Journal of Computer Security 00 (2004) 1–15
IOS Press

Research on automated rollbackability of intrusion response

Jian Zhang, Jian Gong and Yong Ding
Department of Computer Science and Technology, Southeast University, Nanjing 210096, China
E-mail: {zhangj,jgong,yding}@njnet.edu.cn

The rollbackable automated intrusion response mechanism, a method whereby an intrusion response can be treated by in the context of the detection/response life-cycle. The idea derives from the observation that most intrusion responses have negative effects. To decrease the cumulative response cost, response rollback could be carried out at some suitable time, for example when the attack has terminated or the attack 'detection' is proved to be a false positive. Additionally, technologies supporting automated response are proposed, such as the structure of a response policy and the way the automated response might be implemented. A proposed implementation structure of rollbackable automated intrusion response system (RAIRS) is also given. With the quantified response cost, the result of our experiments show that response rollback is promising as a way to decrease the expected cumulative intrusion response cost.

Keywords: Intrusion detection system, intrusion response system, response rollback, atomic response

1. Introduction

IDS (Intrusion Detection System) is intended to monitor computer networks and systems so as to discover any violation of security policy. The main function of early IDS ranged from detecting and analyzing user behavior in the network, recognizing known attacks, making statistical analysis of abnormal behavior, to inspecting system configuration for vulnerabilities. Increasingly, the purpose of an IDS includes responding to intrusions as well as identifying them. Intrusion response is the countermeasure taken against intrusion events, whose goal is to restrain, assess and recover from the damage of intrusions. An IDS without suitable response is a passive one which is not enough in practice for the purpose of enhancing system security.

According to their different mechanisms, response systems can be divided into three classes [1], i.e., notification systems, manual response systems, and automated response systems. A *notification system* can only generate intrusion reports and alarms, while further intrusion response is left as the responsibility of the security administrator. This kind of system potentially has a long delay between notification and response, and may even overwhelm security administrators if a large number of notifications are received. *Manual response systems* can provide security administrators with additional help besides the functionalities of the notification system, such

as enumerating possible responses of current events for the security administrator to make decisions. This mechanism for response still possesses the shortcoming of notification system. An *automated response system* is able to make response decisions and take action, according to the current security status automatically without the intervention of the security administrator. Notification and alarm subsystems are a necessary but not sufficient component of any intrusion response systems. Similarly, manual response systems are necessary but not sufficient given the current threats to computer systems. Automatic intrusion response systems are in their infancy but appear to provide a decided advantage in protecting against certain types of attacks. All of the most recent intrusion response systems feature some form of automatic intrusion response.

Curtis and Pooch [1] proposed a classification of intrusion response in order to improve the functionality of automated response. The authors hold that appropriate responses should be selected according to the time of intrusion, the type of attack, the type of attacker, the strength of suspicion, the implication of attack and the environmental constraints. The response timing may be defined as preemptive, concurrent with an attack, or after an attack. The implication of attack refers to the criticality of the target system, and the environmental constraints include legal, ethical, and resource limitations. Based on this classification of intrusion responses, Curtis then put forward a system architecture for automated intrusion response system AAIRS. Geib and Goldman [2] proposed to apply plan recognition in intrusion response system, that is, inferring the next step or plan of the attacker based on his current behavior to make preemptive response. In the DARPA project of "Automated Intrusion Traceback and Response", Schnackenberg et al. [3] examined the integration of intrusion detection systems, firewalls, routers and hosts in order to construct an automated defense system in the Intranet environment of enterprises. The key of this project is to devise an intrusion detection and isolation protocol (IDIP) through which all system components cooperate together to detect intrusions, exchange intrusion events, and dynamically reconfigure firewalls, routers and hosts. These systems only respond to the existence of intrusion, but do nothing to address "nonexistence of intrusion" which means false positive or intrusion termination. That will lead to the following problems:

(1) When false positive is proved, the response system cannot always withdraw or halt the response to this event.
(2) When an intrusion stops, the response system lacks a mechanism to withdraw the response to the intrusion automatically.
(3) Because of the resource limitation, most IDS can not keep the status of attacks, so that there is no way for them to realize the termination of an attack.

