ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 2, Issue 2, February 2015

Securing Industrial Control Networks with Cyber System Physical Security Method by Using Virtual Honeypots

Divya Ambrita R.L[1], Sujithra Jenifer.M[2], Vinu.J[3]
Department Of Information Technology, Francis Xavier Engineering College, Tirunelveli, India[1]
Department Of Information Technology, Francis Xavier Engineering College, Tirunelveli, India[2]
Assistant Professor, Department Of Information Technology, Francis Xavier Engineering College, Tirunelveli, India[3]

Abstract— This paper presents a design and implementation for self-configuring honeypots that passively examine control system network traffic and actively adapt to the observed environment. In contrast to prior work in the field, six tools were analyzed for suitability of network entity information gathering. Ettercap, an established network security tool not commonly used in this capacity, outperformed the other tools and was chosen for implementation. Utilizing Ettercap XML output, a novel four-step algorithm was developed for autonomous creation and update of a Honeyd configuration. This algorithm was tested on an existing small campus grid and sensor network by execution of a collaborative usage scenario. Automatically created virtual hosts were deployed in concert with an anomaly behavior (AB) system in an attack scenario. Virtual hosts were automatically configured with unique emulated network stack behaviors for 92% of the targeted devices in the AB system alerted on 100% of the monitored emulated devices.

Index Terms—Industrial control, intrusion detection, network security, self configuring honey pots ,deceptive systems

I. INTRODUCTION

Many modern complex control systems are interconnected via Ethernet networks. These networks, found deployed in areas such as chemical facilities or energy production, are utilized to deliver status and control information vital to the operation of physical systems. A compromised control system could have security, public safety, industrial or economical consequences [1], [2]. The need for resilient adaptive security systems, specifically developed for critical cyber-physical systems, is increasing with the elevated levels of cyber security threats in the modern world [3], [4]. Furthermore, with the advent of the smart grid, the number of configurable devices to be deployed is relatively high. For example, in a typical advanced metering infrastructure (AMI) system, 1500 wireless sensors report to one or multiple wireless access points (WAP) nodes [5]. As of April 2010, almost 69 million of these meters were planned for deployment in the United States [6]. Assuming a uniform deployment of sensors, this plan calls for 46 000 WAPs. So, in addition to protecting existing networks, a large-scale deployment of new devices will soon be prevalent. Network security monitoring systems are a significant part of a solution to protecting control systems. In most contexts, they are rarely capable of providing perfect intrusion detection [7], [8]. Deceptive systems, called honeypots, that emulate critical network entities have been deployed in tandem with monitoring solutions to improve detection accuracy and precision rates [9], [10]. It is difficult to list the definitive attributes of a network host necessary to attract an attacker's attention. This requires analysis of attackers' motivations, which may vary in depth and details depending on the situation. However, a reasonable assumption can be made that if any of the real devices on the network are a desirable target, than emulation of those systems would be a productive exercise. Given this premise and the issue of a large device deployment, a relevant concern is reducing the human effort involved while providing an improved security posture.

In addition to a honeypots faithful reconstruction of a host's network presence, automation is a key capability. According to John Ouster host, there are four common steps for turning deployments from an enemy into a friend [11].

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 2, Issue 2, February 2015

First, and most important, is automation. This is essentially a question of economy. It is usually cheaper to build better tools than manually manage the configurations of individual devices in a large system.

In this paper, the collaborative use of dynamic virtual honeypots in a control system network is introduced. Aspects of effective tools for identifying network host characteristics are examined. 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. Finally, a usage scenario is examined to show how imitating a real network is useful when combined with an anomaly detection routing.

Honeyd simulates the network stack and generally provides only superficial services. Because of this, an attacker is never able to gain access to the host by compromising a service but would quickly realize that something is amiss. The primary goal is not to entrap the attacker into spending all his effort on the deceptive system. It is to attract his attention, for at least a short time and gather information that helps identify the attacker and a possibly compromised internal attack platform.

