ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Self Adaptive Security Configurations in Dynamic Decision Networks

G.Kishore
M.Tech (CSE),
Department of CSE,
AITS, Tirupati.

D.Murali, (Ph.D)
Associate Professor & HoD
Department of CSE,
AITS Tirupati.

ABSTRACT

Cloud security is one of most important issues that have attracted a lot of research and development effort in past few years. Particularly, attackers can explore vulnerabilities of a cloud system and compromise virtual machines to deploy further large-scale Distributed Denial-of-Service (DDoS). DDoS attacks usually involve early stage actions such as multi-step exploitation, low frequency vulnerability scanning, and compromising identified vulnerable virtual machines as zombies, and finally DDoS attacks through the compromised zombies. Within the cloud system, especially the Infrastructure-as-a-Service (IaaS) clouds, the detection of zombie exploration attacks is extremely difficult. This is because cloud users may install vulnerable applications on their virtual machines. To prevent vulnerable virtual machines from being compromised in the cloud, we propose a multi-phase distributed vulnerability detection, measurement, and countermeasure selection mechanism called Self Adaptive Security, which is built on attack graph based analytical models and reconfigurable virtual network-based countermeasures. The proposed framework leverages Open Flow network programming APIs to build a monitor and control plane over distributed programmable virtual switches in order to significantly improve attack detection and mitigate attack consequences. The system and security evaluations demonstrate the efficiency and effectiveness of the proposed solution.

Index terms- ***DDoS, IaaS, APIs***

I INTRODUCTION

The protection of computer based resources that include hardware, software, data, procedures and people against unauthorized use or natural Disaster is known as System Security, System Security can be divided into four related issues: Security, Integrity, Privacy, and Confidentiality.

SYSTEM SECURITY: Refers to the technical innovations and procedures applied to the hardware and operation systems to protect against deliberate or accidental damage from a defined threat. Data Security is the protection of data from loss, disclosure, modification and destruction. System Integrity Refers to the power functioning of hardware and programs, appropriate physical security and safety against external threats such as eavesdropping and wiretapping. Privacy Defines the rights of the user or organizations to determine what information they are willing to share with or accept from others and how the organization can be protected against unwelcome, unfair or excessive dissemination of information about it. Confidentiality is a special status given to sensitive information in a database to minimize the possible invasion of privacy. It is an attribute of information that characterizes its need for protection.

II EXISTING SYSTEM

Cloud users can install vulnerable software on their VMs, which essentially contributes to loopholes in cloud security. The challenge is to establish an effective vulnerability/attack detection and response system for accurately identifying attacks and minimizing the impact of security breach to cloud users. In a cloud system where the infrastructure is shared by potentially millions of users, abuse and nefarious use of the shared infrastructure benefits attackers to exploit vulnerabilities of the cloud and use its resource to deploy attacks in more efficient ways. Such attacks are more effective in the cloud environment since cloud users usually share computing resources, e.g., being connected through the same switch, sharing with the same data storage and file systems, even with potential attackers. The similar setup for VMs in the cloud, e.g., virtualization techniques, VM OS, installed vulnerable software, networking, etc., attracts attackers to compromise multiple VMs.

Disadvantages:

1. No detection and prevention framework in a virtual networking environment.
2. Not accuracy in the attack detection from attackers.

III PROPOSED SYSTEM

In this article, we propose SELF ADAPTIVE SECURITY to establish a defense-in-depth intrusion detection framework. For better attack detection, SELF ADAPTIVE SECURITY incorporates attack graph analytical procedures into the intrusion detection processes. We must note that the design of SELF ADAPTIVE SECURITY does not intend to improve any of the existing intrusion detection algorithms; indeed, SELF ADAPTIVE SECURITY employs a reconfigurable virtual networking approach to detect and counter the attempts to compromise VMs, thus preventing zombie VMs.

