A Mobile Honeypot for Industrial Control Systems

Master-Thesis von Shreyas Srinivasa aus Bangalore, India
Tag der Einreichung:

1. Gutachten: Emmanouil Vasilomanolakis
2. Gutachten: Prof. Dr. Max Mühlhäuser

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik
Telekooperation
Prof. Dr. Max Mühlhäuser

A Mobile Honeypot for Industrial Control Systems

Vorgelegte Master-Thesis von Shreyas Srinivasa aus Bangalore, India

1. Gutachten: Emmanouil Vasilomanolakis
2. Gutachten: Prof. Dr. Max Mühlhäuser

Tag der Einreichung:

Abstract

The number of attacks towards the cyberspace is constantly increasing with their complexity. Over the last years Enterprise networks are targeted with hazardous malware to exploit their data and communication. Industrial Control Systems (ICS) are responsible for distribution of basic necessities like electricity, water and gas. Such systems are highly complex, critical and huge. ICS systems comprise of sensory devices, controllers, real time units, command servers connected with an internal network. For better control and management, these devices are connected to the Internet.Failure of such critical systems lead to devastated environments and hence there is a need to secure them.

ICS systems are further classified to Supervisory Control and Data Acquisition (SCADA) systems. SCADA forms a standard for majority of ICS systems deployed worldwide. These systems have been targeted previously with malware such as Stuxnet. Stuxnet was capable of controlling the Programmable Logic Controllers (PLC) and infecting them with malicious code to change their behavior. There is a need to detect the propagation of such malware. ICS systems are hard to protect using passive systems like Intrusion Detection System (IDS) as they rely on a signature database. Also, strategized off-beat attacks cannot be detected using IDS. *Zhung et al.* showed that although the detection accuracy of the IDS is high the number of false positives are also high as these systems are sensitive to noise[67]. An active mechanism that provides alert mechanisms on malicious packets and features an analysis framework with lesser false positives is necessary.

The advent of better smartphones has changed the way people interact and also leverage its resources to constantly provide better productivity to users. We propose HosTaGe ICS, a mobile ICS honeypot which actively detects attacks with respect to ICS SCADA systems. HosTaGe, an Android based mobile honeypot is extended to HosTaGe ICS, to accommodate ICS profiles like a nuclear power plant.

The protocols and the target systems that are specific to ICS SCADA systems like Modbus and S7 are implemented to simulate the behavior of end systems. The behavior and response mechanisms are also implemented to facilitate the attacker to pursue his attacks. Malware detection is a crucial challenge that involves studying the behavior and its propagation techniques. We create the environment for the propagation of Stuxnet, a popular malware that can bring down ICS systems and detect its propagation using HosTaGe ICS. The results are further verified by the Virustotal [54] database.

The attack strategies used by attackers nowadays involve distributed, single level protocol, multi level protocol/multistage and payload level attacks. Multistage attacks refer to attacks that originate from the same source and attempt to exploit different types of protocols within a small window of time. It is one of the prevailing and effective strategies used by attackers these days to perform social engineering and compromise the end systems. During the course of the thesis, multistage attack patterns were observed. The detection of such attacks was incorporated into the design of our honeypot. HosTaGe ICS features the detection mechanism for multistage attacks. Interesting results obtained during the evaluation period are briefly summarized in the Evaluation section.

HosTaGe ICS incorporates a signature generator module where signatures are generated based on the attacks recorded with Modbus, Siemens S7 protocol (S7) and HTTP protocols. The signatures that are generated by Hostage ICS can be incorporated into the Bro Network Intrusion Detection System (NIDS) to check incoming packets for malicious payloads. This feature enables HosTaGe ICS to be deployed inline with Bro NIDS to provide better attack detection capabilities.

Contents

1 Introduction

Power, water and gas form basic necessities for human life today. ICS are used to automate the generation and distribution of these vital necessities. These are control systems used in industrial production that include SCADA[5] and DCS[6] systems. These systems manage critical infrastructure and are deployed in industrial sectors. SCADA systems are used to control distributed assets using centralized data acquisition and supervisory control. DCS manage active production systems within a local area such as a factory using supervisory and regulatory control. PLCs are devices used for discrete control of specific applications and generally provide regulatory control for the critical machinery[62]. PLCs automate the control and management of critical systems in an ICS environment.

SCADA systems are distributed systems used to manage and control geographically dispersed assets, often scattered over large distances. The data acquisition and control are centralized in these environments and are critically managed. PLCs are major components in such operations. They are deployed to regulate the sensor information and provide an interface to convert signals from human-to-machine and vice versa. PLCs are continuously being evolved in order to have better performance and simplified communication. The protocol responsible for communication between these devices is Modbus[36]. PLCs today are able to communicate to the Internet through the advancements in the field of PLC and Modbus-TCP[50]. This makes the devices being controlled through the Internet. SCADA systems were mainly designed for interaction between devices in a distributed network without considering security. This made the systems and the communication vulnerable.

PLCs handle critical systems, but have lesser computational resources as compared to standard desktop hosts. As modern day PLCs have Internet communication capabilities, they are open for attacks from adversaries. This is due to the lack of security measures employed in securing both the device and the communication. As these devices have limited resources, ensuring security mechanisms may prove expensive. The manufacturers focused on providing better compatibility for communication rather than securing it. This vulnerability was exploited early by the attackers. In 2009, Stuxnet[28] a malware infected a nuclear enrichment facility in Iran. Stuxnet is still considered to be the most complex engineered malware until today for its stealthy and propagation techniques[68]. It was considered to be powerful enough to cause a devastative impact. There are also recently discovered worms like Flame [59], that are observed to have similar impact. This opens up new challenges in securing SCADA networks.

Many enterprises and infrastructure use PLCs today to automate processes. It is extensively used in airports, packaging industries, fuel stations, frozen food storage units and even in super markets. Securing the networks that use such devices is necessary. IDS form front line defense systems against external attacks. They are widely deployed in enterprise networks to safeguard their internal networks. NIDS[45] are passive devices that are used to monitor networks or systems in enterprise infrastructure. The detection mechanism is either anomaly or signature based. Signature[65] based IDS monitors the packets and compare them with a database of signatures. Anomaly[18] based IDS detect intrusions by monitoring the system activity and classifying it as normal or anomalous.

The signature based approach can be easily tricked by introducing varied payloads to defy the signatures checked by the IDS. A well strategized and tailored attack can easily pass through the system to gain access into the network [24]. A more active approach towards detecting strategized attacks with lesser false positives is the use of honeypots[42]. Honeypots are decoys that pose as target systems for the external viewer. It simulates specific services and reply mechanisms to emulate the complete system behavior. Any traffic directed towards honeypots can be considered as an attack. The number of false positives are comparatively lower in honeypots than in NIDS. They can also work as an additional line of defense along with IDS in traditional as well as in ICS networks [64].

A study made by DELL[9] showed that the attacks on Industrial components like PLCs doubled over the years, and even more dangerous is the fact that such incidents are unreported. The research found a hundred percent increase in attacks against ICS environments like SCADA.

Figure 1 gives an understanding of the key SCADA attack methods. It shows that about half of the total attacks were based on improper assignment on bounds of a memory buffer, improper input validation and vulnerabilities in credentials management. These vulnerabilities pose as a huge threat to ICS. Figure 2 represents the number of attacks performed over the months. There is a steep increase in the number of attacks performed over the months, expressing the need to safeguard ICS systems and also detect these attacks.

The majority of industrial systems today use SCADA for controlling and automating their processes. Securing these devices are as much important like any other hosts in the network because these devices are programmable and could affect the normal automatized working. An overview of the above survey indicates various vulnerabilities and the need to secure the ICS environment. As PLCs are not only used in industrial infrastructure, but also in smaller environments

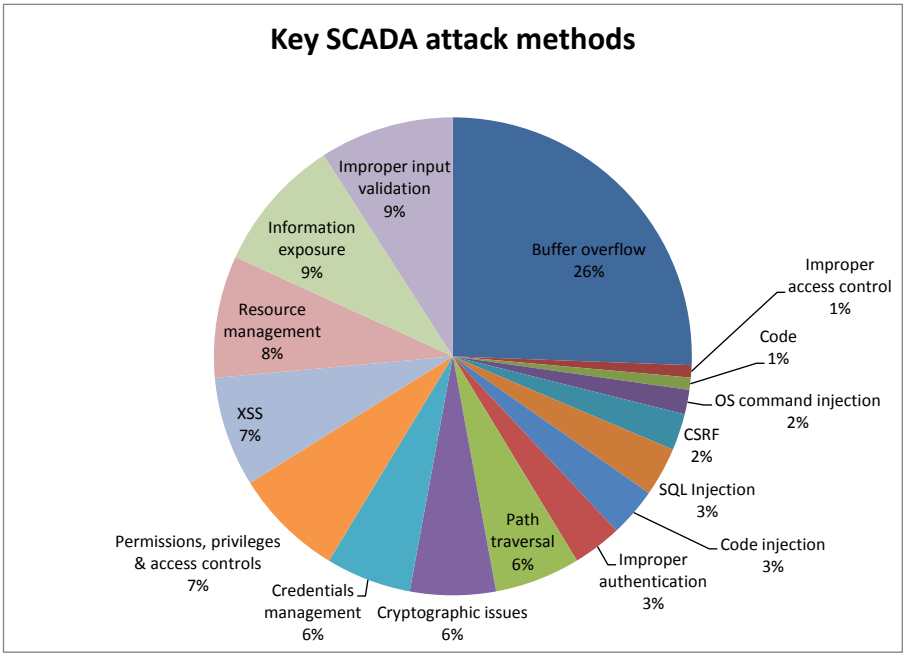

Figure 1: SCADA attack methods[9]

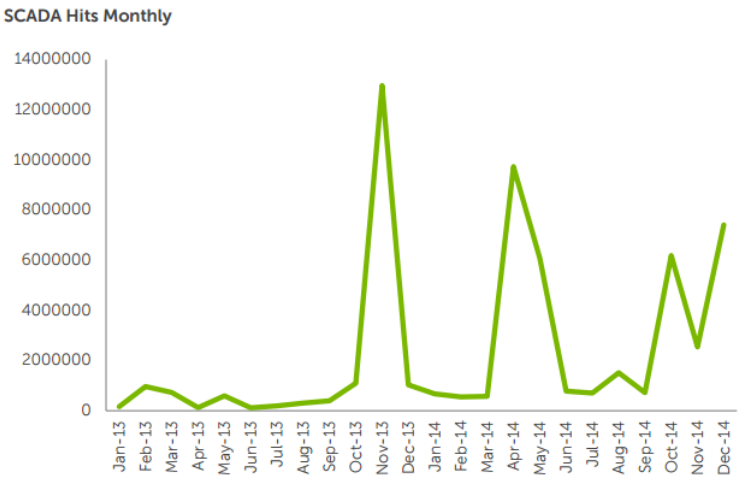

Figure 2: SCADA hits on a monthly basis.[9]

where automation is required. It is very important and crucial to have less complex, usable and robust monitoring system. Such systems can be used to identify malicious packets and provide a flexible and analytical framework.

Smart phones provide enormous computing capability with high flexibility. They form a platform for developing light weight and reliable applications. These devices commute with the user and are highly available. They are able to connect to wide area networks and offer various modes of operational connectivity. We propose a lightweight, low interaction mobile based honeypot HosTaGe ICS which leverages the computing power and the above features of modern day smart phones to detect ICS specific attacks.

1.1 Contribution

This thesis aims at identifying and detecting the ICS SCADA specific attacks using HosTaGe ICS, a low interaction mobile honeypot platform. The master and slave profiles, which are a part of ICS infrastructure are simulated by emulating the constituent protocols. An analysis of the communication paradigm and the security loopholes of an ICS SCADA system is made, to simulate the services offered by the system.

The thesis also concentrates on contributing to many security related research questions of ICS SCADA systems like identifying ICS specific malware , its propagation and devising a method to detect malware through our honeypot. A detailed study about the malware and its propagation is made to model the detection capability into HosTaGe ICS.

During the course of the thesis, a pattern of attacks was observed and inferred to be Multistage attacks. We formally model the multistage attack strategy used by attackers to compromise the targets. Through an effective model, we further implement the detection of such multistage attacks.

Honeypots today are used inline with IDSs for increased detection accuracy. We also contribute towards this paradigm by a signature generation module that generates signatures for attacks received on specific protocols. The attacks identified through our multistage attack detection module are also leveraged to create policies for the Bro Intrusion Detection Monitor. The signatures and policies generated from HosTaGe ICS can be deployed on the Bro NIDS for effective monitoring of malicious packets.

Search engines are an effective tool for looking up resources in the Internet. Shodan a search engine provides its users with information regarding vulnerable devices in the Internet. Shodan probes the Internet to find vulnerable devices. The effective probing strategy of Shodan fetches as much information required to determine the vulnerabilities of a target system. Our honeypot setup received these probes during the evaluation period. We further investigate the impact of these probes on our honeypot and make interesting observations.

1.2 Outline

The thesis mainly focuses on detecting attacks with respect to ICS environment. The outline of the thesis is as follows. Section 2 provides a pre-study of ICS SCADA systems, security concerns with respect to ICS SCADA systems, the Modbus protocol and honeypots.

A study about the Siemens Simatic PLC family is presented in section 2.4. Section 2.6 provides an overview of the related work in the field of honeypots and also SCADA specific honeypots. Multistage attacks are briefly discussed in section 2.6.5 and section 2.6.6 giving an introduction to Signature Generation module of HosTaGe ICS.

Section 3 proposes a solution with a system design involving discussion of HosTaGe as an ICS perspective in section 3.2. The protocols, formal models and detection mechanisms are briefed in their corresponding sections.

We present the implementation ideas of HosTaGe ICS in section 4 as a honeypot with discussion related to detecting internal attacks, malware detection, multistage attacks in the forthcoming sections. The implementation of the signature generation module is elaborated in section 4.5.

The evaluation of the thesis is discussed in section 5. It focuses on evaluation of a single protocol attack and detection of Multistage signatures. The impact of Shodan probes and the observation is open in section 5.4. Section 5.5 provides an overview of the performance of HosTaGe ICS as an Android application. The limitations of our work are discussed in section 5.6.

Section 6 aggregates the work done on HosTaGe and the future enhancements are specified in section 6.1.

2 Background and Related Work

The following sections provide a background on ICS SCADA, its security perspectives, the Modbus and S7 protocols, Siemens Simatic S7 PLC and honeypots. Section 2.6 discusses the related work in the area of honeypots, further classified to mobile honeypots and SCADA honeypots.

2.1 Background

ICS form a dominant portion in present day industries. ICS components include actuators, sensors, networking devices, controlling systems and PLCs . The sensors form a major portion of ICS as they provide continuous feed of critical information which is used to automate and control other systems. The PLC is another such important component for the ICS . This interface allows a programmer to implement a logic to automate the systems based on the data received from sensors. There are mainly two different types of ICS. The major type is SCADA, which are deployed on geographically widespread and are controlled using a central location. Examples include nuclear power plants, water and power distribution where there is a need for constant monitoring and critical automation. The second type of ICS is the Distributed Control Systems (DCS). On the contrary, DCS are not centralized, but distributed across a network. We focus more on ICS SCADA systems as they are being deployed in major infrastructures as a standard today.

The SCADA infrastructure have a lot of components and devices which need constant communication between them. The ICS SCADA systems involve the interaction of many sub-modules and devices that will be discussed below in the following sections.

2.1.1 ICS SCADA

SCADA is an industrial automation control system at the core of several industries today including energy, oil and gas, power, water and recycling, manufacturing and many more. SCADA provides the benefit of, simple configuration and usability.

The basic architecture of SCADA involves communication of information from sensors or manual inputs to PLCs or Real Time Unit (RTU)s. PLCs process the information as per the logic deployed in them and then forward this information to workstations/servers running SCADA applications. Figure 3 describes the basic architecture of a SCADA system which consists of sensors, PLCs, master hosts and the controller hosts. Data is fetched from the sensors and is processed on PLCs. The Human Machine Interface (HMI) displays this processed data to the user.

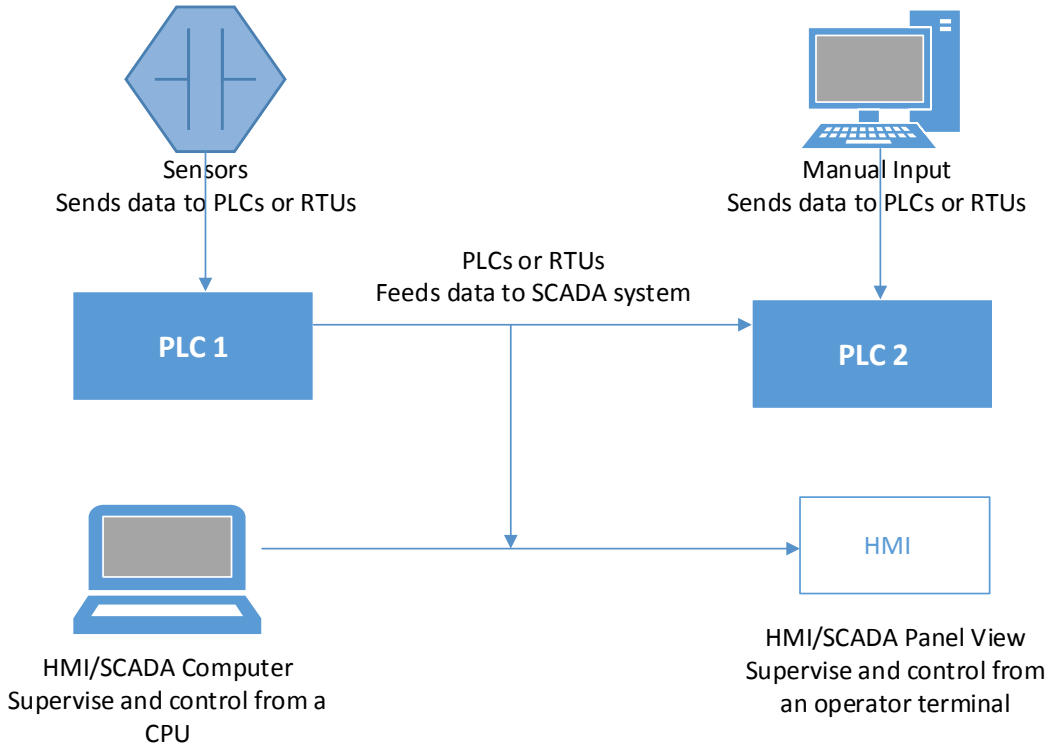

Figure 3: SCADA Architecture

SCADA systems involve control and network components. The following are the list of control components in SCADA systems:

- **RTU:**These devices are time constraint hardware that fetch data from sensors and convert it to digital data. The converted data is further sent to the master system or the supervisory system. RTUs receive commands from the supervisory systems. The devices possess arithmetic processors to perform logical operations on the data received from sensors.

- **PLC:** These devices fetch data from sensors that are embed to heavy machinery such as spinning centrifuges. Further, they convert the analog data received to digital data. PLCs possess higher processing power than RTUs to perform complicated operations on the data received. The controllers can embed logic into PLCs for a specific operation. PLCs are preferred over RTUs in certain critical environment due to their flexibility and customization features. PLCs can be made compatible for interaction with devices from different manufacturers. These devices also posses expansion slots for scalability.

- **Telemetry systems:** They are used to connect PLCs and RTUs with supervisory system centers, and controllers in enterprise networks. In a typical scenario, they include leased lines, LAN cables with switches and Wide Area Network (WAN) circuits lines. Some networks also have wireless transmission capabilities. Such networks have wireless telemetry via satellite uplink and broadband radio of microwaves.

- **Data and Control Server:** Data acquisition systems are software that are used to achieve connectivity to the device interfaces through telemetry lines with the deployed PLCs and RTUs. The software enable users to access remote device data from field devices through SCADA protocols.

- **HMI:** The HMI is responsible for presenting the processed data in human readable form. Users interact with the field devices and controllers through HMI to issue commands and control them. The HMI acts a client interface to receive data from remote data acquisition servers.

- **Historian software:** This software stores and represents the data graphically for statistical evaluation. It contains a database that stores data which is time stamped. The data can also be stored as a part of event handling. This software is also capable of retrieving data from the data acquisition servers.

Different network characteristics exist for every layer within the control systems. The network topologies vary with vendors or manufacturers and also on different implementations. Modern day SCADA systems are open to the Internet and as a result, enterprise integration can also be achieved. The control networks work in hand with the corporate enterprise networks to manage more efficiently and control the systems from outside networks. The following are the major network components of an ICS network:

- **Fieldbus Network:** This is used to achieve connectivity between PLCs and other controllers in the network. The need for point to point cabling is avoided between the controller and every slave using the fieldbus network. Fieldbus supports various communication protocols. Every device has a unique identifier for the messages sent. This way there is no collision of messages during processing.

- **Control Network:** The controller networks connect the supervisory systems and master systems to the slave devices.

- **Communications Routers:** The basic functionality of a router is to transmit messages between networks. Routers in SCADA systems are used to connect Local Area Network (LAN) to a WAN and also connecting huge enterprise networks to bridge long-distance networks. They also connect field devices to outer networks for command and control.

SCADA applications help in monitoring and analyzing the data. The application help the device controllers and operators to work efficiently. Modern SCADA systems allow real time data from the plants to be accessed from anywhere in the world. This also means that it provides attackers an opportunity to exploit the data and its availability. Exploiting SCADA systems can cause catastrophe as it may result in huge damage to the environment and people in the plant. We try to identify the attacks and exploits and detect them using our mobile honeypot.

2.1.2 Security Overview of SCADA ICS

ICS SCADA systems are highly distributed. They are used to control and manage geographically dispersed plants, often scattered over thousands of kilometers. In these areas, centralized data acquisition and control are critical to system operation. They are applicable in circulation systems such as water distribution and wastewater collection systems, oil and natural gas pipelines and electrical power grids on systems. A SCADA control center provides centralized monitoring and control for long distance separated sites with monitoring systems. The systems also process the data acquired from these sites. The data received from remote sites that are driven by automation or by an operator, supervisory commands can be pushed to remote station control devices, which are often referred to as field devices. These devices manage local operations such as switching control units and pumps, retrieving data from sensors and local environment monitoring.

The devices at the field sites are controlled and managed through the control center. This requires a secure and robust communication line between the control center and the field site. This network is critical. This is usually established through the Modbus TCP/IP over the Ethernet. It is usually advised [49] to place the SCADA devices on a network that is not physically connected to any other networks.

Bailey et al. [10] proposed honeypots as a solution to SCADA security monitoring. SCADA systems today have evolved from complex infrastructure to simple and well packaged network architecture. This makes SCADA systems usable for critical systems as it is simple to manage. Further, the threat landscape of SCADA is huge due to the various critical environments under which it is deployed. Below are some of the underlying threat areas that SCADA systems are subjected to today.