Therefore, the unknown of attack termination will unnecessarily bring about negative effects on the systems being protected. For example, users may not be able to access a service, because it has been shut off by the response system, despite the termination of the attack.

This paper puts forward a Response Rollback mechanism and a Rollbackable Automated Intrusion Response System (RAIRS). The response rollback mechanism automatically decides whether the response measure should be rolled back, then generates the response rollback command, and translates the command into a script, being executed to withdraw the response measure. Thus, RAIRS can handle the false alert caused by false positive and roll back any unnecessary response measures. This feature can effectively overcome the shortcoming of traditional response system.

Section 2 discusses the theory of response rollback. Section 3 introduces the rollbackable automatic intrusion response system. The corresponding experimental results are shown in Section 4, and conclusion is presented at last.

2. Response rollback

The main idea of response rollback is to retract responses which are unnecessary or even cause negative effects. In the most common cases, the response system should withdraw previous response measures when the events detected by IDS prove to be false positive or the intrusion behavior terminates. Otherwise, the response will probably only have negative effects on the system being protected. The discussion of response rollback involves two key issues:

(1) The formal definition of response and response rollback. Since response rollback is based on response itself, it cannot be implemented without integral and precise description of response. Furthermore, in automated response systems, it is required to automatically transform response into response rollback, which makes the formal definition necessary.
(2) The detection of intrusion termination and false positive shown as false alert. Without these capabilities, response rollback is impossible.

The following discussion is focused on the above two points.

2.1. The formal definition of response rollback

Definition 1. Response Subject (RS) is the basic object which is accessed by a response. RS has the following form:

$$RS := \text{<}Domain\text{>}[!\text{<}Sub\text{-}domain1\text{>}[!\text{<}Sub\text{-}domain2\text{>}\ldots]]$$

where *domain* and *Sub-domain* are location symbols of RS, *Domain* $\supset$ *Sub-domain1* $\supset$ *Sub-domain2* $\supset \ldots$, "!" is the separator between domains.

For example, the access list (AL) in router 192.168.2.1 is described as 192.168.2.1!*AL*.

Definition 2. Atomic Operator is the basic and independent action which is exerted on RS. Atomic Operators can be divided into two classes. In one class, each operator has a corresponding operator which can exert the reverse effect. This kind of operator is called an Invertible Atomic Operator. Its corresponding operator is called the Inverse Atomic Operator of the invertible atomic operator. We indicate the invertible atomic operator as o_r, and its inverse atomic operator set as $\neg\ o_r$. In the second class, each operator does not have an inverse operator. This kind of operator is called Uninvertible Atomic Operator. We note the uninvertible atomic operator as o_n, and its inverse operator set $\neg\ o_n = \phi$, where ϕ refers to empty set.

For example, generally, *add* and *delete* are invertible atomic operators, while *send* and *kill* are uninvertible atomic operators. However, if the program has no side-effect, {*kill, restart*} is still an inverse atomic operator set.

In practice, to determine the invertability of an atomic operator and its inverse atomic operators, two more issues must be addressed:

(1) The semantics of an atomic operator
Generally, if an RS can't restore to the status after an atomic operator A is exerted to it by any atomic operator, $\neg\ A = \phi$, in which the status can be defined according practical requirement.
(2) The capability of IRS.
It means that the capability of IRS can decide the invertability of an atomic operator. For example the semantics of atomic operator *alert* is to send alert to the console or the security administrator. If the IRS can send an alert withdraw, *alert* can has the inverse atomic operator *alert-withdraw*.

Table 1 shows the commonly used atomic operators and their inverse operators under some IRS environment:

Definition 3. Operand *Ou* is the content of the action exerted on RS. An empty operand is noted as null.

