International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-4, Issue-12)

Research Article | **December 2015**

Self Protection in Autonomic Computing

I Naveen
Research Scholar, CVR College Engineering,
Telangana, India

V Sowjanya
Assistant Professor, CVR College Engineering,
Telangana, India

Abstract–

Autonomic Computing systems are the self managing systems according to the goals designed by system administrator. New components in Autonomic Computing systems integrates as simple and effortlessly as a new cell establishes itself in human body. The increasing complexity of computer systems also increased the complexity and risk in securing precious assets of organizations, such as data, hardware, and network. In autonomic computing human intervention is reduced, hence the system must be designed to make strategic decisions to protect autonomously. The self-protecting mechanism must be able to protect the system, defence itself and also must be extensible and scalable to adapt the changes in computing system and high level security policies. This report presents a an algorithm for addressing the protection of the autonomic elements of autonomic computing systems.

Keywords– Autonomic computing, self protection, self awareness, context awareness, Self-Configuration, Self-Optimization, Self-Healing.

I. INTRODUCTION

In mid-October 2001, IBM released a manifesto observing that the main obstacle to further progress in the IT industry is a looming software complexity crisis. The company cited applications and environments that weigh in at tens of millions of lines of code and require skilled IT professionals to install, configure, tune, and maintain. The manifesto pointed out that the difficulty of managing today's computing systems goes well beyond the administration of individual software environments. The need to integrate several heterogeneous environments into corporate-wide computing systems, and to extend that beyond company boundaries into the Internet, introduces new levels of complexity.

Computing systems' complexity appears to be approaching the limits of human capability, yet the march toward increased interconnectivity and integration rushes ahead unabated[1]. Autonomic computing aims at the construction of self-managing and self-adapting computing systems. Autonomic Computing systems are the self managing systems according to the goals designed by the system administrator. New components in Autonomic Computing systems integrates as simple and effortlessly as a new cell establishes itself in human body.

The idea of Autonomic Computing is not a fiction, is the grand challenge in creation self-managing and self-protecting computing systems. It has become an emerging research topic in the face of the ever-increasing complexity of computing systems and distributed environment. The presence of bugs and security holes are statistically unavoidable in complex distributed computing environment of today's world. Self-protected system is an approach to make the computing environment as safe and secure from the issues, such as bugs and security holes are to implement in complex computing environment.

1.1 Threat Model

To defend themselves, autonomic systems must be aware of the environment that surrounds them and their internal state. The behaviour and security policies for Autonomic Computing are defined through high level policies by the owners or system administrators of Computing System. If Attackers are capable to gain access and able to modify such policies, then the damage cause to such an environment potentially high as compared to the attacks that are perpetrated to non autonomic computing systems. Through self-awareness and context-awareness features, these systems can defend themselves by adhering to the embedded policies that have been set by their owners in order to assure proper operations. This means that self-aware and context-aware autonomic computing systems are able to provide the right services at the right time to the right clients. They can detect hostile intrusive behavior and take action to protect the integrity and operational status of the system. They can also correctly fulfil policy requirements for secure environments [3].

1.2 Frame work introduction/Introduction to proposal

This work introduces a framework for self-protection in autonomic computing. The main aim of the self protection frame work is to provide the security to the autonomic elements from internal and/or external threats and to provide the security to high-level security policies that have been set by the system administrators and/or owners of the computing system.

1.3 Organization of the report

Section 2 describes, what is Autonomic System? what is the need for Autonomic System? and also the aspects to be achieved in establishing the fully Autonomic System. The work related to self-protection, which was already carried out in Autonomic Computing Systems discusses in section 3. The proposed method of Self-Protection in Autonomic Systems introduced and described in detail in section 4. Section 5 concludes the work and gives the directions for future work.

II. PRELIMINARIES

As systems become more interconnected and diverse, architects are less able to anticipate and design interactions among components, leaving such issues to be dealt with at runtime. Soon systems will become too massive and complex for even the most skilled system integrators to install, configure, optimize, maintain, and merge. And there will be no way to make timely, decisive responses to the rapid stream of changing and conflicting demands[1].