- **Supply Chain:** A product is engineered and manufactured by a group of vendors in different manufacturing units. The final product is aggregated from all the units later for assembly before it is provided for the requester. This makes some room for unintentional error and also induced error like, the insertion of malicious functionality with respect to a specific customer, usage of low quality, less secure electronics and finally, poor engineering practices. The errors may also occur at the final assembly parts where these parts are not assembled or tampered for specific induced intentions. The requesters have no idea of these errors until the casualties occur as they do not have quality check units to investigate the products once they are received or deployed.

- **Vendor Backdoors:** These are the most serious threats that are hard to detect. Vendors install backdoors[1] on products to gain access to control end systems. Backdoors sometimes also have covert data leaking channels to remote command and control servers. These backdoors are not documented in any catalogs provided to the customers. Securing from such attacks is very difficult as they are not accounted for. In such cases honeypot acts as a good solution as it is capable of storing all the connection activities which make later investigations possible.

- **Protocol Vulnerabilities:** The protocols that are associated with SCADA like Modbus and S7 communication were not designed keeping security as a requirement. They do not support authentication, encryption and non-repudiation as they were proved to be expensive operations on the data communicated. In such environments, unauthenticated hosts can form legitimate packets and establish connection with any devices. Many of the protocol interaction takes place in plain text thereby making the communication completely visible. Unauthenticated systems running tools like sniffers and monitors can misuse this for knowing and probing the environment.

- **Malware Propagation Techniques:** SCADA systems form the backbone of ICS. The initial infrastructure of the intranet involves different components and devices as discussed in section 2.1.1. Through continuous probing and sniffing techniques, adversaries learn more about the infrastructure components and also the interaction between systems. This is a useful technique to design strategies for malware propagation in SCADA networks. It could also be called Social Engineering [53].

  The other form of designing strategies would be the use of attack trees to assess and determine the vulnerabilities and flaws of a target system. This was initially proposed by *Bruce Schneier* [46]. Attack trees provide a structured and flexible means of pursuing analysis on protocols, applications and networks. They form a strong representation of the attack strategies designed for compromising systems in a SCADA network communication using the Modbus protocol. The research by *Eric J. Byres et al.* [7]provides a deeper insight of attack trees for Modbus protocol. We discuss the vulnerabilities of Modbus protocol in further sections.

- **Botnets and Search Engines:** The Modbus-TCP protocol enabled ethernet communication capability to SCADA systems. As a result SCADA systems are able to connect to the Internet. Google is the most widely used and popular search engine. Through crafted Google queries, critical information can be retrieved .This information might be some accidental information revealed by employees about the company network infrastructure or critical information. This technique is referred to as Google Dorks. *Trend Micro*, a security software company provides an overview of how Google Dorks revealed critical information of ICS systems[63].

  Search engines like Shodan which is acclaimed to be the scariest search engine [19]on the Internet expose vulnerable devices that are connected to the Internet. Shodan is specialised in indexing the headers in contrary to the other search engines that match the content of the page. Shodan database is like a goldmine for attackers which provides enough information like the services, protocol information, Internet Protocol (IP) subnets and also location of the systems. *Roland et al.* [3] provide an evaluation of Shodan's ability to identify Internet-facing ICS devices.

  Bots constantly probe the Internet for vulnerable devices. The information retrieved by bots are sold in the black market. These prove to be very effective bots with fewer false positives. Botnets [21], a networked collection of bots can be rented specifically to perform such activities. Botnets also act as strong probing tools to identify vulnerable SCADA devices connected to the Internet.

- **Espionage and Cyber Exploitation:** Amidst adversaries, there are also espionage planned by governments to keep track of developments on developed countries. This motivation vary from simple curiosity to national level attacks. An example would be of the Stuxnet worm which was discovered to be developed by America and Israel together and was deployed in a nuclear enrichment facility in Iran. The actual motive is unclear but by the aspects it may look a cyber espionage designed by one of the nations. It is very important to note that malware like Stuxnet remained undetected over a long duration and was capable enough to create a environment hazard. Hence, these kind of threats cannot be ignored and rather must be focused in order to ensure better security.

In addition *Igure et al.*[22] specify three challenges that must be focused to secure SCADA networks. The first challenge is to improve the access controls to the SCADA networks. An access control solution will ensure restricted access for an attacker to enter into the SCADA network. The second challenge is to improvise the security in the SCADA intranet and to develop efficient security-monitoring tools. The third challenge involves improvising the overall management of the devices in the SCADA network by identifying loopholes and vulnerabilities. These security mechanisms ensure better safety in the SCADA network. The monitoring tools like IDS identify malicious packets and also monitor the network for suspicious activity.

The above specified threat areas provide an insight into the security perspective of SCADA systems. Further as specified, we discuss the Modbus protocol which forms the backbone of SCADA systems and its security aspects.

2.2 Modbus Protocol

Modbus denoted by Internet Engineering Task Force (IETF) Request For Comments (RFC) 2026 is a serial communication protocol. It was first published by Modicon for communication in its PLCs. Modbus is now a standard that connects industrial devices together. The basic configuration involves connecting a SCADA supervisory control system to a PLC or RTU. The data types are derived from its use in production devices where a single-bit physical output is called a coil, and a single-bit input is discrete input or a contact. The device requesting the information is called the Modbus Master and the devices rendering the data are Modbus Slaves. In a standard Modbus network, there is one Master and up to 247 Slaves, where each Slave device has a unique address from 1 to 247. The Master can also write data to the Slaves. Modbus TCP/IP specification was introduced to Modbus to integrate corporate intranet with PLC systems. This made the network better manageable, scalable and also cost-effective. Modbus TCP/IP offers many advantages:

- **Simplicity:** The Transmission Control Protocol (TCP) is wrapped with the Modbus instruction set. The setup involves simple driver initialization at end devices for communication. Low development cost, hardware and compatibility with different Operating System (OS)s makes it simple.

- **Standard Ethernet:** Ethernet ingrates easily into simple chipsets and boards. The cost of implementing Ethernet to Modbus is low and also provides ample resources as there are many developers working on optimizing the technology. Ethernet port 502 is used by the Modbus TCP/IP protocol.

- **Open:** The Modbus protocol has been open source since 2004 and has a dedicated organization working towards development,optimization and maintenance.

- **Compatibility:** Modbus provides interoperability among various vendors and also compatibility with devices of other manufacturers.

Modbus TCP/IP is an Internet protocol. This makes the devices open to the Internet. This was a particular feature that was incorporated to facilitate better control and making device maintenance through remote systems over the Internet. Modbus is an industrial networks protocol. Modbus TCP/IP helps in better management of distributed ICS. It is also necessary to understand the security aspects of Modbus as SCADA systems rely on it for communication between field devices. The underlying security issues of Modbus are discussed below:

- **Lack of Confidentiality:** The Modbus traffic is transmitted in plain text and does not involve any encryption. This makes man-in-the-middle attacks easily possible. The traffic can be viewed using simple sniffing and traffic monitoring tools.
- **Lack of Integrity:** There are no integrity checks build into the Modbus application and as a result it relies on lower level layers to preserve integrity.
- **Lack of Authentication:** There is no authentication at any level of the Modbus protocol, with the exception of some commands that explicitly introduce an authentication mechanism. It is observed from vulnerability analysis that this authentication mechanism is hard-coded and can be easily retrieved by simple packet forging techniques.
- **Simplistic Framing:** The Modbus-TCP frames are carried over established TCP connections. Although TCP is a reliable protocol for data transmission, the drawback of not preserving record boundaries for the Modbus protocol, makes it easy to inject arbitrary frames. This provides an opportunity for man in the middle attacks.
- **Lack of Session Structure:** Modbus-TCP has short span sessions like SNMP and HTTP protocols. The master host initiates a request to the slaves which results in a single action. The lack of authentication and poor TCP Initial Sequence Number (ISN) generation in embedded system frameworks, enables attackers to inject packets without caring for existing sessions.

2.3 S7 Protocol

The SCADA systems were developed basically for simplified serial communication considering that the devices operating on it are legitimate and are programmed to perform the logic specified. Thus, there is no actual security measure to enforce secure communication. There is no authentication check to verify if the data received is from a valid source. Moreover, all the data is in plaintext making it possible for any machine in the network to read all the intercepted data.

The Siemens S7 protocol is widely deployed in SCADA networks in the Siemens S7 PLCs, which are the leading controllers used in ICS environments. The S7 protocol structure is complex when compared to the Modbus protocol as it allows parallel access to multiple variables of different data datatypes. Related information regarding the protocol structure of Siemens S7 protocol is hard to find as it is a proprietary protocol designed by Siemens for communication between its PLCs. However, we try to fetch as much information regarding this protocol by the traffic traces found on the Internet [1] and some related information.[25]

The Siemens Simatic S7 product line was introduced in 1995 for models S7-200, s7-300 and S7-400 and later for models S7-1200, S7-1500. The Step 7 software is responsible for the HMI of these PLCs. TCP/IP based connectivity for the protocol is provided by the Siemens Ethernet Driver. However, connectivity can also be achieved through third party drivers. The protocol communicates over TCP port 102 and interacts over the Connection Oriented Transport Protocol (COTP) and TPKT[40]. These protocols add their own headers in the TCP segment and thereby encapsulate the S7 packet within the COTP packet.

S7 provides a common communication interface where the device can choose to be clients, master or peers. There are two implemented versions of the S7 protocol. Simatic S7 PLCs implement the S7 implmentation and the later devices implement the *0x32* variant. A newer variant *0x72* introduces some security features.

The maximum length of a S7 Protocol Data Unit (PDU) lies between 112 to 960 bytes. The packet is divided into three parts. The first part is a fixed header which includes the COTP, TPKT and TCP header. The second part is the parameter that indicates the PLCs variables that are accessed and lastly the data part which includes the data to be written.

[1] https://wiki.wireshark.org/S7comm

The S7 protocol does not provide security features similar to Modbus. The protocol was designed to establish communication between Siemens Simatic PLCs. The protocol is also not available as open source for a detailed study. The packets sent through ISO TSAP [44] are in plaintext. These packets can be easily forged by capturing them and modifying it to exploit the PLCs. Thus the attacker can intercept any packet and modify it to issue commands like powering off the CPU and changing the ladder logic of the PLC. This also enables an attacker to secretly place a code which acts as a backdoor that helps is gaining remote access to the PLC directly. A man-in-the-middle attack is also possible by intercepting the packets that flow from the software to the PLC. *Dillon Beresford* in his paper [2] shows various ways a Siemens Simatic S7 PLC communicating through a S7 protocol can be compromised. In his report, he elaborates the vulnerabilities of Siemens S7 PLCs and simplistic tools to exploit them. The five attacks portrayed are:

1. The TCP Replay attack over the ISO TSAP
2. Bypassing the basic authentication of S7 protocol
3. Powering the CPU on and off of the PLCs.
4. Read and write of the memory registers.
5. Gaining shell access on the PLC

The exploit is performed using penetration testing tools like Metasploit [35] and Nmap [33] that help in providing pre-defined modules for the attack. Listing 5 shows the Modbus discovery script for the detection of the Modbus service status on the target system. HosTaGe ICS received many attacks which were based on this script. Listing 6 shows the Metasploit script for detecting the Modbus service on the target system. Listing 7 shows the Metasploit script that can be used to fetch the values stored in Modbus registers. The script can also be used to push values into the Modbus registers and coils. We also leverage the capability of these tools to study the attack and probing strategies to model out attack detection vectors.

2.4 Siemens Simatic Series

The Siemens S7 200 is a micro-programmable logic controller which can control a wide variety of devices to support various automation needs. The S7-200 features monitoring, inputting values and outputs as validated by the logical program, which can include Boolean logic, counting, process timing, complex mathematical operations, and communications with other intelligent devices. It can control and communicate with devices like automatic pressure controllers, centrifuge pumps and water cooling systems. The STEP 7 programming package provides a user-friendly application programming interface to develop, edit, and monitor the logic needed to control the application that monitor devices. The Siemens Simatic S7 PLC's use PROFINET which is based on Ethernet for communication. There are over 3 million PROFINET devices deployed worldwide.

Siemens S7 200 PLCs boasts of a compact design, powerful performance, optimum modularity and open communications. This Micro PLC has been in successful use in millions of applications around the world,in both standalone and networked solutions.

This PLC uses communication protocols such as PROFINET, an advanced version of Modbus communication protocol. This protocol is also based on Ethernet. It also supports TELNET, Hyper Text Transfer Protocol (HTTP), File Transfer Protocol (FTP), Simple Network Management Protocol (SNMP), Simple Mail Transfer Protocol (SMTP), Modbus and S7 protocols. Though this PLC is designed to be used to control critical systems, security was not a part of its design. The above mentioned protocols were not customized to facilitate secure communication. The standards were defined to create an interconnected environment between industrial automation devices and common networking protocols.Security was either ignored or rather was thought to be expensive on these devices. This makes it an easier target for attackers.

The Simatic S7 PLC is also subjected to various vulnerabilities and attacks including the Stuxnet as discussed earlier. We simulate the Siemens Simatic S7 200 PLC as a target system in our honeypot.

2.5 Honeypots

A honeypot is a decoy server or a system in a network which is closely monitored for adversaries. It is also defined as: *A honeypot [48] is an information system resource whose value lies in unauthorized or illicit use of that resource*. They are mostly deployed inside firewalls, but they could be deployed in any part of the network. It is designed to be a system with vulnerabilities and services that are offered by a real target system. Any attempt to connect to these systems could be considered as an attack. All the activities are logged and further traced. The general idea is that once an adversary

detects a vulnerable system and tries to attack it, he would come back with more sophisticated attacks. The initial part of discovery and knowing the general services and loopholes is called system social engineering. Honeypots provide active monitoring components that wait for attacks and respond to the attacks by luring the attacker to pursue more. There are certain main functionalities that the honeypots must possess in order to perform their main functionality.

1. Honeypots must simulate the system that they are intend to focus on. This gives the attacker a feeling of approaching a real system. The honeypot may simulate the complete functionality of the system or just the services offered by the system.
2. A proper response mechanism which keeps the attacker engaged to the honeypot. This facilitates better logging of the attack and also provides more data to analyze the attacks.
3. It mainly has three perspectives. First, an attacker perspective, by posing as a vulnerable system; second, an administrator who can identify and log the attacks made by the attacker and third, being able to present and analyze the attacks logged by the administrator.
4. Honeypots must not induce additional load on the infrastructure. Honeypots must be designed to be lightweight and having no influence on the production systems.
5. Honeypots are basically exposed to exploits, threats and malware. It is very important to see that this data is not leaked through the network which can later infect other systems in the network.
6. Based on the previous condition, honeypots must be robust to withstand the attacks and exploits and not fail.

It is clear that honeypots are valued because of the interaction mechanism that they provide for any communication request. They can be used to study and gather exploits, malware and threats in an early attack phase. There are many advantages that one could consider for using honeypots as an additional security monitor. There are various advantages of honeypots:

- **Effective Data Sets:** Honeypots collect data only when there is a communication requested with it. The data collected at honeypots may not be immense but is good enough to analyze and detect attacks. The logs provide information about the attacker IP, time of attack and protocol used to carry out the attack. This ensures lesser false positives.
- **Reduced False Positives:** Among other security approaches like IDS and firewalls, false positives are quite common. The biggest challenge is to reduce false positives. Honeypots could be designed to reduce false positives. Any communication with the honeypot is unauthorized. This makes honeypots efficient in detecting attacks.
- **Catching False Negatives:** Honeypots have advantages over signature based detection systems. Signature based systems do not categorize unknown attacks. They rely on a signature system to be updated on their local database to identify and detect unknown attacks. The probability of a detecting a new exploit is low. Honeypots detect all attacks irrespective of their signatures, thereby increasing the possibility of detecting new attacks.
- **Encrypted Communication:** The current standards in Transport layer includes using encrypted Transport Layer Security (TLS) communication between nodes. Some attacks fail to detect because of the encrypted data and communication. All enterprise networks employ secure protocols like Secure Shell (SSH),Internet Protocol layer security (IPSec), Secure Hypertext Transfer Protocol (HTTPS), TLS in their infrastructure. This may cause problems in detecting exploits and analyzing the attacks later. Honeypots solve this issue as they are end points in the communication. The hosts directly interact with the node and hence all the traffic and data can be decrypted and analyzed later.
- **Compatibility to new architecture:** Technology evolves every moment. It is very essential to consider future compatibility with newer standards and technology. Most of modern day IDS or firewalls are not compatible with Internet Protocol Version 6 (IPv6) which promises to be the next standard on Internet addressing. Honeypots can be made compatible to newer standards and technology as they are not mediators or devices but act as end points. However, devices could be simulated by honeypots.
- **Flexibility:** Honeypots can be deployed locally or open to the external network. Honeypots could be deployed on any environments based on the requirements such as specific servers, host systems or protocol level emulation. It could be used to simulate any software, hardware, servers, workstations and devices.

- **Minimal Resource Consumption:** Honeypots can run on low resource machines as they are just simulations and are may not depict full functionality of the system simulated. Honeypots today can run on smartphones as they possess the required resources which are good enough to run a honeypot.

2.6 Related Work

There has been extensive research going on in the field of honeypots and honeypot-based Signature Generation. We also discuss multistage attacks, an attack strategy employed by attackers towards attacking multiple protocols. Multistage attacks are effective in obtaining more vulnerabilities of the system at the protocol level. Detecting these attacks is crucial as it forms a basis for analyzing strategic attacks. In the following sections we discuss related work in these areas.

2.6.1 Types of honeypots

Honeypots can be classified into two types based on the attacker ability to interact with the application or services. They can be categorized to *high-interaction honeypots* and *low-interaction honeypots*. This classification is mainly based on the honeypot's interaction with the attackers. High interaction honeypots typically composed of the actual device, its operating system and all the applications that run on that device. In short, the exact machine is used as a honeypot with all its services. This provides better interaction as we are using the device itself as a honeypot. There are also better chances that based on the vulnerability known, all the exploits work on the device. The main advantage of such honeypots is that it is the machine itself being exposed and has greater chances of attracting attackers. The disadvantage would be that if the honeypot is completely compromised, then it has to be rebuilt in order to log other attacks. The validity of such honeypots is not guaranteed.

A low interaction honeypot on the other hand is a software based or simulation based honeypot approach. The system to be subjected to an attack is simulated by the honeypot along with its main services. The honeypot can run on any system, for example it can run on a Linux machine and simulate a honeypot for a Windows Internet Information Server (IIS) server. It can simulate or mimic the network stack and the operating system of the target host. All connections and communication with this device is logged. The advantage of low interaction honeypots is that they are completely flexible and easy to maintain. Low interaction honeypots are also likely not to get compromised as they just mimic the services or in short the basic communication mechanism. It is on the researcher to design these honeypots accurately to get productive results.

2.6.2 Honeynets

Honeynets [20] are a networked collection of honeypots that look like common network services and servers. They provide an overview of the simulation of server and network components. This overview provides better modeling of the honeypot and the services to be simulated. The simulation could be a collection of honeypots depicting as a Domain Controller, web server, application server, file server and so on which provide a facade of a enterprise network. Honeynets usually consist of high -interaction honeypots, low - interaction honeypots, or a combination of both. Using high interaction honeypots only for this approach would be more expensive. Honeynets are placed behind a Honeywall , which acts as a bridge to the honeynet. It includes network monitoring, packet capture, and IDS capabilities. Honeynets provide an insight into design of low interaction honeypots along with other devices in the network like NIDS.

Honeynets require complicated setup and resources to deploy them to the network. It also involves complex configurations as it pertains to simulation of many target hosts through honeypots and a separate logging mechanism. However, Honeynets provide an idea of simulating the infrastructure and also providing a framework for further analysis. We look forward to implement a similar framework for the analysis of malicious packets in our honeypot.

2.6.3 Mobile honeypots

Smart phones have huge computing capability to host resource intense applications. The phones also have efficiently built software kernels that are capable of processing huge data. We are also able to stay online every moment and to connect to various network infrastructures. We plan to leverage the capabilities of a smartphone to deploy our honeypot.In this section we discuss mobile honeypots that relates to our idea of designing a mobile honeypot for detecting ICS specific attacks.

The power of mobility, computing resources, usability and flexibility make Mobile devices a good platform to host low interaction honeypots. Some researchers believe that mobile honeypots are still not well defined and could be used to define either a probe deployed on a mobile device or on a mobile operating system. It can also be defined for a system that is controlled in the network of mobile devices [61].

Early research on mobile honeypots focused only on Bluetooth communications[5,17]. The continuous advances in the field of smartphone technology has enabled better opportunities towards honeypot research on smart phones.

There has been existing work that focused on detection of mobile specific malware. The first to discuss the idea of a honeypot for smartphones were *Mulliner et al.*, by providing the initial ideas, challenges and an architecture for their proposed system[37]. Nomadic Honeypots[32] concentrates on mobile specific malware and also trades off with a lot of personal information.

- **HoneyDroid** HoneyDroid [37] is a smartphone honeypot for Android operating system which claims to be the first ever honeypot in the mobile honeypots category which makes use of smart phone hardware to host the honeypot.It is built on a Linux micro-kernel and is customized to impose restrictions on the Android operating system for monitoring its activities. The architecture is comprised of an event monitor, to monitor active connection requests and also system calls in the kernel level; filters to mitigate any attempts of malware trying to affect the system and a log software to log all the activities.

  This honeypot is also focused on detecting attacks from apps installed in the device which try to infiltrate the kernel for gaining unauthorized access. The system involves virtualization which enables simulation of various services. This could also result in an overhead, hereby causing a signature which can be detected by attackers and malware. However, the direction of HoneyDroid was to introduce the concept of mobile honeypots.

- **Cellpot:** Cellpot [31] concentrates on detection and defense of attacks in the cellular network. It comprises of a collection of honeypots, or honeynets that are deployed on mobile phones. Cellpot consists of applications like SMS spam prevention, mobile phone theft and malware protection. The honeypot mainly is concentrated towards Small Cells[31], wireless infrastructure deployed in customers site and operated in licensed bands. The main use of Small cells is to support the need of coverage and capacity. These points are a good place to deploy the honeypots to detect malware and other intrusion attacks.

  Denial Of Service is the most common category of attack in the area of cellular networks, and with the help of few devices,this attack can be executed successfully. Introducing a honeypot approach for detecting such attacks at small cells is a feasible solution.The concept of Cellpot is to detect, collect intelligence and mitigate threats based on the cellular network that are operated on the base stations. Further, it has the ability to deploy countermeasures against detected threats, and enables a wide area of applications. It provides a good platform for mobile network operators to easily deploy the application and run additional applications to reduce signaling overhead.

- **Nomadic Honeypots:**Nomadic Honeypots [32] propose a concept that provides a framework to enable mobile network providers to collect threat intelligence data on smartphones. It was the first proposed honeypot on a smartphone platform. It places the smart phone user as a centric role and requires the device to be used in a normal way. The honeypot is deployed on the smartphone in a separate partition which serves the purposes of providing all communication of the mobile OS, hosting infrastructure for data collection, facilities for snapshots and logging and lastly provides a secure backchannel for the operator. The main partition hosts the mobile OS and a strict line of isolation is followed between the partitions to ensure the robustness of the honeypot infrastructure.