For example, in the response measure of adding a rule to the access list of a router, the rule is operand. For example, *Ou*="access-list listnumber deny sourceaddr".

Definition 4. An Atomic Response is a ternary with the following form:

$$UR = \text{<}Op,RS,Ou\text{>}$$

where *Op* is the atomic operator, *RS* is the response subject, and *Ou* is the operand. An atomic response is a basic response measure, and an inverse atomic response is

Table 1
Commonly used atomic operators and their inverse atomic operators

	Description	Inverse Atomic Operator Set	Illustration
Add	Add an operand to RS	{remove}	Block the source IP of attacker (in the router)
Create	Create a RS	{delete}	Create system backups
Run	Execute programs	{shutdown}	Run other detection tools
Hangup	Suspend a specific process	{resume}	Hangup the attacked service
Send	Send information to RS	ϕ	Alarm to security administrators
Enable	Enable parameters in RS	{disable}	Enable extra logs
Lock	Block RS	{unlock}	Block an user account
Shutdown	Shut down RS	{startup}	Shut down the host
Kill	Terminate running programs	{restart}	Interrupt a session

the one which has the reverse effect. We note atomic response as *UR* and its inverse response as $\neg$ *UR*, where $\neg$ *UR* = <$\neg$ *Op,RS,Ou*>. If $\neg$ *Op* = ϕ, then $\neg$ *UR* = ϕ, and it is called the empty response.

Definition 5. A response $\boldsymbol{R}$ is a sequence of atomic responses with the following form:

$$\boldsymbol{R} = <UR_1, UR_2, \ldots, UR_n>, n = 0, 1, 2, \ldots$$

An inverse response is the response which has the reverse effect. Note the inverse response of $\boldsymbol{R}$ as $\neg \boldsymbol{R}$, then $\neg \boldsymbol{R} = < = \neg UR_n, \neg UR_{n-1}, \ldots, \neg UR_1>$.

The value of this definition of response lies in the notion that every response can be decomposed into a sequence of atomic responses, so the response rollback is implemented by taking the inverse atomic response from the last step of the sequence. If some inverse atomic response set is ϕ, then it is not executed.

2.2. The detection of false positive

False alerts could be detected automatically, or by human intervention.

Zhang et al. [4] puts forward a novel IDS. Its detection module does not comply with traditional inference rules of prepositional logic, but is based on the inference mechanism of non-monotonic logic. This inference mechanism is capable of inferring security results with some certain confidence under insufficient facts. It not only improves its detection sensitivity and capability of analyzing large-scale data, but also can discover false positives, which leads to the correction of the previous alert and the rollback of previous responses.

To implement this model, intrusion plan recognition could be used. As more facts are collected, the plan recognition conclusion may change. Identification of such changes may lead to discovery that a previously alert was a false alert. This means a false positive was issued and the associated response is unnecessary, and thus response rollback should be taken.

Security administrator might also discover an alert to have been a false positive. In this circumstance security administrator can invoke response rollback as appropriate.

2.3. The automatic detection of the termination of intrusion

An Intrusion Session is composed of compound intrusion actions which is intended to achieve the ultimate goals of the intrusion. The Intrusion Session can be described as a sequence of intrusion events among which every event is carried out for a sub-goal.

Generally, to detect the termination of a compound intrusion, events belonging to this compound intrusion need to be correlated and no following event should be observed. Therefore two problems need to be solved. The first one is how to judge whether one event is related to this compound intrusion, or more generally, if two events are related. The other is how to judge the termination of the compound intrusion. We now consider both of these issues.

An intrusion event can be described as a triple <Context,Prerequisite, Consequence>, where Context is the set of environment parameters when the intrusion happened. For example, Context might include Source IP Address, Destination IP Address and time. Prerequisite represents the prerequisite of the intrusion action, and Consequence represents the consequence of the intrusion action. Prerequisite and Consequence are described as the set of atomic logic expressions.

