International Journal of Research in Advent Technology, Vol.3, No.12, December 2015
E-ISSN: 2321-9637
Available online at www.ijrat.org

A New Approach towards Design and Development of Portable Honeypot

Rohan Thakur[1], Ashu Singla[2]
SSIET, Dera Bassi, Punjab, India[1], Hartron Chandigarh, India[2]
thakur577@gmail.com[1] , 4s.ashu@gmail.com[2]

Abstract-A honeypot is a system that pretends to be an attractive target to attract malware and attackers. A honeypot has no productive use; each attempt to connect it can be interpreted as an attack. Based on honeypot deployment topology, if it is deployed in front of a firewall, it serves as an early warning system, if deployed behind the firewall, it serve as part of defense-in-depth system, in such case it helps to detect attackers who bypass the firewall and IDS(intrusion detection system) or it can be an insider threat. According to the level of interaction between the attacker and the honeypots, the honeypots are generally divided into three categories: low, medium and high. Configuring and maintaining a high interaction honeypot is always a tough task for new security researcher and network administrators. The purpose of this study is to design and implement easy to configure, easy to deploy, portable high interaction honeypot. To reduce the burden on the deployment of honeypot, we implemented the system on the USB as a live USB system, which gives the system features of easy installation, high portability and plug-n-play operation. In this paper an idea is presented on portable honeypot on a USB device that aims at fast detection of malicious network activity and thus boosts the security awareness of its user.

Index Terms- Early Warning Security System, Live USB Honeypot, Portable USB Honeypot

1. INTRODUCTION

A honeypot is a closely monitored computing resource that we want to be probed, attacked, or compromised. More precisely, a honeypot is "an information system resource whose value lies in unauthorized or illicit use of that resource". The value of a honeypot is weighed by the information that can be obtained from it. Monitoring the data that enters and leaves a honeypot lets us gather information that is not available to NIDS. For example, we can log the keystrokes of an interactive session even if encryption is used to protect the network traffic. To detect malicious behavior, NIDS requires signatures of known attacks and often fail to detect compromises that were unknown at the time it was deployed. On the other hand, honeypots can detect vulnerabilities that are not yet understood. For example, we can detect compromise by observing network traffic leaving the honeypot, even if the means of the exploit has never been seen before.

Because a honeypot has no production value, any attempt to contact it is suspicious by definition. Consequently, forensic analysis of data collected from honeypots is less likely to lead to false positives than data collected by NIDS. Most of the data that we collect with the help of a honeypot can help us to understand attacks.

Honeypots can run any operating system and any number of services. The configured services determine the vectors available to an adversary for compromising or probing the system. A high-interaction honeypot provides a real system the attacker can interact with. In contrast, a low-interaction honeypots simulates only some parts — for example, the network stack [1]. A high-interaction honeypot can be compromised completely, allowing an adversary to gain full access to the system and use it to launch further network attacks. In contrast, low-interaction honeypots simulate only services that cannot be exploited to get complete access to the honeypot. Low interaction honeypots are more limited, but they are useful to gather information at a higher level — for example, to learn about network probes or worm activity. They can also be used to analyze spammers or for active countermeasures against worms; neither of these two approaches is superior to the other; each has unique advantages and disadvantages.

Further honeypots can be described in two more types; physical and virtual honeypots. A physical honeypot is a real machine on the network with its own IP address. A virtual honeypot is simulated by another machine that responds to network traffic sent to the virtual honeypot**.**

When gathering information about network attacks or probes, the number of deployed honeypots influences the amount and accuracy of the collected data. A good example is measuring the activity of HTTP-based worms [2]. We can identify these worms only after they complete a TCP handshake and send their payload. However, most of their connection requests will go unanswered because they contact randomly chosen IP addresses. A honeypot can capture the worm payload by configuring it to function as a web server or by simulating vulnerable network services. The more honeypots we deploy, the more likely one of them is contacted by a worm.

