Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2013, Article ID 134760, 8 pages
http://dx.doi.org/10.1155/2013/134760

Research Article

A Retroactive-Burst Framework for Automated Intrusion Response System

Alireza Shameli-Sendi, Julien Desfossez, Michel Dagenais, and Masoume Jabbarifar

Département de Genie Informatique et Génie Logiciel, École Polytechnique de Montréal, P.O. Box 6079, Succ. Downtown, Montreal, QC, Canada H3C 3A7

Correspondence should be addressed to Alireza Shameli-Sendi; alireza.shameli-sendi@polymtl.ca

Received 14 December 2012; Accepted 20 February 2013

Academic Editor: Rui Zhang

The aim of this paper is to present an adaptive and cost-sensitive model to prevent security intrusions. In most automated intrusion response systems, response selection is performed locally based on current threat without using the knowledge of attacks history. Another challenge is that a group of responses are applied without any feedback mechanism to measure the response effect. We address these problems through retroactive-burst execution of responses and a Response Coordinator (RC) mechanism, the main contributions of this work. The retroactive-burst execution consists of several burst executions of responses with, at the end of each burst, a mechanism for measuring the effectiveness of the applied responses by the risk assessment component. The appropriate combination of responses must be considered for each burst execution to mitigate the progress of the attack without necessarily running the next round of responses, because of the impact on legitimate users. In the proposed model, there is a multilevel response mechanism. To indicate which level is appropriate to apply based on the retroactive-burst execution, we get help from a Response Coordinator mechanism. The applied responses can improve the health of Applications, Kernel, Local Services, Network Services, and Physical Status. Based on these indexes, the RC gives a general overview of an attacker's goal in a distributed environment.

1. Introduction

Multisteps cyberattacks are common problems in distributed systems. Many security tools or system loggers may be installed in distributed systems and monitor all events in the network. Security managers often have to process huge numbers of alerts per day produced by such tools [1].

The Linux Trace Toolkit next generation (LTTng) [2] is a powerful software tool that provides a detailed execution trace of the Linux operating system with low impact. Its counterpart, the User Space Tracer (UST) library, provides the same trace information from user mode for middle-ware and applications [3]. The Target Communication Framework (TCF) agent collects traces from multiple systems. After collecting all traces, we need a powerful tool to monitor the health of a large system continuously such that system anomalies can be promptly detected and handled appropriately.

Intrusion Detection Systems (IDSs) are tools that monitor systems against malicious activities. We use network-based IDS (NIDS) to monitor the network and host-based IDS (HIDS) to locally monitor the health of a system. IDSs are divided into two categories: *Anomaly-based* and *Signature-based* [4, 5] techniques. Anomaly-based detection is interesting to detect unknown attack patterns and does not need predefined signatures. On the other hand, it suffers from the fact that it is difficult to define normal behavior and that malicious activity may look like normal usage pattern [6, 7]. In signature-based techniques, we compare captured data with well-defined attack patterns. Because of the pattern matching, this technique has the advantage of being deterministic and can be customized for each system we want to protect. Moreover, signature-based techniques are stateless; once an attack matches a signature, an alert is emitted and the detection component does not record it as a state change.

One solution, to tackle the limitation of detection based only on stateless signatures, is to use a finite state machine (FSM) to track the evolution of an attack. That way, while an attack is in progress, the state changes and we can trigger appropriate responses based on a confidence level threshold, which leads to a lower false positive rate. Based on the previous discussion about the advantages and disadvantages between anomaly-based and signature-based techniques, and since LTTng produces accurate traces, we decided to develop our framework with the FSM approach in order to track multistep attacks.

The main contributions of this work are the following. The proposed framework has a novel response execution organization named *retroactive-burst*. The term retroactive refers to the fact that we have a mechanism for measuring the effectiveness of the applied responses. The term burst refers to the fact that each retroactive execution consists of several bursts each consisting in relevant responses to apply. The idea is that each burst mode execution of responses in each retroactive execution must mitigate the progress of attack and avoid the need to run another burst. In contrast to previous models, this model is round-based. The online risk assessment measures the risk index of an applied round of responses instead of one applied response. Also, a multilevel response selection mechanism is implemented in our model. The higher level corresponds to strong responses. This helps to control the cost in performance and increases the intelligence of the Intrusion Response System (IRS). A Response Coordinator helps to select the appropriate level based on a global overview of past history of applied responses and attacks.