Definition 6. For intrusion event

I_1 = <Context_1,Prerequisite_1,Consequence_1> and

I_2 = <Context_2,Prerequisite_2,Consequence_2> if $\exists p \in$ Prerequisite_2, s.t.

$c_1 \wedge c_2 \ldots \wedge c_n \rightarrow_{FCFS} p$,

$\{c_1, c_2, \ldots, c_n\}$ = Consequence_1 and

Context_1 can be matched to Context_2, the relation of I_1 and I_2 is denoted as:

$I_1 R I_2$ or $I_1 I_2$

i.e., I_1 is related ot I_2. R represents "existing the causal relation".

To judge the termination of compound intrusion, we suppose:

Hypothesis 1. Intrusion events with causal relation are subjected to one compound intrusion.

For intrusion events A and B, if A R B, it means that "A happened" is the necessary condition of "B happened", which shows that A and B have the same ultimate goals.

The idea of Intrusion Termination Detection Algorithm (ITDA) is as following:

For an intrusion event sequence IS = $I_1 I_2 .. I_k$, IS implies a compound attack A. If during the period of T, $\neg\exists$I, s.t. I $\neg$R I_i, $\forall i \in 1..k$, that means A has terminated. T is a experience threshold of the time gap between adjacent intrusion events belonged to A.

3. The implementation model for response rollback

In this section, we will discuss two critical points to support automated response, and then propose an implement architecture for Response rollback, i.e., RAIRS.

3.1. Platform dependant execution script for automated response

Currently the communication between IDSs and IRSs is prone to adopt a self-contained protocol which also adapts itself to the communication among any IDS components, for example CIDF, IAP and IDIP. In the paragraph following another communication method between IDSs and IRSs is proposed, which can implement automated response and is better than the protocol method in some aspects.

3.1.1. The comparison of automated response execution methods

Responses are carried out differently on devices with different operating system platforms. These devices include routers, firewalls, hosts, etc. There are two approaches to distribute response commands to these devices.

(1) Devising a specific protocol. This protocol is responsible for the transmission of commands between IDS and response devices. The response device will execute a response script or modify its own configuration once it receives the response command. A typical example of this approach is IDIP, which can not only transmit response commands, but also can support the cooperation of intrusion detection among all the components of IDS.
(2) Using an automatic interacting script. The script is able to automatically log into response devices, and execute response programs or modify system configuration. This approach is similar to manual response in that every statement in the script corresponds to a step of manual response, but the script is executed automatically by a translator. Expect, which is currently used in areas such as IDS assessment, is a kind of automatic interacting script language suitable for such an approach. This method is platform dependent, and is not so complicated as mobile agent.

Table 2
The comparison of two approaches for executing response

	Protocol approach	Script approach
Standardization	Good. The methods of information exchange and information execution are standard process	Bad. The methods of information exchange and information execution are proprietary
Compatibility	Bad. Take IDIP as an example. IDIP has already been implemented and it is now in the test phase with some functionalities still requiring further improvement. Moreover, it requires all components IDIP-compatible, which makes it impossible to incorporate currently used routers into the framework	Good. The script is executed by a translator without any support of the response devices. Current response devices can be easily added to the response system
Scalability	Good. For a self-contained and robust protocol, it can transmit any kind of response and meet all kinds of requirements in information exchange	Fair. New responses are easily added to the response system by adding corresponding response scripts. However, more cost will be required when the system expends because the interaction is not standard
Support for response rollback	Current protocols don't support	Good. Response rollback script can be built like response script

The comparison of the above two approaches is shown in Table 2.

From Table 2, it can be seen that the disadvantages of the protocol approach lie in compatibility. From the practical point of view, despite its problem of standardization and scalability, the script approach has the advantages of simplicity, ease of implementation, compatibility with current response devices, and supporting response rollback. Therefore, it is used in RAIRS to realize automated response and response rollback.

