Anti Network Scanning Honeypot System

Aung Chan Min
University of Computer Studies, Yangon
mgdipa@gmail.com

Abstract

In today computer security field, honeypot technology is rapidly maturing and various types of honeypots are establishing their role of countermeasure as viable and useful in modern network defense system. In this paper, we proposed the anti network scanning honeypot system that can deceive to potential hackers' remote network scanning attempts. The proposed system dynamically creates virtual honeypot machines with configured virtual TCP/IP networking stack of different OS and these virtual honeypots can response different types of network scanning attempts correctly. The proposed system hides the real topology of internal networks design and displays the layout of virtual machines with configured virtual routing topology to remote hackers. The proposed system can detect different types of network scanning attempts and produce log for further analysis.

Keywords - Network Security, Intrusion detection, Hacker Distraction

1. Introduction

Honeypot is a security system resource whose value lies in unauthorized or illicit use of that resource. This means that a honeypot can be anything - a program sitting on a computer logging all the users who log into the system and by means they log into, just a dummy account on the system which when logged into generates an alarm, and to some very extent it could even be a fake record with interesting name to attract invalid intruders.

In security circles honeypots are often thought to be as bait-and-capture systems. Since there are various configurations of honeypots, it is hard to define what a particular honeypot does and how far it can meet its objective. The definition of a honeypot is as "**a** security resource whose value lies in being probed, attacked or compromised" [1]. This means that whatever users designate as a honeypot, it is general expectation and goal to have the system probed, attacked, and potentially exploited. How users use honeypot is up to users and depends on what users are attempting to achieve.

Honeypots are closely monitored network decoys serving several purposes: they can distract adversaries from more valuable machines on a network, they can provide early warning about new attack and exploitation trends and they allow in-depth examination of adversaries during and after exploitation of a honeypot**.**

Honeypots are a highly flexible security tools with different applications for security. They don't fix a single problem. Instead they have multiple uses, such as prevention, detection, or information gathering [2]. All Honeypots share the same concept: a security resource that should not have any production or authorized activity. In other words, deployment of honeypots in a network should not affect critical network services and applications.

1.1Types of honeypots

There are two types of honeypot systems low-interaction honeypots and high-interaction hoenypots [3]. A high-interaction honeypot simulates all aspects of an operating system. A low-interaction honeypots simulates only some parts, for example the network stack. 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, e.g., learn about network probes or worm activity. They can also be used to analyze spammers or for active countermeasures against worms.

Honeypots can also be classified as physical and virtual honeypots [4]. A physical honeypot is a real machine with its own IP address. 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. Virtual honeypots are attractive because they require fewer computer systems, which reduce maintenance costs. Using virtual honeypots, it is possible to populate a network with hosts running numerous operating systems. To convince adversaries that a virtual honeypot is running a given operating system, we need to simulate the TCP/IP stack of the target operating system carefully, in order to deceive TCP/IP stack fingerprinting tools.

Physical honeypots are often high-interaction, so allowing the system to be compromised completely, they are expensive to install and maintain. For large address spaces, it is impractical or impossible to deploy a physical honeypot for each IP address. In that case, we need to deploy virtual honeypots.

2. System Overview

The proposed system can be classified as low-interaction virtual honeypot system; it sits among production systems and emulates virtual honeypot machines with configured OS networking stack within simulated virtual routing topology. The proposed system creates virtual networks by using unused IP subnet addresses of the local network and distracts potential hackers from real production systems.

2.1 Simulation of virtual honeypots

The proposed system is implemented as a low-interaction honeypot system that simulates virtual honeypots with TCP and UDP services. It also understands and responds correctly to ICMP messages. Instead of simulating every aspect of an operating system, the proposed system simulates only its network stack. The proposed system uses TCP/IP stack spoofing techniques to create virtual honeypots with different Operating Systems. The main purpose of the TCP/IP stack emulation is to assure adversaries that they are probing remote network with real target machines running different Operating Systems. Another advantage of network stack emulation is that virtual honeypots simulated by the proposed system can handle different scanning probes such as, ping scan (ICMP ping, TCP ping, UDP ping), port scan (connect scan, stealth scan) and remote OS detect scan.

2.2 Simulation of Virtual Networks

The proposed system simulates virtual honeypots on multiple IP addresses simultaneously, in order to populate the network with numerous virtual honeypots with different operating systems and services. To increase the realism of simulation, the proposed system simulates arbitrary network routing topologies with configurable link characteristics such as latency and packet loss. When networking mapping tools like traceroute are used to probe the virtual network, they discover only the topologies simulated by proposed honeypot system.

