing the participating parties IP address. The compiler translates SFDL program to a Boolean Circuit. Cryptographic engine executes a protocol which is based on a protocol by Beaver, Micali and Rogoway (BMR protocol)[11], computes a circuit securely. FairplayMP allows user to separate input, computation, result party. Architecture of FairplayMP is presented as Figure 2.

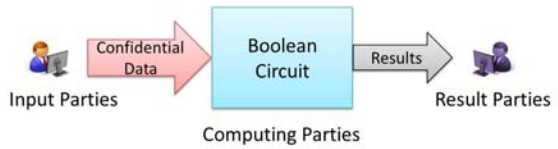

Figure 2: FairplayMP

2.2.4 SMC Protocol

Secure Multi-party Computation has various kinds of computation operation protocol, such as, addition, multiplication, equality, greater-than, division, etc.

In our research, we use SMC *equality* operation to replace matching value operation in prolog.

3 Privacy Preserved Risk Analysis Proposal

3.1 Applying SMC to Risk Analysis

In this section, we propose 3 privacy preserved risk analysis patterns based on secure multi-party computation. We give 3 figures to show our proposed system. The red parts and red arrows means the parts where need privacy preservation in each figure.

1) Only information related to network system, such as host configuration, host access list, policy, etc. is privacy preserved. The analysis rules are provided by third party who has professional risk analysis knowledge base. In this proposal, risk analysis provider also provides SMC engine to do analysis, as Figure 3 depicts.

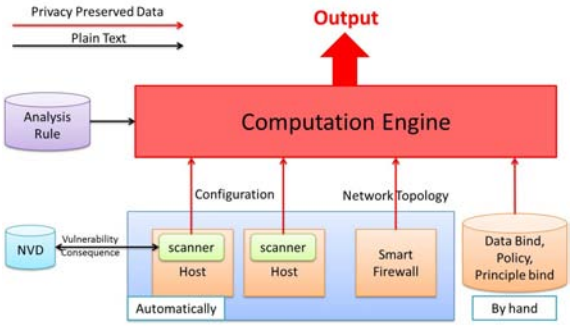

Figure 3: Privacy Preservation Pattern 1

2) In this propose, only analysis rule is privacy. Risk analysis rules are sent to user who has network system information and needs to do risk analysis. Different with pattern 1, in this proposal, system information holder provides SMC engine to do analysis, as Figure 4 depicts.

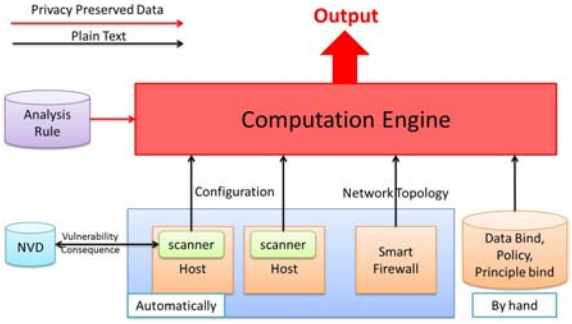

Figure 4: Privacy Preservation Pattern 2

3) Different with pattern 1 and 2, we separate SMC engine from risk analysis knowledge provider or network system information provider. No only analysis rule but also system related information is privacy preserved. Both of them send data to SMC engine, as Figure 5 depicts.

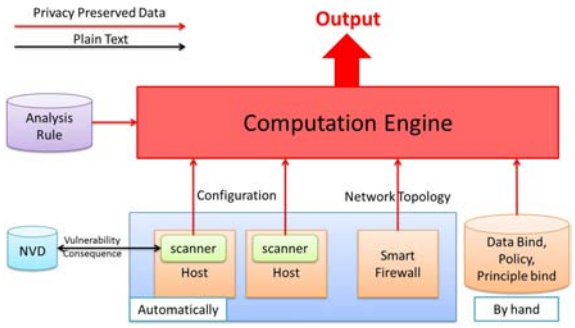

Figure 5: Privacy Preservation Pattern 3

4 Prototype System Development

4.1 Architecture of prototype system

We implemented pattern 1 proposed in section 3.1. As Figure 6 shows, there is only one input party in our system. In our system, analysis rule is not private. Input party collects all network system configurations, topology, binding information, policy and sends it to computation parties. Meanwhile, make sure that computation engine not know or derive the input value after it gets the information from input party is important. Computation parties send result to input&result party. Output is also should be privacy preserved, because output will cause privacy leakage or output is also very confidential to system information provider. Unconditional security against a semi-honest model can be achieved with less than $n/2$ of n parties are corrupted [12]. In 3 computation parties (CP) case, corruption in at most one computing party is tolerated.