3.1.2. Automated execution of response and response rollback by script approach

The automated response script is similar to other shell scripts, and its advantage is the capability of invoking interacting programs such as telnet, ftp and etc., without human intervention.

Section 2.1 points out that every response could be decomposed into a sequence of atomic responses. Thus response can be realized by executing each of its atomic response script in turn. In addition, response rollback can be realized by executing each script of inverse atomic response in the reverse order.

For $\boldsymbol{R} = <UR_1, UR_2, \ldots, UR_n>,\ n = 0, 1, 2, \ldots, UR_i = <Op_i, RS_i, Ou_i>$. Note S_i is the automated response script of UR_i. Therefore, to implement UR_i is exactly to execute $S_i(RS_i, Ou_i)$, which RS_i and Ou_i are the parameters of S_i. To implement R is exactly to execute $S_1(RS_1, Ou_1), S_2(RS_2, Ou_2), \ldots, S_n(RS_n, Ou_n)$. Note that before

automated response and response rollback can be executed, each automated response script corresponding to a kind of response measure must be built.

3.2. The hierarchy of response policy

The response policy of intrusion events can be divided into three layers, that is, goal, plan and action. From the upper layer to the lower layer, response policies go from abstract ones to specific ones, from guidance to specific implementations. The hierarchy of response policy is shown in Fig. 1.

The response goal is the goal that the responses of the security events want to achieve, such as the goal of minimum cost. The response plan is the abstract schedule that is made based on security events and the response goal, such as the schedule of "first terminate user session, then lock the account of this user". Response is the actions which realize one step of the response plan, that is, the ternary atomic response. Last, responses are translated into response scripts to be executed.

When the response system receives an intrusion event, its transaction process is as following:

(1) Query the response goal base according to the intrusion event.
(2) Query the response plan base according to the intrusion event and the response goal, and assess each response plan to choose the optimal one that satisfies the response goal.
(3) Map response plan and the intrusion event into atomic response sequence by querying the response action base. Therefore $\boldsymbol{R} = <UR_1, UR_2, \ldots, UR_n>$ is gotten, $n = 0, 1, 2, \ldots$; If now there is a command to rollback $\boldsymbol{R}$, then
I Transform $\boldsymbol{R}$ to $\neg$ R, i.e., $\neg$ R = $<\neg\ UR_n, \neg\ UR_{n-1}, \ldots, \neg\ UR_1>$.
II Remove the atomic responses in $\neg$ R each of which is ϕ, i.e., $\neg\ UR_i = \phi$, $i \in 1..n$. Then $\neg$ R= $<\neg\ UR_{j1}, \neg\ UR_{j2}, \ldots, \neg\ UR_{jk}>$, $\neg\ UR_{ji} \neq \phi$, $j1 > j2 > \cdots > jk$, $i = 1..k$, $k \leqslant n$.
(4) Find the execution script of the atomic response in the response script base.

Note that instead of eliminating the effect of the previous response, the purpose of response rollback is to restore the status of the system to a suitable one under which the system can operate normally, so that the rollback script need not be symmetric to its corresponding one.

This description of response policy has the advantages of clarity and agility, and can be applied to the definition of response policies in various environments.

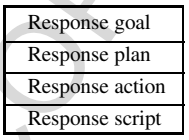

Fig. 1. The structure of response policy.

3.3. The system architecture of RAIRS

RAIRS should satisfy several requirements:

(1) Having the previous structure of response policy.
(2) Being capable of detecting intrusion termination and rolling back responses of false alert.

To emphasize the mechanism of Response Rollback and automated response, a simplified realization model of RAIRS is given in Fig. 2. It behaves as follows.

(1) The IDS at the top of Fig. 3 sends intrusion event reports to RAIRS one by one. The IDS can be any type, provided the same event semantics are compatible with RAIRS.

