ISSN:0975-9646

Sumaiyya Z. Khan et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 3250 -3253

Honeydoop - A System for Creating Virtual Honeypots Using Hadoop

Miss. Sumaiyya Z. Khan , Prof. D.M.Dakhane , Prof. R.L.Pardhi

Sipna College Of Engineering and Technology, Amravati, Maharashtra, India.

***Abstract*— System security personnel fight a seemingly unending battle to secure their digital assets against an ever-increasing onslaught of attacks. Honeypots- A security resource whose value lies in being probed, attacked, or compromised, provides a valuable tool to collect information about the behaviors of attackers in order to design and implement better defenses. Any commander will often tell his soldiers that to secure yourself against the enemy, you have to first know who your enemy is. This military doctrine readily applies to the world of network security. Just like the military, you have resources that you are trying to protect. To help protect these resources, you need to know who is your threat and how they are going to attack. On demand allocation of honeypots at right places on the network and at right time would considerably make the network more secure and harder to sneak into. This system is based on an idea of dynamically creating, modifying and managing virtual honeypots. This system combines the concept of honeypots and uses big data analyzer, Hadoop for quick information retrieval and analysis. The goal of this system is to create evanescent honeypots at right places and times, on demand.**

***Keywords*— Honeypots, Virtual Honeypots, Hadoop, Dynamic Honeypot Construction.**

I. INTRODUCTION

"A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource [1]." This means that whatever we designate as a honeypot, it is our expectation and goal to have the system probed, attacked, and potentially exploited. The honeypot contains no data or applications critical to the company but has enough interesting data to lure a cracker- a programmer who cracks (gains unauthorized access to) computers, typically to do malicious things.

Most current configurations are static setups consisting of either low interaction or high-interaction environments. Low-interaction honeypots have limited interaction, they normally work by emulating services and operating systems. Attacker activity is limited to the level of emulation by the honeypot. High-interaction honeypots are different, they are usually complex solutions as they involve real operating systems and applications. Nothing is emulated, we give attackers the real thing. That is, some of the vulnerable or important systems are identified beforehand, and their corresponding honeypots are maintained. It is unfeasible to maintain honeypots pertaining to the entire network.

To solve this problem, dynamic honeypots came to rescue. Dynamic Honeypot is a solution, you simply plug into your network, it learns the environment, deploys the proper number and configuration of honeypots, and adapts to any changes in your networks [7]. Although there are some dynamic Honeypots, deployment of right number of virtual Honeypots at right places and at right time on demand is the need of the hour.

A physical honeypot is a real machine with its own IP address. Deploying a physical honeypot is often time intensive and expensive as different operating systems require specialized hardware and every honeypot requires its own physical system. A virtual honeypot is a simulated machine with modeled behaviors, one of which is the ability to respond to network traffic. Multiple virtual honeypots can be simulated on a single system [10].

Hadoop is a "flexible and available architecture for large scale computation and data processing on a network of commodity hardware" [9]. It is an open source framework for processing, storing and analyzing massive amounts of distributed unstructured data. It was designed to handle Petabytes and Exabyte's of data distributed over multiple nodes in parallel. Hadoop clusters run on inexpensive commodity hardware so projects can scale-out without breaking the bank.

II. ANALYSIS OF PROBLEM

There are two systems A and B in a network (See Fig. 1) System B was found to be important and had its equivalent honeypot B'. System A did not have its equivalent honeypot. If an attacker tries to exploit A without falling for honeypot B', the main purpose of having a honeypot in the network is unused. It is expensive to maintain honeypots that yield us no information whatsoever. It is imperative to maintain only those honeypots that could be potential targets for the attacker. Had there been a honeypot for A, it could have provided us a great deal of information.

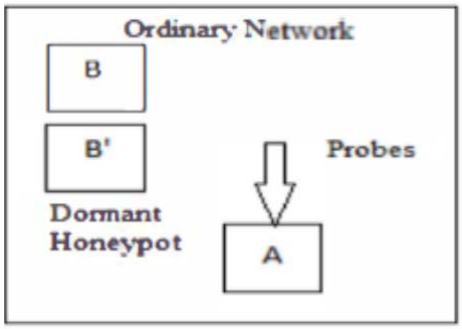

Fig. 1: Honeypot deployed in an Ordinary Network.

