Volume 4, Issue 6, June 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Self Protection Using Access Detection

Priyadarshini Patil, A. K. Bongale
Department of Computer Engineering
Dr.D.Y.Patil College of Engineering
Ambi, Pune, India

Abstract*— The complexity of today's distributed computing environment is such that the presence of bugs and security holes is statistically unavoidable. A very promising approach to the present issue is to implement a self-protected system. Self-protection refers to the ability for a system to detect illegal behaviors. This article demonstrates the implementation of self-protection manager which targets clustered distributed systems. Our approach is based on the global database of the clustered distributed applications. This knowledge permits to detect known and unknown attacks if an prohibited (illegal) access is performed. The prototype is designed using access detection.

***Keywords*— Cluster security, self-protection, LAN device**

I. INTRODUCTION

The assumptions correspond to the point of view of a machine provider which rents his cluster infrastructure to different customers. It is assumed that each customer has a set of machines completely allocated to the applications. However, the native network and the Internet access are shared by all the applications. Therefore, the threat may arrive from outside of the cluster through the web. The main aim of this paper is to present the improved method for self-protected system in the context of cluster-based applications. It is considered that the hardware environment is composed of a cluster of machines interconnected through a local area network with an Inter net access via a router. The software environment is composed of a set of application components deployed on the cluster.

The approach is based on the access provided to the particular user in a cluster to access a particular process. Any attempt to use a process which is not allowed for a particular user is trapped and the access to that process is rejected. Legal access for different process to different users in the cluster is maintained by the Deployment Manager. The main characteristics of the system are: 1) to reduce the perturbation on the managed system whereas providing high reactivity, 2) to change the configuration (and reconfiguration) of security components when the system evolves, and 3) to keep the protection manager (which implements the protection policy) independent from the protected legacy system. The purpose of the work is not to replace the existing tools but rather to provide a systematic approach that allows more closely-coupled interactions between them, so that the cluster wide, coordinated reaction against an attack can become automated, and thus, more efficient. The main limitation relates to the scope of the detected attacks and to the allowed process; the current system can only detect attacks which use illegal process based on the information in the global Catalog. In order to validate our approach, we applied it to the self-protection of a cluster of machines. The remainder of the article is organized as follows: Section 2 presents the related work. Section 3 presents our Implementation details. Section 4 presents Flowchart. The evaluation is reported in Section 5. We conclude in Section 6.

II. LITERATURE SURVEY

This section reviews the main tools and techniques currently used by security experts to fight against intrusions and the existing systems which implement a self-protected behavior.

A. Intrusion Detection

In [20], two main approaches have been explored to ensure intrusion detection: misuse intrusion detection and anomaly intrusion detection. These approaches have been used in the case of Firewalls and Intrusion Detection Systems (IDS).

Snort[19] is an example of such systems. This approach induces alittle range of false-positives however cannot notice unknown attacks. Anomaly intrusion detection tries to identify irregular behaviors of the system by shaping the traditional behavior of the system (instead of attacks). The system is discovered and any misdeed is signaled. An early work [6] modelled and verified behavior correctness at the level of system calls. Recent examples of anomaly-based detection can be found in [17], [8], [9], and [5]. This approach can detect unknown attacks but at the price of a lot of false positives.

B. Backtracking Tools

In [14], Backtracking tools record detailed data about the system activity so that once an intrusion attempt has been detected; it is possible to determine the sequence of events that led to the intrusion and the potential extent of the damage (e.g., data theft/loss).

The Taser system [10] provides the ability to restore the system in a trusted state. It enhances the file system with a selective self-recovery capability.

C. Self-Protected Systems

Self-protected systems avoid miss communication between systems and provide security to the system. Self-protected systems are systems which are able to autonomously fight back intrusions in real time.

Rootsense [15] is an example of self-protected system. It differs from classical IDS within the sense that it detects and blocks intrusions in real-time. It audits events within different level of the host operating system and correlates them to comprehensively capture the global system state.

MLIDS [1] (multilevel intrusion detection system) is another example of self-protected system. MLIDS automates the detection of network attacks and proactively protect against them. The Self-cleansing system (SCS) [11] is another solution to build self-protected software.

It targets replicated servers which are stateless(e.g., web servers) involving a load-balancing strategy. This bearish approach makes the assumption that all intrusions cannot be detected and blocked. In fact, after a certain time, the system is considered to be compromised. Hence, it periodically reinstalls a part of the system from a secure repository. However, the solution only applies to stateless components.

