Self-adaptive SSH Honeypot Model Capable of Reasoning

Adrian Pauna, Victor Valeriu Patriciu
Military Technical Academy
George Cosbuc Blvd, No. 81-83, Sect. 5, Bucharest, ROMANIA
adrian.pauna.ro@gmail.com, vip@mta.ro

Abstract

The increased number of SSH stolen credentials on the "black market" used for targeted attacks proves the opportunity of SSH Honeypots. Corroborated with recent trends in honeypot research area regarding self-adaptive systems, capable of interacting with attackers on their own, leads us to the necessity of self-adaptive SSH Honeypots. Several Artificial Intelligence techniques have been used in order to achieve "adaptability" such as Reinforcement Learning and Game theory. In this paper we propose a model that uses Case Based Reasoning techniques to enforce the adaptive capabilities of a SSH Honeypot.

KEYWORDS

Honeypot, case based reasoning, SSH, artificial intelligence, information security

1 INTRODUCTION

Recent reports on the proliferation of SSH brute force attacks with the scope of acquiring super user credentials that afterwards are sold on underground internet websites, present the picture of a system used for targeted attacks in which the work is done in subtasks [1]. From this finding we can depict two aspects: the first is the aspect regarding the process of collecting SSH credentials and the second in regards to the usage of these credentials for initiating targeted attacks that have as source the compromised machines. One of the demonstrated ways of learning as much as possible about these attacks has been the usage of honeypot systems.

Since the development of the first SSH honeypot [1] things have evolved relatively slow and the cornerstone has been represented by the development of Kippo. Kippo is a SSH honeypot developed to log brute force attacks and the entire shell interaction performed by the attacker [3].

Wagener et. al has proposed a model of a high interaction SSH honeypot capable of interacting with the attackers based on Reinforcement Learning techniques [4]. The SSH honeypot was developed under the form of a Linux operating system with a modified kernel running as User-mode Linux [5]. The honeypot system interacts with the attacker by issuing actions such as blocking commands, substituting the commands or insulting them. The agent has the ability to perform a set of actions (blocking, executing the command, returning errors, or insulting) in various situations (states). Each action is awarded with a positive or negative reward. The purpose of reinforcement learning is to find the optimal policy to select the most promising actions in given states [6].The honeypot system is modeled as a learning agent that was configured with two defined reward functions.

We propose in this paper a new approach in which we use Case Based Reasoning to leverage adaptation capabilities for an SSH honeypot system.

We consider this article adds two main contributions. First we propose and describe a new approach for adaptive honeypots that become capable of reasoning while interacting with an attacker. Then, we propose the usage of Case Based reasoning methods to let the honeypot decide which action to take. Afterwards we propose an adaption of existing SSH honeypot systems or a development of new ones so to achieve interaction capabilities.

Secondly, we describe the initial modeling of the CBR system that uses as input cases generated by an improved SSH honeypot.

The article is organized as follows: in Sect. 2 we present a short description of the Honeypot systems with an emphasis on the auto-adaptive and dynamic ones. In Sect. 3 we present the paradigm of Case Based reasoning relevant to our model and the jCOLIBRI [7] framework used to create the initial model of the CBR model is described in Sect. 4. We present elements of interest regarding the profiling of

attackers after the compromise of an SSH server in Sect. 5. We continue by presenting in Sect. 6 the basic composition of our proposed Honeypot CBR system followed by a more detailed workflow debrief of the entire model in Sect. 7 . We end up with the presentation in Sect. 8 of the simulations we achieved and wrapping up with conclusions and future work in Sect. 9 .

2 THE HONEYPOT SYSTEMS

The last years increasing level of cyber-attacks has raised the issue of whom, from where and how the hackers are acting. Having this in mind, the honeypot systems have proved to be a valuable resource on gathering the necessary information. But, since the level of expertise of the attackers has evolved and since their attacks have evolved in ones that are more targeted and more sophisticated, the need of systems that can unveil in a proper time and manner their techniques has become stringent.

