International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463
Vol. 3 Issue 3, March-2014, pp: (569-573), Impact Factor: 1.252, Available online at: www.erpublications.com

Deploying Honeynet according to Perception of Attacker

Ankush Kohli[1], Vikas Malik[2], Manjeet Singh[3], Gurjant Singh[4]
[1234]Faculty of Communication Engineering, Military College of Telecommunication Engineering, Mhow, Madhya Pradesh

Abstract: A modern technology in the area of intrusion detection is honeynet technology that unlike common IDSs tends to provide the attacker with all the necessary resources needed for a successful attack. Honeynets provide a platform for studying the methods and tools used by the intruders (blackhat community), thus deriving their value from the unauthorized use of their resources. Honeypots and Honeynets are particularly very useful for learning from attackers. This knowledge is required to update database and security policies of various network security tools available with the administrators. Almost all network security tools require updation of their database with new attack signatures as and when these attacks are developed. Honeynets are a powerful way of extracting these attack signatures. However, most of the current honeynets are manually configured and managed. This manual approach requires a prior knowledge about attacker and his behaviour. Also, such honeynets are vulnerable to identification by the attacker. In this paper we propose a model to automatically identify the requirements of an attacker and dynamically direct him to a machine providing these services. The attack and its target services are identified and the attacker is directed to a high interaction honeypot configured dynamically to provide the required service.

I. INTRODUCTION

Honeypots are valuable resources, configured to lure an attacker and then gather information about his ethnological and technical background. A network of such honeypots is referred to as a honeynet. In recent past, a large number of honeynet designs and deployment methodologies have been proposed. However, the capability of these honeynets to deceive the attacker is very limited and hence extraction of attack signatures using honeynet remains a challenge for network security planners. On one hand, if a honeynet is made too restrictive, little information can be gathered. On the other hand, if attackers have too much of freedom, they can easily perform a takeover of the honeynet and lock out the honeynet operator. This dilemma adds to the security management problem. Only a few of research activities have focused on configuring honeynets to serve the requirements of attacker. The current options, which are real systems with real security holes, it is possible to prevent such takeover by an attacker. Also, proposals have been made to implement a predefined behavior for any given application based on a security policy. However, these techniques are not suited to design effective autonomous honeynets because attackers' behavior is barely known in advance and may evolve. In this research, we have used artificial intelligence of an attack classifier as a driving force for implementing deployment of services in a honeynet as per the requirements of the attacker.

