ISSN 2395-1621

Implementation on Intrusion Detection System Using Virtual Nodes

[#1]Prof. B.B. Gite, [#2]Mohan Yelpale, [#3]Priyanka Salve, [#4]Rushikesh Gondake, [#5]Prathamesh Shinde

[1]bbgite.sae@sinhgad.edu
[2]mohansaeit@gmail.com
[3]priyankasalve29@gmail.com
[4]rushigondake@gmail.com
[5]panther.cw.8888@gmail.com

[#1]Prof. Department of Computer Engineering
[#2345]Student. Department of Computer Engineering

Sinhgad Academy Of Engineering Pune,
Savitribai Phule Pune University
Ganeshkhind Pune-411007, India.

ABSTRACT

A honeypot is a closely monitored network decoy serving several purposes: it can distract adversaries from more valuable machines on a network, provide early warning about new attack and exploitation trends, or allow in-depth examination of adversaries during and after exploitation of a honeypot. 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. This paper presents Honeyd, a framework for virtual honeypots that simulates virtual computer systems at the network level. The simulated computer systems appear to run on unallocated network addresses. To deceive network fingerprinting tools, Honeyd simulates the networking stack of different operating systems and can provide arbitrary routing topologies and services for an arbitrary number of virtual systems. This paper discusses Honeyd's design and shows how the Honeyd framework helps in many areas of system security, e.g. detecting and disabling worms, distracting adversaries, or preventing the spread of spam email.

ARTICLE INFO

Article History

Received: 28th May 2016

Received in revised form :

28th May 2016

Accepted: 31st May 2016

Published online :

1st June 2016

Keywords: **Honey pot, Extended Honey pot, Malicious Objects.**

I. INTRODUCTION

When securing a network, there are many different aspects, tools and approaches that can be utilized. One tool in particular offers a large amount of useable flexibility for administrators; the honeypot. Honeypots differ and can be deployed in a number of different forms as well as filling the roles of other tools that are not available based on the needs of the network. The main role filled by honeypots of course would be detection. Keep in mind though that each individual honeypot must be configured and tailored to the individual network and what they need to detect. This paper will take a short look into honeypots, and then more specifically focus on the program Honeyd. Afterwards, it will go into the basics of how it works, some examples of usage, and how to realistically implement it into a network. Frequently, network intrusion detection systems are the common form of threat detection seen in networks, also known as an NIDS. NIDS's passively monitor the traffic on the network and log any alerts they find of suspicious or unauthorized activity they view. They (NIDS) work well enough, but the main issue with them is their inability to differentiate between malicious attempts on the network, and false-positives. Not only is this a problem when a network is small, but the effect this has on the NIDS as the network grows is exponential since it will be required to monitor larger amounts of traffic creating much larger logs, and requiring more resources to continue running it. Also, this tasks the network administrator with parsing huge files to even begin searching for malicious attempts. On the other hand, looking at this in a more positive light, this does not mean NIDS are not at all effective. They, like any other piece of defense are an added layer or protection for the network. No security overlay of a network should be reliant on just one method. Multiple tools and methods should be employed to best defend the network.

Honeypots can be classified into two categories:

1) Low interaction honeypots
2) High interaction honeypots.

Low interaction honeypots are designed to imitate vulnerable services and investigate attacks without exposing full operating system functionality. Although they have evolved in many ways over the past 15 years, understanding their limitations and sometimes inherent design weaknesses is important when you consider deploying them. While the attacker is busy gaining access to this machine, information on them can be gathered allowing the administrator to plan how to deal with them, and protect the network from similar threats in the future.

High-Interaction honeypots are time-consuming to design, manage and maintain. The goal of the high interaction honeypots is to give the attacker access to a real operating system where nothing is emulated or restricted.

Deploying a physical honeypots is often time intensive and expensive as different operating systems require specialized hardware and every honeypot requires its own physical system.

Honeyd is a program that allows a user to organize and simulate virtual hosts on a computer network system. These virtual hosts can follow as a model many different types of servers, allowing the user to simulate an infinite number of network configurations. Honeyd is primarily used in the field of cyber security by professionals & cyber crime investigators for detecting and disabling worms, distracting adversaries, or preventing the spread of spam email etc.