  The operator can further use the collected data to gain intelligence on mobile threats. Snapshots of the mobile OS file system to do an offline forensic analysis of attacks could be easily provided which can gain an thorough insight on the nature of the threats and use the findings to protect his customers.

- **HosTaGe:**[56],[57] is an Android application which acts as a Mobile honeypot, determined to detect malicious networks and probe for attacks. It is user centric and aims at creating security awareness to its users. The results obtained in this process are synchronized with a global repository and also can be shared locally through bluetooth. The current version has capabilities of emulating as Windows, Unix, Apache Server, Structured Query Language (SQL) and Paranoid host. Attacks through HTTP, Server Message Block (SMB), SSH, HTTPS, Telnet and FTP can be identified. HosTaGe is one of the mobile honeypot capable of simulating different targets on a mobile platform.

2.6.4 SCADA honeypots

Analysing the security concerns of ICS SCADA systems and the advantages of honeypots, a solution could be implemented to combine the needs and features. SCADA honeypots could be deployed in ICS Networks for monitoring and analysis. They act as an additional line of defense providing warnings and notifications for attacks. Designing a SCADA honeypot involves studying the architecture of the SCADA systems and the components, protocols involved in communication and processing of data. Further, as discussed before, SCADA networks comprise of hardware devices like PLCs and RTUs which play a very critical role in processing and communication of data. SCADA systems rely on PLCs for data processing. If PLCs are targeted by attackers to compromise their working, it could bring down the entire plant, hereby resulting in a huge catastrophe.

Modern day PLCs offer TCP/IP communication which can be used to control and manage the data flow between other PLCs and control servers. On investigating attacks that have occurred in the past, Stuxnet a malware, was found to be injected in a Nuclear Enrichment Facility in Iran in the year 2009. Stuxnet was found to be injected into the internal network using a USB drive to one of the host control systems. The malware spread from that system to other systems through intranet and remained hidden from operators. Stuxnet was able to interfere with the working of a PLC that controlled centrifuges and managed to compromise the conditions on which the PLC depends. It was only by the observation of an operator that the PLC was causing the centrifuges to run more fast than usual was detected. But nobody could determine what caused the centrifuges to run abnormally.

Detecting such kinds of attacks is not only complex but also very necessary. Such kind of attacks cannot be detected neither by signature based systems, nor by firewalls. Some organizations took initiative to design honeypots for SCADA systems. They are elaborated as follows:

- **SCADA Honeynet** SCADA Honeynet Project[16] is a project aimed at building honeypots for industrial networks. It was the first of its type. SCADA Honeynet was designed to simulate the PLCs and detect attacks performed on them.The short-term goal of the project was to determine the feasibility of building a software-based framework to simulate a variety of industrial networks such as SCADA, DCS, and PLC architectures. It provided scriptable industrial protocol simulators to test actual protocol implementation. The design was an integration of stack level, protocol level, application level and hardware level. The honeypot was carefully designed to cover all the services offered by the SCADA systems, including the networking devices like routers and a direct serial device. The setup of the honeypot is complex as it involves the configuration of two virtual machines as per the network infrastructure.

- **Trend Micro SCADA honeypot** Trend Micro a global security software company conducted an experiment[2] to detect attacks on SCADA by setting up 12 honeypots in 8 countries. The honeypots obfuscated a public municipal water control system based on SCADA that was connected to the Internet. Attacks were basically focused on fondling with the pump system. The objective of this experiment was to analyze the attacks of the Internet-facing ICS SCADA devices and the reason behind them. Further, the research aimed at identifying if the attacks performed on these systems were intentional by specific interests.

  The honeypot architecture design used, is a combination of high-interaction and device based production honeypots. A total of three honeypots were deployed to ensure as much of the target surface as possible. All three honeypots were made public, facing the Internet. The honeypots were assigned different public IP addresses of different subnets distributed in the United States. The honeypot provided a full simulation of the pumps depicting a real scenario. The design used high interaction honeypots that were deployed on server instances and the real hardware was used to also simulate the environment.

- **Digital Bond** A security research and consulting firm created a honeypot system that comprised of two virtual machines. The honeypots are open source. One of the virtual machine acts as a PLC honeypot and the other is a monitoring engine that logs all the traffic information. This system is also called a Honeywall. Honeywalls can also be used to monitor high interaction PLC honeypots. The Honeywall comprises of Snort IDS and signatures with respect to PLC. The services that are simulated are FTP, TELNET, HTTP, SNMP and Modbus TCP. This honeypot focused on utilizing the signature set of IDS to monitor the network.IDS.

- **Conpot** Conpot[3] is a low interactive server side ICS honeypot designed to be easy for deployment, modification and extension. It provides a range of common industrial control protocols capable of emulating complex infrastructures to convince an adversary .To improve the deceptive capabilities it also provides the possibility to host

[2] http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-whos-really-attacking-your-ics-equipment.pdf

[3] http://conpot.org/

a customized HMI to increase the honeypots attack surface. The default configuration of Conpot simulates the Siemens Simatic S7-200 PLC with the Modbus, S7 and HTTP protocol simulation. Conpot provides ideal SCADA honeypot capabilities. The logging mechanism observed does not provide enough information for the attack analysis.

The research works discussed involve complex setup and simulation of a specific target system. They also need more resources and computing power to host the honeypots. The existing research work on mobile honeypots concentrate on designing a honeypot to detect mobile phone specific attacks. We aim to create a honeypot that is less complex, provides greater analytical capabilities, high detection ability and flexibility.

2.6.5 Multistage Attacks

In multiple networks there occurs a large number of stealthy scans, worms outbreaks and distributed denial-of-service attacks simultaneously. These attacks are very difficult to identify using IDS which monitors only some part of the Internet. *Zhou et al.* present survey[69] of co ordinated attacks and collaborative intrusion detection. Such attacks can be detected using Collaborative Intrusion Detection System (CIDS). This part of the report summarizes the detection of attacks using CIDSs. However there are two main challenges to be faced in CIDS research, which includes its architecture and alert correlation algorithms.

Attackers have recently been able to perform complicated attacks by targeting or leveraging large number of hosts that are distributed over large geographic area. For instance attackers can scan large number of hosts simultaneously to look into vulnerable systems or softwares, which are stealthy scans. Also they can use worms, which are self replicating computer programs to spread and multiply malicious code on many vulnerable systems quickly. Distributed denial of service(Distributed Denial of Service (DDOS)) can also be performed by overloading the target link by thousands of compromised hosts to disrupt and deny its service. These different ways of attacks could be integrated by an attacker which can result in a significant threat to the Internet security.

There exists four groups of alert correlation techniques used by CIDSs [69]. They are similarity based, attack scenario based, multi-stage and filter based. The similarity based technique correlates alerts based on the similarity between the alert attributes. The similarity between the alerts is calculated using a function, the resulting score tells if the alerts will be correlated or not. The attack scenario based technique correlates the approach based on predefined attack scenarios. The attack scenarios for this technique can be specified by the user or it can be trained by the datasets. The multistage technique is based on correlating the attacks based on the causality of earlier or later alerts. In this approach , there could be reconstruction of some of the complex attack scenarios by linking individual steps which are a part of same attack. In filter based technique, the alerts are prioritized based on the criticality of the protected system.

The main focus is made on multi-stage approaches in this thesis. We focus on the approach where there are possible logical links between the post-condition of an attack A and pre-condition of attack B. Therefore, executing a given attack can contribute to executing another attack. Whenever an alert is raised, it will be compared with the earlier alerts to check if correlation conditions are satisfied or not, which results in set of correlated pairs of alerts. These pairs will be verified if they belong to existing attack scenarios. If the scenario is verified then it is joined with the earlier existing scenario, or else a new scenario will be started. We model this correlation approach formally through Extended Finite State Machine (EFSM)s and discuss it further in section 3.4.

In this approach of multistage , the attacks are correlated based on the causal relationship between alerts, and most of them can detect unknown attack scenarios. This approach mainly focus on correlated alerts and discards others that cannot be correlated. However, the reason for this discarding of the alerts that are not correlated is not analyzed rigorously and the accuracy of the correlation is affected by the false alarms generated by individual IDSs. Also, the complete library of attack steps is expensive as there are large number of attack types. A language called LAMBDA is used to correlate the alert from different IDSs and CIDS. The attack is described by four main components:

1. Pre-condition and post-condition: The condition of the target to be satisfied for performing an attack, and the impact on the target system after the attack is successful.

2. Scenario: combination of attack events or steps for completing an attack.

3. Detection: steps for attack detection. This events sets may be different from scenario as some attack steps cannot be observed in IDSs.

4. Verification: some conditions on target system in order to check an attack is succeeded, like vulnerabilities existing in the system.

2.6.6 Signature Generation

Signature-based Intrusion Detection Systems rely on a signature database for the detection of malicious payloads. This signature database has to be updated with signatures for newer malware. Some attacks, as discussed in the previous section are target specific, for example specific to an organization. Detection of such attacks is very complex. Major IDS fail to detect such attacks due to the absence of signatures to detect such kind of attacks. Honeypots act as active entities that capture such attacks with no impact to production systems. The attacks can be analyzed and studied offline to determine the its complexity and impact towards internal systems. The attack information captured by honeypots could be used to generate signatures that IDS depend on to detect attacks. This is possible by either deploying a honeypot is the same network as in the IDS, or could be a collaborated honeypot that is capable of generating signatures that popular IDS can integrate.

A similar approach was proposed with Honeycomb[27] . It focuses on a honeypot system that automates generation of attack signatures for network intrusion detection system. The system uses this pattern matching technique and checks protocol conformance on various levels in protocol hierarchy to the captured network traffic in a honeypot system.The attack signatures are required to describe the characteristic elements of attacks. The system explained in this part of the report supports signatures for Bro and Snort NIDSs.

The system makes use of Honeycomb, a system that generates signature for malicious attacks on network traffic automatically. The honeypot honeyd[42] is extended by a subsystem that checks the traffic inside the honeypot at different levels in protocol hierarachy. Honeypots mainly monitor and log the activities of entities that attack or probe the system. The paper describes the extension of *honeyd*, a low level interaction open-source honeypot. honeyd simulates the hosts with individual networking personalities. It intercepts on the nonexistant hosts and use the simulated systems to respond to this traffic.

The idea is to keep the system free of any particular knowledge related to certain application layer protocols. An overview of the Signature Generation Algorithm is shown in Figure 4. Every received packets results in initiating the same sequence of activities in Honeycomb. These activities involves:

1. If any connection state is existing for new packet, that state is updated, or else a new state is created.
2. If the packet is outbound, processing is stopped in this state.
3. The protocol analysis is performed by Honeycomb at network and transport layer.
4. For every stored connection, header is compared to detect the matching IP networks, initial TCP sequence numbers, etc. If the connections have the same destination port, Honeycomb does a pattern detection on the exchanged messages.
5. If there are no useful signatures created, processing stops. Or else the signature is used to augment the signature pool.

HoneyAnalyzer [51] is an analysis tool that extracts signatures of intrusion detection patterns using honeypots. The basic idea is to analyze *honeyd* c.f [41] logs stored in an Relational Database Management Systems (RDBMS) using a web interface. The signature extraction consists of three parts.

1. **Data Capture:** A logging component which consists of *honeyd* and *tcpdump* [23] for collecting data.
2. **Data Analysis:** Tool for analysis and extraction which involves analyzing data of signature extraction mechanism to identify specific attack signatures.
3. **Signature Extraction:** Extract refined attack signatures.

The data capture component logs all the communication of an attacker through a honeypot. The *honeyd* honeypot has a log mechanism for reporting the connections that are attempted and completed. To analyze the attacks completely we need the payload information of the connections established or attempted. *tcpdump* is responsible for this activity. It captures the packets with their payload. The data analysis component is used to extract specific attack signatures. This component has a web interface that provides a graphical output which provides the administrator details about the most attacked port and the respective IP address. The strategy for extracting specific attack signature in honey-analyzer is as explained below:

1. Simulating the network using *honeyd*.
2. Start traffic analysis through *tcpdump*
3. The attacks received on the *honeyd* log file is parsed and pushed into the database using a script.
4. The web interface provides an overview of the attack patterns and the data for analysis. The web interface provides packet information, realtime network traffic, attack ports and connections per second.

This approach relies on the experience of a user to determine the malicious traffic and does not generate signatures. However, it identifies and extracts attack specific signature data. This data has to be composed into meaningful signatures that can be deployed on IDS. Honey-analyzer proposes the idea of using the attack related data that is captured on honeypots to create signatures.

In this thesis, we aim to create ICS attack specific signatures for IDS. The related work forms an important basis for our design. We also argue to be the first honeypot to generate signatures for ICS environment.

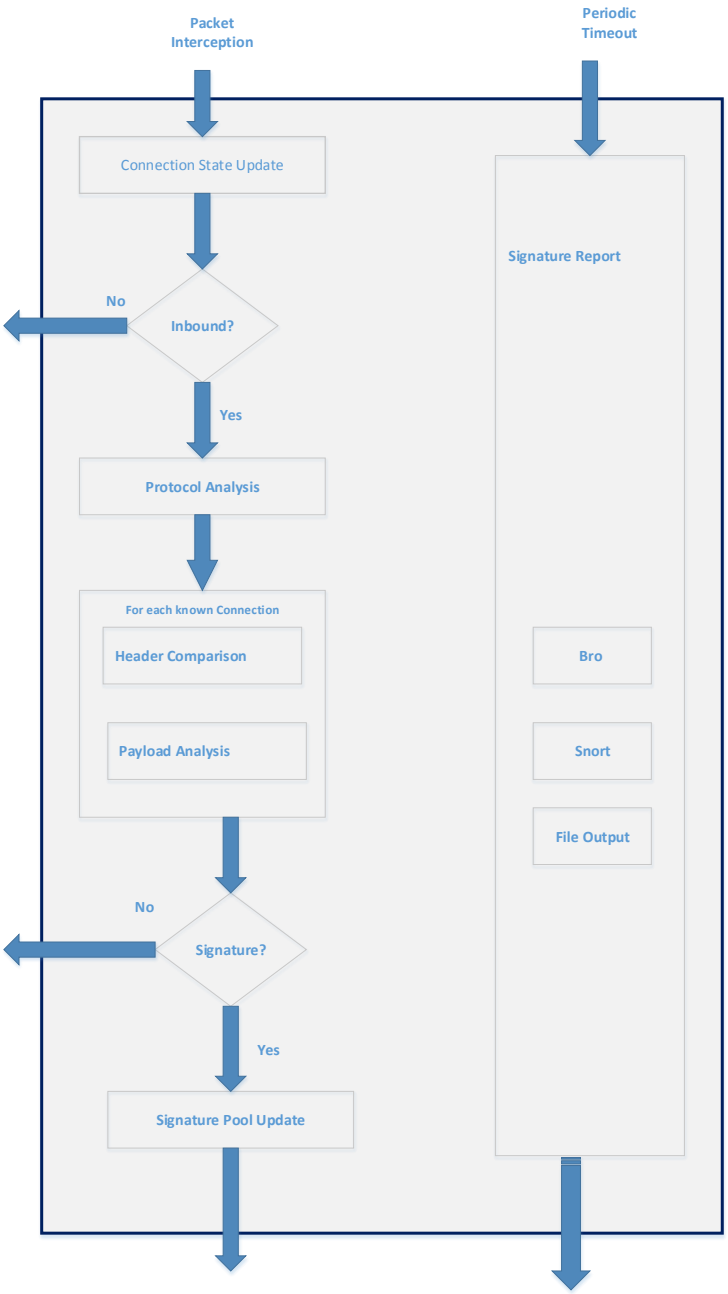

Figure 4: Honeycomb Signature Generation Overview.[27]

2.7 Summary

This section discusses about ICS SCADA. The section also consists of a brief explanation about ICS SCADA components which include network and control components. It explains the architecture of SCADA and shows the data communication and its processing on PLCs. We also see security aspect with respect to ICS SCADA and various threats that SCADA is subjected to. The section also explains about Modbus protocol, its advantages and security issues. We also saw about honeypots, its main functionalities and its advantages. We provide an overview of the Siemens Simatic series which controls and provides automation needs in ICS infrastructure today. Furthermore, the related work section explains about different types of honeypots such as Honeynets, mobile honeypots and SCADA honeypots. The chapter also briefs us on related work on multistage attack and signature generation through honeypots.

3 Proposed System and System Design

We consider the all of the requirements to design and develop a mobile honeypot system for the detection of ICS specific attacks. We also propose a honeypot capable of detecting malware and generating signatures for increasing the detection capability of IDS. In what follows we discuss the proposed system and propose a design for our honeypot.

3.1 Proposed System

In this thesis, a low interaction mobile honeypot mechanism to simulate an industrial PLC will be designed and implemented. The design also aims at detecting attacks and making inferences about the adversaries and attacks. The final implemented version will be integrated to the HosTaGe app along with advanced mechanisms that HosTaGe already provides to its users. As the proposed system deals with implementing a low interaction honeypot, the challenge involves implementing only the essential components or services, that satisfy the discovery and vulnerability to attack them, for example, the network stack. Along with basic attack detection, the system must also have a short response time, robust design to withstand the attacks and also maintain a log of the exploit for further analysis and backtracking. An attempt will be made to detect attacks forged with popular identified worms like Stuxnet. The conclusions on the attacks made will be pushed on to a central repository where the details of the attack are made public for users worldwide.

The proposed system also aims to identify Multistage attacks and generate signatures for IDS. The attacks containing malware are formally modeled to analyze their propagation through the network and their dropping mechanism. This model helps in identification and detection of such malware and also generating respective signatures.There are several systems that propose the idea of Multistage attacks and their detection.

In the following sections we discuss the design of the proposed HosTaGe ICS mobile honeypot. The discussion involves the ICS perspective for HosTaGe, the simulated Siemens Simatic S7-200 PLC system , protocols supported by the PLC, the formal model, detection mechanisms and signature generation mechanisms.

3.2 HosTaGe ICS Perspective

HosTaGe has implemented mechanisms to emulate different kind of hosts like a windows host, linux host, webserver, FTP server, SSH server and more. The simulation of industrial level SCADA based PLC will be added to the the existing list of simulated hosts and services. To simulate PLCs it is important to understand their communication and control infrastructure. PLCs have network interfaces that support Ethernet, TCP/IP, Modbus[17], DeviceNet[30], ControlNet[30], Foundation Fieldbus[52]. The manufacturers have their own in built shells to support FTP commands. The communication capability is realized through the Ethernet module which is deployed on an embedded OS. This OS includes the network protocol implementation for protocols such as Modbus/TCP. The Telnet and FTP server have the required information to identify the vendor and the firmware version deployed on the device. The components in the network stack of the PLCthat have to be simulated are the basic TCP/IP mechanism, Modbus TCP, FTP server, HTTP server and a Telnetd server. The HTTP server simulates the controlling portal to manage the PLC.

The discovery and identification of the PLC in the network can be through a network nmap scan that reveals information about the host name, ports 21, 80 and 502(Modbus) open. The main objective is to detect attacks made using the protocols offered by the Siemens Simatic S7 200 PLC . A logging mechanism logs the information about the attacker in pursuit. The architecture of HosTaGe ICS is as shown in Figure 5. A discussion of the components is followed in the below sections 3.2.1, 3.2.2, 3.2.3, 3.2.4 and 3.2.5.

3.2.1 HosTaGe Core

The HosTaGe Core forms the basic core mechanism of HosTaGe ICS. It is responsible for running the core mechanism and functionality of HosTaGe. It provides an interface for the activation and deactivation of implemented emulated protocols. The core interacts directly with the Graphical User Interface (GUI) to provide notifications to the user of the connections made. The main components or sub modules of HosTaGe Core are Emulator and Connection Guard.

- **Emulator** The Emulator is responsible for the emulation of protocols in the Protocol Emulation . It is a multi threaded module which dedicates a thread for every protocol to be emulated and also actively listens to the incoming malicious traffic for the respective service ports. It calls the Logger module to log all the activities occurring at that port. The emulated protocol can accept multiple simultaneous connections at the same instance. A Connection Handler is started for every incoming connection request. Every Connection Handler communicates with the initiating client, providing a basic protocol interface based on the selected protocol for emulation. The Port Binder module is responsible for binding the ports .

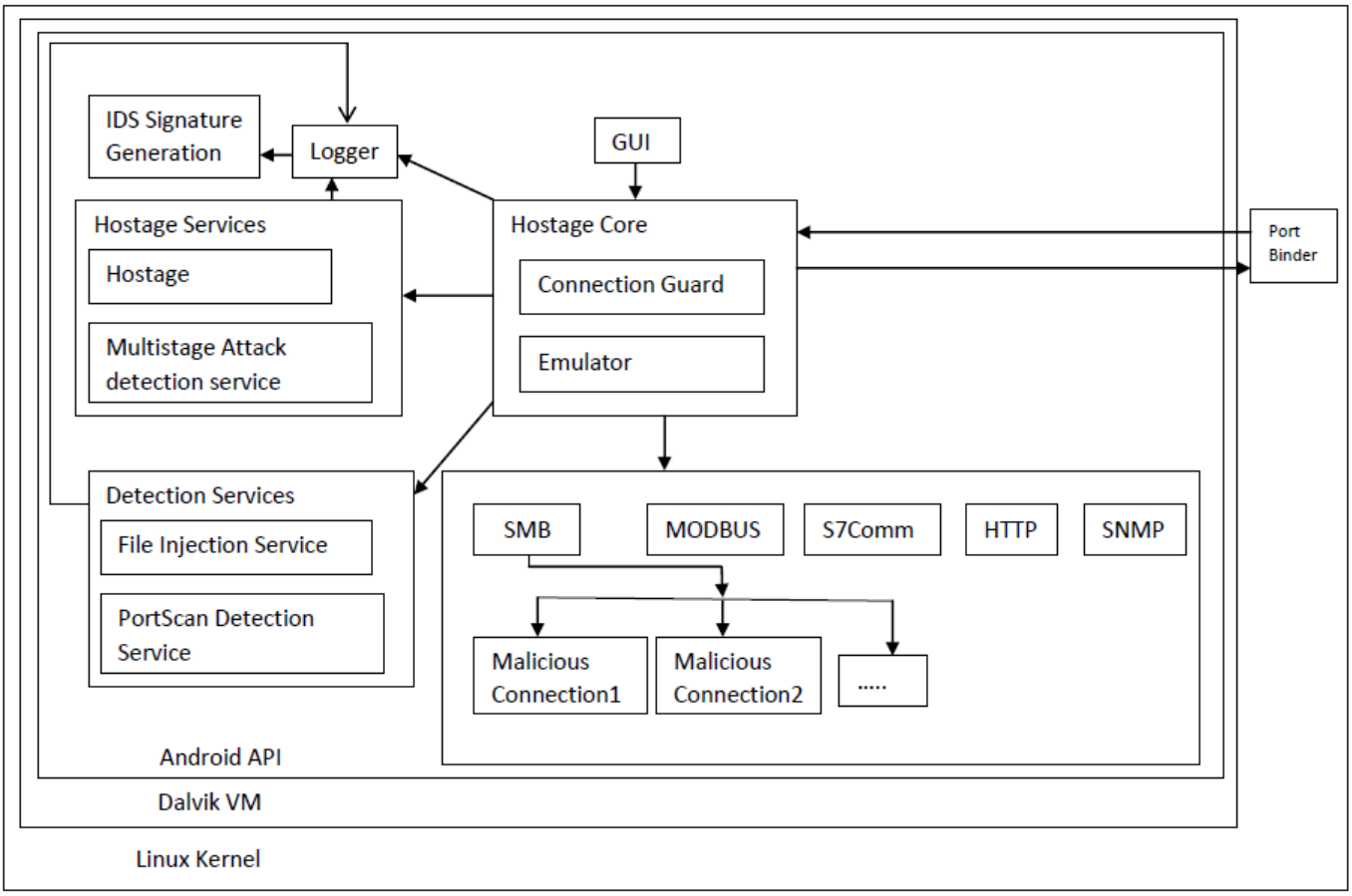

Figure 5: HosTaGe ICS Architecture

The emulator provides a selection interface for the user to choose the protocol to be emulated. For the ICS perspective we provide additional protocols like Modbus, S7, SNMP, SMTP and also a dynamically adapting version of SMB and HTTP which form a mainstream for attacking in ICS systems.Adding new protocols to the Emulator module is hassle free.