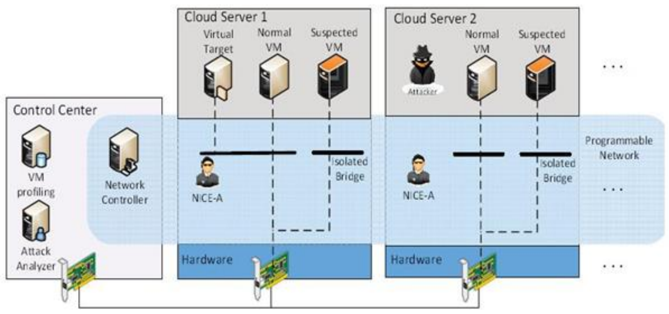

Figure 1: Architecture with one cloud server cluster

1. Implementation

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

2. User Module:

In this module, Users are having authentication and security to access the detail which is presented in the ontology system. Before accessing or searching the details user should have the account in that otherwise they should register first.

3. Countermeasure Selection:

Countermeasure Selection To illustrate how SELF ADAPTIVE SECURITY works, let us consider for example, an alert is generated for node 16 (*vAlert* = 16) when the system detects LICQ Buffer overflow. After the alert is generated, the cumulative probability of node 16 becomes 1 because that attacker has already compromised that node. This triggers a change in cumulative probabilities of child nodes of node 16. Now the next step is to select the countermeasures from the pool of countermeasures *CM*.

4. Attack Analyzer:

The major functions of SELF ADAPTIVE SECURITY system are performed by attack analyzer, which includes procedures such as attack graph construction and update, alert correlation and countermeasure selection. The process of constructing and utilizing the cenario Attack Graph (*SAG*) consists of three phases: information gathering, attack graph construction, and potential exploit path analysis. With this information, attack can be modeled using SAG. Each node in the attack graph represents an exploit by the attacker. Each path from an initial node to a goal node represents a successful attack.

5. False Alarms:

A cloud system with hundreds of nodes will have huge amount of alerts raised by Snort. Not all of these alerts can be relied upon, and an effective mechanism is needed to verify if such alerts need to be addressed. Since Snort can be programmed to generate alerts with CVE id, one approach that our work provides is to match if the alert is actually related to some vulnerability being exploited. If so, the existence of that vulnerability in SAG means that the alert is more likely to be a real attack. Thus, the false positive rate will be the joint probability of the correlated alerts, which will not increase the false positive rate compared to each individual false positive rate. Moreover, we cannot keep aside the case of zero day attack where the vulnerability is discovered by the attacker but is not detected by vulnerability scanner. In such case, the alert being real will be regarded as false, given that there does not exist corresponding node in SAG. Thus, current research does not address how to reduce the false negative rate. It is important to note that vulnerability scanner should be able to detect most recent vulnerabilities and sync with the latest vulnerability database to reduce the chance of Zero-day attacks.

IV OPERATIONAL FEASIBILITY

Operational feasibility is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. Schedule feasibility A project will fail if it takes too long to be completed before it is useful. Typically this means estimating how long the system will take to develop, and if it can be completed in a given time period using some methods like payback period. Schedule feasibility is a measure of how reasonable the project timetable is. Given our technical expertise, are the project deadlines reasonable? Some projects are initiated with specific deadlines. You need to determine whether the deadlines are mandatory or desirable.

1. Technical Feasibility:

Technology and system feasibility. The assessment is based on an outline design of system requirements in terms of Input, Processes, Output, Fields, Programs, and Procedures. This can be quantified in terms of volumes of data, trends, frequency of updating, etc. in order to estimate whether the new system will perform adequately or not. Technological feasibility is carried out to determine whether the company has the capability, in terms of software, hardware, personnel and expertise, to handle the completion of the project. When writing a feasibility report the following should be taken to consideration:

- A brief description of the business
- The part of the business being examined
- The human and economic factor
- The possible solutions to the problems

At this level, the concern is whether the proposal is both *technically* and *legally* feasible (assuming moderate cost).

2. Feasibility Study:

Feasibility studies aim to objectively and rationally uncover the strengths and weaknesses of the existing business or proposed venture, opportunities and threats as presented by the environment, the resources required to carry through, and ultimately the prospects for success. In its simplest term, the two criteria to judge feasibility are cost required and value to be attained. As such, a well-designed feasibility study should provide a historical background of the business or project, description of the product or service, accounting statements, details of the operations and management, marketing research and policies, financial data, legal requirements and tax obligations. Generally, feasibility studies precede technical development and project implementation.

3. Economical Feasibility:

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system. More commonly known as cost/benefits analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system. An entrepreneur must accurately weigh the cost versus benefits before taking an action. Cost-based study: It is important to identify cost and benefit factors, which can be categorized as follows: 1. Development costs; and 2. Operating costs. This is an analysis of the costs to be incurred in the system and the benefits derivable out of the system. Time-based study: This is an analysis of the time required to achieve a return on investments. The future value of a project is also a factor.