The suitable option to handle the complex computing system is autonomic computing-computing systems that can manage themselves given high-level objectives from administrators. IBM's senior vice president of research, Paul Horn, introduced this idea to the National Academy of Engineers at Harvard University in a March 2001 keynote address, he deliberately chose a term with a biological connotation. Autonomic computing is a grand challenge that reaches far beyond a single organization. Its realization will take a concerted, long term, worldwide effort by researchers in a diversity of fields. A necessary first step is to examine this vision: what autonomic computing systems might look like, how they might function, and what obstacles researchers will face in designing them and understanding their behavior.

The essence of autonomic computing systems is Self-Management, the intent of which is to free system administrators from the details of system operation and maintenance and to provide users with a machine that runs at peak performance 24/7.

Four aspects of self-management are [1]:

- ✓ Self-Configuration
- ✓ Self-Optimization
- ✓ Self-Healing
- ✓ Self-Protection

2.1 Self-Configuration

Installing, configuring, and integrating large, complex systems is challenging, time-consuming, and error-prone even for experts. Most large Web sites and corporate data centers are haphazard accretions of servers, routers, databases, and other technologies on different platforms from different vendors. It can take teams of expert programmers months to merge two systems or to install a major e-commerce application such as SAP.

Autonomic systems will configure themselves automatically in accordance with high-level policies representing business-level objectives, for example—that specify what is desired, not how it is to be accomplished.

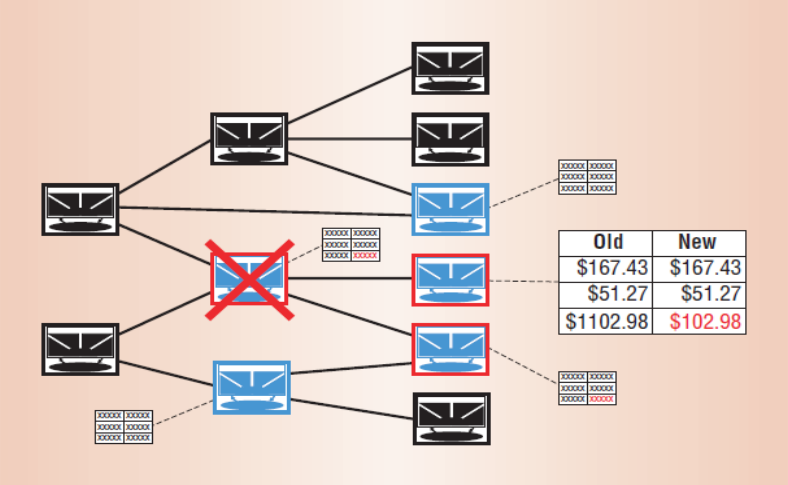

Fig 1: Problem diagnosis in an autonomic system upgrade.

When a component is introduced, it will incorporate itself seamlessly, and the rest of the system will adapt to its presence, much like a new cell in the body or a new person in a population.

For example, when a new component is introduced into an autonomic accounting system, as in Figure 1, it will automatically learn about and take into account the composition and configuration of the system. The upgrade introduces five software modules (blue), each an autonomic element. Minutes after installation, regression testers find faulty output in three of the new modules (red outlines), and the system immediately reverts to its old version. A problem determiner, an autonomic element, obtains information about inter element dependencies (lines between elements) from a dependency analyzer, another autonomic element that probes the system periodically (not shown).

Taking into account its knowledge of inter element dependencies, the problem determiner analyzes log files and infers which of the three potentially bad modules is the culprit (red X). It generates a problem ticket containing diagnostic information and sends it to a software developer, who debugs the module and makes it available for future upgrades.

2.2 Self-optimization

Complex middleware, such as WebSphere, or database systems, such as Oracle or DB2, may have hundreds of tuneable parameters that must be set correctly for the system to perform optimally, yet few people know how to tune

them. Such systems are often integrated with other, equally complex systems. Consequently, performance-tuning one large subsystem can have unanticipated effects on the entire system. Autonomic systems will continually seek ways to improve their operation, identifying and seizing opportunities to make themselves more efficient in performance or cost.