The paper is organized as follows. First, we investigate earlier work and several existing methods for intrusion response. The proposed model is discussed in Section 3. Experimental results are given in Section 4. Finally, we conclude and future work is discussed.

2. Related Work

Automated response systems try to be fully automated using decision-making processes without human intervention. The major problem in this approach is the possibility of executing an improper response in case of problem [8]. It can be classified according to the following characteristics.

Response Selection. There are three response selection models. (a) static model maps an alert to a predefined response. This model is easy to build, but the major weakness is that the response measures are predictable [9]. (b) Dynamic model responses are based on multiple factors such as system state, attack metrics (frequency, severity, confidence, etc.) and network policy. In other words, the response to an attack may not be the same depending for instance on the targeted host. One drawback of this model is that it does not consider intrusion damage [10, 11]. (c) Cost-sensitive model is an interesting technique that tries to attune intrusion damage and response cost. To measure intrusion damage, a risk assessment component is needed [8].

Adjustment Ability. There are two types of adjustment ability. (a) The first is non-adaptive: In this model, response selection mechanism remains the same during the attack period. It does not use the response history to order responses. (b) The second is daptive. In these approaches, the system has an appropriate ability to automatically adjust the response selection based on success or failure of response in the past [12].

Response Execution. There are two types of response execution [13]. (a) The first is burst: In this model, there is no mechanism to measure the risk index of the host/network once the response has been applied. The major weakness in this model is the cost in performance caused by applying all responses, while a subset of the responses may be enough to mitigate the attack. (b) The second is retroactive: in these approaches, there is a feedback mechanism which has the ability to measure the response effect based on the result of the last applied response. There are some challenges in adaptive approaches. For example, how can we measure the success of the last applied response and how multiple concurrent malicious activities can be handled [12]?

Foo et al. [12] proposed a graph-based approach called ADEPTS. The responses for the affected nodes are based on some parameters such as confidence level of attack and previous measurements of responses in similar cases. Thus, ADEPTS uses a feedback mechanism to estimate the success or failure of an applied response.

In [14], Stakhanova et al. proposed a cost-sensitive preemptive intrusion response system. It monitors system behavior in terms of system calls. The authors presented a response system which is automated, cost-sensitive, preemptive, and adaptive. The response is triggered before the attack completes. There is a mapping between system resources, response actions, and intrusion patterns which has to be defined in advance. Whenever a sequence of system calls matches a prefix in a predefined abnormal graph, the response algorithm, based on confidence level threshold, decides whether to repel the attack or not. If the selected response succeeds in mitigating the attack, its success factor is increased by one, while on the contrary, it is decreased by one.

In [15], Lee et al. proposed a cost-sensitive model based on three factors: damage cost that characterizes the amount of damage that could potentially be caused by the attacker, operational cost that illustrates the effort for monitoring and detecting the attacks by an IDS, and response cost that is the cost of acting against attacks.

Retroactive approach was first proposed by Mu and Li [8]. They presented a hierarchical task network planning to repel intrusions. This model is able to avoid unnecessary responses and reduce the risk of false positive response by adjusting risk thresholds of subtasks. The interesting idea in this paper is response time decision-making. It can estimate the execution time of each response. Each response has a static risk threshold associated. The permission for running each response is the current risk index of the network.

In case of response execution, our technique closely relates to [8]. They try to measure the risk index after running

each response. Our experience shows that this measure is not enough to make the decision of running the next response and cannot be applied in a production environment. To tackle this issue, we have defined a retroactive-burst execution mechanism. For adjustment ability, our technique closely relates to [14], but there are many distinguishing features. They used a static damage cost for each node in abnormal graph. In other words, their risk assessment is static. By contrast, we have implemented a dynamic (online) risk assessment component that helps our response component to attune intrusion damage and response cost over time. Another distinguishing feature that separates our model from previous models is that the majority of the proposed response selection mechanisms focus on the local view of threats and responses to select a set of responses and do not have a general view of the network status. In the proposed model, we designed a novel approach named response coordinator to tackle this weakness.

3. Proposed Model

To design a strong intrusion response system, we should have a flexible response mechanism to handle different malicious activities. Figure 1 illustrates the proposed structure for such an intrusion response system. The proposed IRS framework consists of five modules.

