International Journal Of Advanced Engineering & Innovative Technology (IJAEIT)
ISSN: 2348 7208
IMPACT FACTOR: 1.04

Cyber-Physical System Security with Deceptive Virtual Hosts for Network

Priyanka Khutwad, Tejaswi Konde, Yogita Ranjane, Pooja Rashinkar

Abstract: Honeypots are gaining importance as a useful security tool alongside firewalls, IDSs and antivirus software. Honeypot are intended for the early detection of attacking activity. Honeypots are intentionally configured to be vulnerable to compromise. Honeypot can be a computer, printer, router or practically any networked device that has value to a potential intruder. Protecting the typically large number of assets for which administrators are responsible is challenging task. In the control system these cyber devices may be coupled with the physical processes. Honey pots are effective and efficient tool used for gain more information about the attacker and observing network intruder activity.This paper introduces a design and implementation for self-configuring honeypots that actively examines network traffic. In this paper a novel four-step algorithm used which was developed for autonomous creation and update of a honeyd configuration. Honeyd when deployed creates virtual hosts. Virtual honeypots are configured in such way that it can dynamically configure itself and actively learn from the network traffic to detect the malicious data packets or users. This proposed system uses an unsupervised network attack detection system based on the multiple clustering algorithms. For unsupervised network attack detection the combination of Subspace and evidence accumulation clustering is used.

Keywords: Intrusion Detection System; Dynamic virtual Honeypot; Honeyd.

1. Introduction:

Honeyd is a low-interaction virtual honeypot that simulates virtual computer systems at the network level. Attacker are deceived by simulating the network stack of various operating systems, thus making him believe that he is interacting with a real system over the network. Honeyd simulates only the network stack rather than the entire OS which ensures that even if the honeyd gets compromised attacker cannot damage to large extent. Honeyd can be combined with a virtual machine (VM) to simulate multiple operating systems. To order to add the realism, honeyd simulates arbitrary network topologies which convince the hacker that he/she is on a real system. [1] The aim of the Honeypot is to collect data on attacks and attackers by monitoring the machine being attacked. Alternatively, it can emulate the operating system services to detect attacks. Because the Honeypot is a provision made by the owner of the network, it should not receive any internet traffic from the network or the outside world. Consequently, any traffic that comes to the Honeypot could be an attempt to attack or scan the system.

Mr Amit Kadam, Asst Professor, Department of Computer Engg., Priyanka Khutwad, Tejaswi Konde, Yogita Ranjane, Pooja Rashinkar, Research Scholar at B.E Computer Engineering, Navsahyadri Education Society's Group of Institutions,Pune.

Generally Honeypots are divided into two main types: passive and active Honeypots (client Honeypots). A passive Honeypot involves setting up a vulnerable system or service, or possibly their simulation, and then monitoring activity to detect any attack on the system. The objective of this process is to then improve the security of the operating system and network,
as well as gather information about the attacker's motives and behavior. By contrast, instead of waiting for attackers passively, an active Honeypot will go and search for the attackers. [2]

A. Passive Versus Active Scanning

The two primary means for gathering the necessary network host information to create a honeypot includes passive and active network scanning. Unfortunately, most research to this point provides minimal analysis on suitable tools for passive information gathering. This is a key enabling capability if the intent is to deceive an attacker into believing an emulated system is real. This paper corrects this deficiency by examining characteristics of six existing tools and consequently recommends a tool, previously not used in this context, called Ettercap [5]. In most of the literature reviewed, passive scanning has been implemented with P0f and occasionally Snort [4], [6]. P0f is a command line tool that utilizes an array of mechanisms to identify

hosts in a network stream. It is a passive OS fingerprinting tool frequently cited in creation of dynamic virtual honeypots. Snort is inherently a rule-based intrusion detection system. The amount of information that may be gleaned from passive scanning is a limited subset of possible information [7]. A passive scanning-based tool is restricted to only gathering data that is offered in the captured stream. If a service on a host is available, but not utilized, this data point will be missed. Active scanning may prove more successful at extracting this type of information. Nmap is an active scanning tool that has proven useful for interrogating hosts on a network [8]. However, a downside to active scanning is the possible interruption of services on hosts. This problem is especially acute in control systems. For instance, ping sweeps on older systems have been known to disrupt normal operation and cause physical damage [9]. Active scanning also provides a beacon of network activity outside the norm and could be revealing for intruders listening in on the traffic. In either case of active or passive scanning, the resulting information may be used to configure a honeypot.[3]

A. Level of interaction Classification is based on the level of interaction which is provided to the anomalous user by the honeypot system. If an interactive environment is presented, then there is more chance of becoming the honeypot as target, which enables to gather more accurate information about attacker.