Honeypot systems are already used on large scale to accomplish this task. Cyber-security researchers, antivirus developers, governments, internet service providers use honeypot systems to collect the necessary information about all the malware that spreads itself all over internet. From the perspective of the level of interaction until this moment the most common types of honeypot systems used are: low interaction honeypot systems and high interaction one. The difference between the two types of honeypot systems resides in the fact the low interaction honeypot systems emulate specific services , such as FTP, SSH, HTTP etc. and the high interaction ones offer to the attackers a complete operating system, usually virtualized. The common element for both of the honeypot systems it the increased level of logging that fulfills, basically, the main target of honeypot systems, the monitoring of the attackers.

One of the most recognized experts in the area of honeypot system, also a pioneer in this area, Lance Spitzner, has recently stated that the future of honeypot systems will be the dynamic and auto-adaptive honeypot system [8]. In his acceptance a honeypot system should adapt to the environment in which it resides. Starting from his idea there are two approaches of models of this kind of honeypot systems. One is the dynamic honeypot system that accommodates to the production network near which it is installed. The dynamicity it is accomplished by the possibility of exposing the services similar to the ones which are in place in the production network.

Basically if an organization has a production network that have exposed on the internet services such as HTTP, SSH, FTP then when the honeypot system is deployed , it scans the production network and discovers this online services and so it automatically generates them so that it will be exposed and logged every interaction with them.

The second approach involves using Artificial Intelligence algorithms, such as Reinforcement Learning, to leverage adaptation capabilities.

3 THE CASE BASED REASONING

Case Based Reasoning (CBR) combines problem-solving and learning and is able to use specific knowledge of previously experienced, concrete problem situations.

One of the primary motivations for CBR is related to cognitive science and basically is the desire to model Human Behavior. The second primary motivation is related to Artificial Intelligence (AI) and its scope is to make AI systems more effective.

Case-based reasoning tasks tend to be divided into two classes, interpretive and problem-solving. CBR Interpretive CBR makes use of prior cases as reference points for classifying or characterizing new situations; problem-solving CBR makes use of prior cases to suggest solutions that might apply to new circumstances [9].

In the case of our proposed Honeypot model we focus on the first motivation, so to be able to emulate the human interactions with an attacker.

In the development of a simple CBR application you have to follow several steps, such as:

- collecting background knowledge,
- modeling a suitable case representation,
- defining accurate similarity measures,
- implementing retrieval functionality, and
- implementing user interfaces.

In comparison with other AI approaches, CBR reduces significantly the effort required for knowledge acquisition and representation, which is why CBR applications have a big commercial success [10].

4 THE JCOLIBRI FRAMEWORK

jCOLIBRI is a JAVA framework that can be used for building Case Based Reasoning (CBR) systems. Designed using flexible and scalable architecture, jCOLIBRI is a "wide spectrum framework able to support several types of CBR systems: from simple applications based on retrieving using the nearest-neighbor approach, to knowledge-intensive ones with complex reuse and retain tasks" [11].
The JAVA framework provides a graphical interface that can be used as guidance for the configuration of a particular CBR system.
In order to build the Honeypot CBR system different tasks have been executed such as the definition of the case structure and of the similarities functions and the nevertheless defining the system's behavior.

5 THE SSH ATTACKER PROFILING

Ramsbrock et. al [12] proposes in his study a profile of attacker behavior after the compromise of an SSH server. We will use his state machine of attacker behavior for the definition of the cases used by CBR. We use this attribute with the scope of having a better definition of a case that will permit a more robust similarity metrics. We will have seven states defined:

- Checking the configuration - **CheckSW** - commands used by the attacker to obtain more information about the system's software or its users(w, id, whoami, last, ps, cat /etc/*, history, cat .bash_history, php -v);
- Installing rogue code – **Install** - attacker installs new software (tar, unzip, mv, rm, cp, chmod, mkdir);
- Downloading a file - **Download** - attacker downloads remote files (wget, ftp, curl, lwp-download);
- Running a rogue program - **Run** - attacker runs a program that was not on the system prior compromise (. /);
- Changing the account password - **Password** - attacker changes the password of the user logged (passwd);
- Checking the hardware configuration - **CheckHW** – attacker uses these commands to discover information about the hardware on top of which the operating system resides (uptime, ifconfig, uname, cat /proc/cpuinfo);
- Changing the system configuration - **ChangeConf** - attacker uses these commands to change the state of the system (export, PATH=, kill, nano, pico, vi, vim, sshd, useradd, userdel)

6 THE BASIC COMPOSITION OF THE HONEYPOT CBR SYSTEM

The honeypot system we propose, emulates a SSH sever that will be capable of interacting with the attackers by leveraging Case Based Reasoning.