3.1. Module (1) Manager. Manager is responsible for getting alerts from the detection and online risk assessment components and eventually run the appropriate responses on the attacked machine. Online risk assessment (ORA) component guarantees that our model is cost-sensitive. ORA measures the risk index of an applied round of responses (R_{index}). The detection component has all the detailed information about the malicious activity such as the severity, the confidence level (C_{level}), and the type of resource targeted. It sends an alert after each state change in the FSM. In our FSM, a weight is associated to each state, and the sum of all weights is 100. The confidence level related to each raised alert is equal to the sum of all weights of all previous states. The confidence level guarantees that our model is preemptive. The `Activator` process gets all alerts from all the components. Once the following condition (1) is true, it starts the `Establish_Remote_Connection` and `Select_Response` processes:

$$R_{\text{index}} \times C_{\text{level}} > \text{Threshold}. \quad (1)$$

The `Establish_Remote_Connection` part generates a connection string and sends it to the `Open_Channel` process. The `Open_Channel` tries to connect to a remote agent running in the target host. Since TCF [16] is a lightweight and extensible communication daemon, we chose it as remote agent. After establishing a channel to the TCF remote agent, `Run_Plans` process applies responses on the target computer as Figure 1 illustrates.

The `Run_Plans` process is the core of our response framework. Unlike usual response systems, it considers the time factor and applies responses using a multistep reaction procedure. Mu and Li [8], tried to measure the risk index after running each response. Our experience shows that this measure is not enough to make the decision of running the next response.

To tackle this issue, we have defined a round-based response mechanism. Figure 2 illustrates six responses for a specific malicious activity which are ready in pending queue in the `Run_Plans` process before starting the first round. Sending the next round of responses is based on (1). Upon running a round of responses, new risk index of network has to be measured from the Online Risk Assessment component after a specific time. As shown in Figure 2, each response has a `Response_Effect` that defines how the selected responses are ordered in the pending queue. Figure 3 shows two possible scenarios after launching the first round of responses. In the first situation, since the risk index of the network is decreasing, the next round is not required. This intelligence prevents overly impacting the network. By contrast, in the second situation, in spite of the application of the first round, the risk index shows that malicious activity is still progressing. Thus, the second round of responses has to be applied.

3.2. Module (2) Strategy. An intrusion can be defined as any set of actions that threaten the Confidentiality (C), Integrity (I), Availability (A), and Performance (P) of host/network resources such as files, kernel, or user accounts. To react against attacks, we have designed four strategies to evaluate all responses.

(1) *MAX-Confidentiality* ensures that any authorized user can have access only to the limited subset of resources required.

(2) *MAX-Integrity* verifies that any authorized user can only modify the resources in a conform manner.

(3) *Availability* means that the resources are always available to the authorized users.

(4) *Performance* means that the system responds within the time expected.

3.3. Module (3) Responses. This module is responsible for managing the set of responses available. A database connects the Strategy and Responses modules. This connection is a structure to measure each response effect. Our evaluation relies on the positive effect and negative impact of responses to the strategies (C, I, A, and P).

3.4. Module (4) Response Coordinator (RC). Most of proposed response mechanisms focus on the local view of threats and responses and do not have a general view of the network status. Thus, they suffer from not having global information about the attacker's goal. To tackle this issue, we introduce the RC module. We divide the system status in five general categories: Applications, Kernel, Local Services, Network Services, and Physical Status. The purpose of RC is to improve the quality of responses over time. Let us briefly describe the RC categories.

Application Status. There are many applications installed in the system like Web applications, desktop applications,