This paper begins with the concept of implementation of High-Interaction Portable Passive Honeypot. Complete Honeypot system is a live system i.e. the system is a complete bootable computer installation, including operating system, which runs in computer's memory, rather than loading from a hard disk drive. It allows users to run an operating system for any purpose without installing it or making any changes to the computer's configuration. At the end of a live USB session the computer remains as it was before. The live system is able to run without permanent installation by placing the files that normally would be stored on a hard drive into RAM, typically in a RAM disk. The computer must have sufficient RAM both to store these files and maintain normal operation. Now the advantage of the above system is that even if our Honeypot is compromised the system will come to its initial state once it is rebooted.

According to the statistics by the Computer Emergency Response Team (CERT), the number of reported security incidents per year is rising and malicious users are increasingly using automated attack tools [3], in order to detect and stop malicious activities, and protect their assets, organizations implement various security tools and methods. Two of the most common security tools that are used today to protect organizations network are firewalls and Intrusion Detection Systems (IDS). Firewalls are most often implemented at the network perimeters where they control network traffic. This control is employed according to a set of rules which define allowed and denied network traffic. IDS monitor network traffic and alert the administrator when a known malicious activity is detected. In order to detect a malicious activity, IDS will use two methods: signature detection and anomaly based detection. These security tools have some inherent shortcomings [4]. A firewall cannot stop malicious users exploiting a new vulnerability in a service to which access is allowed by the firewall rules. IDS cannot reliably detect a previously unknown attack, especially if only signature detection is used. If anomaly based detection is used, it is based "on the assumption that intrusive activities are necessarily different from non-intrusive activities at some level of observation." [5] None of these methods of detection can guarantee that the IDS will report all attacks, so false negative detections will exist.

This motivated us to create this system to capture these unknown attacks and study the attacks in order to help security agencies and researchers. We studied references for already existing high interaction systems and noted the limitation of existing systems. Then we created, and designed our own system and tried to improve on the drawbacks of other existing system. For this first we made it a live system (always running in memory). Generally when high interaction honeypots are compromised, the state changes becomes persistent and it is difficult to bring back the system to its original state. Making the system as live system overcomes this problem such that the system comes to its original state just by rebooting. Then we added data control module, data control prevents attackers from using a compromised honeypot system to attack other external computer systems. To hide the network interactions from attackers, we created a different system to perform data capturing activities, the data will go through this new machine, which is hidden to attacker and attacker will only see victim machine having vulnerabilities.

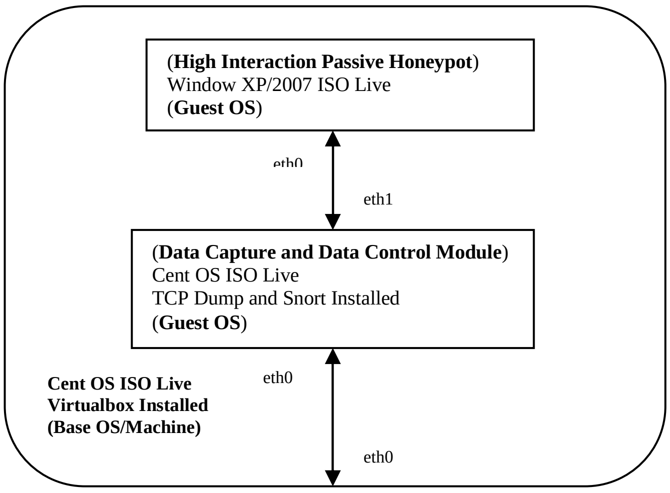
1. OBJECTIVE AND SCOPE

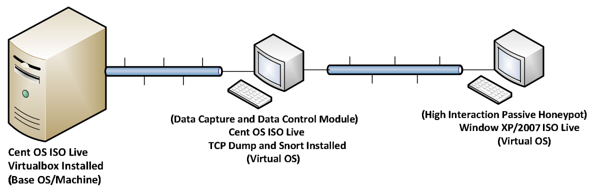
Scope of this work includes portable high interaction USB passive Honeypot, its design, development, configuration and installation. Low Interaction, Medium Interaction and Client Honeypots are out of this scope. Analysis and Classification of collected PCAP data is also out of the scope of this work.