II. PROPOSED SYSTEM

Honeyd's architecture consists of several components: a configuration database, a central packet dispatcher, protocol handlers, a personality engine, and an optional routing component; see Figure Incoming packets are processed by the central packet dispatcher. It first checks the length of an IP packet and verifies the packet's checksum. The framework is aware of the three major Internet protocols: ICMP, TCP and UDP. Packets for other protocols are logged and silently discarded. Before it can process a packet, the dispatcher must query the configuration database to find a honeypot configuration that corresponds to the destination IP address. If no specific configuration exists, a default template is used. Given a configuration, the packet and corresponding configuration is handed to the protocol specific handler. The ICMP protocol handler supports most ICMP requests. By default, all honeypot configurations respond to echo requests and process destination un-reachable messages. The handling of other requests depends on the configured personalities. For TCP and UDP, the framework can establish connections to arbitrary services. Services are external applications that receive data on stdin and send their output to stdout. The behavior of a ser-vice depends entirely on the external application. When a connection request is received, the frame-work checks if the packet is part of an established connection. In that case, any new data is sent to the already started service application. If the packet contains a connection request, a new process is created to run the appropriate service. Instead of creating a new process for each connection, the frame-work supports subsystems and internal services. A subsystem is an application that runs in the name space of the virtual honeypot. The subsystem specific application is started when the corresponding virtual honeypot is instantiated. A subsystem can bind to ports, accept connections, and initiate net-work traffic. While a subsystem runs as an external process, an internal service is a Python script that executes within Honeyd. Internal services require even less resources than subsystems but can only accept connections and not initiate them. Honeyd contains a simplified TCP state machine.

The three-way handshake for connection establishment and connection teardown via FIN or RST is fully supported, but receiver and congestion window management is not fully implemented. UDP datagrams are passed directly to the application. When the framework receives a UDP packet for a closed port, it sends an ICMP port unreachable message unless this is forbidden by the configured personality. In sending ICMP port unreachable messages, the framework allows network map-ping tools like trace route to discover the simulated network topology. In addition to establishing a connection to a local service, the framework also supports redirection of connections. The redirection may be static or it can depend on the connection quadruple (source ad-dress, source port, and destination address and destination port). Redirection lets us forward a connection request for a service on a virtual honeypot to a service running on a real server. For example, we can redirect DNS requests to a proper name server. Or we can reflect connections back to an adversary, e.g. just for fun we might redirect an SSH connection back to the originating host and cause the adversary to attack her own SSH server. Evil laugh.Before a packet is sent to the network, it is processed by the personality engine. The personality engine adjusts the packet's content so that it appears to originate from the network stack of the configured.

SYSTEM ARCHITECTURE

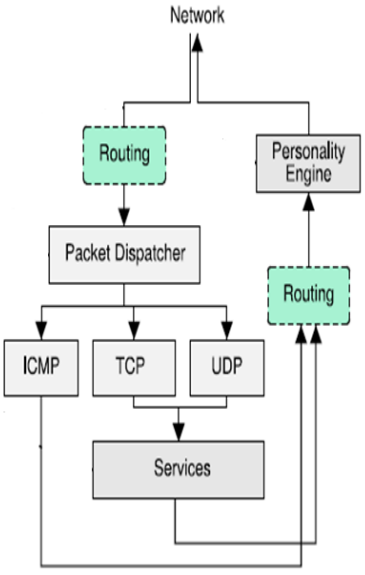


Fig 1. System Architecture

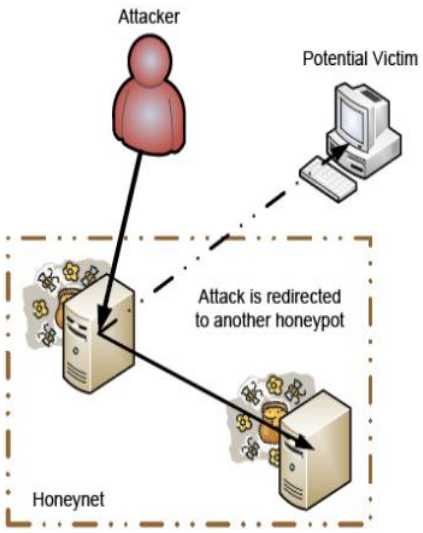
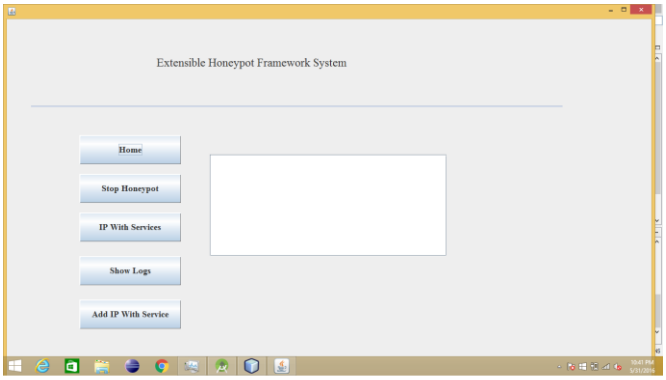
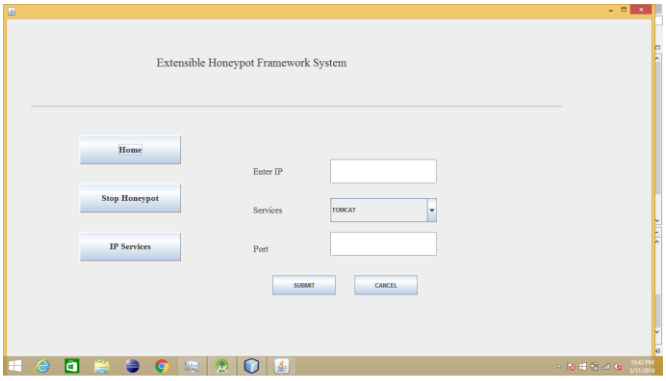
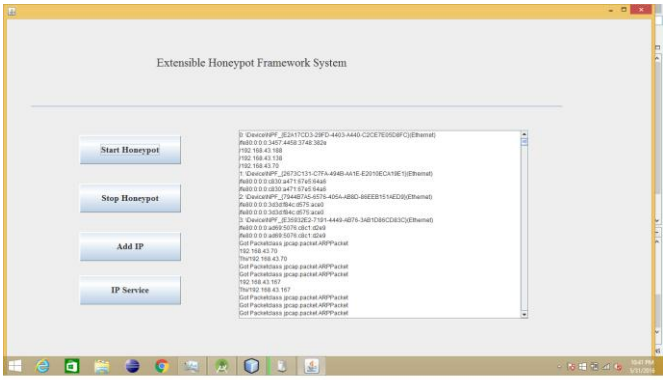