Just as muscles become stronger through exercise, and the brain modifies its circuitry during learning, autonomic systems will monitor, experiment with, and tune their own parameters and will learn to make appropriate choices about keeping functions or outsourcing them. They will proactively seek to upgrade their function by finding, verifying, and applying the latest updates.

2.3 Self-healing

IT vendors have large departments devoted to identifying, tracing, and determining the root cause of failures in complex computing systems. Serious customer problem scan take teams of programmers several weeks to diagnose and fix, and sometimes the problem disappears mysteriously without any satisfactory diagnosis. Autonomic computing systems will detect, diagnose, and repair localized problems resulting from bugs or failures in software and hardware, perhaps through a regression tester, as in Figure 1. Using knowledge about the system configuration, a problem diagnosis component would analyze information from log files, possibly supplemented with data from additional monitors that it has requested.

The system would then match the diagnosis against known software patches (or alert a human programmer if there are none), install the appropriate patch, and retest.

2.4 Self-protection

Despite the existence of firewalls and intrusion detection tools, humans must at present decide how to protect systems from malicious attacks and inadvertent cascading failures.

Autonomic systems will be self-protecting in two senses. They will defend the system as a whole against large scale, correlated problems arising from malicious attacks or cascading failures that remain uncorrected by self-healing measures. They also will anticipate problems based on early reports from sensors and take steps to avoid or mitigate them.

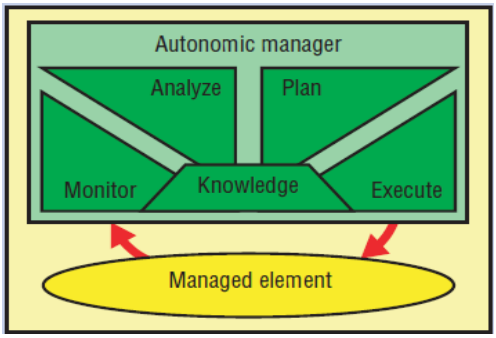

Fig 2: Structure of an autonomic element.

Elements interact with other elements and with human programmers via their autonomic managers. Each autonomic element will be responsible for managing its own internal state and behavior and for managing its interactions with an environment that consists largely of signals and messages from other elements and the external world.

III. LITERATURE SURVEY

The notion of embedding policies into computer systems has been applied to several domains. Many commercial software scanning and monitoring tools have been developed. Their major deficiency is that they run independently of the system they monitor. In addition, the policies are under the control of the system administrator who can change their parameters at will making the system vulnerable to attacks. In [5], the authors addressed the concept of trust as an important factor in the success of new autonomic features and in [3] the authors developed a security model based on the context in which the autonomic system functions. In addition, they believe that the implementation of autonomic computing techniques offer the promise of making systems more secure by effectively and automatically enforcing high-level security policies.

G. Jabbour and D.A. Menasce proposed a two-fold framework for securing autonomic elements of autonomic computing. The approach in this framework is twofold. The first is to partition the building blocks of a security policy into several layers thus adding complexity to the overall architecture of the policy. This serves the purpose of adding perceived ambiguity, which is intended to retract potential hackers from deciphering the policy's contents and mandates.

The second is to inject into the different partitions of the policy false elements (parameters and their values) whose characteristics have the sole purpose of confusing a hacker and giving a false sense of achieving his/her objective[2].

The four main aspects of the Securing the Security Policy (SSP) architecture presented are: partitioning & obfuscation, verification, awareness, and monitoring. This approach mainly focusing on securing the security policy by making fields as redundant in defining security policy. The attacker may not be stopped from attacking onto autonomic element. And security breach to the security policy will make the entire system as a compromised system.

IV. PROPOSED APPROACH

System security and network security are vital parts of any autonomic computing solution. The ability of a system to react consistently and correctly to situations ranging from benign but unusual events to outright attacks is key to the achievement of the goals of self-protection, self-healing, and self-optimization.

In this work, we propose a step by step algorithmic approach to develop Self-Protection in Autonomic Computing Systems.