3. System Design and Implementation

To fulfill the desired goal, the main operational steps that are performed by the proposed system are:

- Redirection of IP traffic destined for IP addresses of configured virtual honeypots to host machine upon which the proposed system is running
- Simulation of TCP/IP stack for virtual honeypot machines with configured Operating System personality
- Simulation of configured virtual static routing topology with virtual networks and virtual routers

3.1 IP traffic redirection

The proposed system is designed to reply to network packets whose destination IP address belongs to one of the simulated honeypots. To receive the correct packets, the network needs to be configured appropriately. There are several ways to do this, e.g., we can create special routes for the virtual IP addresses that point the proposed system machine, or we can use ARP spoofing, or we can use network tunnels.

The proposed system uses the ARP spoofing method to redirect IP traffic. When an adversary sends a packet from the Internet to one of configured virtual honeypots, local entry router receives and attempts to forward the packet. The router queries its routing table to find the forwarding address for destination machine. If no special route has been configured, the router makes ARP requests to determine the MAC address of the destination machine. As there is no corresponding physical machine, the ARP requests go unanswered. The proposed system host replies to

ARP requests for configured virtual honeypots with its own MAC address. This is called ARP spoofing and allows the router to send packets for virtual honeypot to proposed system MAC address.

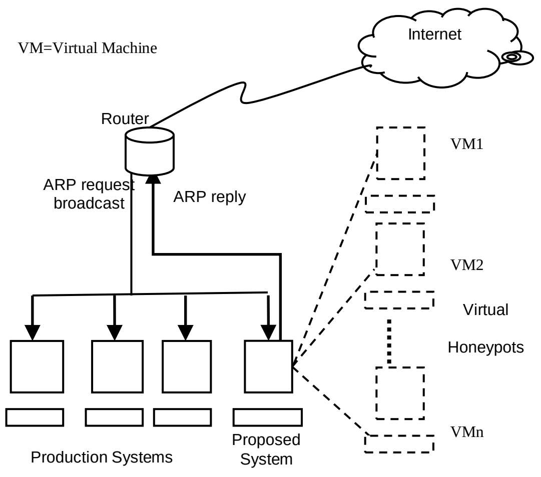


Figure 3-1 ARP spoofing to accept IP traffic for virtual honeypots

3.2 TCP/IP stack simulation

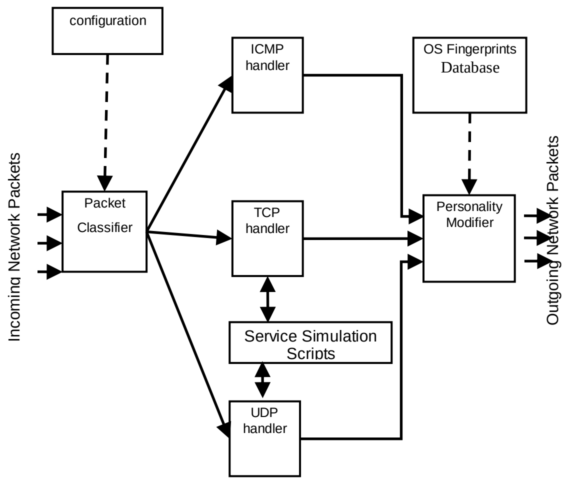


Fig 3-2 Components of TCP/IP stack simulation process

Adversaries commonly run fingerprinting tools like Xprobe [5] or Nmap [6] to gather information about a target system. It is important that virtual honeypots do not stand out when fingerprinted. To make them appear real to a probe, proposed system simulates the TCP/IP network stack behavior of a given operating system. We call this the personality of a virtual honeypot. Different personalities can be assigned to different virtual honeypots. The personality modifier makes a honeypot's network stack behave as specified by the personality by introducing changes into the protocol headers of every outgoing packet so that they match the characteristics of the configured operating system. The proposed system uses Nmap fingerprint database as its reference for TCP and UDP behaviors.

3.3 Virtual routing topology simulation

Proposed system's virtual routing topology is constructed with virtual router nodes and virtual network links. Starting from root entry router; which is located on physical network, acts as default gateway for one virtual network link. Another router resides on that virtual network and acts as default gateways for lower virtual sub networks. In this way rooted tree static virtual routing topology is implemented. Multiple entry routers scheme can be used to simulate virtual routing topology with more than one rooted tree structure.