However the self-protected tools are invaluable for system administrators as they are not powerful enough to ensure good levels of security, for several reasons. First of all, most detectors can only protect the system against known attacks. Therefore, pirates are always a length ahead with the resort to new “exploits”, which are able bypass filters and scanners. The purpose of our work is not to replace the existing tools but rather to provide a systematic approach that allows more closely-coupled interactions between them, so that the cluster-wide, coordinated reaction against an attack can become automated, and thus, more efficient.

D. Summary

From this work, it is analyzed that a self-protected system should be 1) be fully automated both in its configuration and its reaction to intrusions, 2) fire near-zero false positive since the response is automated, and 3) induce a low-performance overhead on an application performance to enable real-time protection.

III. IMPLEMENTATION DETAILS

The approach is based on the access provided to the particular user in a cluster to access a particular process. Any attempt to use a process which is not allowed for a particular user is trapped and the access to that process is rejected. Legal access for different process to different users in the cluster is maintained by the Deployment Manager. Such applications are generally organized in 3 tiers: a presentation tier, a business tier (optional), and a database tier. This organization is summarized as follows:

Presentation Tier. The function of the presentation tier to manage the execution of Servlets, which drive the execution of the application and synthesize its results in the form of dynamic pages.

Application Tier (optional). The function of the business tier is. to implement the application logic (data access and processing) if it is not directly implemented in the presentation tier.

Database Tier. The function of the database tier) is to provide persistent storage and access functions for the information needed by the application.

The different tiers may run on distinct nodes and be replicated for increased performance and robustness.

A.Deployment Manager

Role of Deployment Manger is to create global database and if required maintain the database. Global database will contain information regarding i) the number of machines in the cluster , ii) the number of process in each machine in the cluster, iii) the user groups allowed / not allowed for each user. Deployment Manager Collects information of machines in clusters and identifies the processes on each individual machines. Identify user groups that allowed access for these process and prepares a global database.

B. Self Protection Manager

Role of Self Protection is to fire Query to the machines which are in the cluster. Query will be asking for processes running in the machine under the specific user.

In reply to the query fired by self-protection manager, the machine replies with the processes running under the specific user. Self-Protection manger in turn will verify the reply given by machine with the global database.

When an illegal access of the processes from an undefined user is detected, the self-protection manager quickly stops that request.

C. Proposed Algorithm

1. Start New thread to get relationship information.
2. Get machine information().
3. Add these machines in queue.
4. Start new thread for Queue processor
5. Stop

Relationship Information
```
1. While (true)
2. Get all relation information from database
3. Create hashtable
4. Sleep()
```

Queue Processor
```
1. While (true)
2. If queue. count>0
3. Get machine from queue
4. Remove from queue
5. Start new thread get process and user information
6. Sleep()
```

Get process and user information
```
1. getWMIConnection()
2. Getallprocess()
3. For each process
4. Get belonging user
5. Check if allowed()
6. If allowed
7. Do nothing
8. Else
9. Kill process
10. Add item in queue
```

Get WMI Connection
```
1. If available in pool(Machine name)
2. Return
3. else
4. Create Connection
5. Add in pool
6. Return
7. End if
```

Check if allowed
```
1. Check hashtable for entry
2. If present
3. Return true
4. Else
5. Return false
```

D. WMI

WMI is used is to get information about the process. The purpose of WMI is to define a proprietary set of environment-independent specifications which allow management information to be shared between management applications. WMI prescribes enterprise management standards and related technologies for Windows that work with existing management standards, such as Desktop Management Interface (DMI) and SNMP. WMI complements these other standards by providing a uniform model. This model represents the managed environment through which management data from any source can be accessed in a common way. Windows Management Instrumentation (WMI) is the infrastructure for management data and operations on Windows-based operating systems. We can write WMI scripts or applications to automate administrative tasks on remote computers but WMI also supplies management data to other parts of the operating system and products, for example System Center Operations Manager, formerly Microsoft Operations Manager (MOM), or Windows Remote Management.

The ability to obtain management data from remote computers is what makes WMI useful. Remote WMI connections are made through DCOM. An alternative is to use Windows Remote Management, which obtains remote WMI management data using the WS-Management SOAP-based protocol.