The Objective of this work is to design and develop completely portable high interaction honeypot. Honeypot should be easy to deploy and configure. If honeypot system gets compromised it should be easy to restore the state of the system to its original state and at the same time we should be able to collect attack data.

In our study of Honeypots and particularly High Interaction Honeypots we found that these Honeypots are quite difficult to install and deploy. In addition to this, maintenance of these honeypots is bit risky, as an attacker can completely compromise the system and use the honeypot to initiate outbound connections/attacks. In comparison low interaction honeypots are not risky, as it is not an actual operating system; it is just like a simulated environment of set of network services. Medium Interaction Honeypot is the combination of real as well as simulated services.

3. BLOCK DIAGRAM OF THE SYSTEM



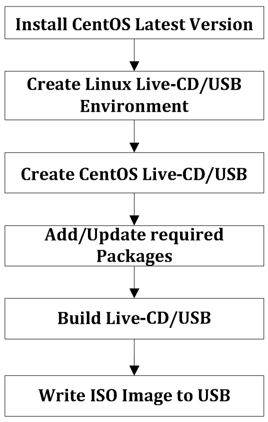


The complete system consists of 3 sub systems, i.e. one Base OS and two Guest OS

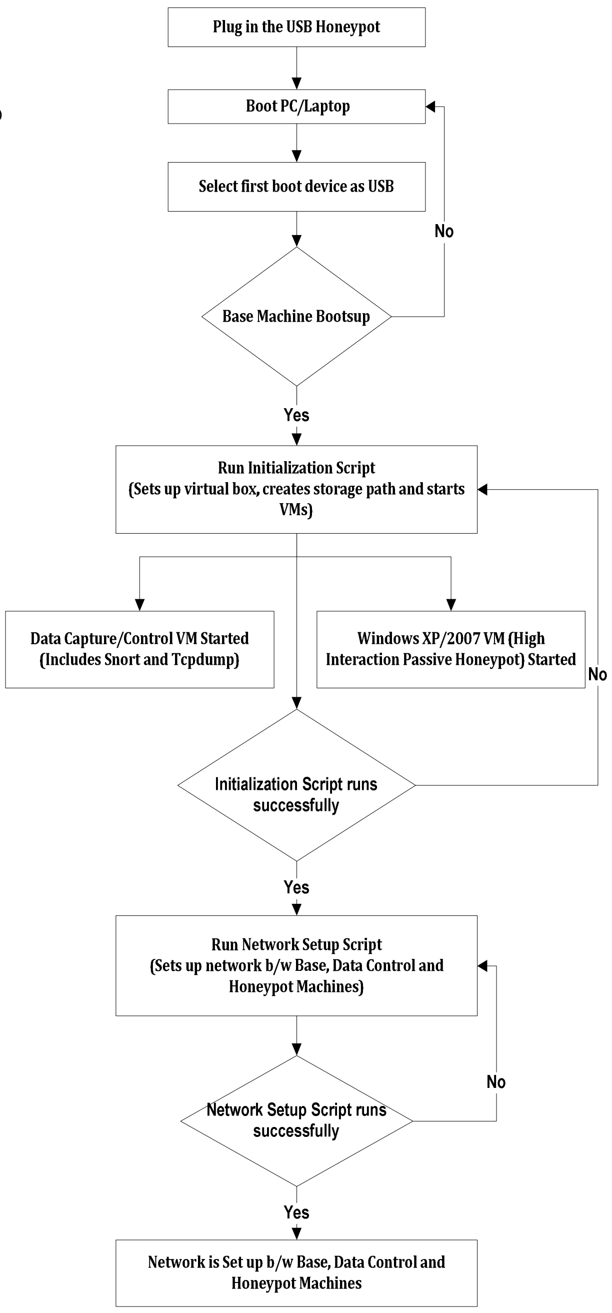
1. Base OS: Cent OS with Virtualbox.
2. Guest OS: (Data capture and Data Control): Cent OS with tcpdump and Snort.
3. Guest OS: (High Interaction Honeypot): Windows XP/2007

4. FLOWCHART TO BUILD THE SYSTEM

The flow chat depicts the high level steps followed to build the system. The system is build based on the Live CD concept. Generally High Interaction Honeypots are standalone desktops or Virtual OS systems running from system hard disk drive. In this case the virtual Honeypot OS runs on USB stick/Pen drive. The complete system is on the USB stick running in Live mode.