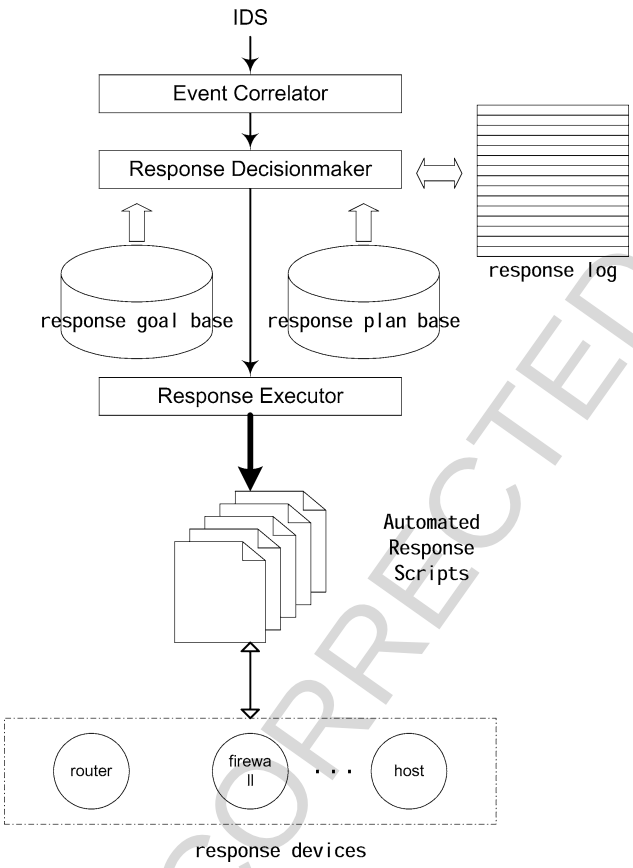

Fig. 2. A simplified realization model of RAIRS.

(2) The Event Correlator analyzes the intrusion event in a report, and decides whether the event can be correlated to an ongoing Compound intrusion or not (see 2.3). If true, the Event Correlator sends the intrusion event sequence to the Response Decision-maker; otherwise it will create a new intrusion event sequence with this event and send the sequence to Response Decision-maker. If the Event Correlator finds a terminated compound intrusion using ITDA, or a report which claims some previous intrusion event is a false positive is received, it sends a response rollback command to Response Decision-maker. The Event Correlator also can be an intrusion plan recognizor which can produce a response rollback command when a previously issued false alert has been found.

(3) The Response Decision-maker has two kinds of input. If the Response Decision-maker receives a response rollback command, it will query its Response Log for atomic response sequence according to the intrusion event Number in the response rollback command and then transform the atomic response sequence to inverse atomic response sequence and send it to the Response Executor. If the Response Decision-maker receives an intrusion event or an intrusion event sequence, it will query the Response Goal Base and the Response Plan Base for a response plan, and then transform the response plan to atomic response sequence. When the Response Decision-maker sends a new atomic response sequence to the Response Executor, it will write it to the Response Log as well.

(4) The Response Executor implements each atomic response in the atomic response sequence in turn. It retrieves the corresponding automated response script according to the Atomic Operator in the atomic response and then refers to Script Interpreter to execute the script.

4. Experiments and results

4.1. Goal and experiment environment

In this section, we show how our methodology works. The goal of this experiment is to make comparison between RAIRS and ARS, which is an automated response system without response rollback. We consider four types of cost in intrusion response systems, that is, operation cost (OCost), response cost (RCost), damage cost (DCost), and response rollback cost (RRCost). OCost is the total amount of resource that is consumed by the execution of responses, including CPU time, memory and network bandwidth. RCost is the negative effects caused by the responses, such as the unavailability of network services or communication channels. DCost refers to the cost of the intrusion after it succeeds and before responses are taken. RRCost refers to the operation cost of rolling back responses. The total cost of intrusion response is the sum of these four types of cost. Let *CumulativeCost(R)* be the expectation of the

total cost of intrusion responses system R, and E be the set of intrusion event types, then

$$CumulativeCost(R) = \sum_{e \in E} \lambda(e)(OCost(e) + RCost(e) + DCost(e))$$

where $\lambda(e)$ represents the proportion of events of type e in all the events.