III. PROPOSED WORK

The problem mentioned above is solved in the following manner. In this endeavor, no honeypots are deployed beforehand. The honeypots are generated 'on-demand', as per the needs generated by the network. This will not only solve the above problem but it is also an efficient way to do so. (See Fig. 2)

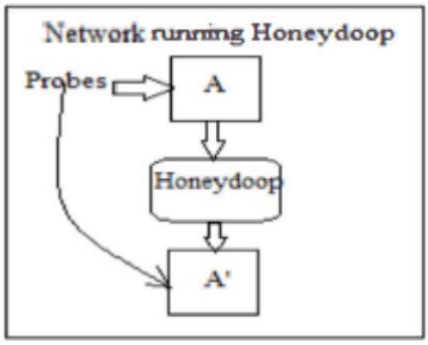

Fig. 2: Network running Honeydoop.

The attacker will experience an obscured network and will be redirected to the newly created honeypot on trying to connect the victim machine. Thus the machine will remain secure from present as well as such other malicious attacks in the future.

IV. SYSTEM MODULES

The proposed system can be broadly divided into four main modules (See Fig. 3):

A. Intrusion detector: This module will detect intrusion attempt in a network and provide IP address of the system being attacked.
B. Enumerator: This module will generate log files of a system on a periodic basis and to store them in database.
C. Auditor: This module will search for log files of the system in the Hadoop database that is under attack.
D. Honeypot Manager: This module will deploy a virtual system that is exactly the same replica of the system being attacked.

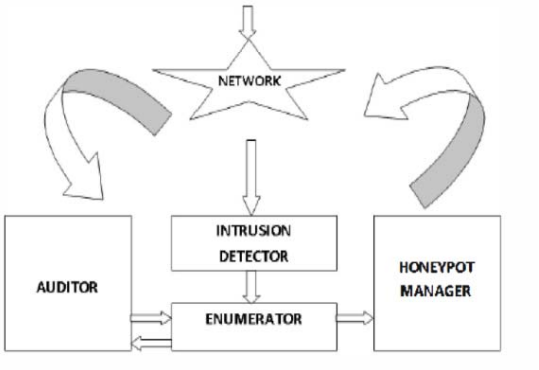

Fig. 3: System Modules.

V. IMPLEMENTATION

All the system modules present in Figure 3 are implemented in the following manner:

A. Intrusion Detector

For detecting intrusion in our system we implemented the Genetic Algorithm (GA) [11]. A Genetic Algorithm (GA) is a programming technique that mimics biological evolution as a problem-solving strategy. It is based on Darwinian's principle of evolution and survival of fittest to optimize a population of candidate solutions towards a pre-defined fitness.

GA uses an evolution and natural selection that uses a chromosome-like data structure and evolve the chromosomes using selection, recombination and mutation operators. The process usually begins with randomly generated population of chromosomes, which represent all possible solution of a problem that are considered candidate solutions. From each chromosome different positions are encoded as bits, characters or numbers. These positions could be referred to as genes. An evaluation function is used to calculate the goodness of each chromosome according to the desired solution; this function is known as "Fitness Function". During the process of evaluation "Crossover" is used to simulate natural reproduction and "Mutation" is used to mutation of species. For survival and combination the selection of chromosomes is biased towards the fittest chromosomes.

Figure 4 shows the operations of a general genetic algorithm according to which GA is implemented into our system.

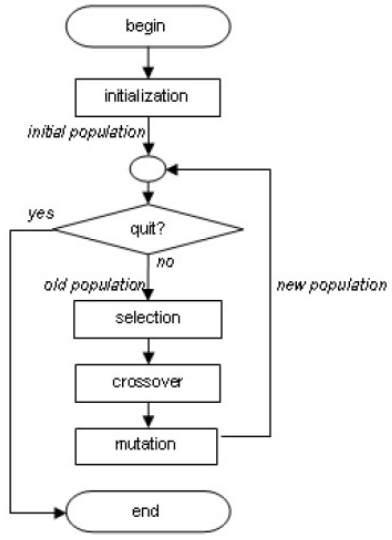

Fig. 4: Flowchart of Genetic Algorithm

We will implement GA on the pfirewall.log file generated by the Windows Firewall. For generating the pfirewall.log file, following steps has to be followed:

1. Go to Start and then search for wf.msc file.
2. Selecting wf.msc file will open Windows Firewall with Advanced Security.
3. Select Windows Firewall Properties.
4. Create pfirewall.log file by customizing following logging settings for Domain, Private and Public profile.
   Log dropped packets: Yes

Log successful connections: Yes

5. This will create the pfirewall.log file of default size as 4096 kBs at desired location.

On opening the pfirewall.log file, it will appear as shown in Figure 5 below.