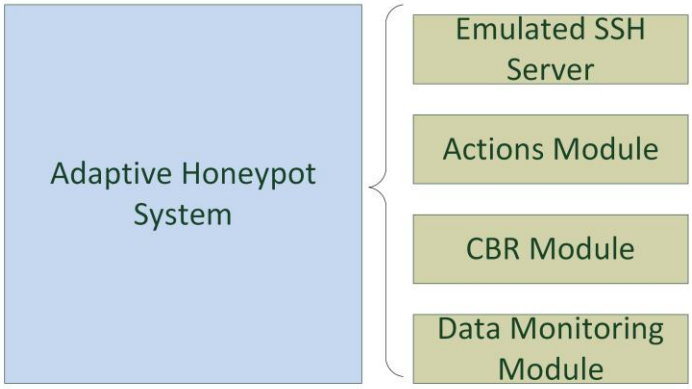

Figure 1. Block schema

As depicted in Figure 1, the Block scheme of the system consists of:

- **An Emulated SSH server** - for this existing solutions such as Kippo can be used or with the help of libraries such as Apache SSHD [13] a new one can be developed;
- **An Action Module** - this module will be responsible for the implementation of the Actions the System will use to interact with the attackers;
- **A CBR Module** – This module is responsible for the representation of cases, derivation of a solution for the action to be taken based on the previous experiences (cases);

- **A Data Monitoring Module** – This module is responsible for the collection of all the data that attackers generate during the compromise phase, post compromise phase, CBR phase and Action Phase.

6.1 The Emulated SSH Server Module

The Emulated SSH server Module can be developed by using existing solution such as Kippo or we can create form scratch a new one using existing Java libraries such as Apache SSHD [13].
We consider Kippo as well suited software for the necessities of our model because of its features: fake filesystem, possibility of adding fake file contents, session logs, trickery, saved downloaded files [3].

6.2 The Action Module

For the implementation of the Action Module we have initiated a separate research but at a first glance the implementation in both of the cases should be easy if we follow the proxy model. By proxy model we propose that the action module will sit between the SSH socket layer and the Linux terminal and by this means handling all the command prior to be sent to the terminal.
The proposed CBR Honeypot System uses for the Action Module the research initiated by Wagener in the area of interacting with an attacker by using several actions [4]. We propose the following improved set of actions by which our Honeypot will be able to interact:

- **BLOCK COMMAND** – an action that blocks the execution of an attacker command;
- **FAKE COMMAND** – an action that fakes the real output of an attacker command;
- **EXECUTE COMMAND** – an action that allows the execution of an attacker command;
- **DELAY COMMAND** – an action that delays with a number of seconds an attacker command;
- **INSULT COMMAND** – an action that presents an insult message after an attacker action.

6.3 The CBR Module

This module has been already implemented using jCOLIBRI framework and follows the simple approach of Retrieve, Retain, Reuse, Revise cases [7].
The development of CBR systems in jCOLIBRI Studio follows a case based approach. The first step requires that the user retrieves a design from a library (case base) of previously designed templates. In some situations the user has to adapt it by adding, removing or substituting components. The last step involves the source code generation using the selected tools. This process is named Template-Based Design [10].
One of the key components of CBR problem is the definition of a case and of its attributes. We have modeled our case by adding the following attributes:

- **CaseId -** unique identifier of the case;
- **InitialCommand** - the command and attacker executes;
- **Profile** - the profile in which that command integrates according to Ramsbrock study;
- **Action** – the action initiated by the Honeypot;
- **NextCommand** – the command executed by the attacker after the action.

As depicted in the Figure 2, the case is generated using the visual GUI provided by jCOLIBRI.