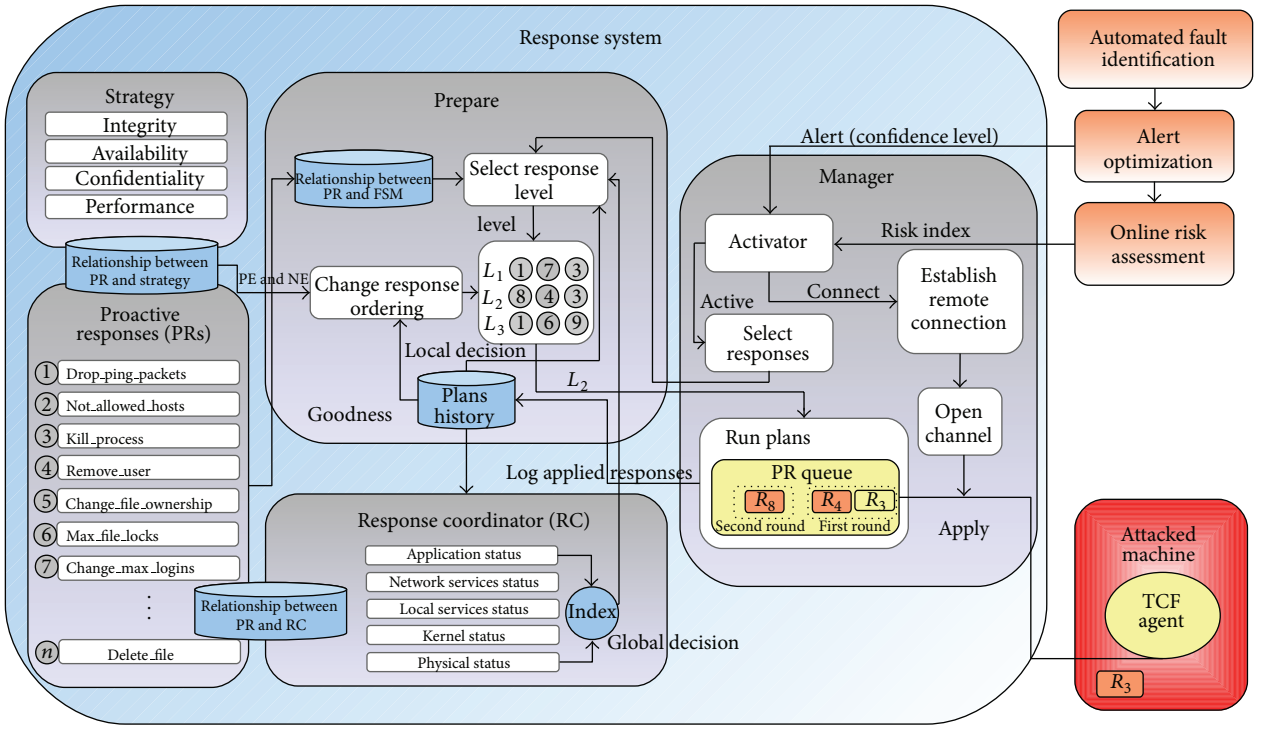

Figure 1: Proposed architecture for automated response system.

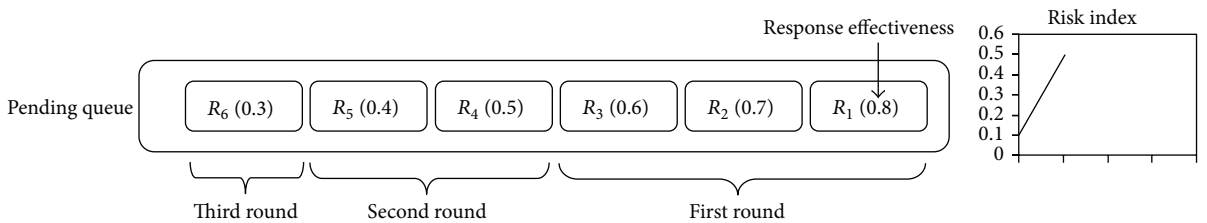

Figure 2: Pending queue in *Run_plans* process before starting the first round of responses.

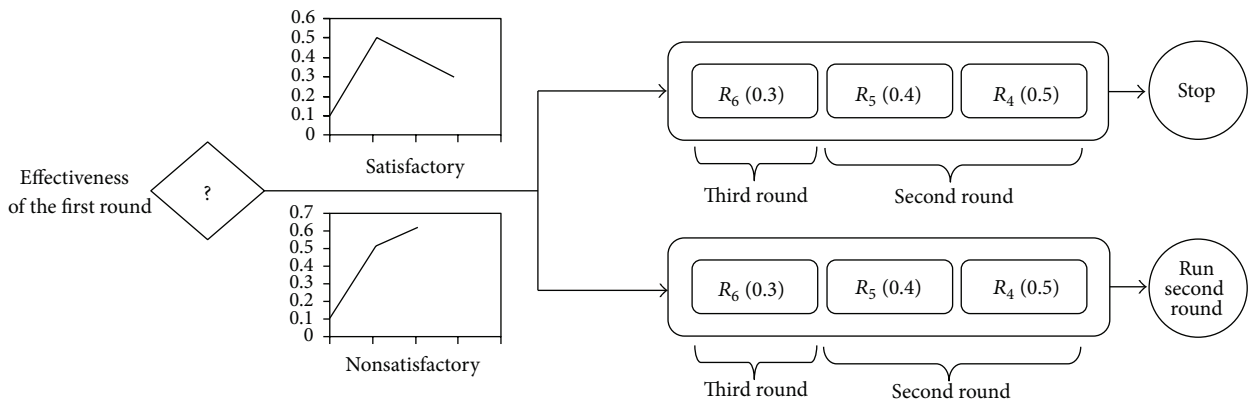

Figure 3: Two possible outcomes for decision-making after running the first round of responses.

programming tools, DBMS, OpenOffice, and web browsers. Malicious hackers are looking for vulnerabilities in these applications. The health of these applications is very important. AppArmor [17] or SELinux [18] are security modules for the Linux kernel that allow associating each program with a security policy that restricts its capabilities. Policy-based tools improve application health.