In this paper, Honeyd was evaluated and logic created to automatically configure it. The resulting configuration is designed to emulate, as close as possible, any user-identified host on the network. This is in contrast to previous work that focused primarily on dynamically creating several honeypots, called a honeynet, that in aggregate are statistically similar to a network of hosts [13].

High-interaction honeypot systems are typically hardware replicas of existing operational components that include the appropriate software. For the purpose of this discussion, virtual machines are included in the high category. These systems do not mimic services, but are deployed with working instances. This type of system provides a high-fidelity solution that is less prone to discovery of its deceptive purpose by network intruders. However, they are at a higher risk for compromise by an attacker and require a more complicated deployment investment. Deploying a virtual machine is simpler than a hardware base system, but still requires complex management scenarios for deploying a wide array of service software. This includes having copies of multiple OS distributions and server software.

Finally, honeypots, high or low interaction, can only detect attacks directed at them. A competent attacker who discovers that a system is a honeypot will avoid any further contact with that system. The fidelity of the deception is in the presentation of the honeypot to the network. How the data is gathered to create this deception is important.

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 [14].

In most of the literature reviewed, passive scanning has been implemented with P0f and occasionally Snort [13], [15]. 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 [16]. 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 [17]. 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 [18]. 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.

II. RELATED WORK

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 2, Issue 2, February 2015

DHP solutions that gather network information, process that information into a configuration, and deploy appropriately have been created as in [13], [15], [19], and [20]. These papers propose monitoring methods that are active [19], passive [13], combined [15], or ambiguous [20]. When passive monitoring is implemented, the chosen tool is typically P0f with no analysis of competing tools provided. Finally, the test implementations are all on noncontrol system networks.

There are two notable projects related to control system honeypots. The supervisory control and data acquisition (SCADA) Honeynet project by Matthew Franz and Venkat Pothamsetty of the Cisco Critical Infrastructure Assurance Group (CIAG) was initially released in March 2004 [21]. The project is not actively maintained, with a last release date of July 15, 2005; however, it is still available from Source forge. The design utilizes Honeyd for simulating a set of services for a PLC. The major contributions of this project are service scripts, which include functionality for file transfer protocol (FTP), Mod bus, Telnet, and a web server. However, the SCADA Honeynet does not consider automatic provisioning of the virtual hosts and is a manually configured project.

Digital Bond, Inc. is a control system security consulting and research group founded by Dale Peterson. Their SCADA Honeynet implementation is an evolution of the original project just described [22]. It utilizes two virtual machines instead of Honeyd. One virtual machine includes network monitoring tools such as Snort with Digital Bond's Quick draw IDS signatures to detect activity. The other virtual machine simulates a PLC with several exposed services. There is no dynamic provisioning of hosts or services, although it is possible to replace the virtual machine PLC with an actual hardware component. This assures complete deception if the PLC is configured correctly with the added expense of an actual hardware device.

III. SOLUTION DESIGN

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). These act in a continuous cycle of processing and updating information represented by the dotted line box.

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. These hosts emulate the actual systems. Honeyd is the popular open source solution which helps for configuration process. As autonomous configuration is an important aspect in this paper and it reduces human involvement the honeyd configuration is an advantageous one. Anyhow the overall goal is the configuration and emulation by network information gathering.

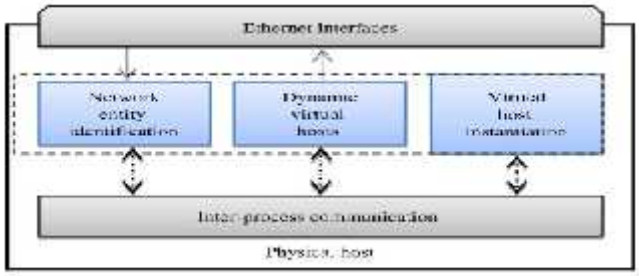

Fig. 1. Conceptual design diagram.