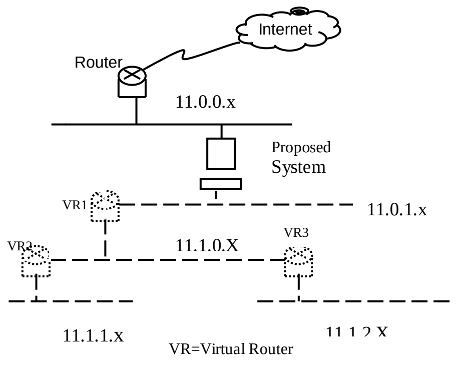


Figure 3-3 Virtual routing topology

4. Test result

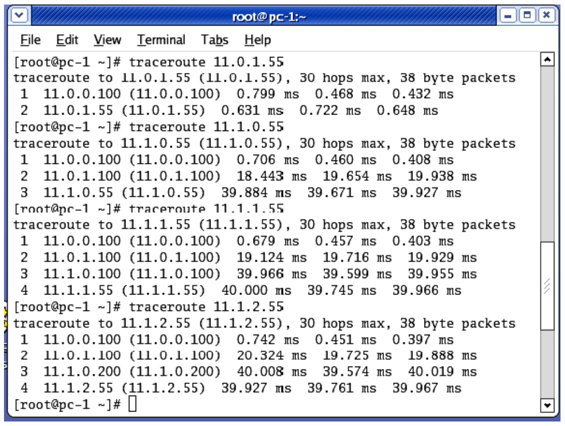
To test the output result of the proposed system, we use two PCs connected with PROLiNK 10/100M Fast Ethernet switch. The specification of the test computers and their Operating Systems are shown in table 1. First we configure the proposed system machine with IP address 11.0.0.1 and tester machine with IP address 11.0.0.5 to form the physical network 11.0.0.x

Role of Computer	OS	Processor	RAM
Proposed Honeypot System	RedHat Linux 9	1GHz IntelCeleron Pentium 3	128 MB
Tester	Windows XP SP2	2.0 GHz Intel(R) Pentium 4	512 MB

Table 1. Specifications of Computers used in Testing

To make testing, we configure the proposed system machine to simulate four virtual networks of class C size to form the virtual routing topology which is shown in Fig 3-3. The simulated virtual routing topology consists of 1014 virtual honeypot machines and 4 virtual routers.

We perform the traceroute testings from probing machine to virtual honeypots located at different depth virtual subnets and get the correct result which is shown in Fig 4-1 and Table 2.



```
[root@pc-1 ~]# traceroute 11.0.1.55
traceroute to 11.0.1.55 (11.0.1.55), 30 hops max, 38 byte packets
 1  11.0.0.100 (11.0.0.100)  0.799 ms  0.468 ms  0.432 ms
 2  11.0.1.55 (11.0.1.55)  0.631 ms  0.722 ms  0.648 ms
[root@pc-1 ~]# traceroute 11.1.0.55
traceroute to 11.1.0.55 (11.1.0.55), 30 hops max, 38 byte packets
 1  11.0.0.100 (11.0.0.100)  0.706 ms  0.460 ms  0.408 ms
 2  11.0.1.100 (11.0.1.100)  18.443 ms  19.654 ms  19.938 ms
 3  11.1.0.55 (11.1.0.55)  39.884 ms  39.671 ms  39.927 ms
[root@pc-1 ~]# traceroute 11.1.1.55
traceroute to 11.1.1.55 (11.1.1.55), 30 hops max, 38 byte packets
 1  11.0.0.100 (11.0.0.100)  0.679 ms  0.457 ms  0.403 ms
 2  11.0.1.100 (11.0.1.100)  19.124 ms  19.716 ms  19.929 ms
 3  11.1.0.100 (11.1.0.100)  39.966 ms  39.599 ms  39.955 ms
 4  11.1.1.55 (11.1.1.55)  40.000 ms  39.745 ms  39.966 ms
[root@pc-1 ~]# traceroute 11.1.2.55
traceroute to 11.1.2.55 (11.1.2.55), 30 hops max, 38 byte packets
 1  11.0.0.100 (11.0.0.100)  0.742 ms  0.451 ms  0.397 ms
 2  11.0.1.100 (11.0.1.100)  20.324 ms  19.725 ms  19.888 ms
 3  11.1.0.200 (11.1.0.200)  40.008 ms  39.574 ms  40.019 ms
 4  11.1.2.55 (11.1.2.55)  39.927 ms  39.761 ms  39.967 ms
[root@pc-1 ~]# 
```