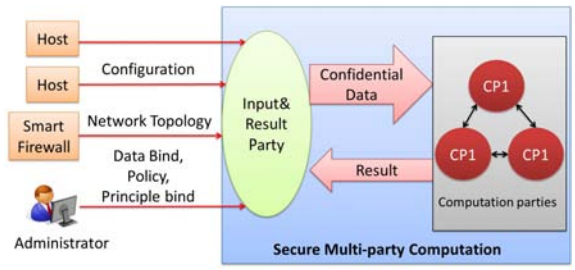

Figure 6: System Architecture

4.2 Function of Prototype System

In this system, user is allowed to ask question about the potentiality of an attack. The result is "true" or "false" computed by computation parties to user's query. "true" means it is possible for an attacker to conduct such kind of attack with given networked system environment information.

4.3 Avoid Infinite Loop

One of the important features of MulVAL is the ability to reason about multi-stage attacks. Cycles are common when it comes to attack graph. For example, an attacker can modify a user's files if he can execute arbitrary code as the user. But it is also possible that he can execute arbitrary code as the user, if he can modify some executables and install a Trojan-horse program in host. These two rules call each other, unfortunately, it will result in infinite loops.

$/* Rule : execCodeimpliesfileaccess */$
accessFile$(H, Access, Path) :-$
  execCode(H, Usr),
  $canAccessFile(H, Usr, Access, Path)$.

$/* Rule : Trojanhorseinstallation */$
execCode$(H, root) :-$
  accessFile$(H, write, _P ath)$.

Prolog is not able to deal infinite loop caused by rules, other than prolog, MulVAL use XSB [13], which supports tabling [14] execution of Prolog to avoid infinite loop. Tabling is a memorial technique. Computation is conducted only once, the computation result is recorded in table for reuse. Therefore, recomputaion and infinite loop could be avoided with high efficiency.

We rewrite MulVAL in SFDL language, FairplayMP, a kind of SMC language. The analysis flow is top-down. We set an attack goal, and check whether this goal could be reachable.

Due to the limitation of SFDL language, recursive function is not supported. Different with tabling execution, we set an upper bound

for rule calls. Namely, each rule is only called within several times, the derivation step must be less than a maximum number, otherwise we regard it as false even if it would become true if we do more derivation. It ensures that the number of attack states is finite. In the experiment, we set the maximum number as 3.

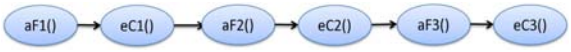

Figure 7: Loop with 3-derivation

Figure 7 depicts loop caused by interaction rule, where "aF" stands for "$accessFile$" and "eC" stand for "$execCode$", $aF1$ calls $eC1$, $eC1$ calls $aF2$, ..., until $eC3$ is called. We stop derivation at 3 that is set as maximum number for function call.

4.4 Target Network System

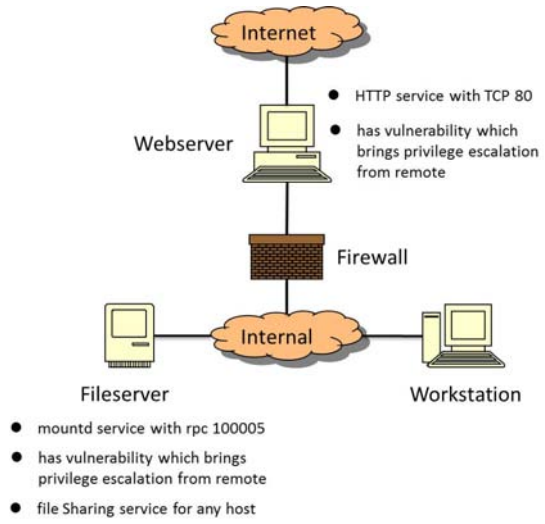

Figure 8: An Example Network

Our experimentation was done using the sample input provided by MulVAL project. Network system is shown as Figure 8, there are 3 hosts in this network system.

Topology information:

- Only webserver could be accessed by external internet. Hosts in network system are allowed to access any host.

Vulnerabilities information:

- Vulnerabilities are found in webserver and in fileserver, whose impacts are remote exploitable range and privilege escalation.

Machine configuration:

- httpd is running in webserver listening on tcp protocol and port 80.
- mountd is running in fileserver listening on protocol rpc on port 10005.
- Fileserver is allow to read/write the file under '/export which is export to workstation or webserver.
- Directory '/export' on fileserver is mounted on path '/usr/local/share' of workstation as an NFS partition.
- fileserver has root privilege to write the file under '/export'.

4.5 Used Rules and an Attack

Below 24 items are used as analysis rules. They are same as MulVAL interaction rules:

1. Insider threat
2. When a principal is compromised any machine he has an account on will also be compromised
3. local exploit
4. remote exploit of a server program
5. remote exploit for a client program
6. Trojan horse installation
7. multi-hop access
8. direct network access
9. direct on-host access
10. Access a host through executing code on the machine
11. Access a host through a log-in service
12. execCode implies file access
13. password sniffing
14. login service through sshd
15. login service through vpn

16. Principal P can access files on a NFS server if the files on the server are mounted at a client and he can access the files on the client side
17. Principal P can access files on a NFS client if the files on the server are mounted at the client and he can access the files on the server side
18. NFS shell
19. Bug description
20. Hypothetical bug description
21. Library bug description
22. Browsing a malicious website
23. Browsing a malicious website
24. Browsing a compromised website

In our experiment, network system background information will result in an attack. Because of the vulnerability, attacker can compromise the webserver to get control of it (rule 4); webserver is allowed to access fileserver, and NFS directory is exported to webserver, so attacker can write files on fileserver after compromising webserver (rule 18); the files of workstation is mounted on fileserver, attacker can install Trojan horse to get the root user on workstation (rule 6).