Install CentOS Latest Version
Create Linux Live-CD/USB Environment
Create CentOS Live-CD/USB
Add/Update required Packages
Build Live-CD/USB
Write ISO Image to USB

5. FLOWCHART TO INITIALIZE THE SYSTEM


Plug in the USB Honeypot
Boot PC/Laptop
Select first boot device as USB
No
Base Machine Bootsup
Yes
Run Initialization Script
(Sets up virtual box, creates storage path and starts VMs)
Data Capture/Control VM Started (Includes Snort and Tcpdump)
Windows XP/2007 VM (High Interaction Passive Honeypot) Started
No
Initialization Script runs successfully
Yes
Run Network Setup Script
(Sets up network b/w Base, Data Control and Honeypot Machines)
No
Network Setup Script runs successfully
Yes
Network is Set up b/w Base, Data Control and Honeypot Machines

The flow chart depicts the USB Initialization steps. To run the system from USB stick, first boot device needs to be marked as USB disk in system BIOS. After system boots up from USB two scripts are executed serially. First script initialize the virtual box, creates storage paths and starts virtual machines. Second script establishes the network setup between base os, data control virtual machine and Honeypot machine. Once both scripts run successfully the system is up and Honeypot gets started.

6. LIVE USB HONEYPOT RUNNING SCREENSHOTS

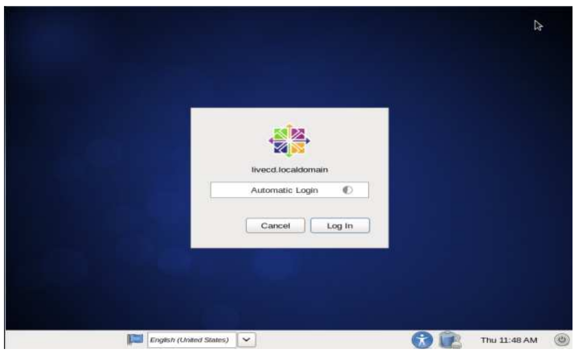


Fig.1. Live USB Honeypot Base OS (Centos)