- **Connection Guard** A honeypot has to be robust itself to detect malicious attacks and survive them. However, it is always possible to compromise a system through tailor made strategies. The connection Guard mechanism limits the number of connections that can be received by HosTaGe ICS. This measure makes sure that our system is safe under the Denial Of Service attacks. In such cases, the number of connections are limited by the source IP or for the destination port. The connections are also terminated over a period of time.

3.2.2 Logger

The Logger module is responsible for logging the attack and connection data into the HosTaGe ICS SQLite database. It also supports export of the logs generated in different formats into the phone memory. The formats include plain text and in Java Synchronous Object Notation (JSON) for data processing by third party applications. The logs are also synchronized with the remote repository for global synchronization.

3.2.3 Graphical User Interface

HosTaGe ICS provides a user friendly and an interactive GUI for its users. The Overview shows the current state of the HosTaGe service with respect to the secure state of the network in which the device is placed. The Overview could be sleep, scanning- without any attack previously detected on the network, scanning- with an attack previously on the

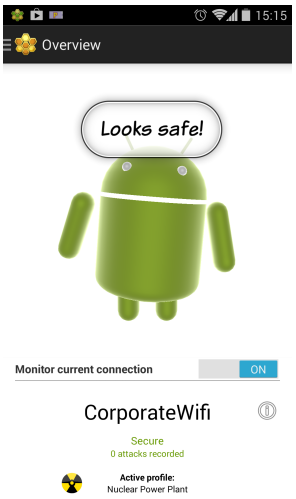

Figure 6: HosTaGe ICS Overview

network and on attack detection. This provides an overview of the network health condition for the users. Figure 6 shows the GUI of the Overview screen of HosTaGe.

HosTaGe offers various functional modes through the Profiles module. This module enables the user to select the Target system type that has to be emulated. Every profile has predefined protocols that have to be emulated which combine together to form the emulation of a target system. However, users can also select from a list of services that they want to include in the profile. HosTaGe also provides protocols to be emulated as per users choice. Users can decide upon the protocols to be emulated. HosTaGe ICS adds Modbus, S7 and SMTP to the list of protocols supported by HosTaGe. Figure 7 shows the Profile view and Figure 8 shows the Services view of HosTaGe ICS.

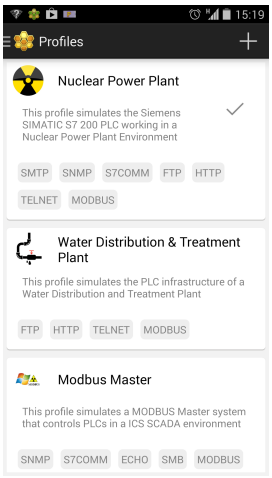

Figure 7: HosTaGe ICS Profile View

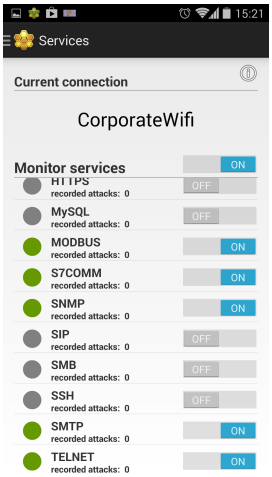

Figure 8: HosTaGe ICS Protocol View

3.2.4 Port Handling

The Android OS has a security policy of allowing only signed applications access to privileged network ports(below 1024). As HosTaGe is a third party application, by default it does not have access to these ports. The policy sounds fair with respect to the security enforcements but adds additional challenges in the implementation of the protocols for HosTaGe. ICS specific profiles include emulation of protocols like Modbus, S7, HTTP which fall in the network privilege ports category. A script was implemented to achieve this challenge in native C and cross compiled for Android OS. This program binds a port passed to it as a parameter and sends the file descriptor back to the caller through a UNIX domain socket[56] . The program uses a Java server sockets to create a socket from a file descriptor which ensures connection to the privilege ports. Some applications already bind to some previlege ports. In this case, HosTaGe notifies user that the port is bound to some other service.

However, a root privilege or a rooted Android device is required for the above mentioned program to work. We also look forward to overcome this situation for the app to work in the future.

3.2.5 HosTaGe Services

HosTaGe relies on services that run in background to achieve its constant attack detection activities. These services ensure monitoring the network, managing the connections, handling of attacks and notifications to the user through the app GUI. The HosTaGe app currently has 3 services running and are described as follows:

- **HosTage service:**The main service that HosTaGe relies on is the HosTaGe Service that controls and co ordinates the mainstream activities of HosTaGe. It is responsible for managing connections, listening on ports, enabling protocol services, attack notifications and logging of data. The HosTaGe service is responsible for the basic functionality of HosTaGe.
- **Multistage Attack detection Service** The Multistage Attack detection service constantly checks the records database for any Multistage attacks. The attack records for a pre-defined time period are retrieved and scanned for Multistage attacks.
- **Synchronization Service** HosTaGe ICS synchronizes the attack records received with a remote repository periodically. This synchronisation is aimed at creation of an updated central repository with blacklisted IPs and payload information. This information is used by a Collaborative Intrusion detection system for monitoring traffic based on the blacklisted IPs.

3.3 Protocols

The ICS SCADA systems include the master and slave devices.Our design must be capable of simulating the services of both the master and slave devices. The Siemens Simatic S7 supports a wide range of protocols which include Modbus/PROFIBUS TCP, HTTP, TELNET, FTP, SNMP, SMTP and S7. Modbus TCP and S7 are the communication protocols and the rest of the protocols are enabled as added features. Considering the other protocols form an important part of the security analysis as malware are usually designed keeping the flaws of the network and the software bugs in mind.

- **HTTP:** HTTP is supported by the majority of PLCs for remote configuration purposes. The HTTP web server in the PLC enables GET/POST messages for information exchange. This HTTP server is simulated by HosTaGe through a dynamic HTTP protocol implementation. A default welcome page is displayed when the adversary tries to navigate to the device's webpage.
- **Telnet:** The Telnet protocol allows accessing a basic shell on the devices in which users are able to dump memory, delete files and execute commands. It provides command and control to the target remote devices. It enables file system based commands and directory listing. Users or applications can communicate with the PLC for file and backup operations.
- **FTP:** FTP provides file transfer and communication between end devices. These are usually files containing sensor readings and logs.
- **SNMP:** The Siemens S7 family of PLCs the configuration of client devices through SNMP. This allows to remotely manage devices on the network.
- **SMTP:** SMTP is mainly enabled for notification servivce in case of device failure or data inconsistency.
- **SMB:** Although SMB is not a part of Siemens S7, it is a main component of the Master devices that control the slave PLCs. The SMB protocol is used to share files in a network of hosts. This protocol is typical on enterprise intra-networks due to file sharing requirements.
- **Modbus/PROFIBUS TCP:** Modbus TCP acts as a strong communication mechanism between the slaves and the master devices. It forms a backbone for industrial systems automation. Modbus has instruction sets for the interaction of devices. PLCs have registers 1 as memory units. The instruction sets are specified as functions which denote Read/Write (R/W) operations on the registers of the PLCs The protocol is used for communication exchange between PLCs and control systems.

- **S7:** The S7 protocol is a Siemens proprietary protocol utilized in PLCs of the Siemens S7 family. It is used for programming the PLCs, communication of data between PLCs, acquisition PLC data from SCADA systems and for diagnostic purposes. The protocol forms as a base for accessing the registers for R/W operations and also programming the PLC for user defined tasks.

3.4 Formal Model

The attack strategies employed by attackers can be interpreted as a sequential approach. This interpretation could be formally represented as a state model. The formal model could be used as a design perspective for the detection of attacks. *Sengar et al.* [47] propose the idea of Voice over Internet Protocol (VoIP) intrusion detection system which is based on an approach of using state machines for the modelling of network protocols and the interaction between them. Using this idea they propose a VoIP IDS based on the protocol state machines. The idea is that to utilize the state transitions made in protocol state machines for intrusion detection. A protocol state machine based IDS can be viewed as a variant of anomaly detection mechanism. Once the protocol state machine is constructed and the relevant attribute features are identified, the approach not only reduces the number of false alarms but also found to be able to detect previously occurred and unseen attacks.

A state machine deals with low-level abstraction of protocol. It can express protocol design in terms of desirable or undesirable protocol states and state transitions. The approach constructs a communicating finite state machine where the output of one machine is connected to the input of other machine. An extended finite state machine called "Mealy" finite state machine is used for the approach. The Mealy machine extends with input and output parameters, context variables, operations and predicates.

We further extend the idea proposed to HosTaGe ICS to detect Multistage and File Injection attacks. We employ the Mealy state machine model [60] to define the stages of attacks and construct formal state machine diagrams to represent attack process.

The detection mechanism of HosTaGe ICS is formalized with an EFSM. An EFSM has all the properties of a normal Finite State Machine (FSM) with an extended design of utilizing *if* -conditions, instead of only boolean conditions, to specify how a state transitions to a new state [15]. The formal model of our proposed detection mechanism is given by Attack Detection EFSM $M = (S, s_0, I, O, V, P, \delta, \lambda)$ [13]

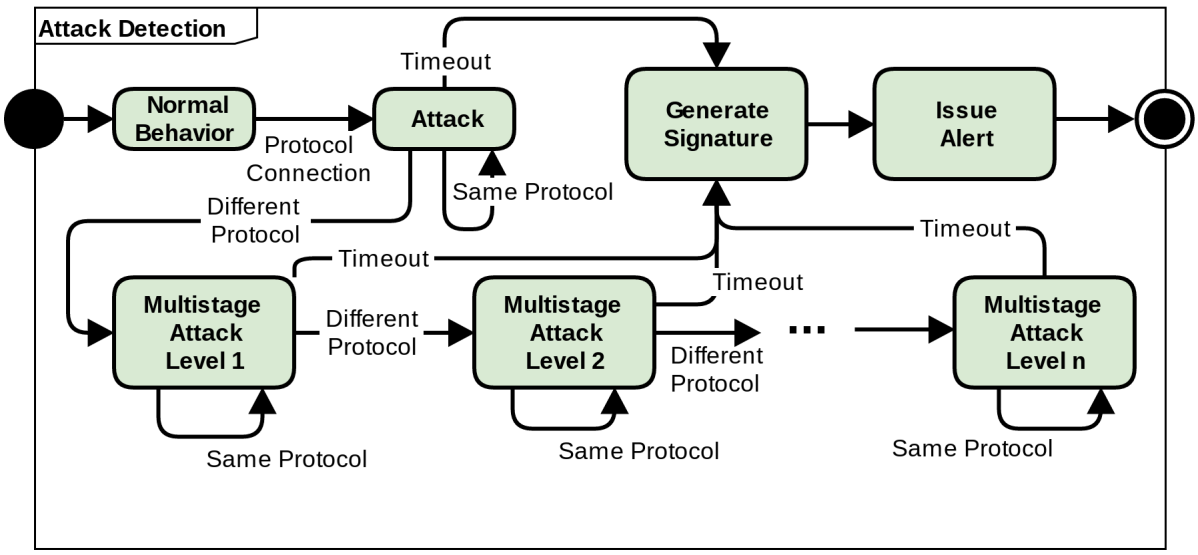

Figure 9: EFSM of the attack detection and signature generation mechanism [13]

and is illustrated in Figure 9. The set of all states are represented with S. The EFSM starts in the *Normal Behavior* state, represented by s_0. If any protocol communication is detected by the honeypot, the EFSM transitions to the *Attack* state. For as long as the same protocol attack is observed, the state remains the same. If a timeout occurs the EFSM transitions to the *Generate Signature* state followed by the *Issue Alert* state. The signature generation is optional and will capture either single attack or multistage attack types. After an initial attack, observing attacks originating from other

protocols (but the same host) that have not yet been observed moves the state to the next *Multistage Attack Level x*, where *x* corresponds to the number of different protocols observed after the first one.

The inputs *I*, outputs *O*, variables *V* and predicates *P* are tightly linked together. State transitions are carried out wherever specific inputs $i \in I$ are received.This transitions may also generate an output $o \in O$. In the *Normal Behavior, Attack* and *Multistage Attack Level x* states, the supported protocols are used as inputs and outputs. As such, $\{Modbus, S7, SNMP, HTTP, Telnet, SMB, SMTP, HTTPS, SSH, FTP\} \in I \in O$ for these states. The inputs of *I* are not limited, however, to only ports. Special activities of interest on a protocol are also considered inputs. For instance, the act of requesting a file through the *SMB* protocol is an input on itself. *V* is a finite set of variables. These variables are used to construct a set of predicates *P* used for determining if a state transitions to another one. Each attack state hold a boolean variable $v \in V$ for each emulated port. If a particular port has been observed in the entire life of the EFSM, the corresponding variable for that port will be set to true. Besides variables, predicates *P* consist of the logic operator *AND* and the arithmetical operator $=$.We define the *Protocol Connection* predicate as the condition where a new protocol is observed without having observed other protocols yet. The *Different Protocol* predicate indicates, as the name suggests, that a new protocol has been observed after having seen at least one other. If any of these predicates is true a state transition takes place.

The final element of our model is the set of transitions $\delta(s_i, i, p) = s_j$ and the outputs $\lambda(s_i, i, p) = o$ generated by the transition itself. The set of transitions specify that whenever state $s_i \in S$ receives the input i and the predicate $p \in P$ is satisfied, the EFSM transitions to state s_j and outputs $o \in O$. The outputs are used by the *Generate Signature* state to create signatures for misuse analysis.

3.5 Detection Mechanisms

HosTaGe ICS is designed to identify three different class of attacks: Single-Protocol Level Detection (SPLD), Multi-Stage Level Detection (MSLD) and Payload Level Detection (PLD).

- **SPLD:** SPLD attacks refer to those that occur on a single-protocol, eg:HTTP connection attempts without observing other protocols or any extraordinary payload-level information. This is the simplest type of detection which still contains interesting analysis potential.

- **MSLD:** MSLD refers to attacks that originate from the same source and attempt to exploit different types of protocols within a small window of time. These type of attacks are identified by the honeypot with the EFSM shown in Figure 9. An important factor in MSLD is the time-window (tw) that determines whether an attack should be mapped as the Single Payload Level Detection (SPLD) or the Multistage Level Detection (MSLD) class. This means that when the EFSM is on the Attack state and no further activity is detected (for a maximum of tw) a timeout will occur and the attack will be identified as SPLD. The tw can be adjusted with respect to the monitored network and its requirements.

- **PLD:** Payload Level Detection is enabled for the HTTP, SMB and Modbus protocols. These protocols carry critical payload in the case of ICS environment. There could be malware in the form of executables injected in the payload which can trick the end systems to execute or process them. Studies show that majority of the malware existing today spread through HTTP payloads. Payload Level Detection (PLD) extends the applicability of the EFSM with respect to the inputs I. Referring back to our formal model, the outputs $o \in O$ from the Attack and Multistage Attack Level x states are used in the Generate Signature state to create signatures. Signatures are also EFSMs that comply with the presented model. As we already mentioned, the input is not limited only to a port or protocol but also to potentially interesting payload- level information. Figure 10 can be considered as an example of an EFSM that represents a signatures generated by PLD. This signature identifies Stuxnet attacks from the set of outputs O obtained from the Attack Detection EFSM shown in Figure 10. The detection of Stuxnet EFSM assumes an initial Normal Behavior state and transitions to SMB Attack if an SMB protocol is observed. Stuxnet tries to inject an infected file through SMB. After a file is received, it (or its hash value) is sent to VirusTotal and, if the file is indeed malicious, the EFSM transitions to the Stuxnet Attack state where its presence can be reported.

3.6 Signature Generation

An Intrusion Detection System relies on its signature set and static response analysis in order to determine malicious packets and traffic. IDS depending on the signature set are called signature based IDs and the one's relying on heuristics

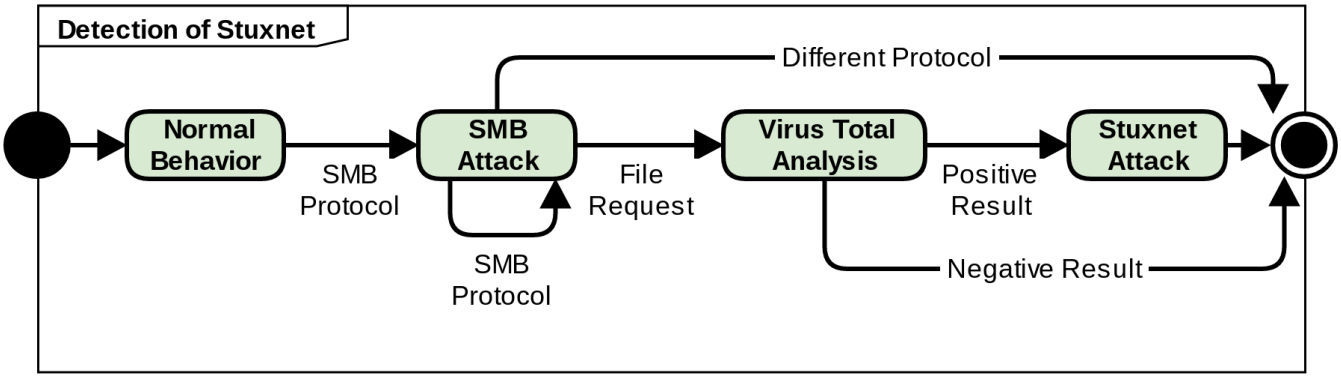

Figure 10: EFSM for PLD in the case of Stuxnet propagation
[13]

are called anomaly based IDS. Signature based IDS monitors the packets and compare them with a database of signatures. An example for signature based systems could be antivirus systems that detect malware by comparing the files with a known set of malware signatures. Anomaly based IDS detect intrusions by monitoring the system activity and classifying it as normal or anomalous. This classification is based on rules or heuristics to determine anything which is different from normal system behavior.This type of IDS also relies on neural networks and artificial intelligence algorithms to classify normal traffic.

Majority of Enterprise Networks include IDS on their network and are dependent on them for identifying and detecting malicious attacks and traffic. It is also true that due lack of expertise or internal constraints, many Enterprise Networks do not prefer having honeypots as decoys to determine malicious attackers and traffic. One such feature why IDS are preferred is because of commercial support and maintenance. IDS developers constantly monitor for newer malware and develop signatures to keep the IDS signature database updated. The administrators have to update the IDS signature database to defend against newer malware. This update system would fail under some circumstances where the developers fail to recognize the malware or if its a tailored attack strategy against an organization.Thus we propose a model of HosTaGe ICS where our honeypot has the feature of generating signatures for an IDS which could be deployed on a active IDS that monitors the network.

The Signature Generation model of HosTaGe ICS concentrates on utilizing the attack results logged by HosTaGe. The active attack correlation techniques determine newer attack strategies followed by attacker, help in understanding the payload of malicious traffic and also instantly generate signatures that can be deployed to an IDS. This model also binds the usage of both an IDS and a honeypot in a network for gaining stronger security aspects by utilizing the advantages and features of both an IDS and honeypot.

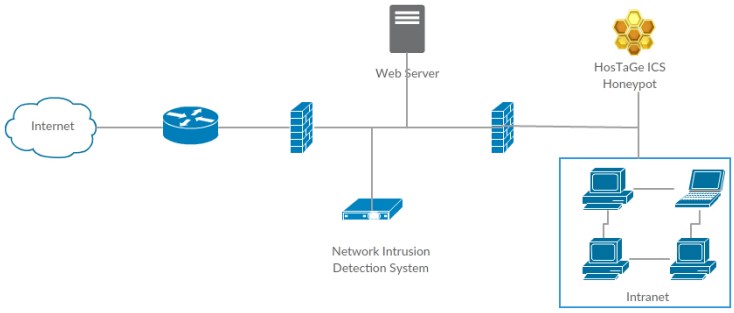

Figure 11: Network Architecture of IDS and HosTaGe

Figure 11 depicts the suitable network architecture for having both an IDS and HosTaGe. The firewall acts as a first line of defense allowing only required traffic. The NIDS comes second monitoring every packet for any malicious entries. HosTaGe can be placed either at the internal network or at the DMZ depending on the required area of interest for monitoring. It functions well at both areas. Any attacks towards HosTaGe is logged. This logged data can be analyzed further and if found malicious, a signature file for matching the content of the payload can be generated. This signature can be mounted on the Intrusion Detection System for monitoring the packets along with other attack signatures.

3.7 SCADA PLC Profiles

ICS SCADA devices can be classified into master and slave device types based on the interaction and functionality. The master system is responsible for controlling the slaves and send them appropriate commands for a task. These systems are usually control servers or host systems connected to PLCs or slaves, that receive critical information and updates from the sensors placed on devices and PLCs. The other most important systems are the automation PLCs. Slave devices interact with many other devices and collectively process information to perform a task assigned by the master. When a Modbus master wants information from a device, it sends a message that contains the device address, the data it needs and the checksum for integrity. The network is typically like a hub structure. The data is broadcast in the network and the device from which the information was requested only responds. The slave devices cannot initiate communication and only can respond to a request made from the master.