1] Low Level Interaction: One or more simple services are made available which log all communication attempts to specific services, like a web or SSH server. These are just simple daemons which provide the person who configured them a passive way to monitor attack attempts. Basically host operating system is not vulnerable to attacks. Therefore low-interaction honeypots are safe to run. But at the same time unable to be used where a more complex, interactive environment is needed, such as SMTP server.

2] Medium Level Interaction: Medium level honeypots begin to emulate collections of software to present more attractive front to the attacker, but still able to protect the host OS. Emulating a collection of software is quite complex task as the emulated programs should respond the same way as their real counter parts. There are more points of attack for the anomalous user, hence chance of system compromise is raised.

3] High Level Interaction: Honeypots presented the complete operating system to the attacker, with actual instances of programs. The goal of high-interaction honeypots is for the attacker to gain root access on the machine and then monitoring the activities. This level of honeypot has the highest risk, with highest potential. [1]

3. Problem Statement

"Cyber-Physical System Security with Deceptive Virtual Hosts for Industrial Control Networks" In this paper, the collaborative use of dynamic virtual honeypots in a control system network is introduced. The presented algorithm focuses on automatically managing the complexity of self-configurable dynamic virtual hosts (DVH) by adapting to an operational network environment. A self-updating model, based on passive monitoring of the network devices, is created and maintained. This model is used to configure deceptive network entities designed to draw the focus of malicious intent.

4. Proposed System

The collaborative use of dynamic virtual honey pots in a control system network is used.

Algorithm focuses on automatically managing the complexity of self-configurable dynamic virtual hosts (DVH) by adapting to an operational network environment. A self-updating model is based on passive monitoring of the network devices, is created and maintained. Network entity identification, DVH configuration and virtual host instantiation are three major functions. The NEI component monitors network traffic.

5. System Requirement

A) Domain: Network Security.

B) Software Interfaces:

Operating System : Windows
Language : JDK 6
Data Base : My Sql
Front End : Java
Back End : MySql

c) Java (JDK 6)

Java is a general purpose programming language with a number of features that make the language well suited for use on the World Wide Web. Small Java applications are called Java applets and can be downloaded from a Web server and run on your computer by a Java-compatible Web browser, such as Netscape Navigator or Microsoft Internet Explorer.

d) MySql Server 5.1

MySQL is a popular choice of database for use in web applications. Many programming languages with language-specific APIs include libraries for accessing MySQL databases. MySQL is primarily an RDBMS and ships with no GUI tools to administer MySQL databases or manage data contained within the databases. Users may use the included command line tools,[citation needed] or download MySQL front-ends from various parties that have developed desktop software and web applications to manage MySQL databases, build database structures, and work with data records.

6. System Architecture:

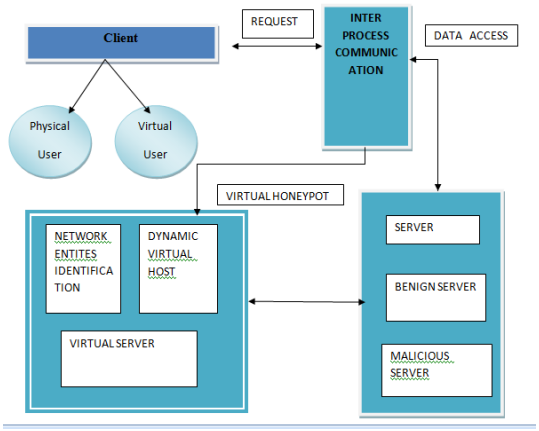

Fig. 1. Conceptual design diagram.

This section describes the software tool evaluation and implementation logic of the solution. Fig. 1 shows the relationship of three key functional areas: 1) network entity identification
(NEI); 2) DVH configuration; and 3) virtual host instantiation (VHI).