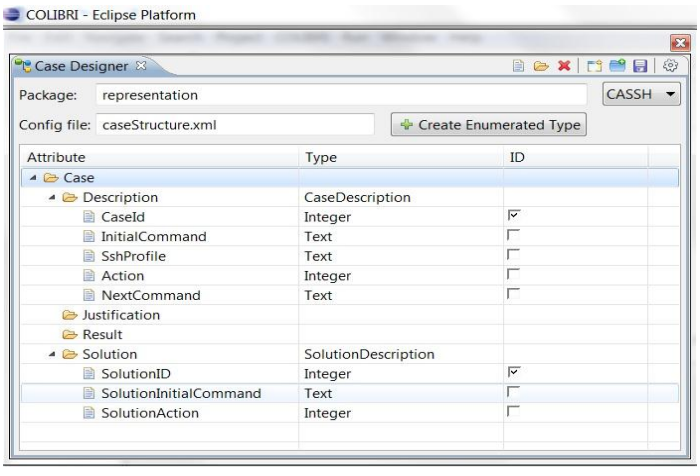

Figure 2. Case definition

Another key aspect regarding the implementation of the CBR system is the usage of the similarity

function. Since it was not our main goal to assess which is the most suitable for our system we have used the already developed: *jcolibri.method.retrieve.NNretrieval.similarity.global.Average*[14].

6.4 The Data Monitoring Module

For the Data Monitoring Module we propose to use the existing capabilities of Kippo or implement them in the new emulated JAVA SSH server. Of course this module will be adapted so that it would log also the data generated by Actions module and the CBR module.

7 WORKFLOW OF THE PROPOSED HONEYPOT SYSTEM

For the workflow of our propose system we consider four phases:

- **SSH Server Compromise Phase** – the attacker brute forces the SSH server so to acquire access,
- **Post Compromise Phase** – the attacker has compromise the server and enters a command in the terminal.
- **Casa based reasoning phase** – the emulated server passed the command entered by the attacker to the CBR module that generate a CBR problem;
- **Actions phase** – the CBR resolves the problem and offer a solution that contains an action to be executes and the Action module applies it.

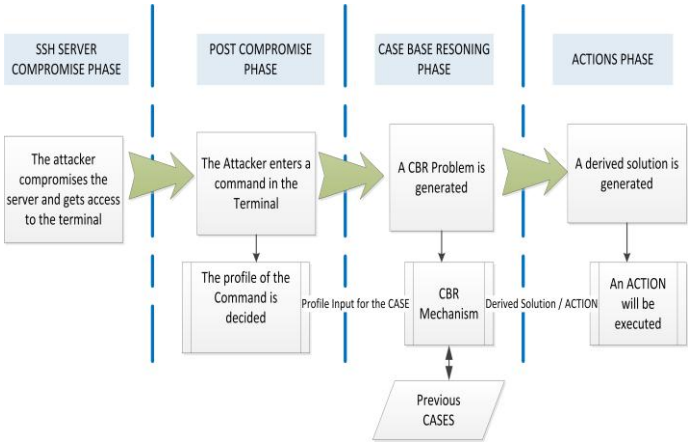

Figure 3. Workflow

As presented in the Figure 3, the workflow assumes two layers one regarding the interaction of the attacker with the SSH emulated server and another one regarding the generation and solving of the CBR problem and the decision of which action to be taken.

8 SIMULATIONS

For the testing of our proposed system we have used collected logs from an active Kippo deployment as reference for our cases and we have modeled a CBR problem using jCOLIBRI Framework. The results where relevant for the status of the research and the system was able to decide based on the existing cases which will be the solution of the CBR problem.

As Díaz-Agudo et. al states "the matching of cases, adaptation of solutions, and learning from an experience may be supported by a deep model of general domain knowledge, by more shallow and compiled knowledge, or be based on an apparent, syntactic similarity only" [15].