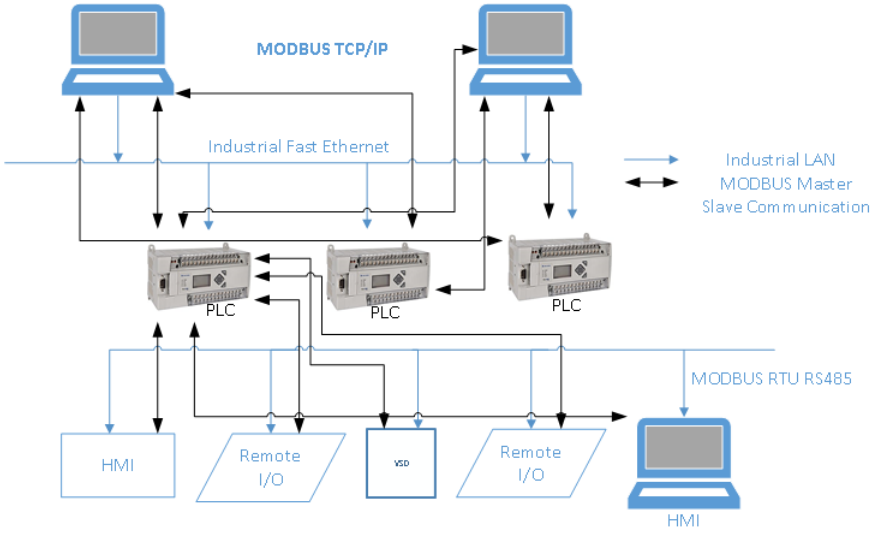

Figure 12: SCADA Master and Slave profile

Modbus/TCP allows multiple masters to poll the same device in parallel. A unit can be either a master or a slave but not both.

The Figure 12 represents devices connected on the industrial LAN and the Modbus master-slave communication. The master devices poll the slave devices and request information. The information is processed and sent back to the master. There is also possibility that a PLC acting as a master polls its data to the other devices like HMI and other PLCs in the network.

In the past there have been attacks both internal and external on SCADA systems. Popular attacks using Stuxnet, were carried out internally by deploying the malware on a host computer with the help of a USB drive. However, the malware made use of the vulnerabilities of the host system to replicate and spread through the network. Detecting such kind of attacks are very important and cannot be ignored. These attacks are more dangerous than the external attacks as there are various mechanisms to detect attacks from external sites. Internal attacks have proved to be more catastrophic. We also concentrate on the slave profile. This is required as the slave devices today have Ethernet communication and can communicate with the Internet. Due to some network configuration loop holes, the device may be accessed due to the Internet or the device itself may be configured to be accessed through the Internet by the administrator. For example, the slave devices also run HTTP servers which can display the sensor information in the form of a webpage. This device may be configured to be accessed through the Internet to check and monitor the sensor readings from a external system. There is no doubt about the possibilities of attack of such systems from the Internet. Thus we concentrate on simulating both the master profile, to check internal attacks and also slave profile to check external attacks.

3.8 Malware Analysis

Malware have been a huge threat to ICS systems. The anomaly is that in ICS environments malware is most likely to originate from internal hosts rather than from external networks. Considering popular malware for ICSlike Stuxnet, which originates from a host in the internal network and then propagates to a control host, it is very much necessary to consider attacks from internal networks. Stuxnet has still been a threat to ICS systems and its detection is imperative as it is known to cause huge hazardous impact. Figure 13 shows the overview number of Stuxnet attacks detected in various countries, just when Stuxnet malware was discovered.

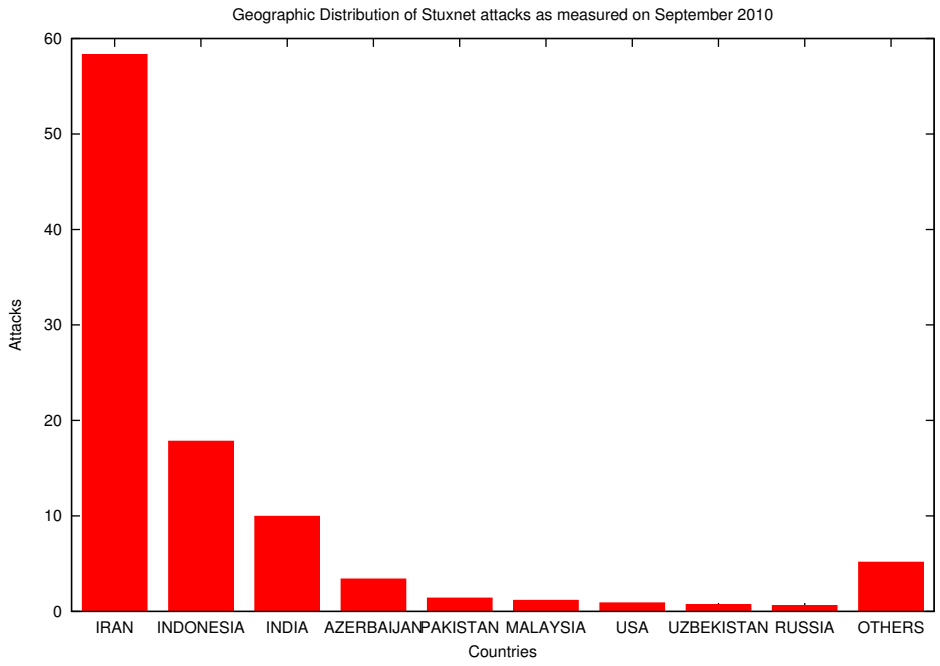

Figure 13: Stuxnet Attacks Worldwide

We first look at Stuxnet malware, its propagation techniques and a proposed design to detect the malware.

3.8.1 Stuxnet malware- A study

Stuxnet was detected in the year 2010 in a Nuclear Enrichment Facility, at Natanz, Iran. It was observed to be the most sophisticated and well engineered malware ever seen.Unlike other malware Stuxnet did not steal, manipulate or wipe information of all hosts in a network. It was a target oriented malware designed to compromise an assigned target. Stuxnet is designed to attack controller systems specifically by using SCADA applications as a means of distribution. The malware was not remotely controlled rather it was stand alone and did not need any Internet access. As a part of validation, the malware contacted some command and control servers and to prove the compromise of the target system.

The controllers are the Programmable Logic Controllers that control the data flow by using the application logic specified by the user. It is real time and is connected to controlling hosts through a fieldbus network. The controllers work on Ladder logic which is a small program that is void of any security metrics like integrity, confidentiality and authentication. The logic on controllers are in turn use to control physical conditions of heavy machinery. Any undesired changes occurring on the ladder logic results in improper functioning of heavy machinery. Heavy machinery refer to complex machinery like the centrifuges, pumps, temperature regulators, pressure systems and so on. The working of these machinery is very critical and any deviations can cause catastrophic disasters.

Stuxnet was designed to manipulate the ladder logic on which the heavy machinery rely. The idea was to inject or manipulate the controller ladder logic of the controllers. This operation was very slow but the main advantage was the stealthy behaviour of Stuxnet. Stuxnet could go unnoticed even bypassing the anti-virus programs. This ability was provided to the malware through some agencies that specialize in this sector. The malware also passed the Microsoft Software Driver Signing authentication system as it had digital signed drivers that could verify the authenticity. The digital drivers were reportedly stolen from popular driver providers. These special capabilities made Stuxnet the most stealthy malware. Stuxnet makes use of client-side applications as attack vectors on contrary to other malware that use browsers, plugins or web applications. The use of client side applications to as an attack vector is expected by the administrators, however Stuxnet manages to remain stealthy and undetected. Below are the features of Stuxnet malaware.

- **Multiple Distribution Vectors:** An attack vector is a technique through which an attacker can gain access to a host or a network to deliver a malicious payload. Stuxnet has multiple attack vectors to facilitate its distribution. It makes use of 5 Zero Days from Microsoft Windows to support its propagation. The malware is first injected to the target environment using a USB device containing the malware itself. This was achieved by social engineering

methods, for example, giving away free USB drives at a Nuclear Science Conference which had the employees of such plants. Stuxnet spread itself through USB media, hereby exploiting the LNK Zero Day vulnerability.

A Zero-Day vulnerability is an undisclosed computer system vulnerabililty which could be exploited to adversely affect the host programs, data and also other hosts in the network . The other attack vectors that Stuxnet leverages [38] are Network shared drives, Windows Printer spooler vulnerabilites, Windows Server RPC vulnerability, Windows Control Center (WinCC) Database servers, Step 7 Project files and P2P mechanism. Additionally, Stuxnet also utilizes the vulnerbilities of the Siemens Step 7 software that is used manage the PLCs. These are basically rootkit techniques aimed at compromising and pulling down the the target system completely in a stealthy manner. Stuxnet infects the libraries that Step 7 uses to manage the PLCs and replaces it with its own version.

- **Malicious Payload** The payload carried by Stuxnet is not huge but highly effective. It carries subtle payload files, that exploit the vulnerabililties of target system. As discussed previously, Stuxnet employs different attack vectors for its propagation. The payload is initially transferred to the transmission media and then, the malware flags to check if the current host could be the target host. If true, Stuxnet actively begins to compromise the host else, it waits for further propagation. Stuxnet payload consists of three parts. The first is a worm that executes all the methods with reference to the base payload of the attack. The second is a *lnk* which executes the propagated worm files. The third component is the *rootkit* which is responsible for stealthy behavior of the malware.

- **Code Obfuscation** Code packing is another term for code obfuscation. It is a technique followed that makes binary and textual data unreadable and very complex to understand. This protects the code to be detected as a malware. Obfuscation also makes it difficult for cyber monitors, anti virus engines and even humans to reverse engineer it, to check if the code is a potential malware. Code packing is also done to hide important declarations that reveal insights about malware behavior.There are several algorithms through which obfuscation can be achieved. Stuxnet follows code packing for its entire payload thereby making it difficult to reverse engineer and also detect it.

- **Anti-AV Functionality** It is the feature where the malware safely bypasses the anti virus scan without any alarm. Stuxnet gains this feature by its signed software signatures and also because of code obfuscation. These features help Stuxnet to look as normal files and system level code.

- **Data Masking** It is described as unauthentic version of an organization's data that is structurally similar. It is used for purposes such as software testing and user training. This results in manipulating original data that may be used by critical systems for controlling and managing heavy machinery. Stuxnet targets data and ladder logic implemented in PLCs inorder to compromise the working of critical systems. The failure of critical systems lead to serious consequences that may put human lives under risk.

- **Robust Malware Architecture**

  Stuxnet has an architecrure as shown in the Figure 14. An ICS network as described in the figure has many hosts which are bound by an internal network. The internal network also connects the hosts to the controllers (PLCs) that manage the industrial machinery like motors and pumps. The propagation steps is as below:

  1. A USB drive which is infected with Stuxnet is provided to employees of the plant through some public events as goodies. Usually, social engineering techniques are employed to perform this step.
  2. The infected USB drive is plugged into a host which is a part of the internal network. Stuxnet propagates itself to the host system by exploiting the zero day vulnerabilities.
  3. Stuxnet checks if the host system is connected to the Internet. If connected, it updates itself from a Command & Control server.
  4. The malware then checks for network shared drives that is mapped into the host machine. If mapped, Stuxnet slowly propagates itself to the shared network drive. Stuxnet also checks if the host has the controlling application installed for programming the controllers.
  5. As discussed earlier, Stuxnet looks for various possible attack vectors like Print spooler service and basically other services that are accessed by many hosts. It then targets such services for propagation to other hosts.

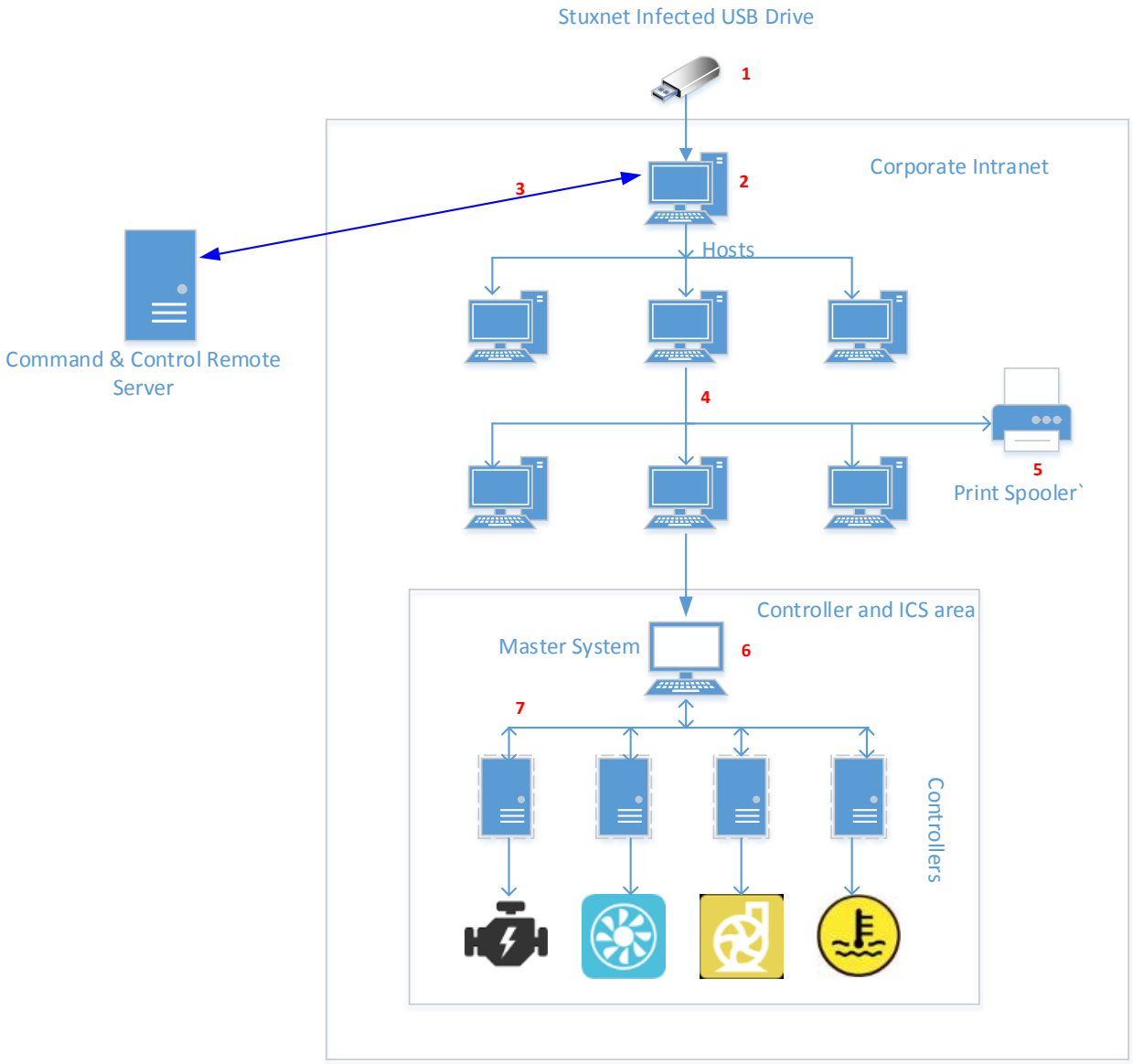

Figure 14: Stuxnet Architecture

6. Once Stuxnet reaches a controller PC, where there is limited or no connectivity observed, it validates the system by checking if there an application like Step 7, installed to program the controllers. Stuxnet now tried to establish itself into the application logic to modify the ladder logic pushed into the controllers. This scenario is similar to that of a Man-in-the-Middle attack where Stuxnet's malicious files act as adversaries.
7. The modified ladder logic now controls the heavy machinery. These machines work based on the values input to them by the controllers. If invalid inputs are passed, these critical systems fail to work normally and lead to a disaster.

The above architecture also depicts the design aspects of Stuxnet. This malware was carefully engineered considering a typical ICS environment to facilitate its propagation. It is also accurately implemented to suit multiple attack vectors.

- **Digital Signature** The applications installed for the Windows operating systems have to be signed with a trusted certificate. The certificates have to be obtained by the vendors to sign their applications. As Stuxnet was to injected to Windows hosts, it had to bypass the signature check mechanism without getting detected. This is possible only by obtaining a legitimate vendor certificate. The developers of Stuxnet stole the certificates of trusted vendors like Realtek, JMicron and imbibed them on the malware files. This way the probability of malware detection is greatly reduced.

3.8.2 Stuxnet-Propagation Techniques

After Stuxnet was detected in June 2010, there were many enthusiastic researchers who studied Stuxnet by reverse engineering, cryptoanalysis, and forensical methods. Security enthusiasts who were curious also conducted research in their private labs to investigate more about Stuxnet as it was declared to be the most sophisticated malware ever. Dissecting Stuxnet was a challenge that many Organizations also looked forward for to optimize their products. Vendors like Siemens and Microsoft were on their peril waiting for researchers to come up with their theories. Nicolas et al [14] from Symantec Labs presented a dossier about Stuxnet and its propagation techniques. The W32.Stuxnet Dossier provides the attack scenario, timeline, a survey of infected hosts and organizations throughout the world and Stuxnet architecture. Stuxnet relies on vulnerabilities to exploit the plant. Stuxnet is dormant without these vulnerabilities. It utilizes the Windows lnk exploit, 4 Zero days of Windows and the Siemens Step 7 software vulnerabililty. In what follows, we discuss various propagation methods that Stuxnet employs to reach its target system.

Stuxnet does not have a dedicated process. It hides behind trusted processes. Table 1 shows a list of trusted processes under which Stuxnet hides. In the course, it injects into these trusted processes. Once established Stuxnet implements a Microsoft Remote Procedure Call (RPC) server and client by exploit the Zero days and performs automatic updates in LAN. The WinCC is used for supervision and control of Siemens industrial systems. Microsoft SQL is used for logging. The password is hardcoded in the SQL server and is publicly available. This vulnerability is leveraged by Stuxnet to gain access as an administrator and modifying the tables with malicious Dynamic Link Library (DLL) representation.

Table 1: Trusted Processes

Product	**Target Process**
Kaspersky KAV	avp.exe
Mcaffee	Mcshield.exe
AntiVir	avguard.exe
BitDefender	bdagent.exe
Etrust	UmxCfg.exe
F-Secure	fsdfwd.exe
Symantec	rtvscan.exe
Symantec Common Client	ccSvcHst.exe
Eset NOD32	ekrn.exe
Trend Pc-Cillin	tmpproxy.exe
Windows	Lsass.exe
Windows	Winlogon.exe
Windows	Svchost.exe

Stuxnet propagates using three mechanisms:

1. It infects the host using infected removable media (such as USB flash drives and external portable hard disks);
2. It propagates through the LAN to distribute itself to all hosts having access to the network drive and print spooler.
3. Injects into the Siemens Step 7 software controller logic files and propagates to the PLCs. It injects into the ladder logic of the PLCs.

Leveraging any of the above three vectors, it uses seven vulnerability exploitation techniques for distributing to new hosts in a network.

1. Exploits a zero-day vulnerability in Windows Shell handling of LNK files.The vulnerability was present in all versions of Windows since Windows NT 4.0.
2. Stuxnet uses several techniques to try to replicate itself to accessible network shares and distribute itself through the network.
3. Injects itself to printer servers using a zero-day vulnerability.
4. It uses an older Conficker RPC vulnerability to propagate through unpatched Windows computers as shown in Figure 15. The RPC infected client requests newer Stuxnet version from an infected Stuxnet RPC server.

5. It contacts Siemens WinCC SQLServer database servers and installs itself on those servers through database calls.
6. Injects copies of itself into Siemens STEP 7 project files to auto-execute whenever the files are loaded.
7. An earlier version of Stuxnet used an old variant of *autorun.inf* exploit [29] to propagate through USB drives.

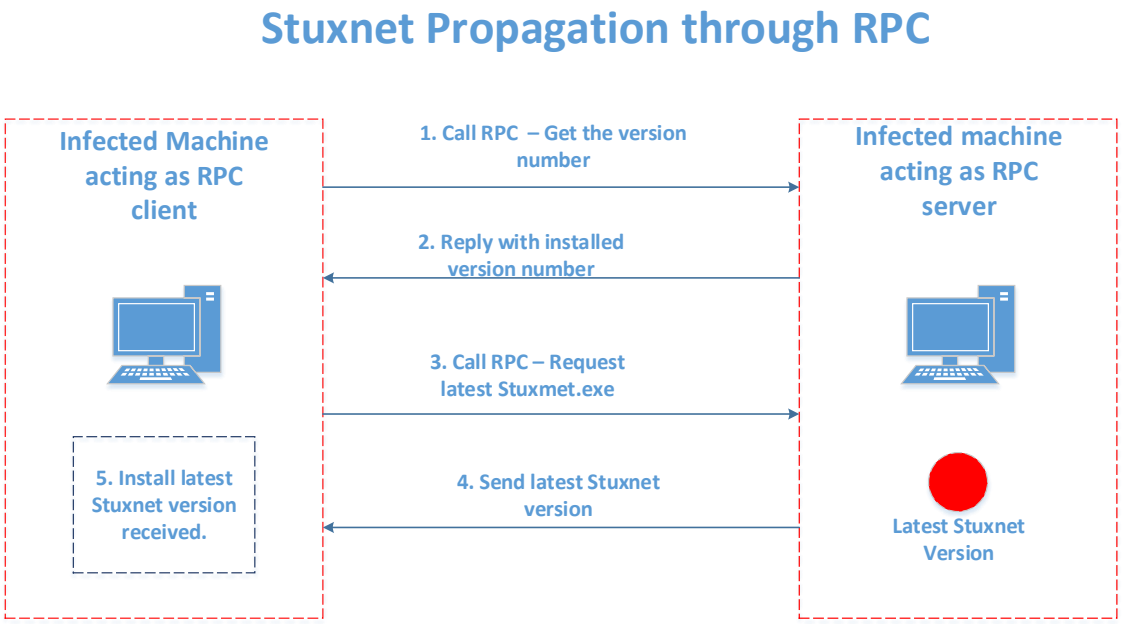

Figure 15: Stuxnet RPC Propagation

3.8.3 Malware Detection Design

After a thorough analysis of the propagation techniques of Stuxnet, identifying a environment to facilitate its propagation considering HosTaGe ICS is challenging. The most feasible technique that could be simulated by HosTaGe is the network share environment. We realise this environment through protocol simulation feature of HosTaGe ICS. HosTaGe ICS will contain the Master profile that emulates a network drive under the same /24 network. Further, the files propagated will be verified for malware. We discuss the actual simuation mechanism in section 4

3.9 Summary

This section describes the proposed system for HosTaGe ICS which aims to detect ICS specific attacks, multistage attacks and generation of signatures for Bro IDS. The section provides architecture for HosTaGe ICS with explanation for components HosTaGe Core, logger, Graphical User Interface and services. The HosTaGe core forms the core mechanism of HosTaGe ICS which consists of Emulator and Connection Guard. The logger is responsible for logging the attacks and connection data in SQLite database. We explain how port binding is handled in HostaGe and also mention the services that HosTaGe offers. The section also explains about protocols HTTP, Telnet, FTP, SNMP, SMTP, SMB, Modbus and S7. The section provides a description on design of Formal model of attacks, that represents attack stages.