6.2 Network Entity Identification

The NEI component monitors network traffic from which it extracts the source, destination, and port activity. Information from the NEI is delivered to an implementation of the logic
tasked with creating a DHP configuration. An evaluation was conducted on six passive network information gathering open source tools to determine their strengths and weaknesses relevant to support of automated configuration. The tools evaluated for providing network host identification are: P0f [10], Tshark [11], Ettercap, Snort [12], Tcpdump [13], and Ntop [14]. Of the six tools, Ettercap and Ntop provide wellformatted structured output as an option. Another tool, called SinFP [15], was removed from consideration because it did not execute correctly on the test sensor system. Ettercap is an extensible network manipulation and reconnaissance tool [5]. In conclusion of this section, the Ettercap tool was selected for identifying network entities. It provides information on host
Internet protocol (IP) addresses, MAC values, and port usage.[3]

6.3. Dynamic Virtual Hosts

This section discusses the configuration creation of the DVH. These host emulate the network signature of actual systems on a physical network. Honeyd is a popular open source solution for virtual honeypots that provides flexible and feature rich configuration capability. As autonomous configuration is a desired aspect for minimization of expensive manual configuration,Honeyd's configuration flexibility is an advantage. The overall goal is the automatic configuration and dynamic update of a variable length list of virtual hosts based on information gathered from actual hosts using Ettercap.The following sections describe four functional areas in DVH:

1) OS selection; 2) OS name mapping; 3) MAC creation; and 4) Service (port) emulation.
1) OS Selection: For any given host on a network, Ettercap may not be able to identify the operating system. If this occurs,for an emulation target, then an OS must be chosen. It is desirable to provide an exact match in network behavior. This does not necessarily require an exact match with the OS name in the database.
2) OS Name Mapping: The Honeyd configuration value for an OS makes use of the Nmap version 1 database defined named values. Similarly, Ettercap utilizes its own defined name values that do not directly match Nmap. To make a functional configuration, a simple algorithm implemented in Map _OS was developed to associate Ettercap names with Nmap names. The algorithm's initial pass

compares the word tokens of the OS names, looking for case-insensitive string matches. The number of word matches were summed and stored. After iterating through each possible OS combination, the one with the largest count total is presented as a candidate. Finally, each OS name combination is written to a file for reference during creation of the configuration.

3) MAC Creation: Honeyd provides two options for specifying the MAC address, either by vendor name or the six-octet string. Because Honeyd has hard coded vendor strings, the six-octet representation was chosen for use in the algorithm. Ettercap captures this MAC octet address for all hosts in The MAC protocol specifies that the first three octets are organizationally unique and should not overlap with any other vendor.

4) Network Service Emulation: The host entries in O contain network ports, previously defined as Ph, that were active during the capture session. Along with the port number, a port service name is available. This service name is a human readable text value that is defined in an Ettercap configuration file called etter. services. Utilizing the service names contained in this file, a new configuration file called serv.conf was created. This file maps the service name to a service emulation script path.

C. VHI and Update

The candidate emulation hosts are provided at startup as a list of IP addresses. It is assumed that if a host in the list disappears from passive sensing, then the user still desires to have an emulated version of it. The overhead to maintain the missing hosts' records is minimal.Of course, the actual system has to

```
create vh1
set vh1 personality "Linux 2.4.xx"
set vh1 default tcp action reset
set vh1 default udp action reset
set vh1 default icmp action reset
add vh1 tcp port 23 "/script/router-telnet.pl"
set vh1 ethernet "00:00:BC:A1:00:23"
bind 192.168.1.125 vh1
```

Fig. 2. Honeyd host configuration.

have appeared in the passive analysis during the monitoring period to create an initial virtual host configuration. Aninitial configuration file is created by Create _Host _Conf .Changes to the configuration of the virtual hosts running under Honeyd are performed while the system is running. After a configurable time period, currently an arbitrarily chosen 60 s,etterlog is called on the ettercap daemon log file. The resulting XML output is saved and compared to an existing output file.Differences in network host activity are noted and stored on a list for possible action. Actions include adding network services,updating OS configuration, and changing MAC addresses. A companion Honeyd executable file, called Honeydctl, providesthis functionality.[3]

7. Algorithm:

A .Apriori algorithm

Apriori is proceeds by finding the repeated inseparable items in the record and extend them to superior and larger item set as extended as those item sets appear adequately frequently in the record. The constant thing sets resolute by Apriori can be used to decide association rules which denotes database management system: this has applications such as transaction method. Apriori is designed to databases containing database management system.