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 [23], Tshark [24], Ettercap, Snort [25], Tcpdump [26], and Ntop [27]. Of the six tools, Ettercap and Ntop provide well formatted structured output as an option. Another tool, called SinFP [28], was removed from consideration because it did not execute correctly on the test sensor system.

In addition to identifying network entities, NEI needs to provide the information necessary to create a representative virtual network presence. The critically required capabilities examined were OS identification, port or service identification per host, and the capture of media access control (MAC) addresses with a resolution to the appropriate vendor [15]. Considering that Ntop and Ettercap fulfill all three criteria. Of the two candidates, Ettercap was chosen for its support of XML output, completeness of information provided from this output, and available functionality for support of future work.

The system, as configured during the test, had 46 physical connections to the network. The second column contains the number of OS identified by each tool. Ntop's identification of 202 hosts in column 3 contains duplicate entries for entities that have both IPv4 and IPv6 addresses. Additionally, records created for broadcast addresses inflate

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 2, Issue 2, February 2015

the host number. Ettercap outperformed or equaled the other tools in three of the four categories.

Ettercap is an extensible network manipulation and reconnaissance tool [14]. It is an established and popular tool in the hacking community. However, this paper is the first to establish its use as a source of information for DHP creation. It was run as a daemon process with unified sniffing. In this mode, it maintains internal network host records and updates them as new information is found. A binary log file is continuously updated as well. An Ettercap companion executable Etterlog is then run on the log file with a -x option to produce an XML file. This data file is the source for communication of the network entity information to the DVH configuration process.

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. When compared with the five tools listed in Table I, it performed as well or better than all of them. An additional key driving capability is Ettercap's formatted XML output that can easily be integrated into other systems. Communication within an automated system requires a defined consistent messaging system. Finally, Ettercap is capable of performing more advanced operations that could be useful for future functional enhancements.

Dynamic Virtual Host

This section discusses the configuration creation of the DVH. These hosts emulate the network signature of actual systems on a physical network. Honeyd is a popular open source solution for virtual honeypots that provides a 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.

Read_data consists of extracting n host records h from the Ettercap entries and forming a record set O such that O=h1,h2……,hn. O then becomes a source of information for creation of virtual hosts. The intention is to examine these records for similarities to an IP address provided in a list of j target IP addresses, where IL=i1,i2….ij. An assumption is being made that the hosts h on the network have an OS similar to a candidate i even if an exact match is not found.

Given that Ph is a set of port values for a host h and a network port set si for target ti ,Find _Closest examines the intersections of Si intersection Ph for all h in O . The integer count of matching ports is stored for each intersection. In addition, the number of ports for the target is calculated. Given these values, a match percentage is calculated, e.g., two candidate ports and an intersection count of two constitute a 100% match. Candidates with a higher percentage were considered to be more similar. Some OSs utilize ports specific to services offered by that OS, and they could be used in identification [16].

If a candidate OS is not identified by examining ports, then the MAC address is examined. Find _Closest compares the vendor identification section of the candidate MAC address of i with the MAC addresses for each host h in O . If a match is found that has an identified OS, then this value is placed on a candidate list.

1)OS selection:

After exhaustively examining, the largest matching value, if one exists, from the candidate list is chosen as the OS. The assumption is that any hosts on the network that have the same NIC vendor may be performing similar functions and thereby have a similar OS. As is described later, several control system vendors have an organizationally unique identifier for their network devices. If no prior step has identified an OS, a random number is generated in the range 0 to where is the cardinality exists. If the host record OS field exists, then this value is utilized. If not, a random value supported by Honeyd is chosen. In other words, a field is possibly selected for inclusion proportional to the relative frequency of its presence in . Given that not all host records contain an OS and possibly none of them.

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

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 2, Issue 2, February 2015

simple algorithm implemented 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. Thus, in order to create a new MAC address that appears to come from a specific vendor, these first three octets were used. The vendor typically assigns the remaining three octets.