Further, the detection mechanisms is classified into three types of attack classes which includes SPLD, MSLD, and PLD. A brief description on the design of signature generation , where HostaGe ICS makes use of attack results logged by HosTaGe is discussed. An overview of the master and slave profiles of SCADA with the necessity to check for internal and external attacks. Further, a brief explanation and study on Stuxnet malware is provided with its features and propagation mechanism. We also discuss the various attack vectors that Stuxnet employs to reach the target system.

4 Implementation

We discuss the architecture, features and protocols offered by the Siemens Simatic S7 200 PLC and also security concerns of ICS SCADA systems in section 3 . There were many exploit areas that were discovered. The PLC was subjected to various exploits and attacks. However, large scale attacks like Stuxnet were successful because of vulnerabilities that existed on the Host controllers as well, that is, Windows OS hosts. It made use of zero day exploits from both Windows OS and the Siemens PLCs. The attack was well designed and strategized considering vulnerabilities on both systems. There are also small attacks like information leakage from an Internet facing PLC, hosting a webserver. Over the years many vulnerabilities have been identified on the Siemens PLCs. It becomes a great challenge to make these systems secure. The PLCs have limited resources and thereby security measures like data encryption may prove expensive. Hence, data encryption was avoided. This decision of ignoring secure features induced several exploits for the device.

The honeypot must be designed keeping all the discovered exploits inorder to be more effective in attracting the attackers. We consider both the external and internal attack approaches, hereby devising strategies to capture both kind of attacks. Before we design our honeypot, it is very important to understand the previous known attacks on PLCs, their impact and the vulnerabilities that caused those attacks.

4.1 HosTaGe ICS honeypot

Based on the design decisions made in the section 3, HosTaGe is implemented to simulate the services of the Siemens Simatic S7 200 PLC. The services include the simulation of HTTP, Modbus, S7, SMTP, Telnet, FTP, SNMP protocols with respect to the PLC. HosTaGe also supports creation of honeypot environment profiles. We leverage this feature to create profiles of nuclear power plant that includes a Modbus slave open to the Internet with the other protocols. The services offered by the other protocols are also customized based on the profile chosen by the user. The services implementation involve handling incoming connection requests on respective ports of the protocols. The main objective of a honeypot is to keep the attacker engaged and allow him to pursue his attack strategy. A proper response mechanism is required which provides satisfying responses to the attacker commands and packets. For example, if the attacker tries to access the webpage by sending a GET request to port 80 of the honeypot, a webpage must be displayed to the attacker. Further, the webpage must also be dynamic depending on the honeypot profile chosen on HosTaGe. This feature has been implemented to keep the attacker engaged to HosTaGe while all the packets are captured and stored as records in HosTaGe attack database.

We also implement the detection of malware propagation through the SMB protocol. This detection forms an important aspect of detecting popular malware like Stuxnet through the network. This forms as an extension to the implementation of the SMB protocol for the Modbus Master profile. As discussed in the exploit areas, the most dangerous incident reported with respect to ICS SCADA systems is Stuxnet. We try to replicate the scenario to facilitate Stuxnet propagation by simulating HosTaGe as a shared network drive. We discuss this module further in the section 5.2.3.

During the course of thesis, an attack pattern was recognized and observed amongst the attacks recorded by HosTaGe. The attacks were later analysed to be Multistage attacks. Multistage attacks are attacks that originate from the same source IP and attempt to attack multiple active protocols on the target system within a time window. This helps in reducing false positives and acts as a means to detect genuine attacks. The detection of Multistage attacks labelled as MSLD in section 3.5 has been implemented based on the formal model shown on Figure 9. The approach to detect MSLD is discussed further in section 4.4

The signature generation module is one of the important component. It focuses on creation of signatures for an IDS based on the attack data captured and stored on HosTaGe. Signatures are generated and exported as files which can be mounted on an IDS for signature based traffic analysis. The creation of policies or rules is also implemented for the detection of MSLD. The implementation is further discusses in section 4.5.

4.2 Detecting Internal Attacks

As discussed previously, ICS SCADA systems have master and slave profiles. Though the devices are subjected to attacks from external attacks, when made open to the Internet, it is proved that major attacks in the past were triggered by systems in the internal network. Attacks from malware such as Stuxnet spread from host systems in the same network. Attacks from internal systems have proved to be more effective and dangerous as they do not leave any fingerprints, also their signature cannot be identified by the anti-virus softwares and other protection tools. The Stuxnet worm was reported to be injected through a USB flash drive. It made use of zero day vulnerabilities of the Windows operating system, the most popular one being how the Windows operating system handles the LNK [34] files, which are used by the operating system to interpret devices capable of AUTORUN functionality, and to detect the software to run the file based on its format.

An anatomy of similar kind of viruses and malware revealed that they made use of as many zero day vulnerabilities as possible to make the malware attack more effective and stealthy. Identifying such malware attacks through our honeypot mechanism is a challenge, as it involves careful design and simulation of services involved in such attacks. To achieve this, the conditions under which such worms propagate and try to sneak into the network is studied. Analysis of the studies made by researchers [28] shows that the worm looks for different attack vectors, exploits the zero day vulnerabilities of the Windows OS and also the DLL of the PLC vendors.

As discussed above Stuxnet exploits the zero day vulnerabilities on a Windows host and is dormant without it. Hence it is required to simulate atleast one of the zero day vulnerability. *Kolesnichenko et al.* [26] in their research try to perform a Quantitative Analysis of Stuxnet to find out its feasibility and the required conditions to increase its precision. The best suited amongst the five was the propagation through the network shared drive. This service could be simulated like on a WebDav server. We could then wait for the virus to propagate itself into this simulated location.

4.3 Malware Detection

Malware detection is a crucial component of HosTaGe ICS. As discussed in section 3.8.1 our main goal is to detect Stuxnet in an ICS environment. To incorporate this, we simulate the required environment for its propagation. We leverage the network share, one of the attack vector used by Stuxnet for its propagation[8].The SMB protocol also known as Common Internet File System (CIFS) is an application-layer network protocol used for file sharing, printer sharing and also certainly in Active Directory services. The protocol was initially based on Microsoft Windows operating system and was called the Microsoft Windows Network as it was a domain binded protocol. SMB provided communication between hosts in an internal network for information exchange. Stuxnet also uses printer spooler service as one of its attack vectors.

We thus infer to implement SMB protocol to simulate the network share drive or the print spooler attack vectors. The protocol emulation is a part of the simulation of Modbus Master profile. This profile emulates a set of default protocols that are open on a Windows host. Along with Modbus protocol and SMB, the target system simulated resembles to a Master host in an ICS SCADA network. Stuxnet propagates from a host in an internal network to all hosts until it finds a host with direct access to a controller. The simulated Modbus Master provides the behavior required by Stuxnet for its infection. An environment is created to evaluate the detection of Stuxnet which is further discussed in the evaluation section 5.2.3. Stuxnet propagates to the network share using the SMB protocol.

Once the malware has propagated into the network share, it is detected by the file injection detection module. Furthermore, a hash of every file injected is calculated. This hash is verified by sending it to the VirusTotal database which returns if the file is a malware. The VirusTotal repository provides an Application Programming Interface (API) to perform this validation. If the injected file is found to be a malware, it also returns the name of the malware, along with the inferences made with popular AntiVirus.

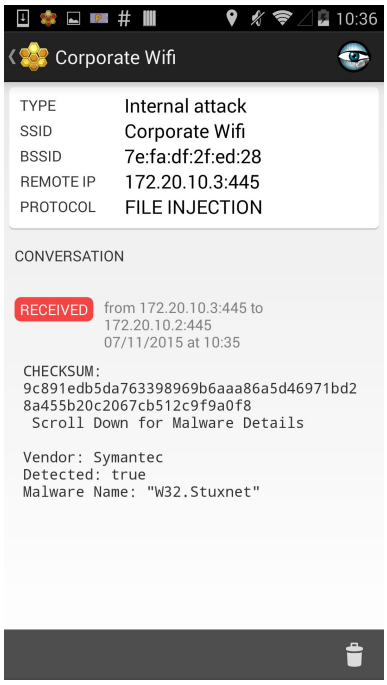

Figure 16: HosTaGe ICS File Injection

Figure 16 shows the File Injection detected by HosTaGe ICS with the attack results. The details of the file injected is shown in the CONVERSATION. It contains the the name of the file injected, checksum and details of the file as identified by Virustotal. If the file is found to be a malware, like in the figure 16, Virustotal responds with the details of the malware based on its malware database with respect to the anti virus vendors. We take into account the major players in the anti virus providers such as Kaspersky, Microsoft, McAffee, Symantec, AVG, TrendMicro and QuickHeal. The hash of the file is calculated to verify the actual contents of the file and not just the name of the file. This also reduces false positives of the attackers who just change the file names to threaten systems with a file injection.

Validation by the various Anti Virus providers ensures better analysis and inference. The Figure 16 shows the identification of a file as a part of the Stuxnet malware. The validation from various Anti Virus also reduces false positives and also provides conformation. The Signature generation module is enabled for file injection. The signature file generated validates incoming packets for the signature, hereby identifying the malware injection and propagation. Identification of malware propagated through internal network forms a good defence mechanism in ICS systems. Popular malware designed for ICS, propagate to the target system by distributing itself to the internal network. The attack record also shows that the attack was from a Internal Network.

4.4 Multistage Attack Detection

Multistage attack detection involves identifying multiple protocol attacks from an attacker in a specific time window. We use the attacker IP address, the time of attack and the protocol targeted to identify a multistage attack. These attacks filter false positives amongst many attacks detected by HosTaGe ICS. As mulistage attacks involve attacks in a sequence, it can be formally modeled. This representation of both the attacking and detection strategy provides an overview to implement the multistage attack detection service. We adapt the formal state machine model discussed in section 3.4 to implement multistage attack detection. We consider time to be an important part in our detection mechanism as it eliminates false positives. Multistage attacks are identified by the attack correlation mechanism.

The steps involved in the detection of multistage attacks is as follows:

1. All the attacks for a specific time period are fetched from the attack database.
2. The attacks are sorted based on the source IP address. (attacker IP address)
3. Once sorted, the attacks are checked for the targeted protocols. If the target protocols are the different, a multistage attack is logged.
4. The attacks are checked on sliding window basis to get lesser false positives.

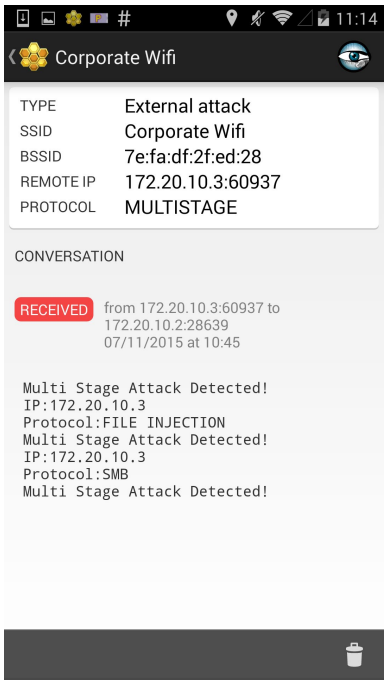

Figure 17: HosTaGe ICS Multistage Attack Detection

The logged multistage attacks contain the attacker IP address and the targeted protocols with the timestamp. An analysis of the individual protocols can be performed with the attack logs of the protocols saved. Significant observations can be made by every multistage attack as they provide a basis of target specific attacks towards an organization. These attacks remain undetected by IDS. Multistage detection service is implemented as a background service in HosTaGe ICS. When it is enabled, the attack logs are constantly scanned for multistage attacks. This service is invoked every fifteen minutes with the attacks fetched for the last twenty minutes. This service can also be disabled through the settings option of HosTaGe ICS.

Figure 17 shows the multistage attack detection by HosTaGe ICS. The record shows an overview of the attacker IP address and the protocols targeted. The attacks can be individually analyzed by selecting the respective attack record from the *Attack Records* fragment. The above record can also be used to generate policies for the Bro IDS. The policy generated checks incoming packets for attacker IP address and the protocols targeted. The time frame is also included to avoid false positives.

Listing 4 shows a policy generated by HosTaGe for a multistage attack detected. Further, Bro provides datatypes such as subnet and vectors which help in implementation of network specific data. Bro has pre-defined modules such as *connection_established* where the task to be performed on a connection establishment could be specified. The logic derives the attacker IP, subnet and the protocols attacked from HosTaGe ICS multistage attack record. HosTaGe ICS provides a method for mapping the protocols attacked to their respective ports. This information is passed to the policy generation module.

Furthermore, we check if the source IP of the connection request is the same as the attacker IP, its subnet and protocol port as identified by HosTaGe ICS. We also specify the time interval to avoid false positives. On receiving the next connection, the policy checks for the next protocol specified in the protocol vector. If the source IP matches the attacker IP and the attacked protocols are as specified in the protocol vector, a multistage attack is detected by the policy. We represent the following as an algorithm below in Listing 1

Listing 1: Algorithm for Multisage attack detection policy

```
Step 1: Start

Step 2: Fetch attacker IP, attacked protocols array from HosTaGe ICS attack record.

Step 3: Set count=0; Wait for new connection.

Step 4: On new connection; for i in attacked_protocols_array
   If (source_ip == attacker_ip) go to step 5,
    else goto step 3

Step 5: If count = 0, go to Step 6. Else, goto Step 7.

Step 6: if {(target_protocol == attack_protocol_array[count])
    count ++;
    goto Step 4
   }

Step 7: if {(target_protocol == attack_protocol_array[count])
    print (Multistage attack detected)
   }
   end for

Step 8: End
```

There were certain significant attacks observed during the evaluation period. A brief description about these results are discussed in the section 5.

4.5 Signature Generation

As discussed previously in section 4.5 Signature Generation module generates signatures for the attack traffic that is logged. We consider the Bro Network Security Monitor for generating signatures. The signature generation module is a

combined model involving modules of HosTaGe. Additional modules have been implemented to support signature generation for the Bro IDS. We highly make use of Bro's features to incorporate the signatures generated through HosTaGe. A brief discussion about the Bro NIDS and the signature generation is provided in the below sections 4.5.1, 4.5.3.

4.5.1 Bro Network Security Monitor

Bro is an opensource, Unix based powerful network monitor that provides a strong analysis framework. It provides a good platform for traffic analysis, signature matching and network forensics. It is originally written by *Vern Paxson*, and now maintained by researchers at International Computer Science Institute, Berkeley, CA and worldwide. Bro is implemented using the Bro scripting language. it is often associated to being a framework and could be used to build effective IDS. Bro is open-source and provides lot of documentation with researchers involved from all over the world.

Bro can analyze traffic both live and offline to perform analysis, take network measurement, forensic investigation and traffic baselining. Bro has many features that aim at building better IDS. The features are discussed below:

- **High-Speed and large volume monitoring:** Bro is capable of handling huge amounts of traffic. It is also quick in analyzing the traffic for the policies and signatures. Bro can also be deployed to work as a distributed security monitor, where it can share states and also propagate event data to notify other distributed peers.

- **No Packet Filter Drops:** As discussed in the previous point, Bro is capable of handling huge traffic flows. There is very minimal packet drops observed. If an application using a packet filter cannot consume packets as quickly as they arrive on the observed link, the filter buffers the packets for later consumption. However, the filter may run out of the buffer and at such point it drops further packets that arrive. From a security monitoring perspective, drops can cause inconsistent monitoring, because the dropped packets might contain interesting traffic that identifies a adversary. Thus, Bro makes sure that the packet drops are minimized, thus making it possible to capture traffic ensuring continuity.

- **Real Time notification:** Detecting attacks online as they happen is very much useful in taking relevant steps for mitigation and notification in real time. Offline detection leads to delay in analyzing the attack and being proactive. Bro supports providing real time notifications of attacks. An attack detected live provides better trace back capabilities and to minimize damage.

- **Independent Policies:** Bro separates the working mechanism from its policies. This offers more flexibilities in customizing the policies and not the mechanism. This also leads to simplicity in analyzing the policies and rewriting them. Policies could also be referred to as rules as in a IDS representation.

- **Language Support:** The Bro language is designed keeping network communications as a base. It offers datatypes for network basics like IP address, subnet, port, source IP, destination IP, source port and destination port. It also supports conditional and non-conditional programming keywords like in , when and match which are very beneficial.

- **Scalable and Extensible:** As Bro can be deployed as a distributed environment, it can easily scale to monitor huge networks and share state information. Bro has coordinators, peers and master systems to manage the Bro deployments. Bro is opensource and this enables to extend the platform for independent researchers. This is also provides the freedom to develop and test new policies and modules.

- **Protocol Support:** Bro processes both TCP and UDP packets. For each TCP packet, the connection handler checks the entire TCP header and validates the TCP checksum for the packet header and payload.

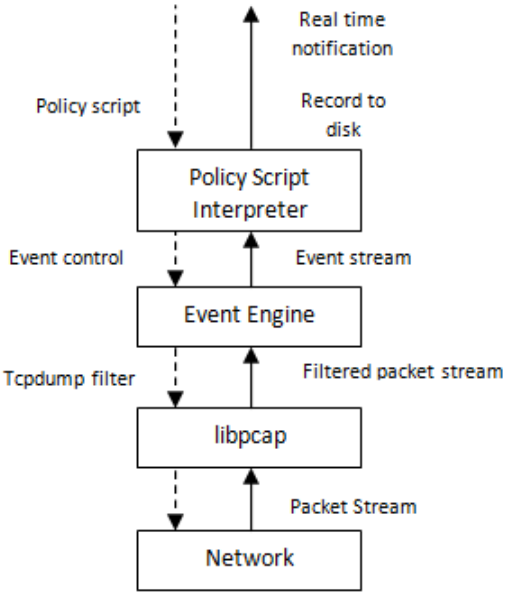

Figure 18: Bro's Architecture [39]

Bro's architecture involves Bro Event Engine, Policy Script Interpreter and libcap. Figure 18 depicts the architecture of Bro and the analysis framework. The role of each component is described below:

- **libcap:** libcap is the packet capture library used by Bro. Libcap isolates Bro from details of network link technology. It also enables to Bro to work offline by accepting packets to analyse threats and malicious content.

- **Event Engine:** This layer performs integrity checks to assure that packet headers are well-formed and also verifies the IP header checksum. Failure of checks generate events indicating the issue resulting in dropping of packets. If the checks succeed, then the event engine fetches the connection state with reference to the tuple of the two IP addresses and the two TCP or UDP ports, creating new state if no states exist. It then dispatches the packet to a handler for the current connection. Bro maintains a tcpdump trace file associated with the traffic it sees. The connection handler decides, if the engine should record the entire packet or just its header to the trace file. Upon return whether the engine should record the entire packet to the trace file or just its header, or nothing at all.

- **Policy Script Interpreter:** The event engine checks if any events have been raised. If any events are generated it processes the events as per the event handler specified until last event. Bro's emphasis on asynchronous events as the connection between the event engine and the policy script interpreter, provides lot of extensibility. Adding new functionality to Bro generally consists of adding a new protocol analyzer to the event engine and then writing new event handlers for the events generated by the analyzer. This holds true also for the policies and the signature. We leverage this feature in HosTaGe to generate policies and signatures for Bro.

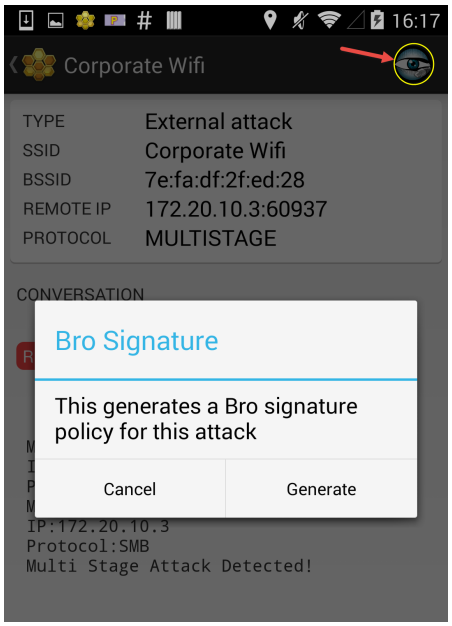

Figure 19: HosTaGe ICS Signature Generation

Figure 19 shows the signature generation Module of HosTaGe ICS. The module has been enabled for Modbus, S7, File Injection and multistage attacks. The screen HosTaGe Records shows an icon towards the top right corner. On press of this icon, a dialog box confirming the signature generation appears. The signature is generated and saved in the phone external memory after the user confirmation. This signature can be mounted into Bro IDS.

4.5.2 HosTaGe Records

HosTaGe stores all the packet data it receives in the form of records. The records include source IP, source port, attack type, protocol and the packet conversation. The data stored as conversation is the payload of the connection packets. This packet data can be considered for pattern matching and signature generation. This data is passed as a parameter for signature generation module.

4.5.3 Signature Generator

The Signature generator contains pre-defined templates which are implemented for specific protocols like Modbus and SMB for File Injection. These templates accept parameters from the conversation information stored. It also accepts the source IP address as a parameter. The source IP address is included to avoid false positives. As discussed previously in Bro's architecture, there are also policies that Bro uses to check the incoming traffic. The signature generator module also generates policies for the multistage attacks detected by HosTaGe. In the case of multistage attacks, the conversation information is not considered for the signature generation. The source IP address and the protocols attacked are considered for the generation of policies for multistage attacks.

Figure 20 depicts the modules involved in the generation of signature. Each module consisting of attributes and methods that are explained in this part of the report. The module Hostage is connected to module Listener. Hostage creates listeners for the protocol enabled by the user. The module Hostage consists of attributes which includes listener, context and connection information. The Hostage defines methods to start and stop listener, to notify UI and to get connection method. The attributes declared in the Listener module includes protocol, port, service, server and connection register. The module defines methods to start and stop service, and to log attacks. Each listener module can have one or more type of records.

There are three types of records which include Attack record, Message record, and Network record. All three types of record merge to form a single module called Record module. An attack record consists of attackId, bssid, device,

protocol, localIP, remoteIP, localPort and remotePort. It defines methods to record the attack and write to Parcel. The module Message record consists of attackId, timestamp, id, type and packet with methods defining the messageRecord and writeToParcel. The Network record has bssid, ssid, timestamp , longitude, latitude and accuracy. It also consists of method that defines writeToParcel, network record and a method to set accuracy.

The three types of records are integrated in the Record module based on their respective attackId. The integrated information is further utilized in generation of signature. The signature generated module consists of methods that defines signature generation, and its file, methods for conversion and to get integrated record.

Listing 2 shows the generated signature for a Metasploit script that checks for Modbus service on the target system. The pattern specified is checked for all incoming packets for specific contents. If the incoming packet matches the signature, an event is raised. The administrator can specify how to handle event.