The experiment is performed in CERNET (China Education and Research Network). The total cost of each system is calculated according to the characteristics of intrusions in this environment and the performance of IDS, i.e., the distribution of intrusion event types and the false positive rate of IDS.

4.2. Experiment method

First, the information about the main types of known network intrusion events in CERNET and the responses to each type of events is gathered. Then, the numerical distribution of intrusion event types, and the total cost of each response are

Table 3
The distribution of event types and response plans

Event type	Class	Intrusion cost *ICost*	Proportion of the events	Response plan
Dos	DOS	30	0.01%	log, isolate the victim host from network (1) log, block the attacker (2)
Buffer overflow	ROOT	100	10.11%	Log, block the attacker, lock the user account (3) log, hang up the process that has the buffer overflow vulnerability (4)
Scan	PROBE	2	59.9%	log, block the attacker (2) log, disable the ICMP echo function of the victim host (5)
Decoding	R2L	50	4.49%	log, block the attacker (2) log, disable the target port of the attack (6)
Web based attack	R2L	50	10.39%	log, block the attacker (2) log, disable the target port of the attack (6)
Virues	R2L	50	0.57%	log, block the attacker (2) log, isolate the victim host from network (1)
BackDoor or Trojan horses	R2L	50	12.04%	log, block the attacker (2) log, disable the target port of the attack (6)

calculated. Finally, the integrated total expectation cost of each automated response system is calculated.

The quantification of each type of cost is quite difficult and is still an open area of ongoing research. This experiment draws upon the quantification results from Lee et al. [5], and is based on the actual conditions of CERNET as augmented by some hypothesis for simplification described below.

(1) Suppose the victim of the intrusion is the device that provides public services in CERNET, such as routers, mail servers, DNS servers and etc. The intrusion responses to these devices will cause considerable negative effects.
(2) The quantification of operation cost and damage cost mainly comes from [5], and response rollback cost is equal to operation cost.
(3) The response cost increases with the time of response, which can be seen in Table 3. Therefore, response cost can be simplified as a linear function of time, that is, $\mathit{RCost} = a \times t$, where a is the response cost in unit time, and t represents the duration from when the response starts.
(4) The quantification of response cost is counted in dollars. For example, suppose there are n users of the Yahoo mail service, and the monthly fee for each user is c dollars, then if the response measure isolates the mail server from the network, the response cost is $\mathit{RCost} \approx (n \times c) \div (30 \times 1440)$ dollars/min.

4.3. Experiment procedures and results

The IDS used in this experiment is Snort which is an open source network IDS and can detect many kinds of network intrusions. All the events detected in one day are recorded in a table whose scale is about 200 to 400 records, and an average false positive rate of 10.23%. Table 3 shows the primary types of intrusion events in CERNET, the daily numerical distribution, and the corresponding response plans. The average false positive rate and "proportion of the events" in Table 3 are calculated based on the observing data in one month of September, 2002.

In Table 3, every type of intrusion event has two response plans, with each one being identified by a unique number.

Table 4 shows the total cost of the above types of events in RAIRS and ARS. In the table, t represents the time interval between the activation of response and the manual elimination of it, and t' is the interval between the activation of response and the rollback of it. In general, $t \gg t'$. DCost is correlated with the type of intrusion event, and we quantify it as the multiplication of *ICost* (Intrusion Cost) and $\varepsilon_1 \in [0, 1]$ (a discount coefficient).