Kernel Status. It represents the higher level of criticality of a local system. In modern operating systems, the kernel and the kernel modules have access to any resource of a local machine. If an attacker gains access to the kernel, it can do anything on the local machine with very little chance of being detected.

Local Services Status. Virtually, each service represents a security threat that could be exploited by an intruder to gain further access to the system. Unlike local applications, the local services run at a higher level of privilege and could give attackers higher privileges if they were compromised.

Network Services Status. Like the local services, network services represent a security risk that could be exploited to gain access to a system. The major difference with local services is that they can be exploited from outside.

Physical Status. It represents the status of the physical devices of the machine in terms of usage and performance. For example, when a machine is under a network flood attack, the network interface health is considered critical. Over time, we gather statistics about how many responses are applied in each category. These statistics help to discover major health problems of each host, detect major health problems of the whole network, help administrator to chose a policy suited for the organization, and it can help the `select_response_level` process to select the more appropriate levels of responses.

3.5. Relationship between Responses and RC DB. Table 1 shows the relationship between responses and RC categories. RC process updates Table 1 based on the `plans_history` database. Each response (R_i) is associated with one or more RC category (Application, Kernel, ...). For example, in Table 1, response R_1 is related to Application, Local Services, and Network Services (Linked). Each RC category has a weight (W_i) which represents the importance of the category for the organization ($W_A, W_K, W_{LS}, W_{NS}, W_P$). In the online risk assessment component, we associate each host (H_i) with a value (V_{Hi}), representing the priority of this host for the organization. In Table 1, *Applied* indicates that a response has been applied and on which hosts. We activate the categories associated with a response when the sum of the values of the hosts which applied this response is greater than a threshold based on the following (n is a subset of hosts that a specific response has been applied on them).

$$\sum_{i=1}^{n} V_{H_i} > \text{Threshold}. \tag{2}$$

The status of each category of RC is computed using

$$\text{Status}\left(W_i\right) = \frac{\text{Count (Activated)}}{\text{Count (Linked)}}. \tag{3}$$

Finally, the RC index is computed using a weighted average by (4). The result of Status and RC calculations is one of the values: *Zero (0)*, *L (Low)*, *M (Medium)*, or *H (High)*. k is five in our model with respect to the number of categories defined for system status. Consider

$$RC_{\text{index}} = \frac{\sum_{i=1}^{k} W_i \times \text{Status}\left(W_i\right)}{k}. \tag{4}$$

3.6. Module (5) Prepare. The prepared module is composed of two processes and two databases.

3.6.1. Relationship between Responses and FSM DB. Each attack pattern is associated with an FSM. For each defined FSM, multiple response actions can be defined in advance. These response actions are organized in levels.

3.6.2. Plans_History DB. It is a log file to store *Target IP*, *User_Name*, *Date*, *Time*, *Resource*, *Alert_Name*, *Level_Id*, *Round_Responses*, and *Round_Success*. As explained in the Manager module, it is possible that all responses are not applied by the *Run_Plans* process; so at the end of each round, it will update the *plans_history* to store the status of the round.

3.6.3. Change_Response_Ordering Process. After selecting the appropriate level of responses required to repel the attack (using the `Select_Response_Level`), we need to order the responses of the selected level. This process guarantees that our model is adaptive. The ordering operation has to be done using

$$\text{RE} = \left[\left(\text{Positive\_effect}\right) - \left(\text{Negative\_impact}\right)\right] \times \text{Goodness}. \tag{5}$$

Positive_effect and *Negative_impact* are static parameters. Goodness is a dynamic parameter that represents the history of success (S) or failure (F) of each response for a specific type of host. The goodness parameter guarantees that our model is dynamic in case of response effectiveness and helps IRS component to prepare the best set of response over time. To measure the success or failure of a round of responses, we use the result of the online risk assessment component. As mentioned in Figure 3, if the risk index of the network is decreasing, the next round is not required and the status of the previous applied round is set to success. By contrast, in the second situation, in spite of the application of the first round, the risk index shows that malicious activity is still progressing. Thus, the second round of responses has to be applied, and the status of the previous applied round is set to failure.