II. SYSTEM ARCHITECTURE

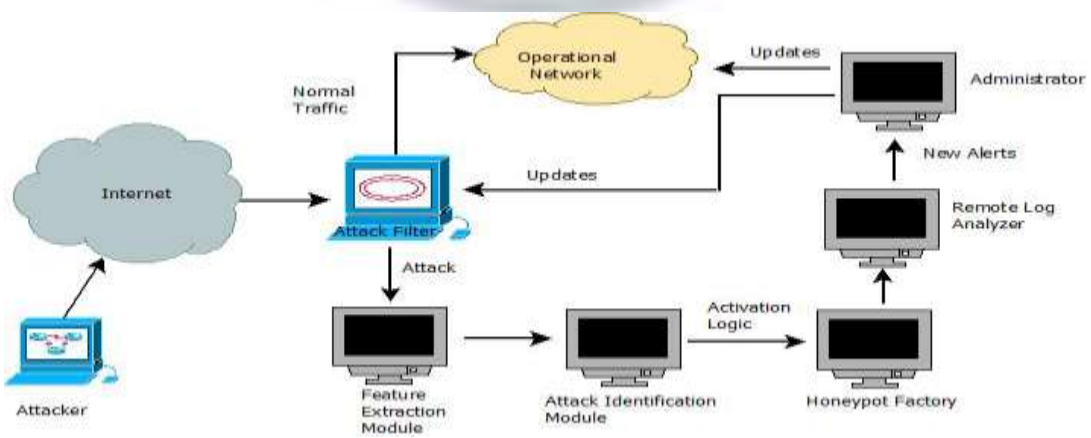

Fig. 1: Logical Model

The proposed model consists of five modules- Attack Filter, Feature Extraction Module, Attack Identification (AI) Module, Honeypot Factory and Remote Log Analyzer. The attack filter is used to segregate incoming attack from normal traffic and direct it to feature extraction module. It is primarily used during the training phase. The Feature extraction module is based on a sniffer to capture traffic and analyze packets. The features extracted by sniffer are used by AI module to classify the attack and accordingly generate an activation logic for honeypot factory. Honeypot factory is responsible for deploying honeynet as per the activation logic and direct the attack traffic to it. The attacker can now play with the services he desired. While he is busy executing his tools to exploit these services, all his actions are being logged at remote log analyzer. The administrator can use this remote log analyzer to profile the attacker. the attack signatures generated in remote log analyser can be used to update attack filter and production networks as well. The state flow Diagram for the same is as shown in Figure2.

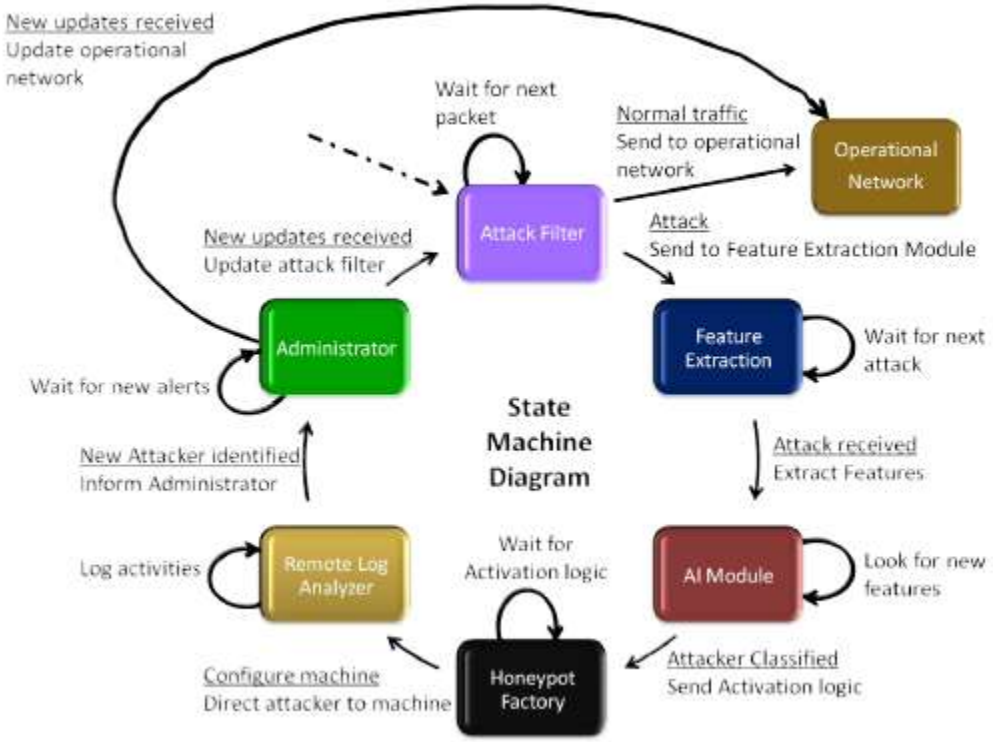

Fig. 2: State Flow Diagram

Table I. List of Extracted Features

S.No	Feature
01	Length
02	TTL
03	Protocol
04	Source Port
05	Destination Port
06	Urgent Flag
07	ACK pointer
08	Push Flag
09	Reset Flaag
10	SYNC Flag
11	FIN Flag
12	Urgent pointer