Suppose each response plan of the same event type has the same opportunity to be executed, $t = 1$ (day) $= 24 \times 60 = 1440$ (min), $t' = 10$ (min), $\varepsilon_1 = 0.1$, then

$$\mathit{CumulativeCost}(\mathit{ARS}) \approx 50217.46,$$

$$\mathit{CumulativeCost}(\mathit{RARS}) \approx 392.10$$

Table 4
The total cost of the response to each type of events

Response plan	Ocost		RCost		DCost		RRCost		CumulativeCost	
	ARS	RAIRS	ARS	RAIRS	ARS	RAIRS	ARS	RAIRS	ARS	RAIRS
(1)	20	20	$228t$	$228t'$	$\varepsilon_1 \times ICost$	$\varepsilon_1 \times ICost$	0	20	$228t + 20 + \cdots$	$228t' + 40 + \cdots$
(2)	20	20	$10t$	$10t'$	$\varepsilon_1 \times ICost$	$\varepsilon_1 \times ICost$	0	20	$10t + 20 + \cdots$	$10t' + 40 + \cdots$
(3)	30	30	$11t$	$11t'$	$\varepsilon_1 \times ICost$	$\varepsilon_1 \times ICost$	0	30	$11t + 30 + \cdots$	$11t' + 30 + \cdots$
(4)	20	20	$150t$	$150t'$	$\varepsilon_1 \times ICost$	$\varepsilon_1 \times ICost$	0	20	$150t + 20 + \cdots$	$150t' + 40 + \cdots$
(5)	20	20	$5t$	$5t'$	$\varepsilon_1 \times ICost$	$\varepsilon_1 \times ICost$	0	20	$5t + 20 + \cdots$	$5t' + 40 + \cdots$
(6)	20	20	$150t$	$150t'$	$\varepsilon_1 \times ICost$	$\varepsilon_1 \times ICost$	0	20	$150t + 20 + \cdots$	$150t' + 40 + \cdots$

From the experiment data, the total cost of RAIRS is much lower than that of ARS for our specific experiment. Although one experiment is not conclusive, these positive results indicate that the response rollback mechanism has the potential to play an important role in decreasing the total cost of response systems.

5. Conclusion

The paper introduces the concept of response rollback in intrusion response systems, which enables the rollback of responses to false positive and the intrusion which has terminated so that the negative effect of responses decreases. The formal description of response is given which facilitate the following discussions. The automated response script method is chosen for executing automated intrusion response. The script approach, compared with the protocol approach, has the advantages of simplicity, ease to implementation, and compatibility with extant devices. These ideas are integrated into the architecture of RAIRS, which supports the realization of automated response and response rollback in IRSs.

The experiment assesses the performance of RAIRS and ARS from the view of cost, based on the detection data observed in CERNET and the cost parameters in reference [5] and our cost quantification method. The experiment shows that RAIRS can dramatically decrease the cumulative cost of responses in our sample environment.

Acknowledgements

This research is supported by National Natural Science Fundation of China (90104031), and the China 973 Program (contract no. 2003CB314803).

References

[1] C.A. Carver and U.W. Pooch, An intrusion response taxonomy and its role in automatic intrusion response, in: *Proceeding of the 2000 IEEE Workshop on Information Assurance and Security*, United States Military Academy, West Point, NY, 2000, pp. 129–135.

[2] C.W. Geib and R.P. Goldman, Plan recognition in intrusion detection system, in: *DARPA Information Survivability Conference & Exposition II*, Hilton Anaheim, California, 2001, pp. 46–55.

[3] D. Schnackenberg, K. Djahandari and D. Sterne, Infrastructure for intrusion detection and response, in: *Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX) 2000*, Hilton Head, S.C, 2000, pp. 1507–1516.

[4] J. Zhang, J. Gong and Y. Ding, Intrusion detection system based on fuzzy default logic, in: *Proceeding of the 2003 IEEE Workshop on Fuzzy System*, St. Louis, 2003, pp. 1350–1356.

[5] W. Lee, W. Fan, M. Miller, S. Stolfo and E. Zadok, Toward cost-sensitive modeling for intrusion detection and response, *Journal of Computer Security* **10**(1) (2002), 318–336.