3.6.4. Select_Response_Level Process. Information coming from the plans_history database and the RC process are used

Table 1: Relationship between Responses and RC.

Response name	RC					Hosts					
	Application	Kernel	Local Services	Network Services	Physical	H_1	H_2	H_3	H_4	...	H_n
	w_A	w_K	w_{LS}	w_{NS}	w_P	VH_1	VH_2	VH_3	VH_4	...	VH_n
R_1	Linked		Linked	Linked				Applied			
R_2	Activated	Activated		Linked		Applied	Applied	Applied			
R_3	Activated	Activated		Activated	Activated	Applied		Applied			Applied
R_4	Linked		Linked	Linked	Linked		Applied		Applied		
⋮											
Status	Medium	High	Zero	Low	Medium						
Index			Low								

Table 2: Policies for dynamic response level selection.

Policy	RC.index	Selected Level
P1 = There is not any information in plans_history	low	1
	Medium	1
	High	2
P2 = (There is related information in plans_history) and (Previous status was successful) and (Time of previous run is far to current time)	Low	current_level
	Medium	current_level
	High	current_level + 1
P3 = (There is related information in plans_history) and (Previous status was successful) and (Time of previous run is near to current time)	Low	current_level
	Medium	current_level + 1
	High	current_level + 2
P4 = (There is related information in plans_history) and (Previous status was not successful) and (Time of previous run is far to current time)	Low	current_level + 1
	Medium	current_level + 2
	High	current_level + 3
P5 = (There is related information in plans_history) and (Previous status was not successful) and (Time of previous run is near to current time)	Low	current_level + 2
	Medium	last_level
	High	last_level

to select the best response level to repel an intrusion in progress. When it receives a message from the Manager module, it tries to find related information in the plans_history database. Depending on the existing knowledge about a similar attack and on the RC index, it selects the appropriate level. Table 2 describes the different policies available to select dynamically a response level.

4. Experimental Results

The Linux Trace Toolkit next generation (LTTng) is a powerful software tool that provides a detailed execution trace of the Linux operating system with a low impact on performance. Using traces, LTTng records computer activities as seen by the kernel and eventually the userspace applications if they are instrumented with UST [7].

Although execution trace contains important and valuable information to detect system faults and network attacks, this information is usually behind the large number of events. Trace-based detection systems usually need a preliminary step to alleviate this problem. Trace abstraction is one possible solution that is used in the literature to reduce the trace size [19], to generate high level synthetic information [20], and to extract complete statistics of system resources usage [21]. This high level information can then be used easily in the detection phase.

In our experiments, Automated Fault Identification (our IDS) [20] and our framework were developed on IP xxx.xxx.72.12; the TCF agent and the attacked machine were developed on IP xxx.xxx.72.131 but in a virtual machine. We have considered the Escaping a chroot jail attack. Chrooting changes the root directory of a process. As Figure 4 illustrates, when a process is in a chroot, it can only access a subdirectory of the file hierarchy and does not have access to higher level directories. If an attacker can exploit a vulnerability in the chroot system, it can access higher level directories.

LTTng traces contain the related events and system calls; the Automated Fault Identification component can detect the attack using the appropriate FSM. It sends an alarm in the IDMEF [22] format to the automated intrusion

Table 3: Plans_history database, after applying the first round of $level_1$.

Target IP	User_Name	Date	Time	Resource	Alert_Name	Level_Id	Round_Responses	Response_success
xxx.xxx.72.131	Smith	2012/10/02	10:05:06	Filesystem Root	Chroot	$level_1$	RS_1, RS_2, RS_4	S

Table 4: Plans_history database, after applying the first round of $level_2$.

Target IP	User_Name	Date	Time	Resource	Alert_Name	Level_Id	Round_Responses	Response_success
xxx.xxx.72.131	Smith	2012/10/02	10:05:06	Filesystem Root	Chroot	$level_1$	RS_1, RS_2, RS_4	S
xxx.xxx.72.131	Peter	2012/10/02	10:10:23	Filesystem Root	Chroot	$level_2$	RS_1, RS_3, RS_4	S

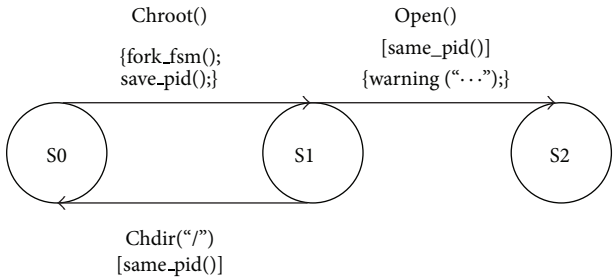

Figure 4: Escaping the chroot jail.