III. DESCRIPTION

Once the classifier is trained adequately and the databases are robust enough to take on live internet traffic the attack filter module can be removed. All the traffic coming from internet or any outside network has to pass through the feature extraction module before reaching the production network. The feature extraction module extracts 12 pre-defined features from the traffic packets and feeds them to Attack Identification module.The features are extracted only from the packet header. The list of packets being extracted is shhown in table1. The Attack Identification module first normalizes the features received from the feature extractor using PCA (Principal Component Analysis) algorithm and then classifies the traffic using K-Mean Clustering algorithm. If it is normal traffic, then it is directed to production network from attack identification module. If, however, the traffic is classified as an attack, it is directed to honeypot factory along with activation logic. The clusters defined in the K-Means Clustering algorithm are self evolving and therefore capable of taking on zero day attacks. Based on the classification of attack, this module decides the type of service to be given to the attacker. This decision is sent to Honeypot factory as Activation Logic. The Honeypot factory is a controlling station which controls all the resources of Honeynet. It is actually an intelligent VM Ware controlling large number of virtual machines. All machines, hosts and servers within the honeynet are controlled by honeypot factory.

Based on the activation logic, the honeypot factory configures one of the machines of the Honeynet to provide the service required by the attacker. Once the honeypot is ready, the attacker is directed to the honeypot. The honeypot factory also maintains a record of ongoing processes and performs load sharing by intelligently tasking the various machines in the Honeynet. As the attacker reaches the configured honeypot and starts playing with the machine, all his activities including the keys pressed, commands executed, files downloaded /uploaded, programs run, files changed and vulnerabilities exploited are logged in a remote log analyzer. The remote log analyzer is an intelligent machine capable of automatically analyzing the log data received from all the machines of the Honeynet and generating attack signatures. If there are any new attack signatures generated by the analyser, an alert is sent to the administrator for verification. These signatures are then used to strengthen the production network and eliminate vulnerabilities present in production network. In addition to this, these new attack signatures can also be used as security patches to update other isolated networks or stand alone computers.