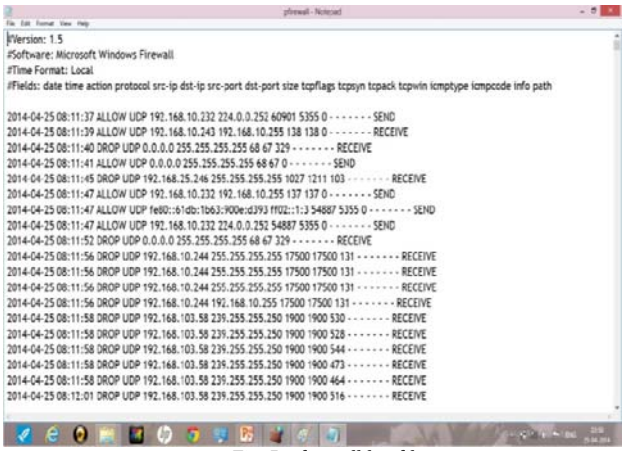

Fig. 5: pfirewall.log file

Genetic Algorithm of Our System:

1. Of the 17 fields present in the pfirewall.log file we will read the contents of 3rd, 5th, 6th and 9th field i.e. the action, scr-ip, dst-ip and size respectively line by line and take them as input for implementing the GA.

2. The rule base for implementing GA.

It consists of following two criteria:

a) If the log entry, in action field= DROP and its dst-ip ends with number other than 255 then these dropped packets can be considered suspicious.

b) If size of the packet > fitness function then, that packet is identified as intrusion packet.

3. Now, if the src-ip and dst-ip for two or more packets are same their respective packet sizes are added up together and that repeated combination is replaced by new combination of src-ip, dst-ip and packet size.
4. Following parameters are fixed on the basis of which GA will work:

   Number of iterations=5
   Mutation factor=0.6
   Detection Threshold=0.1
5. Fitness function for GA is calculated using following formula:

   Fitness (Mean Size) = (addition of cumulative packet size)/(total number of packets present in the pfirewall.log file)
6. Size of each packet is compared with that of the fitness function.

   If the size of packet < mean size then the packet is safe. Crossover operation is done i.e. the packet goes from current iteration to next iteration.

   If the size of packet > mean size then that packet is moved to next iteration.
7. Step 5. and 6. are repeated.
8. If the outputs for two iterations are same then the fitness function is multiplied by the mutation factor and step vi) is repeated.
9. Even after doing step 8. the outputs for two iterations are same then the mutation factor is increased by 1/(Number of iterations) and step 6. is repeated.
10. In the last iteration, the packets present are identified as intrusion packets based on the detection threshold. If the ratio of number of packets present in the last iteration is greater than detection threshold, then that packet is identified as intrusion packets.

B. Enumerator:

All the ip's identified as intrusion ip's by the GA are sent to the Hadoop MapReduce cluster. Hadoop MapReduce cluster works as an enumerator by keeping record of all detected ip's. We will keep this on the machine which will act as a server.

C. Auditor:

The Auditor is one of the main backbones of the system. It keeps track on the ip's of all the clients sending request to our server. If the client ip matches with that present in the Hadoop MapReduce cluster then it is redirected to Honeypot server. Otherwise the client is directed to normal server.

D. Honeypot Manager:

A dynamic virtual honeypot is generated by the honeypot manager when the request arrives from the ip present in Hadoop MapReduce cluster, as this ip is identified as suspicious by our intrusion detector. The attacker without his/her knowledge is then redirected to the honeypot server.

VI. RESULTS

We saved 5 different pfirewall.log files generated by the Windows Firewall in a period of 10 days and gave these files individually as input to GA in our intrusion detector. After that we kept record of the number of ip's identified as intrusional by GA and plotted these number of ip's against its corresponding individual pfirewall.log file.

We obtained the following results from this, which is as shown in Figure 6.

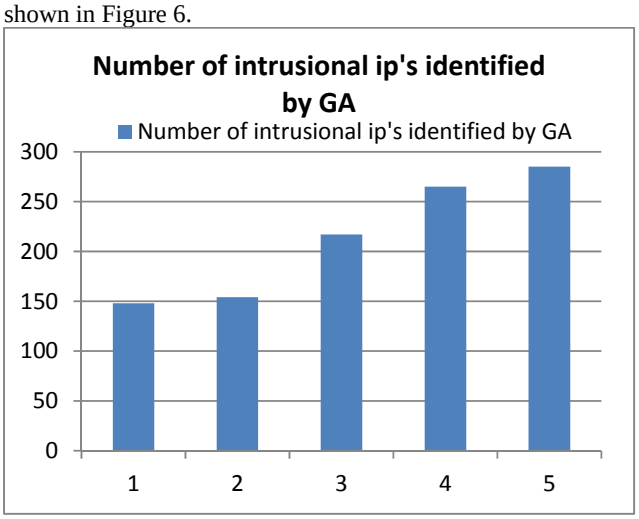

Fig. 6: Number of intrusional ip's identified by GA using 5 different pfirewall.log files