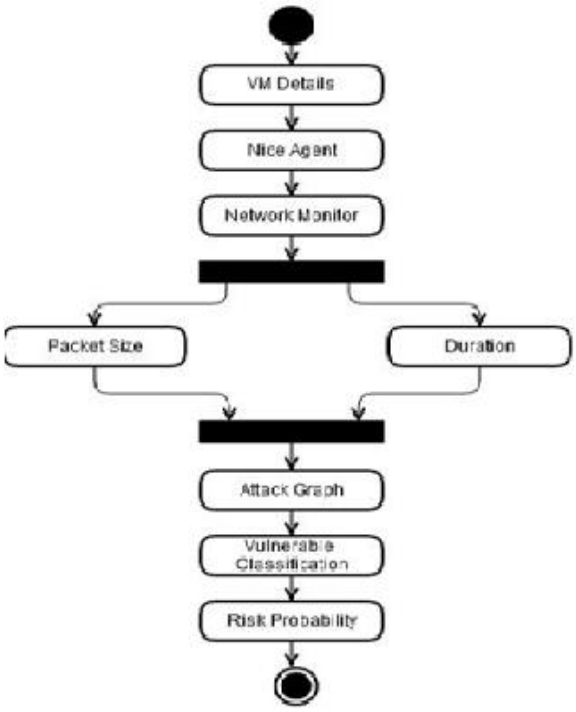

Fig 3: The workflow mechanism of the proposed system

8. Advantages

- Detecting attacks: these systems can easily detect attacks, which have escaped from the firewall .
- Such a system detect real-time intrusion

9. Conclusion

This work has identified several areas of possible future research. The use of virtualized networks and devices derived from the automated system presented could subsequently be used as a standard test bed for a variety of IDS systems. An algorithm was proposed and demonstrated to automatically deploy deceptive virtual network entities in a control system network. Six open source passive network-monitoring tools were evaluated and Ettercap was chosen for host identification.

10. References

[1] International Journal of Innovative Research in Computer and Communication Engineering *(An ISO 3297: 2007 Certified Organization)* Vol. 3, Issue 7, July 2015

[2] Analysing Web-based Malware Behaviour through Client Honeypots Yaser Alosefer February 2012.

[3] Cyber-Physical System Security With Deceptive Virtual Hosts for Industrial Control Networks Todd Vollmer, Senior Member, IEEE, and Milos Manic, Senior Member, IEEE May 2014

[4] J. Hieb and J. H. Graham, "Anomaly-based intrusion detection for network monitoring using a dynamic honeypot," Intell. Syst. Res. Lab., Univ. Louisville, Louisville, KY, TR-ISRL-04–03, Dec. 2004.

[5] Ettercap network sniffer [Online]. Available: http://ettercap.sourforge.net/

[6]C. Hecker and B. Hay, "Securing E-government assets through automating deployment of honeynets for IDS support," in Proc. 43rd Hawaii Int. Conf. Syst. Sci., Koloa, Kauai, HI, USA, 2010, pp. 1–10.

[7] F. Gagnon and B. Esfandiari, "A hybrid approach to operating system discovery based on diagnosis," Int. J. Netw. Manage., vol. 21, pp. 106–119, Mar. 2011.

[8]G. Lyon, Nmap Network Scanning. Palo Alto, CA, USA: Insecure.org, 2008 [Online]. Available: www.nmap.org

[9] D. P. Duggan, "Penetration testing of industrial control systems," Sandia National Lab., Albuquerque, NM, USA, Tech. Rep. SAND2005-2846P,Mar. 2005.

[10] P0f [Online]. Available: http://lcamtuf.coredump.cx/p0f.shtml

[11] Tshark Network Analyzer [Online]. Available: http://www.wireshark.org/

[12] M. Roesch, "Snort: Lightweight intrusion detection for networks," in Proc. 13th Conf. Syst. Admin., Berkeley, CA, USA, Nov. 7–12, 1999,pp. 229–238.

[13] Tcpdump Packet Analyzer [Online]. Available: http://www.tcpdump.org/

[14] Ntop Network Traffic Probe [Online].Available:http://www.ntop.or/

[15] P. Auffret, "SinFP, unification of active and passive operating system fingerprinting," J. Comput. Virol., vol. 6, no. 3, pp. 197–205, Aug. 2010.