E. Relevant Mathematics associated with the project

System:

$S = \{ M,P,U \}$

$M = \{M_1, M_2 \ldots\ldots M_m\}$ = Machines

$p = \{p_1,p_2 \ldots\ldots p_n \}$ = Processes

$U = \{ U_1, U_2 \ldots\ldots U_n \}$ = Users

Input:
$M = \{M_1, M_2 \ldots\ldots M_m\}$
$p = \{p_1, p_2 \ldots\ldots p_n\}$

Ouput:
Average number of processes running on single machine
$P = m \times n$ (1)
where m is number of machines and n is number of processes.

Detection time for Single Machine is
$T_D = T_{m1} + t_p$ (2)
where T_{m1} is time required for connection of machine
and t_p is time required to fetch process and their respective users.

Total Detection time for every single fetch on each machine is
$T = (T_{m1} + t_p) + (T_{m2} + t_p) + \ldots\ldots\ldots + (T_{mn} + t_p)$ (3)

$T \propto P$ and $T \propto M_m$

As multithreding is used total time will be

$T_n = \text{avg}(T_{m1+} T_{m2+\ldots\ldots+} T_{mn}) + t_p$

F. Control Loop Reactivity

This experience evaluates the time between the detection illegal access and the termination of compromised process. Our objective is to keep the time delay at its minimum level.

Below Figure demonstrates the time required to check process running concurrently under different machines. First it will be the learning curve to setup connection between different machine , this is incremental as number of machine will increase. As per design it will store these connection in connection pool and use same connection time for future fetches. This will save connection time for future fetches. So graph will be nearly stable, with a very little inclinment after learning curve.

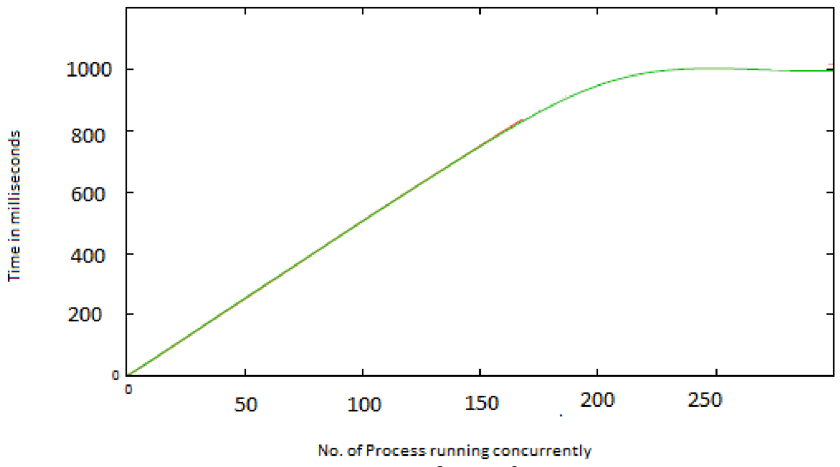

Fig 1. Result Graph

IV. SCENARIO BASED RESULT ANALYSIS

	Communication Channel	Access Detection
External Threats	Yes	Yes
Internal Threats	No	Yes
Centralized Configuration	Yes	Yes

V. CONCLUSION

Today, distributed computing environments are increasingly complex and difficult to administrate. This complexity is such that the presence of bugs and security holes is statistically unavoidable. Therefore, access control policies become very difficult to specify and to enforce. Following the autonomic computing vision, a very promising approach to deal with this issue is to implement a self-protected system which is able to distinguish legal (self) from illegal (nonself) operations.

The detection of an illegal behavior triggers a counter-measure to isolate the compromised resources and prevent further damages. In this we have designed and implemented self-protection system whose main characteristics are: 1) to reduce the perturbation on the managed system whereas providing high reactivity, 2) to change the configuration (and reconfiguration) of security components when the system evolves, and 3) to keep the protection manager (which implements the protection policy) independent from the protected legacy system. In this when an illegal access of the processes from an undefined user is detected, the self-protection manager quickly stops that request. For the moment the scope of detected attack is limited to illegal access to the processes. We are thus unable to spot intruders respecting the expected control flow or targeting different protocols. Our approach is well suited for windows platform, as we are using WMI for getting information about the process.