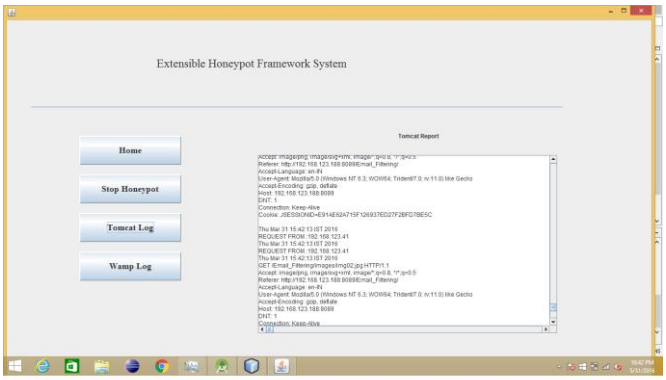
Fig 2. Redirecting an outbound attack in a honeynet

III. RESULT









IV. OBJECTIVE AND SCOPE

1. *Objective*
   a. Honeypot is used to capture information about attacks.
   b. It is used to expose system vulnerabilities and drawbacks.
   c. Honeypot gives us idea about system response during an attack.
   d. It also captures information about attackers. For example, attackers IP address.
   e. It used to identify and recognise attack methods used by attacker.
   f. We can identify attack and system response patterns and make changes accordingly.
   g. Identities.
2. *Scope*
   a. Honeypot is used to implement Middle Interaction.
   b. It also used to Increase the number of operating system services per PH.
   c. Experiment with honeyds and physical honeypots on same network.
   d. Increase stability and robustness of personality engine.

.

V. CONCLUSION

Honeyd is a framework for creating virtual honeypots. Honeyd mimics the network stack behavior of operating systems. We gave an overview of Honeyds design and architecture and showed how Honeyds personality engine can modify packets to match the ngerprints of other operating systems and how it is possible to create arbitrary virtual routing topologies. Honeypots are a valuable tool for intrusion and malware infection analysis. we present Timescope, a honeypot record and replay system that greatly enhances existing ways to perform forensic analysis of honeypots.

ACKNOWLEDGEMENT

Making this project a reality takes efforts of many dedicated people. We are thankful to Savitribai Phule Pune University for providing the guidance. Also we are thankful to Sinhgad Academy Of Engineering, project coordinator Prof. Kiran Avhad and H.O.D. Prof. B.B.Gite for providing the necessary facilities during the working of this paper.

REFRENCES

[1] "Cyber-Physical System Security With Deceptive Virtual Hosts for Industrial Control Networks" By Todd Vollmer, and Milos Manic, Senior Member, IEEE, IEEE TRANSACTION ON INDUSTRIAL INFORMATICS, VOL. 10, NO.2 MAY 2014.

[2] "A Virtual Honeypot Framework", NielsProvos*Google, Inc.niels@google.com.

[3] "Wireless Honeypot: Framework, Architectures and Tools", RadhikaGoel, Anjali Sardana, and R. C. Joshi, International Journal of Network Security, Vol.15, No.5, PP.363-373, Sept. 2013.

[4] "EXTENDED HONEYPOT FRAMEWORK TO DETECT OLD/NEW CYBER ATTACKS",HemrajSaini, BIMAL KUMAR MISHRA, et al. / International Journal of Engineering Science and Technology (IJEST), ISSN : 0975-5462 Vol. 3 No. 3 March 2011.

[5] "Design and Implementation of Linux Based Hybrid Client Honeypot Incorporating Multi Layer Detection",Atinder Pal Singh, Birinder Singh / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 2, Issue 5, September- October 2012, pp.1135-1142.

[6] Web reference: www.honeyd.com