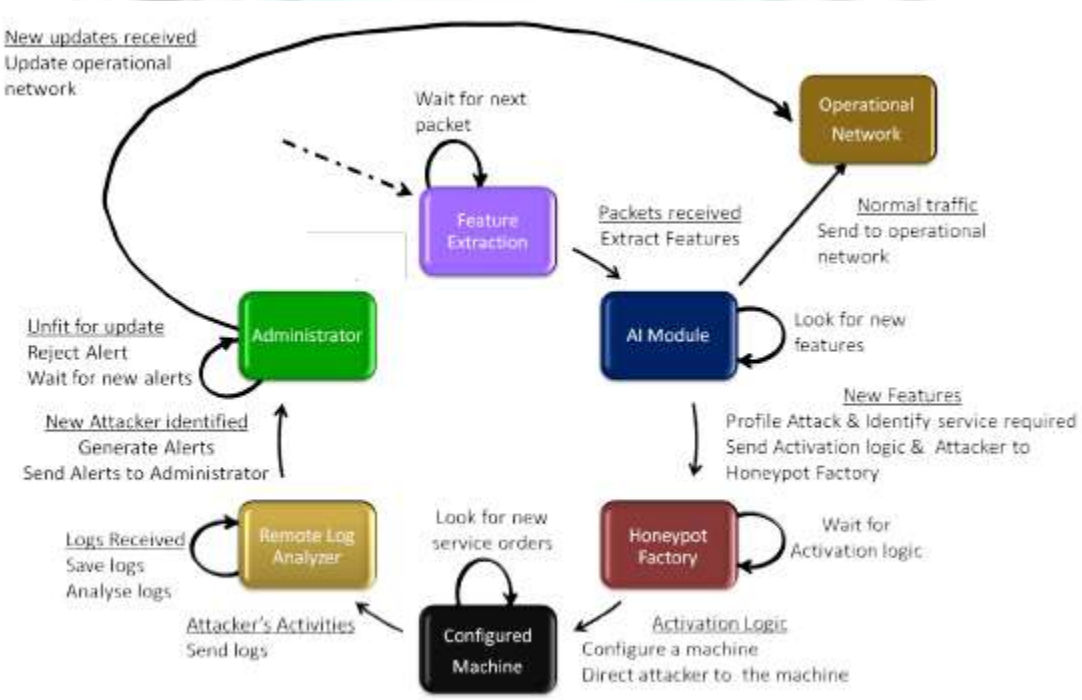

Fig. 3: State Flow Diagram

Figure.3 shows the state flow diagram for the system when after removing the attack filter. As explained above feature extraction module initially waits for new traffic. On arrival of new packets, it extracts the pre-defined features and feeds them to attack identification module. It then continues to wait for next packet. The attack identification module is continuously looking for new features from feature extraction module. As it receives new features along with the traffic, it first normalizes the features using PCA. This normalized data is used by KMeans Clustering algorithm to plot the incoming packets and classify them as either a normal traffic of a specific type of attack. If it is classified as normal traffic it is sent to production network. If the traffic is identified as an attack, the attack identification module decides the required type of service based on classification of attack. This decision is referred to activation logic. Activation logic is sent along with the attack traffic to honeypot factory.

The activation logic triggers the honeypot factory to configure a machine in honeynet to provide service to the attacker as per the activation logic received with the traffic. Once the machine is up, it directs the attack vector to it and allows it to play with it. The machines are preconfigured to send logs of attackers activities to a remote log server which is intelligent enough to analyse them.

Table I. List of Extracted Features

S.No	Feature
01	Length
02	TTL
03	Protocol
04	Source Port
05	Destination Port
06	Urgent Flag
07	ACK pointer
08	Push Flag
09	Reset Flaag
10	SYNC Flag
11	FIN Flag
12	Urgent pointer

The attack signatures generated as a result of this automatic analysis are used to update the actual production networks.

IV. FUTURE SCOPE

This is a unique and novel approach in the field of network security. The concept can be used not only to protect networks against existing attacks but also to protect it against new attacks. At the same time, this methodology will form the basis of attack identification and attack signature and security patch generation by various anti-virus, IDS, IPS and firewall companies. The methodology provides an easy, simple and cost effective solution to zero day attacks. It is a major breakthrough in the field of honeynets and boosts the capability of honeynets to deceive an attacker by manifolds. Since the system can now provide every service desired by the attacker, it removes the possibility of losing any attacker.

ACKNOWLEDGMENTS

The success of any project depends largely on the encouragement and guidelines of many others. This research project would not have been possible without the support and guidance of a large number of people from within and outside the Faculty of Comminication Engineering. First and foremost, we would like to thank to our project guide, Lt Col Ankush Kohli for the valuable guidance and advice. He inspired us greatly to work in this project. We are grateful to Brig Abhay A Bhat, Commander, FCE, for giving us the permission to use all required equipment and the necessary material to complete the task. We are obliged to staff members of Faculty of Communication Engineering, for the valuable information provided by them in their respective fields. We also take this opportunity to express a deep sense of gratitude to our group officer Lt Col B K Shrivastava for his cordial support, valuable information and guidance, which helped us in completing this task through various stages. Most importantly, we are blessed to have the oppurtunity to work under the able guidance of Lt Gen Rajesh Pant, AVSM, VSM, Commandant MCTE and Colonel Commandant, Corps of Signals. It was for his vision and wisdom that a small concept could fructify into a project. We express our sincere thanks and deepest regards to him for his encouragement without which this assignment would not be possible.