V SYSTEM DESIGN

It is an UML diagramming application written in Java and released under the open source Eclipse public license. By virtue of being a java application, it is available on any platform supported by Java. Argo UML does not yet completely implement the UML standard.

1. Unified Modeling Language Diagrams: The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

2. User Model View: This view represents the system from the user's perspective. The analysis representation describes a usage scenario from the end-users perspective.

3. Structural model view: In this model the data and functionality are arrived from inside the system. This model view models the static structures

4. Behavioral Model View: It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

5. Implementation Model View: In this the structural and behavioral as parts of the system are represented as they are to be built.

6. Environmental Model View: In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

UML is specifically constructed through two different domains they are

1. UML Analysis modeling, this focuses on the user model and structural model views of the system.
2. UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

In UML has 14 types of diagrams divided into two categories. Seven diagram types represent structural information, and the other seven represent general types of behavior, including four that represent different aspects of interactions. UML is a notation that resulted from the unification of Object Modeling Technique and Object Oriented Software Technology .UML has been designed for broad range of application.

VI CLASS DIAGRAM

1. Identification of analysis classes: A class is a set of objects that share a common structure and common behavior (the same attributes, operations, relationships and semantics). A class is an abstraction of real-world items. There are 4 approaches for identifying classes:

1. Noun phrase approach
2. Common class pattern approach.
3. Use case Driven Sequence or Collaboration approach.
4. Classes , Responsibilities and collaborators Approach

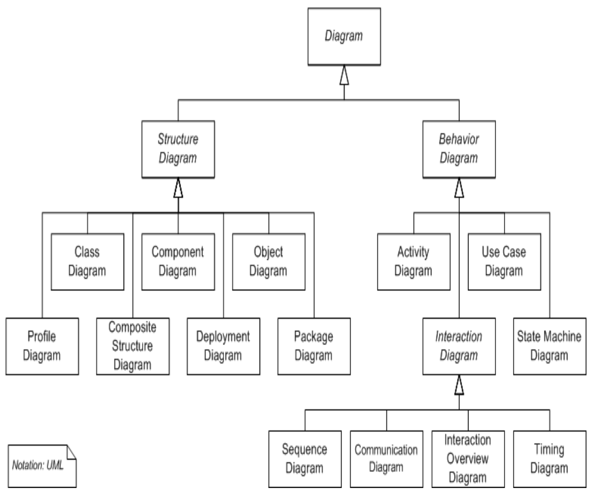

Figure 2: UML design modeling

2. Noun Phrase Approach: The guidelines for identifying the classes:

a. Look for nouns and noun phrases in the use cases.

b. Some classes are implicit or taken from general knowledge.
c. All classes must make sense in the application domain; Avoid computer implementation classes – defer them to the design stage.
d. Carefully choose and define the class names.

After identifying the classes we have to eliminate the following types of classes:

- Redundant classes
- Adjective classes

3. Common class pattern approach: The following are the patterns for finding the candidate classes:
a. Concept class.
b. Events class.
c. Organization class
d. Peoples class
e. Places class
f. Tangible things and devices class.

4. Use case driven approach: We have to draw the sequence diagram or collaboration diagram. If there is need for some classes to represent some functionality then add new classes which perform those functionalities.

5. CRC approach: The process consists of the following steps:
a. Identify classes' responsibilities and identify the classes
b. Assign the responsibilities
c. Identify the collaborators.

6. Super-sub class relationships: Super-sub class hierarchy is a relationship between classes where one class is the parent class of another class (derived class).This is based on inheritance.

7. Guidelines for identifying the super-sub relationship, a generalization are:

i. Top-down: Look for noun phrases composed of various adjectives in a class name. Avoid excessive refinement. Specialize only when the sub classes have significant behavior.

ii. Bottom-up: Look for classes with similar attributes or methods. Group them by moving the common attributes and methods to an abstract class. You may have to alter the definitions a bit.

iii. Reusability: Move the attributes and methods as high as possible in the hierarchy.

iv. Multiple inheritances: Avoid excessive use of multiple inheritances. One way of getting benefits of multiple inheritances is to inherit from the most appropriate class and add an object of another class as an attribute.

v. Aggregation or a-part-of relationship: It represents the situation where a class consists of several component classes. A class that is composed of other classes doesn't behave like its parts. It behaves very difficultly. The major properties of this relationship are transitivity and anti symmetry.