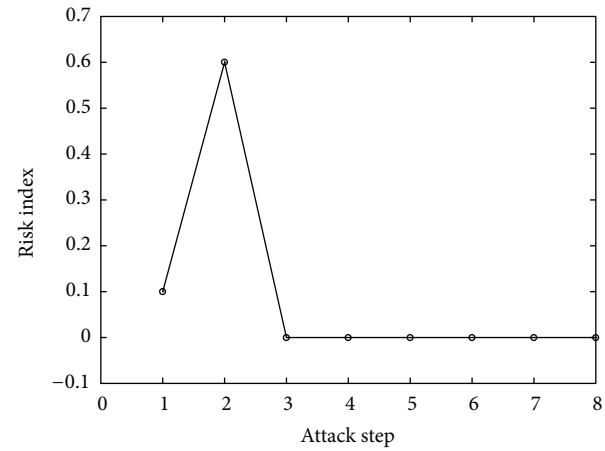

Figure 5: The online risk assessment result for escaping the chroot jail.

response framework. While this attack is in progress, the state changes and we can trigger appropriate responses based on a confidence level threshold. Our threshold value for (1) is 0.5. If state S1 sends an alert, since its confidence level is 0.5 and risk index (output of online risk assessment component) is 0.1, the rule of (1) is not triggered. If state S2 sends an alert, since the confidence level of S2 is 1 and the value of risk index is 0.6, the IRS component creates a channel to the target computer (xxx.xxx.72.131) and applies a first round of responses through the TCF agent. Sending the next round of responses is based on the new risk index and calculating (1). Our framework has two response levels for this attack:

List of Responses = {

RS1: KILL_PROCESS, RS2: LOCK_USER,

RS3: REMOVE_ALL_USERS, RS4: LOGOUT,

RS5: RESET}

Level1 = {Round1 (RS1, RS2, RS4), Round2 (RS5)}

Level2 = {Round1 (RS1, RS3, RS4), Round2 (RS5)}

4.1. First Scenario. Since the `plans_history` database does not have any information about this type of alarm, `select_response_level` process selects $level_1$ (Policy = P_1, RC.index = Low). RS_1 is the first response applied by the `Run_Plans` process. It kills the problematic processes. Then, RS_2 is executed to change the current user's password. RS_4 is then executed to logout the problematic user. From this moment, the user cannot login anymore to the machine using its system account. Table 3 illustrates the content of the plan_history database after these actions have been applied. Since $round_1$ causes that risk index to decrease, $round_2$ is not applied and this framework keeps the impact on service availability to a minimum. The risk curve caused by Escaping the chroot jail is shown in Figure 5. As seen, this attack has two steps. The risk index of the first and second step is 0.1 and 0.6, respectively. Since the IRS component will control this attack, the risk index will be zero after the second step.

4.2. Second Scenario. If other accounts are available on the victim machine, the attacker may login with another account. Suppose, at this time, that the RC index is *Medium*. The next time the same attack pattern reappears, the system will adapt its response and $level_2$ is selected (Policy = P_3, RC.index = Medium). Once again the problematic process is killed and, this time, all users on the system are removed. Table 4 illustrates the updated `plan_history` database after applying $level_2$. At this point, all users of the system have been removed and logged out.

5. Conclusion

Network services are becoming larger and increasingly complex to manage. It is extremely important to maintain the users QoS, the response time of applications, and critical services in high demand. On the other hand, we see impressive changes in the ways in which attackers gain access to systems and infect computers. An intrusion response system has to accurately assess the value of the loss incurred by a compromised resource and has an accurate evaluation of

the responses cost. The aim of this paper is introducing a novel framework for automated intrusion response system. In our model, unnecessary responses are controlled by an online risk assessment component. Each response is put in a multilevel mechanism for each FSM and run in a retroactive-burst mode execution. This is the main contribution of this paper. The response coordinator named RC enables us to have a general view of applied responses history. Taking control of the network status by RC leads us to have a reliable IRS which keeps the network quality of service.