Listing 2: Bro Signature for Modbus service through Metasploit script

```
signature modbus-signature1 {
    ip-proto == tcp
    dst-port == 502
    payload /\x21\x00\x00\x00\x00\x06\x01\x04\x00\x01\x00\x00/
    event "Modbus attack detected!"
}
```

4.6 Logging Mechanism

Data analysis of the attacks received also form an important part of a honeypot. This feature enables administrators to carry out forensics on the attack data and check for new malware types. Through attack analysis the attack vectors can be mapped which help us understand the strategies followed by attackers to compromise systems. The Attacks Log has an entry of all the incoming and outgoing connection information. Each connection received to HosTaGe is further divided as attack record, network record and the message record.

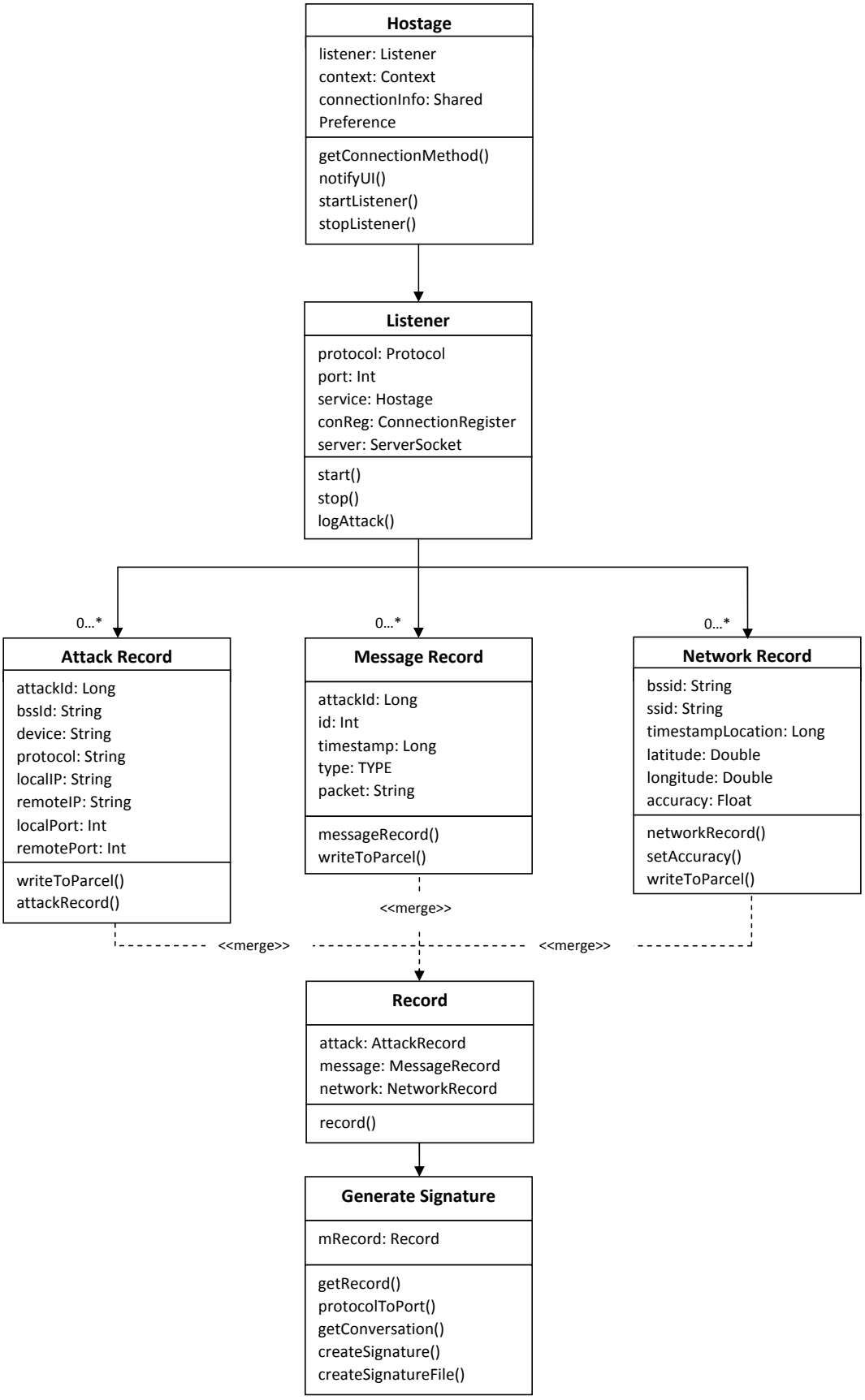

Figure 20: Signature Generator Class Diagram

4.7 Summary

In this section we focused on implementation of the HostaGe ICS honeypot. HosTaGe ICS is implemented to simulate the services of Siemens Simatic S7 200 PLC. The detection of malware through SMB protocol is also implemented which explained about detection malware like Stuxnet in the network. Furthermore, we looked into detection of malware and its validation which was one of the crucial component in HosTaGe ICS. In this part of the section we explained the detection of malware once it is propagated into network share.

We also overview how multistage attack is detected by the HostaGe ICS which involved detection of multiple protocol attacks from an attacker in a particular time window. We briefly explain the algorithm of multistage attack detection which involved four main steps. We explain the implementation of signature generation which generated the signatures for the attack traffic that was logged. A brief discussion on Bro IDS is made, which discusses some features of Bro .

5 Evaluation

In this section, we evaluate the detection capability and performance of HosTaGe ICS. Evaluation of the detection capability involves the analysis of the attacks received on individual protocols emulated by HosTaGe ICS. Further, the results obtained from the comparison of HosTaGe ICS with Conpot is evaluated. Detection of multistage attacks and signature generation were also the focus of the thesis. We evaluate the interesting results obtained through the multistage attack detection service and the signatures generated for the Bro NIDS by HosTaGe ICS. We also evaluate the environment setup for the propagation of Stuxnet and detecting it through HosTaGe ICS. HosTaGe ICS was deployed on a mobile device with resource constraints. We discuss the evaluation and performance of HosTaGe ICS in the performance evaluation. We also evaluate the impact of Shodan services and probes received and thereby determine the detectability of HosTaGe ICS.

5.1 Experimental Setup

There were two experiments carried out that involved setting up of required environments for the evaluation. The two environments are discussed below.

The first experiment was to setup an environment for the evaluation and comparison of Conpot with respect to HosTaGe ICS. Figure 21 shows the environment setup for the evaluation of HosTaGe ICS with Conpot. HosTaGe ICS was deployed on a rooted Samsung Galaxy S4 mobile phone. Conpot an ICS specific honeypot was deployed on a Raspberry Pi using an image file, Honeeepi [4]. Honeeepi has a collection of honeypots and profiles which can be customized based on the network. Both honeypots were setup on a /24 network and exposing the devices to the Internet by lifting rules on a firewall. The services emulated on these devices were publicly open to the Internet. The devices were behind NAT and all the packets were routed to the honeypots from the router using port forwarding. This helped us to have a static IP address on both devices that could be accessed publicly. The IP addresses of the devices were on the subnet. This gave a feeling to the attackers of a serial line connected to two devices that are open to the Internet.

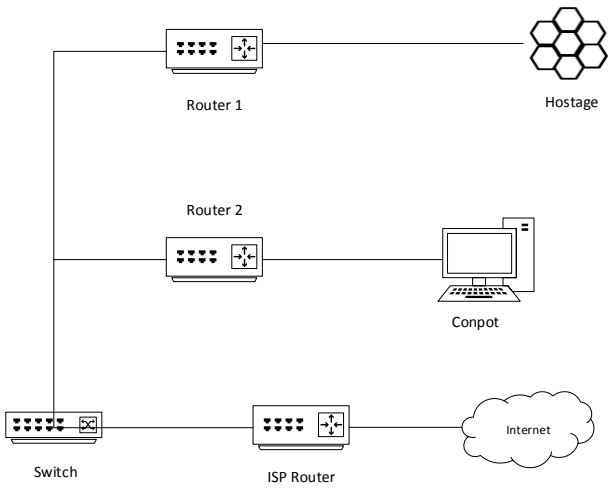

Figure 21: Conpot and HosTaGe environment setup

Conpot logs all the connection data onto a log file. The attack based on HTTP, Modbus, S7 and SNMP protocols is logged and all other connection requests that is not relevant were discarded. The logs contain the source IP address, the timestamp and the protocol attacked. The logs of protocols like Modbus and S7 contain header information of the attacks.

HosTaGe ICS logs all the attacks in its database. HosTaGe ICS has two ICS specific profiles. These profiles emulate the master and slave devices with the respective protocols. The master profile extends simulation of a Windows XP host with Modbus, S7 and SMB protocols, while the slave simulates a PLC with Modbus, S7, SNMP, SMTP, FTP, HTTP and Telnet protocols. All the connection attempts to these protocols are logged as attack records in HostaGe ICS database. Furthermore, HosTaGe ICS is also able to detect file injection, portscan and multistage attacks. The attack information logged by HosTaGe provides lot more information when compared to Conpot. The attack records contain the conversation or the message exchange taking place between the attacker and the honeypot hereby providing better analysis opportunities. The detailed summary of the attacks logged by Conpot and HosTaGe ICS are described in the forthcoming sections.

[4] https://redmine.honeynet.org/projects/honeeepi/wiki

The second environment was setup for the evaluation of the signature generation of HosTaGe ICS. Bro, a powerful network analysis framework that could be customized to be an IDS. It offers powerful customization and logged mechanisms for administrators. The signatures generated by HosTaGe is deployed on the Bro network security monitor.

Figure 22 shows the environment setup of Bro with HosTaGe ICS. Bro was setup on a Debian virtual machine. Conpot also was installed in the virtual machine to attract more attacker traffic. Bro logs all the connection attempts to the system. Logs are further classified based on TCP, UDP and also on various protocols. Logging can be customized by the administrator based on events raised by signatures and policies. Bro is shipped with many active policies and signatures by default. There are policies for HTTP, HTTPS, Modbus, SNMP and many more [39].

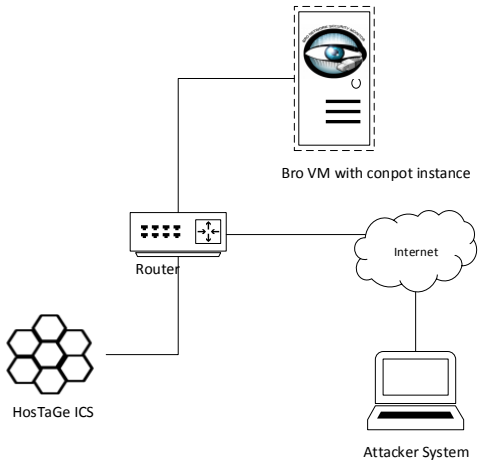

Figure 22: Bro and HosTaGe ICS environment setup

There are also policies that are classified as weird traffic based on some experiments by Bro. Bro has two traffic operation modes. Firstly, Bro can be setup to monitor live traffic, thereby applying desired policies and signatures. This option is dynamic and works like an active NIDS. Malicious packets that are identified can also be dropped before entering the internal network.Secondly, Bro can be used to analyze offline traffic which can be fed as pcap files. The pcap files can be checked for policies and signatures. Hence Bro supports both online and offline traffic analysis. We leverage this to evaluate the signatures generated by HosTaGe ICS.

Bro provides a flexible framework for evaluation and analysis of traffic. The signatures generated for Bro were also enabled for live traffic monitoring to check for accuracy by attacking the system in realtime. The signatures were accurate in detecting the attacks.

5.2 Detection Evaluation

The main aim of HosTaGe ICS is to detect ICS specific attacks efficiently. HosTaGe ICS provides mechanisms to detect attacks with respect to individual protocols and also to detect popular attack strategies such as portscan, file injection and multistage attacks. It is very important to evaluate the results obtained using these mechanisms inorder to determine the efficiency and productivity of the application. We compare HosTaGe ICS with Conpot, an interactive ICS honeypot, to evaluate the detection capability. Further, HosTaGe offers detecting malware that is propagated through protocols that are open in ICS networks. We evaluate the detection of Stuxnet, a popular and devastating malware in ICS SCADA systems. The evaluation of the featured capabilities discussed above is summarized in the following sub sections.

5.2.1 Analysis of Individual Protocol Attacks

Table 2: Attack results per Protocol in HosTaGe ICS

Week	HTTP	MODBUS	TELNET	S7 Comm	SMTP
	HosTaGe ICS	HosTaGe ICS	HosTaGe ICS	HosTaGe ICS	HosTaGe ICS
Week 1 July 7th-12th	83	9	177	NA	NA
Week 2 July 13th-19th	174	10	283	NA	NA
Week 3 July 20th-26th	127	11	514	NA	NA
Week 4 July 27th-August 2nd	79	11	260	NA	NA
Week 5 August 3rd-August 9th	83	9	225	NA	NA
Week 6 August 10th-August 16th	94	11	154	2	12
Week 7 August 17th-August 23rd	116	9	198	1	24
Week 8 August 24th-August 30th	88	13	256	1	17
Week 9 August 31st-September 6th	109	7	512	4	9
Week 10 September 7th-September 13th	89	14	239	0	14
Week 11 September 14th-September 20th	104	10	212	2	18

This section presents analysis of various protocols emulated by HosTaGe ICS. It consists of results for 11 weeks from 7th July to 20th September 2015. We need to note that HosTaGe is placed outside the firewalls, by making it face the Internet. The HosTaGe and Conpot had similar IP address which means they were involved in the same/24 sub-network. The analysed result of protocols is shown in the Table 2.

The table 2 consists of protocols emulated by HosTaGe ICS which includes HTTP, Modbus, Telnet, S7 and SMTP protocols. The Telnet protocol results have been gathered as it is considered to be an important attack factor for ICS networks. Note that the S7 and SMTP protocol is considered for later weeks as its implementation was completed in a later stage.

We also identified some genuine attack results for the Modbus protocol during the evaluation period. These were identified to be genuine as the attackers came back and tried to probe the target multiple times. The Table 3 shows a list of attacker IPs with the location and the type of attacks carried out. This information forms a basis of detection evaluation of HosTaGe ICS. We provide only a overview of the attacks in the Table 2 with respect ICS profiles. There were a lot of interesting results that were observed during the evaluation period. One such interesting result was the detection of an attack pattern which led to the identification of multistage attacks which we discuss in further section.

Table 3: ICS Protocol Attacks Overview

Attacker IP	**Number of Attacks**	**Type of attack**	**Location**
5.200.120.251	3	Modbus service check	Iran.
71.6.165.200	2	Modbus port scan, with read coil	San Diego, California, United States.
198.20.69.98	6	Modbus service check	Chicago, Illinois, United states.
188.138.1.218	2	Modbus read coil	Germany.
101.69.178.234	5	Modbus service check	Hangzhou, Zhejiang Sheng, China.
37.153.181.219	4	Modbus port scan	Iran.
169.54.233.116	2	Modbus service check	United States.
52.10.40.42	4	S7 Service detection	Boardman, Oregon, United States
80.13.27.236	3	Modbus service check	France

5.2.2 Conpot and HosTaGe ICS attack comparison

In the section 2.6.4 Conpot is described briefly. Conpot is an ICS honeypot which simulates the behavior of a Siemens Simatic S7 200 system. It has the S7, Modbus , HTTP and SNMP protocols implemented to support the simulation of the PLC.

The purpose of this experiment is to compare the two honeypots (HosTaGe and Conpot) and to detect the automated attacks that targets the ICS networks and also the reason for not advertising the honeypots. However, it is shown that both honeypots are probed by well known search engine Shodan. c.f 5.4

Conpot was deployed on one of our servers and was exposed to the Internet for attacks. All the connection packets received for HTTP, SNMP, S7 and Modbus are logged. However, the complete packet with the payload information is not logged by Conpot. The logging mechanism is different for each protocol. For evaluation, HosTaGe ICS was also deployed and exposed to the Internet for attacks [58]. Both honeypots were kept running for a specific evaluation period to analyze and compare the results at the end of the evaluation period.

The honeypots were deployed in controlled environments with no firewalls between them and the Internet in the same /24 subnet. The results of the analysis are shown in Figure 23. The results gathered from the two honeypots for the HTTP, Modbus and S7 protocols are compared. The Telnet protocol is a part of HosTaGe ICS simulation. In addition to the mentioned protocols, HosTaGe ICS also received a lot of Telnet attacks. Telnet is considered important as it provides a shell access on the PLC devices, through which command and control is possible directly. We also show the Telnet attacks in Figure 23.

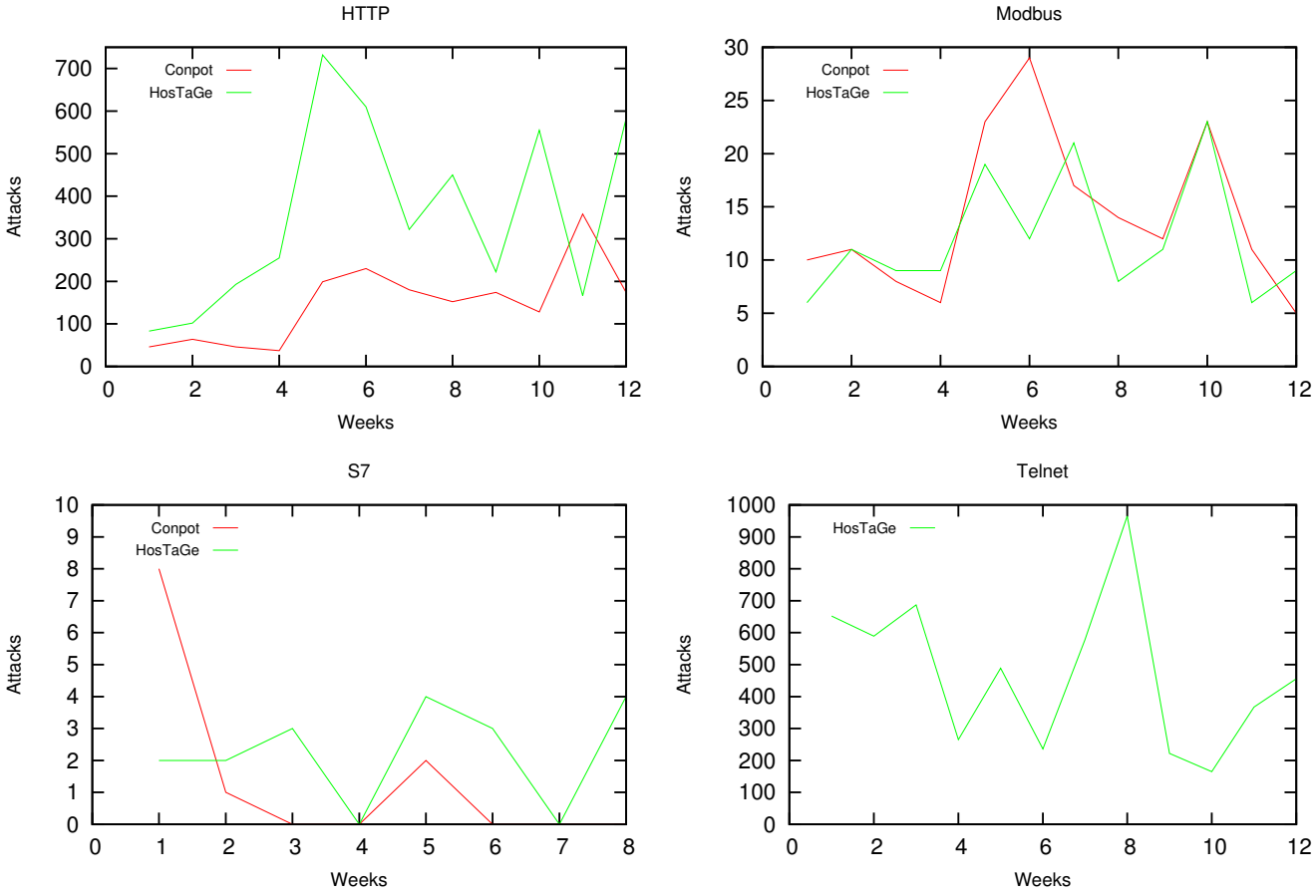

Figure 23: HosTaGe and Conpot Comparison

Inferring some results from the Figure 23 we can note that HosTaGe exhibits good and on-par detection accuracy when compared to Conpot. In the case of HTTP, HosTaGe ICS is observed to have more attacks than Conpot and remains on par with Conpot on Modbus protocol. It is also observed that the results obtained through the Modbus protocol are a part of wide scans which are operated through research communities. For example [12]. It is important to mention that neither of the honeypots were advertised in any form on the Internet. However, Shodan the online vulnerable device database and search engine managed to find and probe both honeypots. This is further discussed in section 5.4.

The main goal of this evaluation was to compare the performance of HosTaGe ICS with Conpot in terms of attack detection and gathering. It is notable to observe that Conpot runs on a full PC environment while HosTaGe ICS performs equally better on a mobile environment.

This inference forms a basis for proving that HosTaGe ICS is a powerful honeypot which runs efficiently on limited resources and is capable of simulating the services of real devices on a mobile device platform.

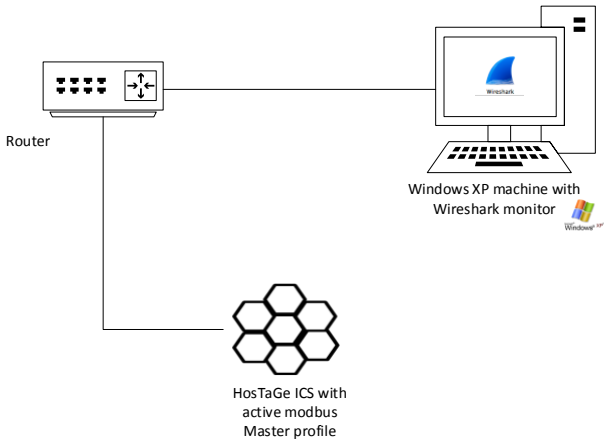

Figure 24: Stuxnet Propagation Environment

5.2.3 Stuxnet Propagation

Stuxnet uses specific attack vectors to propagate to its target system. It remains dormant if it does not find the required attack vectors for distributing itself to the network. The most feasible attack vector that can be simulated in our evaluation environment is to simulate the network share service.

Figure 24 shows the environment setup for the evaluation of Stuxnet propagation and detection.We create a suitable environment for the propagation of Stuxnet through HosTaGe ICS and a vulnerable Windows XP box. The experimental setup for the evaluation is performed as below:

1. Stuxnet requires a vulnerable host with zero days to initially get active. A host system with vulnerable Windows XP operating system that is vulnerable to Zero Days is setup in the evaluation network.

2. HosTaGe ICS is deployed in the subnet with Modbus Master profile active. This profile enables the SMB protocol emulated that simulates a shared network drive in the network. This shared drive has a structure that resembles the SMB/CIFS file share of the earlier Windows systems.

3. We deploy a host system with a packet capture tool like Wireshark as a tap mechanism. Wireshark monitors all the information that is communicated between the vulnerable host and the target system.

4. The shared network drive simulated by HosTaGe ICS is mapped in the Windows XP machine.

5. The Windows XP host is injected with Stuxnet using a USB Flash drive. Stuxnet leverages the LNK[5] exploit of the Windows Zero Days to inject itself into the System files.