In this function, the last three octets are randomly generated and appended to the end of the captured candidate vendor portion. This new MAC is then compared with all other MACs noted in the Ettercap host list . Any collision of addresses instigates a recreation of another random set of values. Given the possible values, the probability of a collision is

Network Service Emulation

The host entries in contain network ports, previously defined as 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. The _ function examines any service ports found in the Ettercap output and loads the serv.conf file. Any service name match to entries in the file results in the appropriate service script value placement in the Honeyd configuration. This enables the creation of service specific behaviors that furthers the goal of deception. Currently, the manual creation of scripts is necessary, although some service scripts are already available from other projects. Automatic creation of these behavior scripts is another future area of exploration.

```
<host ip="192.168.1.25">
  <mac>00:0C:29:77:61:78</mac>
  <os>D-Link DWL-900AP</os>
  <port proto="udp" addr="68" service="dhcpclient"/>
  <port proto="udp" addr="123" service="ntp"/>
  <port proto="udp" addr="137" service="netbios-ns"/>
</host>
```

In addition to services found during passive scanning, a variable number of ports associated with the common services are randomly activated. A common service mapping file for control system devices is utilized by the function. It consists of a hierarchical MAC mapping structure. Generally, in the case of a control system device, the vendor portion of the MAC is directly tied to the device manufacturer, enabling usage of the mapping file to find relevant services. Constructed utilizing XML, the file maps the vendor MAC to a list of common services that are possible to find activated on a device of this type. Each service in the file is described by the following attributes: port number, protocol, service description, and action script. The action script specifies which script Honeyd should utilize, if any, when it sees traffic to this port. A value in this field will overwrite any previously defined default script found in server configuration. This provides the capability to customize a response to this specific device type while still retaining generic service emulation functionality.

Each service description has an "include" value. This is a floating-point value between 0.0 and 1.0. This value is compared to a randomly generated value in the appropriate range. If the random value is less than the include value, then the port is added to the honeypot configuration. The intention is to vary port inclusion to represent the variability in device configurations.

An analysis of available vendor product specifications was
used to create this file. For example, the test system contains a Rockwell Micrologix 1100 PLC and the possible services listed for this consist of Ethernet/IP, web services, simple mail transfer protocol (SMTP) email (outbound), and simple network management protocol (SNMP) [29].

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

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 2, Issue 2, February 2015

system has to have appeared in the passive analysis during the monitoring period to create an initial virtual host configuration. An initial configuration file is created by 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 executable file, called Honeyd provides this functionality.

USAGE SCENARIO AND RESULT

In the following test scenario, scans and probes are directed at all devices on the network representing the reconnaissance phase of an intrusion. This assumes that the attacker is an outsider and does not have a network map. The goal of the security system is to generate informational alerts about the anomalous presence. A secondary effect is the diversion of attention and effort of the. attacker to a virtual honeypot system. Keys for success include: a faithful imitation of real devices on the network, a mechanism for monitoring activity directed at the honeypots, and appropriate communication of emulated IPs and alerts. To improve the cyber security of network systems, various approaches can be applied [30]–[32]. One of the most common approaches is anomaly detection. An anomaly detection system is trained on a set of normal network behaviors. The extracted behavior model is then used to detect anomalous behavior in any subsequently observed traffic. One of the difficulties of this approach is building a comprehensive normal behavior model for a specific network communication system. Typically, a user-defined period of activity is designated as “normal”. However, by definition, any network activity directed at a honeypot can be considered abnormal. This provides a definitive source of information for classifying traffic that does not require direct user interaction. Anomaly behavior (AB) implementation details are not covered in this paper, but may be found in previous work of the authors [32], [33]. For this test scenario, an AB system was configured to monitor the virtual honeypot IP addresses and send alerts on any activity.