Figure 4-1 Traceroute testing snapshot

Virtual Honeypot IP	Hop Count	1stRTT (ms)	2ndRTT (ms)	3rdRTT (ms)
10.0.1.55	1	0.631	0.722	0.648
10.1.0.55	2	39.884	39.671	39.927
10.1.1.55	3	40.000	39.745	39.966
10.1.2.55	3	39.927	39.761	39.967

Table 2. Traceroute testing results

We carried out to visualize the configured virtual routing topology by using Zenmap utility and get the visual routing topology as shown in Fig 4-2.

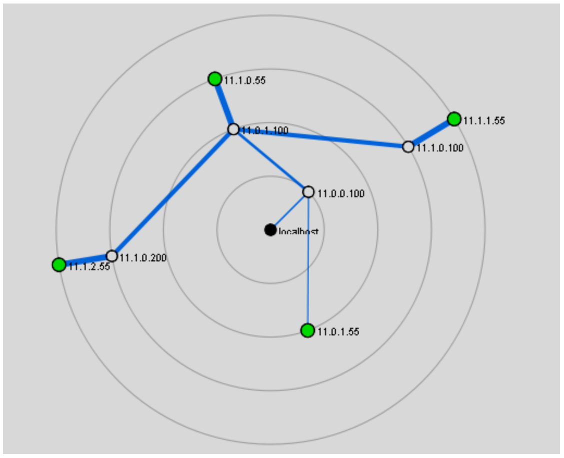


Figure 4-2 Zenmap topology discovery result

The next step to test the performance of the proposed system is to measure the response time speed to various scanning types using Nmap network scanner. To get the response time of the proposed system, we laugh different types of scanning probes on first level virtual network link 10.0.1.0/24 and Nmap shows all virtual honeypots (254 hosts) are up. The response times for various scanning types are shown in table 3.

Scan Type	Response Time for 254 hosts
Connect scan	244.478
SYN Stealth scan	258.850
ACK Stealth scan	211.453
FIN/ACK Stealth scan	209.962
FIN Stealth scan	219.439
NULL Stealth scan	210.246
Xmas Tree Stealth scan	211.927
TCP window scan	212.277

Table 3 Response time results

5. Conclusion

The proposed system presents a lightweight honeypot system for creating virtual honeypots. Proposed system allows us to instrument thousands of IP addresses with virtual machines and corresponding network services. The proposed system limits adversaries to interacting with virtual honeypots only at the network level. Instead of simulating every aspect of an operating system, the proposed system simulates only its network stack. The proposed system can handle virtual honeypots on multiple IP addresses simultaneously, in order to populate the network with numerous virtual honeypots simulating different operating systems and services. To increase the realism of simulation, the proposed system simulates arbitrary virtual network topologies.

5.1 Future work

The proposed system has some weakness in its implementation issue and future modifications to proposed honeypot system should address following facts:

- Currently available fingerprinting tools are usually stateless because they neither open TCP connections nor explore the behavior of the TCP state machine for states other than LISTEN or CLOSE. There are several areas like congestion control and fast recovery that are likely to be different between operating systems and are not checked by fingerprinting tools. An adversary who measures the differences in TCP behavior for different states across operating systems would notice that they do not differ in proposed system and thus be able to detect existence of virtual honeypots.

- Another method to detect virtual honeypots is to analyze their performance in relation to other hosts. Sending network traffic to one virtual honeypot might affect the performance of other virtual honeypots but would not affect the performance of a real host

References

[1]http://www.securityfocus.com/infocus/1897/1
[2] http://www.honeypots.net/
[3] Honeypots: Tracking Hackers http://www.tracking-hackers.com/book
[4] Know Your Enemy http://www.honeynet.org/book/
[5] Ofir Arkin and Fyodor Yarochkin. Xprobe v2.0: A “Fuzzy” Approach to Remote Active Operating System Fingerprinting. www.xprobe2.org, August 2002.
[6] Fyodor. Remote OS Detection via TCP/IP Stack Fingerprinting. www.nmap.org/nmap/ nmap-fingerprinting-article.html, October 1998.