4.6 System Environment

Real Machine for Input Party & Result Party

- CPU: Intel Core i7 2.20GHz
- Memory: 4G
- Operating System: windows 7 (64 bit)
- Java Version : 1.7.0_45
- Ruby Version : 1.9

Real Machine for 3 Computation parties

- CPU: Intel Core i7 2.67GHz
- Memory: 6G
- Hypervisor: ESXi 5.5.0
- vSphere Client: 5.5.0

Virtual Machine Information

- Operating System: Ubuntu 14.04.1 (64 bit)
- Memory: 3GB
- Java Version : 1.7.0_65
- Ruby Version : 1.9

Execution Runtime Environment

- Runtime Environment: FairplayMP
- Programing Language: SFDL

Packet Analyzer

- Analyzer : Wireshark 1.6.5

In our experimentation, three computation parties are run on virtual machine, and one input party is run on real machine. Input party play a role of result party as well.

4.7 Performance

We set that attacker gets root privilege on workstation in target system (Section 4.4) as attack goal; privacy preserved risk analysis system output "true". Our system found an attack successfully.

Time measurement is start from input data input by input party, and stop when input & result party receives the result from computation parties. We used Wireshark to capture the packets that related to secure multi-party computation to get beginning time and ending time. The time interval is the computation time which contains time of data transmission between parties and secure multi-party computation. Experimentation shows that it spent 1.4754s to analysis attack goal in target system under secure multi-party computation.

5 Future Work

Our proposed system is the first step to do privacy preserved risk analysis. There are several remain problem, like computation party corruption.

Based on unconditional security result in [12], Input information would be safe if at most one computation party is corrupted when there are only 3 computation parties. How to prevent computation party corruption is also significant to input party.

We only implemented privacy preserved risk analysis pattern 1 in this paper. Implementations for pattern 2 and 3 are greater challenges.

6 Conclusion

We propose privacy preserved network system rule-based risk analysis system based on secure multi-party computation in this paper. 3 patterns of privacy preservation risk analysis method are presented in section 3.1. We implemented first pattern by using FairplayMP to protect network system information. Our system separates input&result and computation. Input information would be safe if at most one computation party is corrupted. The performance looks good when network size is small.

References

[1] Phillips, Cynthia, and Laura Painton Swiler. "A graph-based system for network-vulnerability analysis." *Proceedings of the 1998 workshop on New security paradigms.* ACM, 1998.

[2] Ou, Xinming, Sudhakar Govindavajhala, and Andrew W. Appel. "MulVAL: A Logic-based Network Security Analyzer." *USENIX Security.* 2005.

[3] Ou, Xinming, Wayne F. Boyer, and Miles A. McQueen. "A scalable approach to attack graph generation." *Proceedings of the 13th ACM conference on Computer and communications security.* ACM, 2006.

[4] OVAL, `http://nvd.nist.gov/` (Aug. 25, 2014)

[5] NVD, `http://nvd.nist.gov` (Aug. 25, 2014)

[6] Horn, Alfred. "On sentences which are true of direct unions of algebras." *The Journal of Symbolic Logic* 16.01 (1951): 14-21.

[7] Ou, Xinming, and Andrew W. Appel. *logic-programming approach to network security analysis.* Princeton: Princeton University, 2005.

[8] Yao, Andrew C. "Protocols for secure computations." 2013 IEEE 54th *Annual Symposium on Foundations of Computer Science.* IEEE, 1982.

[9] FairplayMP,`http://www.cs.huji.ac.il/project/Fairplay`, (Aug. 25, 2014)

[10] Ben-David, Assaf, Noam Nisan, and Benny Pinkas. "FairplayMP: a system for secure multi-party computation." *Proceedings of the 15th ACM conference on Computer and communications security.* ACM, 2008.

[11] Beaver, Donald, Silvio Micali, and Phillip Rogaway. "The round complexity of secure protocols." *Proceedings of the twenty-second annual ACM symposium on Theory of computing.* ACM, 1990.

[12] Chaum, David, Claude Crpeau, and Ivan Damgard. "Multiparty unconditionally secure protocols." *Proceedings of the twentieth annual ACM symposium on Theory of computing.* ACM, 1988.

[13] XSB, `http://xsb.sourceforge.net`, (Aug. 25, 2014)

[14] D. S. Warren. *Programming in Tabled Prolog.* Department of Computer Science SUNY @ Stony Brook, July 1999.