The role of the automatically created honeypots is to attract and possibly delay an intruder on the network. This usage is similar to that proposed in [7] and [34]. The intended deployment is an operational control system network with a heterogeneous mix of hosts. There are two possibilities for timing when the honeypots are instantiated. The first approach, used in this test scenario, is to create the virtual hosts in advance of any anomalous situations. This would increase the probability of a network scan identifying the hosts. It removes the race condition between recognizing an anomaly and getting the hosts instantiated in time to get noticed. The second approach, with the race condition, would be to instantiate the hosts after some indication of intrusion has occurred. This indication could come from a traditional intrusion detection system or some other security mechanism. Given the DVH use of virtual hosts with its reduced hardware requirements, a dedicated integrated host and low-network impact, there is little benefit to delaying instantiation until after detection. At the beginning of the scenario, all hosts are running and a sensor host with the virtual host logic is connected to the control network. As the NEI component becomes aware of changes in the host characteristics, the honeypots are automatically reconfigured to include the new behavior. The emulated hosts become more authentic appearing, in the service ports offered, over time. As already mentioned, this early instantiation reduces the risk of a stealthy intruder bypassing the honeypots, as they will most likely be present prior to the malicious activity.

Test Network

An existing small campus grid (SCG) and sensor network that physically exists in the Center for Advance Energy Studies in Idaho Falls, Idaho was used to test the algorithm. The network includes a suite of wireless sensors targeted at environmental conditions in the building, wind and solar renewable resources, and a variety of control system devices. The SCG is connected to a small wind turbine, a solar power station, and a wireless AMI. Additionally, the network has several Windows-based computers, web camera’s, a Rockwell Automation PLC, and a National Instruments PLC. The SCG network contains wireless systems from Emerson, Honeywell, and Arch Rock. Each system connects wirelessly to the sensors via a wireless access point. These WAP gateways have a wired connection on one side of the network and wireless interfaces to remote environmental sensors. The network sensor device has visibility on the wired side of the connection. Each wired WAP connection has a variation in the method of Ethernet network protocols utilized that makes each one a unique challenge to emulate. For instance, the

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 2, Issue 2, February 2015

Emerson device transports data at the raw Ethernet level using a custom protocol.

The software for the implemented algorithm was deployed on a test host platform. This platform runs a 32-bit Ubuntu 12.04 OS on a dual core Intel Atom 330 processor with 2GBof double data rate 2 (DDR2) RAM, a 250-GB hard drive and three gigabit Ethernet (GigE) network ports. One of the Ethernet ports was dedicated for use by the honeypot. Honeyd is capable of running multiple virtual hosts on one physical network interface. The second port was used to perform passive monitoring by NEI. The final port was connected to a second separate network used for management of the devices.

Test Steps and Result

A PERL implementation of the algorithm was run on the test sensor platform attached to the operational test network. In addition to OS emulation performance, seven network test probes were completed. The ID column is used as a reference identifier and corresponds to the last octet used in the emulated IP address. For completeness, the Honeywell wireless access point is included. Because it does not utilize an IP address for communication, Honeyd cannot emulate this device.

Initiate Honeypots:

An input text file for the DVH component contained two sets of space delimited IP addresses labeled R and E. List R contains the unordered IP addresses of real hosts. List E contains the list of IP addresses to be assigned to the emulated hosts. The lists represent a bi adjective function, in that f: > is a one-to-one and onto mapping of set R to set E. The same message was initially sent to DVH. Three virtual honey pots were created and verified by sending Internet control message protocol (ICMP) echo messages. After 60 s, a newly updated input text message was sent containing 12 test IP addresses. The software automatically created configurations for all of the devices. Each emulated host was assigned its own unique IP and MAC address and was instantiated on the test sensor hardware. These actions verified that the integrated communication mechanism works and virtual hosts are instantiated. Specifically, the NEI component created a network model stored as an XML file. The message passing mechanism is a simple text file dropped into a specific directory. The application continuously monitors the appropriate directory for a new file. After receipt of the message, the AB commenced passive monitoring of the 12 virtual hosts. Of the 13 devices initially chosen for emulation, 10 specific OS were configured autonomously, two were “random,” and the Honeywell device was undetermined.

```
<host ip="192.168.1.25">
  <mac>00:0C:29:77:61:78</mac>
  <os>D-Link DWL-900AP</os>
  <port proto="udp" addr="68" service="dhcpclient"/>
  <port proto="udp" addr="123" service="ntp"/>
  <port proto="udp" addr="137" service="netbios-ns"/>
</host>
```