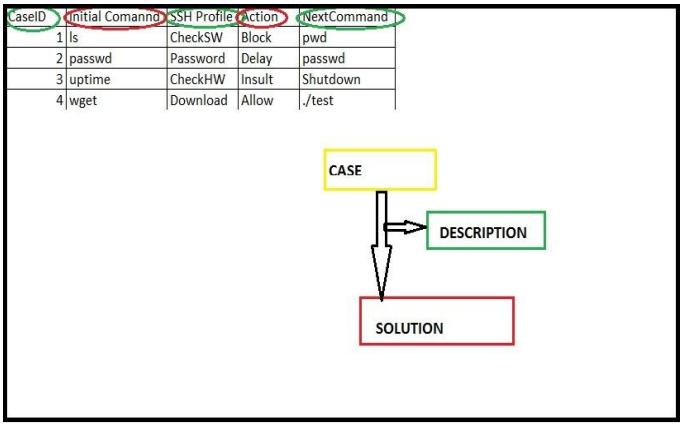

Figure 4. Solution derived from a case

As we can see in the Figure 4, in our simulations we have used syntactic similarity as the mean by which to trigger a solution. Obviously the system can be improved using more elaborate CBR methods.

9 CONCLUSIONS AND FUTURE WORK

From the beginning of the development of the honeypot systems one of the big issues has been the problem of engaging attackers in unveiling as much information about their intention, skills and methods. On the other hand, facts show that attackers unveil this information in a direct correspondence with the level of interaction the honeypot system poses. The system proposed tries to accomplish two important things in this struggle and those things are: increased interactions and adaptation.

We consider opportune a real implementation of the proposed system. Also we consider that the Case Based Reasoning Module can make use of more advanced solutions that should be related to the attacker's typology.

10 REFERENCES

[1] A. Ortega, Take care of your server, or it will be hacked and sold : http://www.alienvault.com/open-threat-exchange/blog/take-care-of-your-server-or-it-will-be-hacked-and-sold#sthash.6uxy1JJr.dpuf

[2] Kojoney : http://sourceforge.net/projects/kojoney/

[3] Kippo: A ssh honeypot. http://code.google.com/p/kippo/

[4] G. Wagener, R. State, T. Engel, A. Dulaunoy: Adaptive and self-configurable honeypots. Integrated Network Management 2011: 345-352

[5] User-mode Linux : http://user-mode-linux.sourceforge.net/

[6] A. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning). The MIT Press, Cambridge, MA (1998)

[7] Jcolibri : http://gaia.fdi.ucm.es/research/colibri/jcolibri

[8] L. Spitzner, Dynamic Honeypots: http://www.symantec.com/connect/articles/dynamic-honeypots

[9] D. In Leake, ed., 1996, Case-Based Reasoning: Experiences, Lessons, and Future Directions. Menlo Park: AAAI Press/MIT Press, 1996.

[10] M. Petridis, T. Roth-Berghofer, N. Wiratunga (eds.), Building Case-based Reasoning Applications with myCBR and COLIBRI Studio in :Proceedings of the 17th UK Workshop on Case-Based Reasoning, Pages 71-82, Cambridge, United Kingdom, School of Computing, Engineering and Mathematics, University of Brighton, UK, 12/2012

[11] A. A. Sánchez-Ruiz, J. A. Recio-García, P. A. González-Calero and B. Díaz-Agudo, Towards semi-automatic composition of CBR systems in jCOLIBRI, in: Proccedings of the 11th UK Workshop on Case Based Reasoning, CMS Press,University of Greenwich, 2006

[12] D. Ramsbrock, R. Berthier, M. Cukier : Profiling attacker behavior following SSH compromises. In: DSN '07: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Washington, DC, USA, IEEE Computer Society, 119–124 (2007)

[13] Apache MINA SSHD : http://mina.apache.org/sshd-project/

[14] Uses of Class jcolibri.method.retrieve.NNretrieval.similarity.global.Average: http://gaia.fdi.ucm.es/files/people/juanan/jcolibri/doc/api/jcolibri/method/retrieve/NNretrieval/similarity/global/class-use/Average.html

[15] B. Díaz-Agudo, P.A. González-Calero An architecture for knowledge intensive CBR systems E. Blanzieri, L. Portinale (Eds.), Advances in Case-Based Reasoning, EWCBR'00, Springer-Verlag, Berlin, Heidelberg, New York (2000)