VII. APPLICATIONS

1. *Intrusion Detection:* Intrusion Detection is the art of detecting inappropriate, incorrect, or anomalous activity. Honeydoop can be used to determine if a computer network or server has experienced an unauthorized intrusion.
2. *Social Networking:* Web-based social systems enable new community-based opportunities for participants to engage, share, and interact. This community value and related services like search and advertising are threatened by spammers, content polluters, and malware disseminators. In an effort to preserve community value and ensure long-term success, we can use proposed system for uncovering social spammers in online social systems.
3. *Network Forensics:* Network forensics deals with the capture, recording and analysis of network events in order to discover evidential information about the source of security attacks in a court of law. Using this system we can gather intelligence about the enemy and the tools and tactics of network intruders.
4. *Campus Net Security:* With the development of digital campus construction, the campus network size has been rapid growth, but there are also many network security problems. If this system is applied to the campus network it can make the security of campus network unobstructed.

VIII. CONCLUSION

Honeydoop makes possible the creation of dynamic, virtual honeypots using Hadoop. It fulfills the goal of creating evanescent honeypots at right places and times, on demand, to achieve better security in this ever changing environment. The use of Genetic Algorithm for intrusion detection yields a stronger firewall, by identifying number of intrusional ip's (Figure No. 6). This complete system would greatly benefit the entire computing community at large by bolstering our defenses.

IX. FUTURE SCOPE

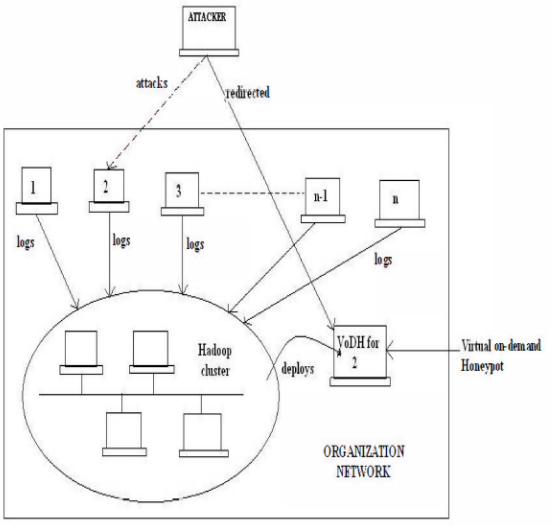

Fig. 7: Future Scope

Figure No. 7 shows the future scope of Honeydoop. In this, our system is implemented on an organization network consisting of 'n' number of systems. Each system creates its own pfirewall.log file and logs it on the Hadoop MapReduce cluster. Now, if any client sends request to any system, here suppose system '2' then it is checked whether the ip of that requesting client is compared with that present in Hadoop cluster. If the ip matches then the client is identified as an attacker and it is automatically redirected to the Virtual on-demand Honeypot for system '2' deployed by the Honeypot Manager. As a result of this the system '2' is safe from all present and future attacks. Similarly all the systems present in the organization network are safe.

REFERENCES

[1] Lance Spitzner, Honeypots: Definitions and value of Honeypots. http://www.tracking-hackers.com.
[2] John P. John, Fang Yuet et al., Heat-seeking Honeypots: Design and Experience. In Proceedings of WWW 2011-Session Web Security, 2011.
[3] Christopher Hecker, Kara L. Nance, and Brian Hay, ASSERT Centre, University of Alaska Fairbanks. Dynamic Honeypot Construction. In proceedings of the 10th Colloquium for Information Systems Security Education University of Maryland, University College Adelphi, MD June 5-8, 2006.
[4] L. Spitzner, 2002, Honeypots tracking Hackers. lsted. Boston, MA, USA: Addison Wesley.
[5] The Bait and Switch Honeypot, http://www.violating.us/projects/baitnswitch/
[6] The Honeynet Project, http://www.honeynet.org.
[7] L. Spitzner, Dynamic Honeypots, http://www.securityfocus.com/infocus/1731, Sept. 2003.
[8] BAIT-TRAP, http://www.cs.purdue.edu/homes/jiangx/BaitTrap, Dec. 2003.
[9] Research paper entitled A Study on "Role of Hadoop in Information Technology era" by Vidyasagar S.D.
[10] A Virtual Honeypot Framework by Neils Provos, Google, Inc.
[11] Research paper on "An Implementation Of Intrusion Detection System Using Genetic Algorithm" by Mohammad Sazzadul Hoque, Md. Abdul Mukit and Md. Abu Naser Bikas.