There are three types of aggregation relationships. They are:

1. Assembly: It is constructed from its parts and an assembly-part situation physically exists.

2. Container: A physical whole encompasses but is not constructed from physical parts.

3. Collection member: A conceptual whole encompasses parts that may be physical or conceptual. The container and collection are represented by hollow diamonds but composition is represented by solid diamond.

VII CONCLUSION

In this paper, we have a tendency to bestowed SELF ADAPTIVE SECURITY, that is projected to find and mitigate cooperative attacks within the cloud virtual networking surroundings. SELF ADAPTIVE SECURITY utilizes the attack graph model to conduct attack detection and prediction. The projected resolution investigates the way to use the programmability of software package switches based mostly solutions to boost the detection accuracy and defeat victim exploitation phases of cooperative attacks. The system performance analysis demonstrates the feasibleness of SELF ADAPTIVE SECURITY and shows that the projected resolution will significantly cut back the chance of the cloud system from being exploited and abused by internal and external attackers.

REFERENCES

[1] H. Ziv, D. J. Richardson, and R. KlŽsch, "The uncertainty principle in software engineering," 19th International Conference on Software Engineering,ICSE'97, boston, Massachusetts, USA.

[2] N. Fenton, W. Marsh, M. Neil, P. Cates, S. Forey, and M. Tailor, "Making resource decisions for software projects," in Proceedings of the 26th International Conference on Software Engineering, ser. ICSE '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 397–406.[Online].Available: http://dl.acm.org/citation.cfm?id=998675.999444

[3] B. H. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, "Software engineering for self-adaptive systems: A research roadmap," in Software Engineering for Self-Adaptive Systems, B. H. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, Eds. Springer- Verlag, 2009, vol. 5525, pp. 1–26.

[4] J. Whittle, P. Sawyer, N. Bencomo, B. Cheng, and J.-M. Bruel, "Relax: a language to address uncertainty in self-adaptive systems requirement," Requirements Engineering, vol. 15, pp. 177–196, 2010.

[5] D. Garlan, "Software engineering in an uncertain world," in Proceedings of the FSE/SDP workshop on Future of software engineering research, ser. FoSER '10. New York, NY, USA: ACM, 2010, pp. 125–128. [Online]. Available: http://doi.acm.org/10.1145/1882362.1882389

[6] N. Esfahani and S. Malek, "Uncertainty in self-adaptive software systems," in Software Engineering for Self-Adaptive Systems 2 (SEfSAS 2). Springer-Verlag, 2012.

[7] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, Non-Functional Requirements in Software Engineering. Springer, 1999, vol. 5.

[8] H. J. Goldsby, P. Sawyer, N. Bencomo, D. Hughes, and B. H. Cheng, "Goal-based modeling of dynamically adaptive system requirements," in IEEE Int. Conference on the Engineering of Computer Based Systems (ECBS), 2008.

[9] K. Welsh, P. Sawyer, and N. Bencomo, “Towards requirements aware systems: Run-time resolution of design-time assumptions,” in ASE, 2011, pp. 560–563.

[10] A. Filieri, C. Ghezzi, and G. Tamburrelli, “A formal approach to adaptive software: continuous assurance of non-functional requirements,” Formal Asp. Comput., vol. 24, no. 2, pp. 163–186, 2012.

[11] S. J. Russell and P. Norvig, Artificial intelligence: A modern approach, 2nd ed., ser. Prentice Hall series in artificial intelligence. Prentice Hall, 2003.

[12] J. Bilmes and J. Bilmes, “On virtual evidence and soft evidence in bayesian networks,” 2004.

[13] J. Pearl, Probabilistic reasoning in intelligent systems: networks of plausible inference. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[14] R. Howard and J. Matheson., “Influence diagrams,” in Readings on the Principles and Readings on the Principles and Applications of Decision Analysis II. Menlo Park CA:: Strategic Decisions Group, 1984.

[15] K. Welsh and P. Sawyer, “Understanding the scope of uncertainty in dynamically adaptive systems,” in REFSQ, 2010.