REFERENCES

[1]. L. Spitzner, Honeypots: Tracking Hackers. Boston, MA, USA: AddisonWesley Longman Publishing Co., Inc., 2002.
[2]. G. Wagener, R. State, A. Dulaunoy, and T. Engel, ”Self adaptive high interaction honeypots driven by game theory.” in SSS, ser. Lecture Notes in Computer Science, vol. 5873. Springer, 2009, pp. 741-755.
[3]. X. Jiang and X. Wang, ” ”Out-of-the-Box” monitoring of VMbased high-interaction honeypots,” in RAID’07: Proceedings of the 10th international conference on Recent Advances in Intrusion Detection. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 198-218.
[4]. S. Smalley, C. Vance, and W. Salamon, ”Implementing SELinux as a Linux Security Module,” NAI Labs, NAI Labs Report 01-043, Dec 2001. [Online]. Available: http://www.nsa.gov/selinuxldoc/module.pdf
[5]. M. Bauer, ”Paranoid penguin: an introduction to Novell AppArmor,” Linux Journal, vol. 2006, pp. 13-17, 2006.

[6]. R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning). The MIT Press, March 1998.
[7]. J. Dike, User Mode Linux. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2006.
[8]. D. Ramsbrock, R. Berthier, and M. Cukier, "Profiling Attacker Behavior Following SSH Compromises," in DSN '07: Proceedings of the 37th AnnuallEEEIIFlP International Conference on Dependable Systems and Networks. Washington, DC, USA: IEEE Computer Society, 2007, pp. 119-124.
[9]. E. Alata, V. Nicomette, M. Kaaniche, M. Dacier, and M. Herrb, "Lessons learned from the deployment of a high-interaction honeypot," in Dependable Computing Conference, 2006. EDCC'06. Sixth European, 2006, pp. 39-46.
[10]. A. Greenwald, "Matrix Games and Nash Equilibrium," 2007, lecture.
[11]. J. Goeree, C. Holt, and T. Palfrey, "Regular Quantal Response Equilibrium," Experimental Economics, vol. 8, no. 4, pp. 347-367, December 2005.
[12]. L. P. Kaelbling, M. Littman, and A. Moore, "Reinforcement Learning: A Survey," Journal of Artificial Intelligence Research, vol. 4, pp. 237-285, 1996.
[13]. B. Banerjee, S. Sen, I. Sen, and J. Peng, "Fast Concurrent Reinforcement Learners," in In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, 2001, pp. 825-830.
[14]. J. Hu and M. P. Wellman, "Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm," in In Proceedings of the Fifteenth International Conference on Machine Learning. Morgan Kaufmann, 1998, pp. 242-250.
[15]. A. L. Coates, "Pessimal Print: A Reverse Turing Test," in ICDAR '01: Proceedings of the Sixth International Conference on Document Analysis and Recognition. Washington, DC, USA: IEEE Computer Society, 2001, pp. 1154-1159.
[16]. G. Navarro, "A guided tour to approximate string matching," ACM Comput. Surv., vol. 33, no. I, pp. 31-88, 2001.
[17]. R. Love, Linux Kernel Development (2nd Edition). Novell Press, 2005.
[18]. G. Wagener, "AHA Source Code." [Online]. Available: http://git. quuxlabs.coml?p=aha-linux-2.6/.git;a=summary
[19]. C. Stoll, "Stalking the wily hacker," Commun. ACM, vol. 31, no. 5, pp. 484-497, 1988.
[20]. S. M. Bellovin, "There Be Dragons," in Proceedings of the Third Usenix Unix Security Symposium, September 1992, pp. 1-16. [Online]. Available: http://www.cs.columbia.edu/smb/papers/dragon.pdf
[21]. P. Baecher, M. Koetter, M. Dornseif, and F. Freiling, "The Nepenthes platform: An efficient approach to collect malware," in In Proceedings of the 9 th International Symposium on Recent Advances in Intrusion Detection (RAID). Springer, 2006, pp. 165-184.
[22]. W. Tillmann, "Honey trap."
[23]. B. McCarty, "The Honeynet Arms Race," IEEE Security and Privacy, vol. I, no. 6, pp. 79-82, 2003.
[24]. K.-w. Lye and J. M. Wing, "Game Strategies in Network Security," in Proceedings of Foundations of Computer Security Workshop 2002, 2002.