Algorithm for implementation Self-Protection:

1. Determine the high-level security policies.
2. Create the Log file.
3. Create the log record for every operation/ transaction made by the Autonomic Element.
4. Monitor Node Behaviour (Autonomic Manager continuously monitors the node).
5. Differentiate the Node normal behaviour with abnormal behaviour by observing the node log file.
6. If in appropriate behaviour identified, then
    - 6.1 Isolate the element from complex Autonomic system.
    - 6.2 Diagnose the problem.
    - 6.3 Restore the node back into Autonomic system.

Self Protection Element: The computing elements are designed to protect themselves from security threats. The elements behaves as per the high level policies defined by the system users/ administrators/ owner.

Security Policy: The security policies are high level policies, which are defined by Owner/ Administrator to determine, who are permitted to access which resources and also the type of operations permitted on various elements of Complex Computing system.Autonomic Manager: Autonomic Manager monitors the node behaviour continuously, which helps to identify unauthorised/ inappropriate access to the elements of Autonomic system. The Autonomic Manager also establishes the communication with Autonomic elements. The Autonomic Manager communicates with other autonomic elements also with human beings to interchange the information about node behaviour and/or actions to be taken when inappropriate behaviour is identified in a node or in its neighbouring nodes.

Log file: Each Autonomic Element maintains a log file to record the operations performed. The log file helps the computing environment to identify the threats. The log file must be kept in a secure manner and should not permit anyone to modify/ edit the log file.

Log Record: Maintains the information about a particular operation/ transaction such as resources (software and hardware) utilized for the operation, the time taken to perform the operation, the start time and end time, the user who initiated or requested for the operation.

Inappropriate behaviour: The monitoring element continuously monitoring the autonomic element, hence it can identify the Inappropriate behaviour of the node (if any).

Effected Autonomic Elements: The autonomic elements communicates with each other in Autonomic Computing system. Hence, the Monitoring Elements of Autonomic Elements will be able to know the behaviour of other Autonomic Elements, through which, the elements effected by the attack made by the attacker can be identified.

Isolate: Isolate the effected Autonomic Elements from Autonomic Computing system, that is, the affected autonomic elements are separated from complex autonomic computing system.

Identifying the cause: Scan the log file of autonomic elements which behaving in inappropriate manner to identify the root cause/ attack made by attacker for this erroneous behaviour.

Diagnose the problem: In this phase, diagnose the problem identified in autonomic element. That is, if any software or hardware components are infected, then rectify the problem. If attacker performs any attack on the element, then identify the type of attack and damage caused by the attack in the system.

Read the Log file to know modifications done by the attacker on the node, then undo the operations made on autonomic element by the attacker.

Restore: Once the autonomic element is repaired, ensure the element is restored to safe state. And then restore element back in to Computing System by establishing connections to this element from other elements of Computing system.

V. CONCLUSION AND FUTURE SCOPE

This work gives the introduction to the Autonomic Computing and describes the four major aspects of Autonomic Computing. In this work, we proposed an algorithmic approach for self protection of an autonomic element in abstract manner. The premise of proposed algorithm is mainly based on Autonomic Manger and Log File. The proposed algorithm is a generalised algorithm, which can be used to implement solutions for any type of problems.

The algorithmic approach can be further improved by defining each step in profoundly with appropriate tasks needs to be carried at every step. A framework can be designed with this algorithmic approach to make the system as self protecting system. The suitable algorithm/ framework can be designed exclusively to address a particular type of attacks in Autonomic Systems.

REFERENCES

[1] Jeffrey O. Kephart, David M. Chess "The Vision of Autonomic Computing", IEEE Computer Society, January 2003

[2] G. Jabbour and D.A. Menasce, "Securing Security Policies in Autonomic Computing Systems", International Conference on Security and Management, Las Vegas, Nevada, USA, July-2008.

[3] Kaiyu Wan and Vasu Alagar, "Security Contexts in Autonomic Systems", IEEE, 2006.

[4] D. M. Chess, C. C. Palmer, S. R. White, "Security in an autonomic computing environment", IBM Systems Journal, Vol 42, No 1, 2003.

[5] Duez, P.P., M.J. Zuliani, and G.A. Jamieson, "Trust by Design: Information Requirements for Appropriate Trust in Automation" Pierre Guez, Greg Jamieson and IBM Canada Ltd., 2006.