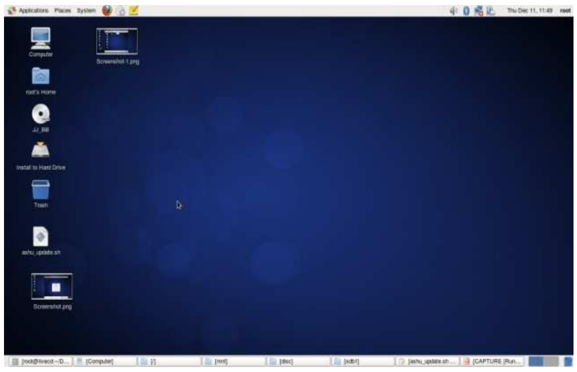


Fig.2. Desktop Base OS

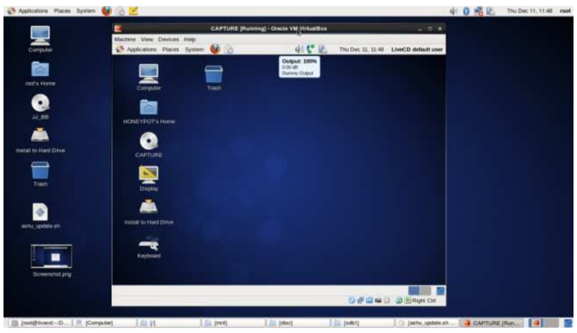


Fig.3. Data Capture Live Image Up

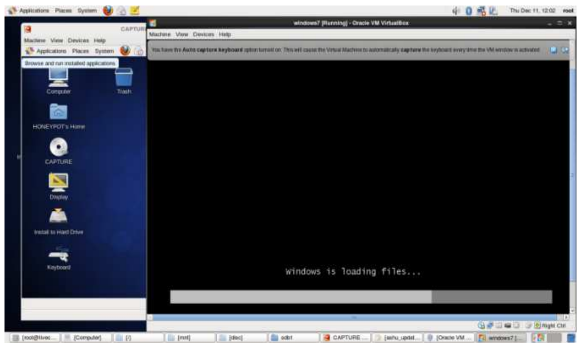


Fig.4. Live Window 7 VM Initializing

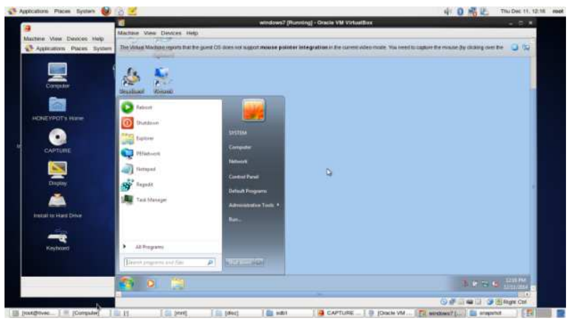


Fig.5. Windows 7 Live Image Up

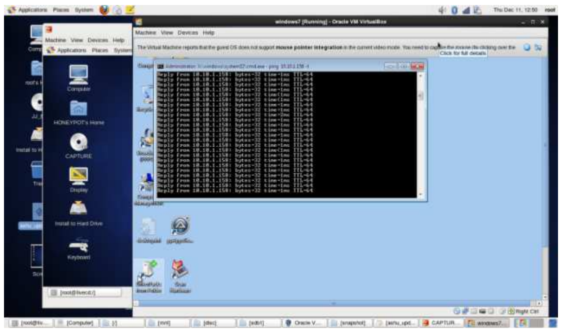


Fig.6. Live Honeypot (Base CentOS, Data Control [Data Capture], Windows 7 Live)

7. CONCLUSION AND FUTURE SCOPE

With this Portable honeypot, we have the opportunity to collect those unknown attacks that are generally missed by traditional security tools like firewalls and intrusion detection system (IDS) and thus collect the valuable attack data. By making it a live system i.e. system always running in memory, persistent state changes are avoided, as system comes to its original state once rebooted. In addition to this, the system is pre-installed with required packages and is a kind of plug and play device; therefore the system can be deployed in any network with great ease.

From this initial work, we have identified some possibilities for future work that could be developed further.

- Develop the Client Server model of the same system in which the collected network data and other attack data such as binaries will be sent to Central Server for data collection and statistical analysis of collected data.
- Apart from High Interaction Honeypot, we can add Low and Medium Interaction

Honeypots like Honeyd and Nepenthes.

- We can add High and Low Interaction Client Honeypots to this system
- Develop a front end to configure Honeypot.
- Develop a front-end for attack data visualization.

REFERENCES

[1] Niels Provos. A virtual honeypot framework. In Proceedings of 13th USENIX Security Symposium, pp. 1–14. USENIX, 2004.
[2] Niels Provos, Joe McClain, and Ke Wang. Search worms. In WORM '06:Proceedings of the 4th ACM Workshop on Recurring Malcode, pp. 1–8, NewYork, 2006. ACM Press.
[3] Computer Emergency Response Center (CERT), "CERT/CC Statistics 1988-2004," http://www.cert.org/stats/cert_stats.html, 2004.
[4] J. Levine, R. LaBella, H. Owen, D. Contis, B. Culver, "The Use of Honeynets to Detect Exploited Systems Across Large Enterprise Networks," IEEE Systems, Man and Cybernetics Society, 18-20 June 2003, (pp. 92-99)
[5] J. McHugh (2001), "Intrusion and intrusion detection," International Journal of Information Security 1, (pp. 14-35)