1) Network Scan Tests and Results

Nine tests, described next, were executed on the virtual hosts using Nmap, the open vulnerability system (Open VAS), and the ping command line tool. Nmap version 5.21 was chosen to test the network presence of the emulated devices. This version utilizes the second generation Nmap OS database that is actively maintained. It uses a more robust guessing implementation for uncertain signatures. OpenVAS is a flexible comprehensive security scanning tool. It is capable of over 30 000 network vulnerability tests. A laptop, with Nmap, Open VAS, and ping installed, was assigned the IP address 192.168.1.15 and attached to the SCG network. The laptop filled the role of network intruder.

a) Test1: This simple test performs a “ping sweep” on all 256 addresses in the range that contains the 12 emulated devices. A combination of an ICMP echo request, transmission control protocol (TCP) SYN to port 443, TCP ACK to port 80, and ICMP timestamp request are sent. Any system that responds to one of these requests is considered available on the network. All 12 of the emulated addresses were found in 2.2 s.

b) Test2: This command line is the first example provided in the Nmap man page documentation. The -A option enables aggressive scan options including OS detection, version scanning, script scanning, and trace route. The -T4 option is a timing template that improves scan time on reasonably stable networks. Note that, by default, Nmap only scans 1000 of the most commonly used ports. It completed in 234 s. OS detection in Nmap is based on a database of signatures. Each fingerprint record in the database contains four fields: vendor, OS family, OS

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 2, Issue 2, February 2015

generation, and device type. Output from detection includes lists of possible OSs and device classes with an accuracy score. The score falls in a range of 0.0–1.0 with the later indicating a perfect match. The OS detection produces large amounts information. For the 12 emulated devices, 223 device types and 40 OS matches were returned. In both cases, accuracy ranged from 0.85 to 0.97. As there were multiple results for most of the emulated devices, any of the entries that matched either the original device or its mapped OS were considered a success. Of the 10 non-random devices, Nmap identified seven for a 70% success rate. Of the three that failed, no information was produced for device 2. Twenty-one incorrect entries were created for device 215. Device 5 was identified by one incorrect entry.

c) Test 3: This scan sends IP packets and iterates through the 8-bit IP protocol field. The emulated hosts responded to only three of the 256 protocols: ICMP, TCP, and UDP.

d) Tests 4: Utilizing the ping command line tool, ICMP echo requests were sent to the 12 emulated and 46 actual devices on the test network. ICMP packets are wrapped in an IP datagram and can contain IP option fields. Three rounds of requests were sent, one with the Record Route (-R) option, one with timestamp only and finally the option with both IP and timestamp address (tstampaddr). All but one of the physical devices responded with varying levels of correctness to the pings. None of the 12 emulated devices responded correctly.

e) Test 5: The OpenVAS framework was leveraged to perform more intensive network probes than Nmap on the virtual hosts. A single large-scale discovery and vulnerability scan was executed against the 12 virtual hosts. Of the available 32 418 plugins, 3778 were enabled for the scan. Plugins are attributed to a wide variety of functional categories and enable specific scanning behaviors. Many of the plugins execute on the target host with the appropriate credentials. Host plugin types were disabled. All 12 devices and their open ports were discovered during the scan. The activity took 21 min and 44 s to complete, and a scan report was produced. At the initial level of detail, the finished scan report looked similar to those reports from scans against the actual hardware. However, several differences were found when looking at the details. All of the devices had a common warning about a multicast address response flaw that could lead to a denial of service attack. This kind of similarity could possibly be leveraged to facilitate identification of virtual hosts. In this particular instance, a configuration change to the virtual hosts would remove the commonality.

f)Test 6—Anomaly Test: As was mentioned earlier in this section, a message with the 12 emulated IP addresses was sent to the AB component. The function of the AB component is to passively monitor host traffic and send alert messages. If the AB component receives an input IP, for which it has not been trained, then it will consider all traffic to it as abnormal. This is a convenient feature for the intended use of honeypots in this system. The AB posted abnormal behavior messages for %100 of the monitored emulated hosts during tests 1–7. A . It contains the source IP address, destination IP address, and the IP layer protocol number involved. In the example, protocol 1 indicates ICMP.