REFERENCES

[1] Noel De Palma, Daniel Hagimont, Fabienne Boyer, and Laurent Broto, "Self-Protection in a Clustered Distributed System", IEEE Transactions On Parallel And Distributed Systems, VOL. 23, NO. 2, FEBRUARY 2012.

[2] G.S. Blair, N. Bencomo, and R.B. France, "Models@ run.time," Computer, vol. 42, no. 10, pp. 22-27, Oct. 2009.

[3] E. Bruneton, T. Coupaye, M. Leclercq, V. Quema, and J.-B. Stefani, "The Fractal Component Model and Its Support in Java," Software— Practice and Experience, vol. 36, nos. 11/12, pp. 1257-1284, 2006.

[4] B.H.C. Cheng, P. Sawyer, N. Bencomo, and J. Whittle, "A Goal-Based Modeling Approach to Develop Requirements of an Adaptive System with Environmental Uncertainty," Proc. ACM/IEEE Int'l Conf. Model Driven Eng. Languages and Systems, 2009.

[5] L. Ertoz, E. Eilertson, A. Lazarevic, P. Tan, J. Srivastava, V. Kumar, and P. Dokas, The MINDS-Minnesota Intrusion Detection System Next Generation Data Mining. MIT Press, 2004.

[6] S. Forrest, S.A. Hofmeyr, A. Somayaji, and T.A. Longstaff, "A Sense of Self for Unix Processes," Proc. IEEE Symp. Research in Security and Privacy, 1996.

[7] S. Forrest, S.A. Hofmeyr, and A. Somayaji, "Computer Immunology," Comm. the ACM, vol. 40, no. 10, pp. 88-96, 1997.

[8] D. Gao, M.K. Reiter, and D. Song, "Behavioral Distance for Intrusion Detection," Proc. Eighth Int'l Symp. Recent Advances in Intrusion Detection (RAID '05), Sept. 2006.

[9] J.T. Giffin, D. Dagon, S. Jha, W. Lee, and B.P. Miller, "Environment-Sensitive Intrusion Detection," Proc. Int'l Symp. Recent Advances in Intrusion Detection, Sept. 2005.

[10] A. Goel, K. Po, K. Farhadi, Z. Li, and E. De Lara, "The Taser Intrusion Recovery System," Proc. 20th ACM Symp. Operating Systems Principles, 2005.

[11] Y. Huang and A. Sood, "Self-Cleansing Systems for Intrusion Containment," Proc. Workshop Self-Healing, Adaptive and Self-MANaged Systems, 2002.

[12] Sun Microsystems, Java 2 Platform Enterprise Ed. (J2EE), http://java.sun.com/j2ee/, 2011.

[13] J. Kephart, An Architectural Blueprint for Autonomic Computing. IBM White Paper, 2003.

[14] S.T. King and P.M. Chen, "Backtracking Intrusions," ACM Trans. Computer Systems, vol. 23, no. 1, pp. 51-76, 2005.

[15] R. Koller, R. Rangaswami, J. Marrero, I. Hernandez, G. Smith, M. Barsilai, S. Necula, and S. Masoud, "Anatomy of a Real-Time Intrusion Prevention System," Proc. Int'l Conf. Autonomic Computing, pp. 151-160, 2008.

[16] B. Morin, O. Barais, G. Nain, and J.-M. Jezequel, "Taming Dynamically Adaptive Systems Using Models and Aspects," Proc. IEEE Int'l Conf. Software Eng., 2009.

[17] D. Mutz, F. Valeur, C. Kruegel, and G. Vigna, "Anomalous System Call Detection," ACM Trans. Information and System Security, vol. 9, no. 1, pp. 61-93, Feb. 2006.

[18] S. Sicard, F. Boyer, and N. De Palma, "Using Components for Architecture-Based Management: The Self-Repair Case," Proc. Int'l Conf. Software Eng., 2008.

[19] M. Roesch, "Snort—Lightweight Intrusion Detection for Networks," Proc. Large Systems Administration Conf., Nov. 1999.

[20] A. Sundaram, "An Introduction to Intrusion Detection," ACM Crossroads Student Magazine, vol. 2, no. 4, pp. 3-7, 1996 S. Chen, B. Mulgrew, and P. M. Grant, "A clustering technique for digital communications channel equalization using radial basis function networks," *IEEE Trans. Neural Networks,* vol. 4, pp. 570–578, Jul. 1993.