Acknowledgments

The support of the Natural Sciences and Engineering Research Council of Canada (NSERC), Ericsson Software Research, and Defence Research and Development Canada (DRDC) is gratefully acknowledged.

References

[1] F. Xiao, S. Jin, and X. Li, "A novel data mining-based method for alert reduction and analysis," *Journal of Networks*, vol. 5, no. 1, pp. 88–97, 2010.

[2] M. Desnoyers and M. Dagenais, "LTTng: tracing across execution layers, from the hypervisor to user-space," in *Proceedings of the Linux Symposium*, Ottawa, Canada, 2008.

[3] J. Blunck, M. Desnoyers, and P. M. Fournier, "Userspace application tracing with markers and tracepoints," in *Proceedings of the Linux Kongress*, October 2009.

[4] N. B. Anuar, H. Sallehudin, A. Gani, and O. Zakari, "Identifying false alarm fornetwork intrusion detection system using hybrid data mining and decision tree," *Malaysian Journal of Computer Science*, vol. 21, no. 2, pp. 101–115, 2008.

[5] A. Lazarevic, L. Ertz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative study of anomaly detection schemes in network intrusion detection," in *Proceedings of the 3rd SIAM International Conference on Data Mining*, 2003.

[6] Yusof, *Automated Signature Generation of Network Attacks [B.S. thesis]*, University Teknologi Malasia, 2009.

[7] "Difference between signature based and anomaly based detection in IDS," http://www.secguru.com/forum/difference.

[8] C. P. Mu and Y. Li, "An intrusion response decision-making model based on hierarchical task network planning," *Expert Systems with Applications*, vol. 37, no. 3, pp. 2465–2472, 2010.

[9] Y. M. Chen and Y. Yang, "Policy management for network-based intrusion detection and prevention," in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium, Application Sessions (NOMS '04)*, pp. 219–232, Seoul, South Korea, April 2004.

[10] G. B. White, E. A. Fisch, and U. W. Pooch, "Cooperating security managers: a peer-based intrusion detection system," *IEEE Network*, vol. 10, no. 1, pp. 20–23, 1996.

[11] P. Porras and P. Neumann, "EMERALD: event monitoring enenabling responses to anomalous live disturbances," in *Proceedings of the National Information Systems Security Conference*, 1997.

[12] B. Foo, Y. S. Wu, Y. C. Mao, S. Bagchi, and E. Spafford, "ADEPTS: adaptive intrusion response using attack graphs in an e-commerce environment," in *Proceedings of the International Conference on Dependable Systems and Networks*, pp. 508–517, July 2005.

[13] A. Shameli-Sendi, N. Ezzati-Jivan, M. Jabbarifar, and M. Dagenais, "Intrusion response systems: survey and taxonomy," *International Journal of Computer Science and Network Security*, vol. 12, no. 1, pp. 1–14, 2012.

[14] N. Stakhanova, S. Basu, and J. Wong, "A cost-sensitive model for preemptive intrusion response systems," in *Proceedings of the 21st International Conference on Advanced Information Networking and Applications (AINA '07)*, pp. 428–435, Washington, DC, USA, May 2007.

[15] W. Lee, W. Fan, M. Miller, S. J. Stolfo, and E. Zadok, "Toward cost-sensitive modeling for intrusion detection and response," *Journal of Computer Security*, vol. 10, no. 1-2, pp. 5–22, 2002.

[16] http://wiki.eclipse.org/DSDP/TCF.

[17] https://help.ubuntu.com/community/AppArmor/.

[18] http://www.nsa.gov/research/selinux/.

[19] N. Ezzati-Jivan and M. Dagenais, "A stateful approach to generate synthetic events from kernel traces," *Advances in Software Engineering*, vol. 2012, Article ID 140368, 12 pages, 2012.

[20] H. Waly and B. Ktari, "A complete framework for kernel trace analysis," in *Proceedings of the 24th Canadian Cference on Electrical and Computer Engineering (CCECE '11)*, pp. 1426–1430, Niagara Falls, ON, Canada, May 2011.

[21] N. Ezzati-Jivan and M. Dagenais, "A framework to compute statistics of system parameters from very large trace files," *ACM SIGOPS Operating Systems Review*, vol. 47, no. 1, pp. 43–54, 2013.

[22] H. Debar, D. Curry, and B. Feinstein, "The intrusion detection message exchange format," http://www.ietf.org/rfc/rfc4765.txt.