ANALYSIS

Although honeypots, physical or virtual, emulate real operations systems at some level, there is no guarantee that attackers would perform a scan of a network. However, if one is conducted having emulated devices similar to actual devices can provide a benefit to the security of the system. Minimally, it makes the attackers analysis of the devices difficult by increasing the amount of data to analyze. Additionally, the attacker will waste time and effort if an emulated device is chosen for further probing. This provides defenders with extended opportunities to identify intruders on the network. Based on information from the tests, industrial control network protocols are a viable candidate for emulation by the presented algorithm. Application ports are fixed, unusual ports that readily identify the use of a particular protocol. Given the passive nature of information capture, active network sessions are needed to discover the ports and nature of the service. For instance, the test system contains a Rockwell Micrologix 1100 processor that uses EtherNet/IP for communication.

The network traffic from the operator HMI to this device occurs on port 44818 using TCP. The TCP connection is maintained for the duration of the session. The traffic between the HMI and control device is regular in size and timing. The packet lengths were as follows: 19.15% between 40 and 79 bytes, 80.82% between 80 and 159 bytes, and 0.04% between 160 and 319 bytes. The average packet size is

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 2, Issue 2, February 2015

95.861 bytes. This regularity benefits the anomaly detection algorithm as well. The background section is a discussion on the choice of passive scanning for host information. One side effect of passive scanning is the inability to directly identify network ports not in use. While the control system network traffic is typically regular and a constant connection, it or other services may not be enabled. However, having these inactive services as part of the virtual hosts is beneficial to the presented deception. The mechanism to support this capability is found in the Device_Features function described in the solution design section. Originally created to add optional services for control devices.

An Nmap scan of the device revealed six open TCP ports. Passive scanning identified four of the ports. One of the missing ports was for a terminal service that was not accessed during the test time frame. This terminal service port was subsequently added to a service file with a probability of addition set to 1.0. All subsequent re-runs of the test scenario then included this port in the configuration for that virtual host. Prior to this configuration change, the passive discovery tool discovered 30 of the 33 of the ports found in an active Nmap scan on the 12 test devices. Tests were designed to evaluate the network presence of the virtual hosts. Test one verified that as a base case 100% of the virtual hosts were discoverable on the network. At a superficial level, they appeared to be legitimate devices. Test 2 provided a more in depth network probe designed to verify the OS representations. The scan correctly identified 70% of the devices. A more intensive OS scan in test 3 correctly identified 80% of the emulated OSs. So given both are superficial. The virtual hosts appear to resemble actual hosts, at a 70% or 80% accuracy rate. This shows the end result of an effective integration of the information gathering, communication, and host creation logic.

Tool	# OS ID	# of hosts	# of MACs	# of IPs
Ettercap	16	45	35	44
Ntop	0[a]	202	43	39
P0f	13	NA	NA	10
Tshark	NA	NA	69	44
Tcpdump	Not tested			
Snort	Not tested			

Scalability and Security Issues:

Scalability of the presented solution relies primarily on the capability of the hardware host. Honeyd is technically capable of emulating65 535hosts. Testing by the Honeyd authors knows that on a modest system thousands of different honeypots are possible [12].To validate this claim,a test with 986 virtual hosts was run on the test platform. The Honeyd OS signature database contains 986 entries. Each host configuration was created similar with a unique OS entry from the database and an IP address. The Nmap command in Test 1 was then executed targeting the 986 IPs. The top command was run on a 1-s interval to capture CPU and memory usage of the Honeyd daemon. At rest, prior to the Nmap tests, 8748 KBs of memory was consumed. 8860 KBs were used at the conclusion of the test. The average CPU utilization was 0.3% with a standard deviation of 1.23% and a maximum of 14.9%. This testing is not comprehensive, but does validate that, at a superficial level, a large number of virtual hosts can be created. Honeyd is single threaded and with more intensive probing, it is possible to maximize utilization of a single CPU. The test system has two CPUs and can continue to function even if this occurs.

The tested hardware host uses a long-term support (LTS) version of Ubuntu 12.04. This OS has a 5-year support cycle that includes security upgrades. As part of the hardware design, three physical Ethernet ports were specified. The ports are all assigned to a specific communication task to avoid a complete denial of service situation. For instance, if a large number of honeypots are active and consuming the entire bandwidth of a single port, then the system can still communicate on another port assigned to the management network. Updates to the host OS, communication of alerts, and IP monitoring/emulation lists are delivered on a separate management network. The second interface is configured as a passive read only interface on the operational network. This means it is not directly addressable from another host on the network. One security concern is a possible flaw in the monitoring software attached to the interface. The third interface is for use by the honeypot software to present its emulated hosts on the operational network. The most likely threat to the host is from this interface. This is a logical outcome considering that the honeypots are designed to attract the attention of those with nefarious intent. This threat is partially mitigated by the design of Honeyd. The software runs as a restricted user and, by default, does not provide any real services to compromise. For instance, on a high interaction honeypot, there are real shell services that might be compromised. Note that this does not rule out a denial of service or exploitation of a possible flaw in Honeyd itself. In addition to the Honeyd features, a host monitoring system such as OSSEC [35] can be utilized to provide self-monitoring. Finally, it is not required that Honeyd and the AB routines reside on the same machine. However, by

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 2, Issue 2, February 2015

condensing the software installs to one platform, it simplifies configuration. This also provides an opportunity to explore the recently expanding computational capabilities of low-power multi-CPU devices.

CPU and Memory Performance Measurements:
The DVH configuration logic, when implemented in Perl and run on the test machine previously described, took 0.7 s clock time to run and utilized 21 MBs RAM. The input Ettercap XML file contained 46 host entries and the resulting Honeyd configuration file included 12 devices. When running this configuration file, Honeyd consumed 5.7 MBs of RAM. During active scanning with Nmap, this would increase to 7.2 MBs. Ettercap was run continuously in daemon and logging mode on the test machine. It utilized 6 MBs of RAM and would utilize up to 60% CPU time when the Ethernet port, it was monitoring, was utilized to transfer data files.

IV. CONCLUSION

We proposed automatic deployment and configuration, a usage scenario was executed. In this scenario, an anomaly detection system monitored the network activity of the honeypots. The role of the automatically deployed honeypots was to attract and possibly delay an intruder on the network. The primary enabling technologies included continual host monitoring, reconfigurable deceptive virtual hosts, and a network AB monitor.

REFERENCES

[1]. Philip Auerswald, Lewis M. Branscomb, Susan Shirk,"critical infrastructure:control systems and the terrorist threat"vol14 no2, pp-120-130,march 2007[2] Yu- Lun Huang , Alvaro A. Cárdenasa, Saurabh Aminb, Zong- Syun Lin, Hsin- Yi Tsai, Shankar Sastrya" Understanding the physical and economic consequences of attacks on control systems"vol 11, no6,pp-110-119,april 2009

[2]. GordonRueff,BryceWheeler,ToddVollmer,TiMcJunkin." INL Control System Situational Awareness Technology Final Report 2013

[3]. IEE report 2012," utility scale smart meter deployments plans and proposals"

[4]. Peter Fanfara, Marek Dufala, Ján Radušovský," Autonomous Hybrid Honey pot as the Future of Distributed Computer Systems Security" Acta Polytechnica Hungarica, Vol. 10, No. 6, 2013

[5]. Dr. Ali M. Al-Khouri United Arab Emirate," e Government Strategies The Case of the United Arab Emirates (UAE)"security issue vol 2, no 8,pp-78-97,may 2014.