6. Once established, Stuxnet updates itself from a Command and Control server. Stuxnet checks if the current host communicates with a PLC. If yes, it infects the PLC through the Siemens Step7 software through a software attack vector. If not, it checks for other attack vectors to propagate itself to other hosts in the network.

7. Stuxnet detects that the host is mapped to a network shared drive and uses this attack vector to propagate itself to the drive. It propagates using the dropper file, dropper.exe into the shared drive. This propagation is captured using the Wireshark tool. The Figure 25 shows the propagation from the host system to the network shared drive which is simulated by HosTaGe ICS.

8. The file propagated is detected by HosTaGe ICS. A hash of this file and its contents is computed and validated for malware from Virustotal malware database. Virustotal returns the name of the malware identified by various antivirus providers to HosTaGe ICS thereby validating the injected file as Stuxnet.

[5] https://technet.microsoft.com/en-us/library/security/ms10-046.aspx

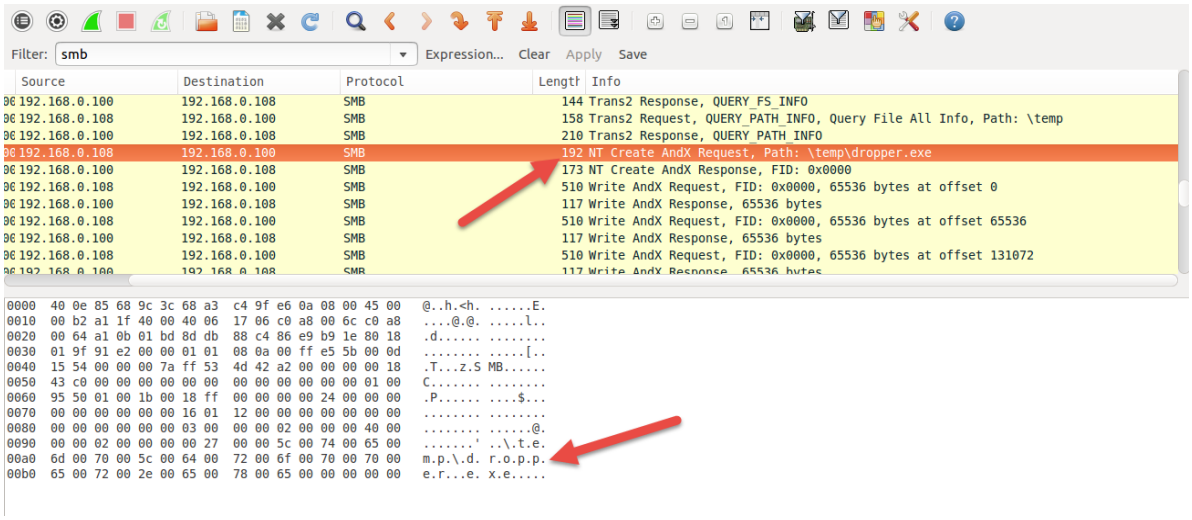

Figure 25: Wireshark capture of Stuxnet Propagation

5.2.4 Multistage attack detection and Inference

There were three significant multistage attacks. Among them the IP '5.200.120.251' was interesting as the location of the IP is approximated to be a Prison Facility in Tehran, Iran(which is also approx 10 kms away from the Tehran Nuclear Research Center). The attacker performed a portscan and sent a HTTP GET request to the IP. The next step was an nmap script to check if a Modbus service is running on the host. Listing 5 shows the script used for this attack. The second interesting attack came from IP '183.131.76.132' which is also approx 15kms away from Zhejiang Nuclear Industry in China. The attacker approached the HTTP protocol first and then targeted the Modbus protocol by sending a service query to the host. The third interesting attack was from IP '181.143.236.179' which is 12 minutes away from EPM Power Plant in Colombia. The connection was first established to HTTP followed by a portscan and then by a Telnet session.

Table 4: Multistage Attacks Overview

Attacker IP	Protocol/Attacks	Latitude, Longitude
183.131.76.132	Portscan, HTTP, Modbus	29.1068, 119.6442
181.143.236.179	HTTP, Portscan, TELNET	6.2518, -75.5636
5.200.120.251	Portscan, HTTP, Modbus	35.6961, 51.4231

5.3 Bro Signature Generation and Evaluation

Generating multistage attack signatures is one of the mainstream features of HosTaGe ICS. We follow a series of steps to evaluate the signatures generated.

1. We pose as an adversary and attack HosTaGe ICS with multistage attacks as a goal. The attacks are captured on a packet capture tool like Wireshark and saved as a 'pcap' file for further reference.

2. HosTaGe ICS detects the attacks from the previous step and generates signatures from both the protocol and payload-level interaction.

3. To determine the applicability of the generated signatures, we perform two different tests. All tests utilize publicly available datasets of network traffic (in the form of pcap files). They consist of synthetic and real network captures[6] , malware focused traffic [7] , and honeypot captured traffic [8].

4. We then determine whether the generated signatures detect false positives. We import the signatures into the Bro IDS and replay the network traffic of the test datasets. We utilize a time-window of tw = 15 (minutes) for our tests. No multi-stage attacks were detected by Bro, as expected.

[6] Small and Big flows datasets:http://tcpreplay.appneta.com/wiki/captures.html
[7] CTU-13 Dataset: https://stratosphereips.org/category/dataset.html
[8] HoneyBot Dataset: http://www.netresec.com/?page=PcapFiles

5. As a next step, we merge each dataset with the network traffic captured by Wireshark in the initial step (that includes our injected multi-stage attacks). Subsequently, we replay each modified network file while Bro is running. In all cases, Bro successfully detects all of the injected attacks without generating any false positives.

5.4 Shodan Evasion

One of the most important and essential features of a honeypot is to remain undetected as a decoy mechanism. This is a very hard feature to achieve and to be evaluated. During our experimental evaluation setup of Conpot and HosTaGe ICS, we received a lot of probes from Shodan. Shodan is one of the biggest search engines to find vulnerable devices on the Internet. It crawls the entire Internet to check for vulnerable devices. On the Shodan website, users can search for IP addresses, specific devices, protocols and also vulnerabililtes like heartbleed [11] or default password. Recently, Shodan started a new service called *Honeypot Or Not?* which can identify honeypots. This service performs a series of probes or checks and subsequently creates a score, which is called *Honeyscore* for each probed device. Shodan determines if the host is a honeypot based on this score.

During the evaluation period both honeypots (Conpot and HosTaGe ICS) received probes from Shodan. The probes targetted HTTP, Modbus and S7 protocols. There were as many as 90 probes received over a period of 8 weeks. The probes were observed to be from 7 different subnets.

After 3 weeks, Shodan identified Conpot to be a honeypot and listed this on its website at the honeypot or not site . Figure 26 shows the Shodan search result of our Conpot instance IP. The open services and protocols were also listed. HosTage ICS was not detected as a honeypot by Shodan. The Shodan probes for HTTP and SSH were of low complexity, where the HTTP protocol involved a GET request and the SSH protocol was just the initial handshake message. The probes for Modbus and S7 protocols looked more sophisticated. An analysis made to the payload reveal that it could be a modified Nmap or Metasploit script to identify ICS.

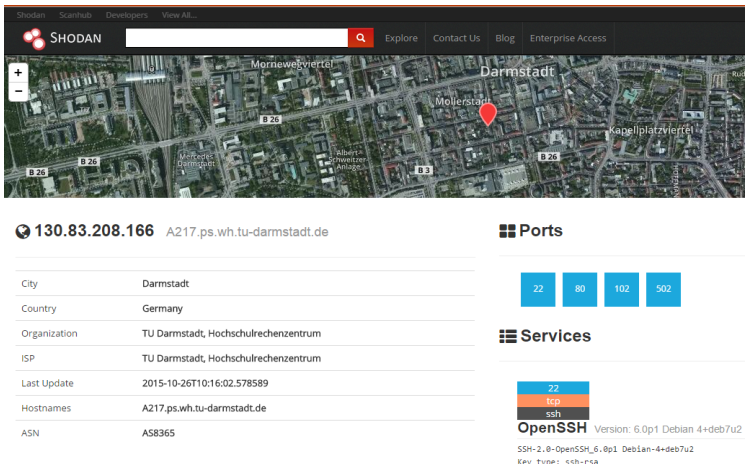

Figure 26: Shodan Search result of Conpot Instance [9]

The S7 attack involved checks for the device type, location, serial number, plant identification and module name. The Modbus attack involved fetching details of certain units (i.e., unit number 0 and 255) and their slave data. Conpot could not respond as expected by Shodan in all the aforementioned requests (either due to static serial numbers or Modbus protocol simulation errors) and thus was classified as a honeypot. HosTaGe ICS managed to respond successfully and hence remain undetected. Figure 27 shows the conpot instance detected as a honeypot in Shodan *Honeypot or Not?* page.

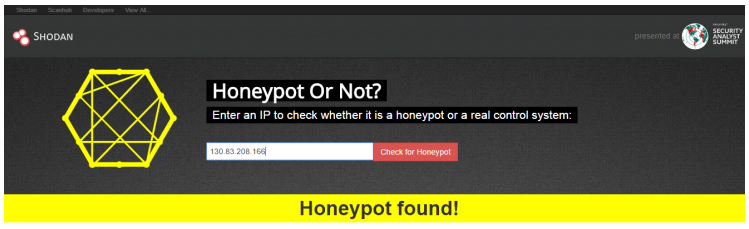

Figure 27: Conpot detected at Shodan Honeypot or Not page [10]

We tried to investigate Shodan's honeypot identification criteria, based on the probes received by Shodan. Finding the probes received from Shodan was a challenge. Shodan does not reveal it probes IP. From a lot of study and mining on the Internet, we could identify some of the Shodan probes and tried to fetch data from our logs based on this information. Table 5 shows the various Shodan probes, hostname, location and the protocols they probe. Though we have the hostname, it is essential to know the packets that the probes use to get information to decide on whether the target system is a honeypot. In Listing 3 we show the packets received from the probes of Shodan. This information was logged by Conpot and Bro NIDS. We aggregate the log information from both to get some interesting information.

Table 5: Shodan Probes Overview

SlNo	Shodan Probe IP	Hostname	Protocol Targeted	Location
1	71.6.167.142	census 9.shodan.io	S7	San Diego. California, United States
2	71.6.135.131	census7.shodan.io	HTTP	San Diego. California, United States
3	66.240.192.138	census8.shodan.io	HTTP	San Diego. California, United States
4	66.240.236.119	census6.shodan.io	S7	San Diego. California, United States
5	93.120.27.62	m247.ro.shodan.io	Modbus	Romania
6	188.138.9.50	atlantic.census.shodan.io	S7, Modbus	Germany
7	85.25.103.50	pacific.census.shodan.io	Modbus	Germany

Further we analyse the impact of the above Shodan probes on our honeypot environments. To determine the impact of Shodan[4] on our honeypots, we consider the attack data obtained before Shodan Probes and attacks after the Shodan probes. We observe a significant decline of attacks after Shodan declares Conpot instance as a honeypot. Shodan probed both the honeypots roughly around the same time period. Figure 28 shows the impact of Shodan probes on Conpot instance before and after the probes attacked Conpot. A significant fall in the number of attacks is observed. On the contrary, Figure 29 shows the impact of Shodan probes on HosTaGe ICS. There seems to be negligible impact on HosTaGe ICS. However, to support our research that Shodan probes do make an impact on the honeypots, we need to prove that there is a realistic impact and that attackers look forward to Shodan for determining if the vulnerable systems on the Internet are actually honeypots. This research is important, considering honeypots to be stealthy detection systems. Such services pose a threat to the working aspect and productivity of a honeypot.

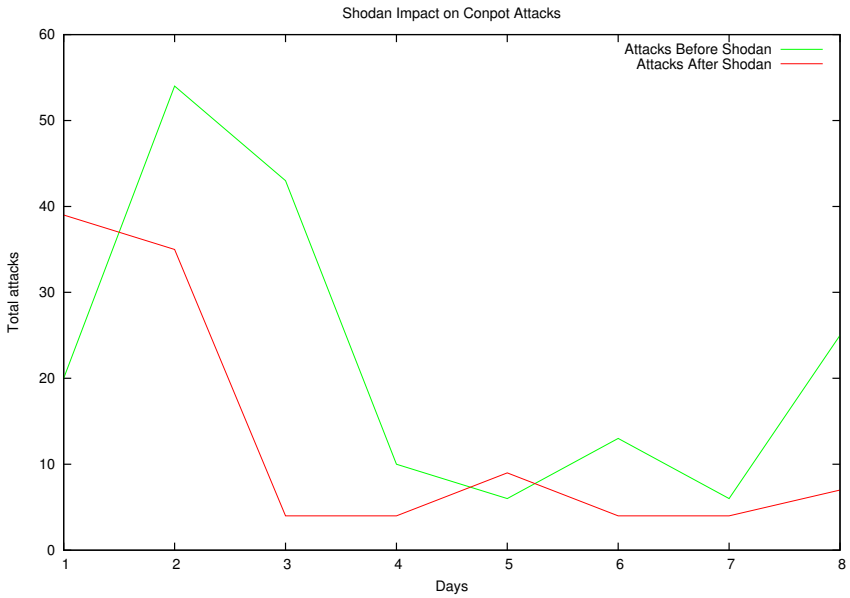

Figure 28: Shodan Impact on Conpot Attacks

We can make use of Gaussian Distributions with the help of curve fitting algorithms to prove the impact is significant, provided we have more live data. This is a continued research and will be focused on the future work of HosTaGe.

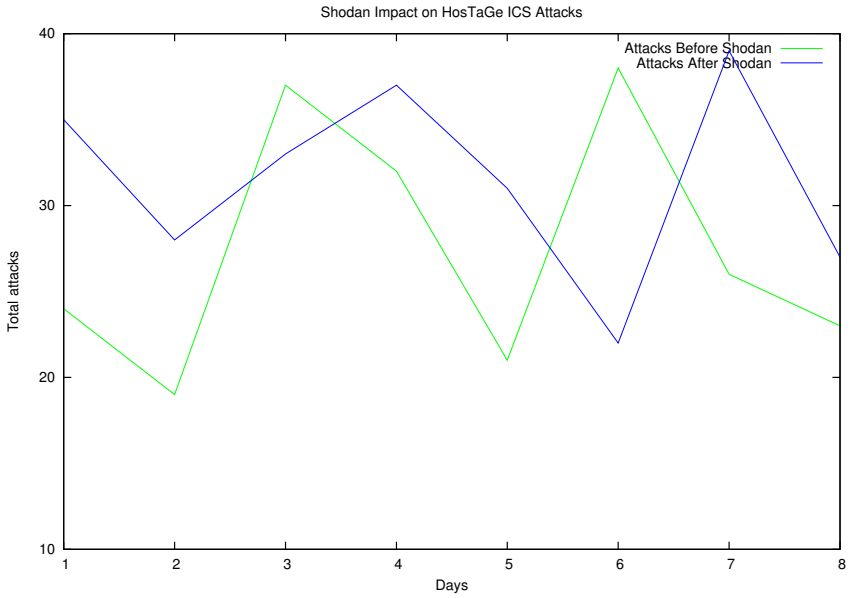

Figure 29: Shodan Impact on Conpot Attacks

5.5 Performance Evaluation

Honeypots are usually deployed on robust systems that can withstand strategized attacks. HosTaGe ICS is a mobile honeypot designed and implemented for the Android OS platform. Android is highly flexible and open source offering a large number of developer APIs and libraries.

We intend to provide a robust environment for HosTaGe ICS through the Android platform leveraging its open source and developer friendly features. We try to achieve robustness through the protocol emulation mechanism. The mechanism has been carefully implemented for reply mechanisms of individual attacks. Further, by studying the behavior of target systems with respect to popular attacks towards individual protocols, a robust response mechanism has been implemented for handling the impact. HosTaGe ICS was deployed on a Samsung Galaxy S4 with a rooted configuration to allow emulation of protocols which have ports below 1024. The evaluation period was done for increasing interval periods of 5 minutes until 1 hour.

HosTaGe ICS performance was evaluated using PowerTutor[66], an android app to monitor the power consumed by major systems like the CPU, network interface, display, memory and group this power consumption by the appropriate applications. The primary goal of the application is to be able to track power consumption changes after modifying the application architecture and implementation details.

HosTaGe ICS was monitored with various performance impact constraints such as monitoring with the app minimized with running background, services stopped and with full functionality where the app runs in foreground and has all services enabled. To achieve the desired ICS specific functionality, we require the device to be rooted. This is necessary to achieve port binding to the protocols to be emulated.

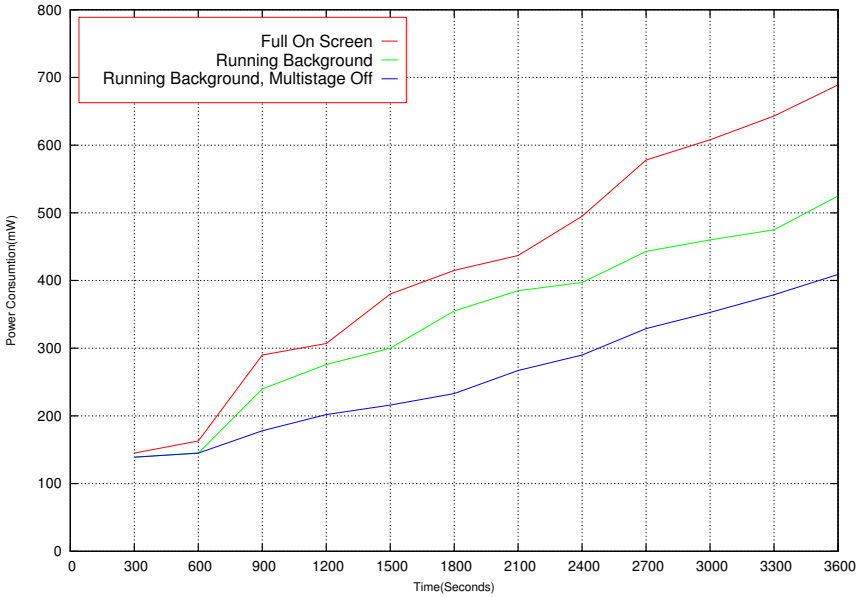

Figure 30: HosTaGe ICS Power Consumption Modes

Figure 30 shows the Power Consumption of HosTaGe ICS at three different modes. We briefly discuss the energy efficiency of HosTaGe ICS during the three modes below.

- **HosTaGe ICS on screen with full functionality:** We evaluate the power consumed by HosTaGe ICS incrementally for a period of one hour. HosTaGe ICS was tested running foreground with multistage Detection service active. The Nuclear Power plant is set as active profile for monitoring. The default screen of the app consists of animation depicting the current state of network.
- **HosTaGe ICS running background:** The power consumption is evaluated with HosTaGe ICS running in background with multistage attack service active. The service is called every 15 minutes. This mode has fair utilization of power as there is no app GUI on foreground.
- **HosTaGe ICS running background with multistage service off:** In this mode, the power consumption evaluation with multistage attack service turned off. This mode provides best utilization of power in comparison with other modes.

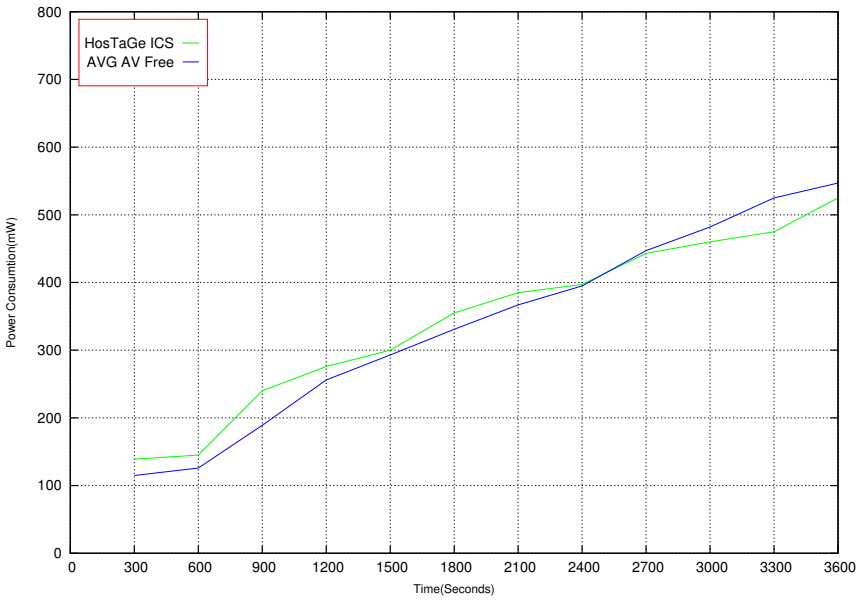

Figure 31: HosTaGe ICS Power Consumption in comparison with AVG

In addition to comparison with different modes, we compare HosTaGe ICS with AVG AV Free app to better evaluate the power consumption. While it was hard to find a suitable app to compare the power utilization of HosTaGe ICS, we chose to compare with AVG on the basis of functionality. As our app is deployed on a device, which would preferably not be used for usual applications, we restrict the comparison to an app which is similar to the resource consumption of HosTaGe ICS. Figure 31 shows the power consumption of HosTaGe ICS with comparison to AVG AV Free. HosTaGe ICS was running in background and minimized with the multistage attack detection service enabled. It is observed that HosTaGe ICS consumes fairly more power. It was also observed that the possible reason for increase in power utilization is due to the varied occurrence of attacks on HosTaGe ICS. The energy consumption depends on the protocol being targeted, the number of connections and packing the response to the request made. Therefore, the utilization changes based on the attacks received.

5.6 Limitations

There were challenges faced during the implementation of some specific modules in HosTaGe ICS. Some design decisions such as rooting of the mobile device, had to be assumed to achieve the functionality of the app as a whole. The Siemens S7 protocol operates over the ISO TSAP transport layer. The regulation was imposed by DHS[11]claiming that TCP's lack of encryption capability. Due to some ambiguties in the design of ISO-TSAP on contrary to regular TCP, the S7 protocol was decided to be implemented as to operate on TCP in HosTaGe ICS. However, this does not deviate the working mechanism of the S7 protocol. The protocol interaction works normally as on with TCP connections but sometimes failed if a connection is attempted from legacy software.

The SNMP protocol is implemented as a basic service because of the absence of support on Android hardware and kernel. This is due to a restriction by the manufacturers enforced on the design. Emulation of the SNMP protocol is very limited and the functionality is not completely achieved. Siemens has its proprietary and customized FTP system. Emulation of FTP as per Siemens could not be achieved due to the lack of documentation. Multistage attacks consider portscans also as attack protocols for inferring the detection. However, signature policies for multistage attacks involving portscans could not be achieved due to development constraints on the Bro language. Nevertheless, a prototype is implemented as a part of an attempt. This implementation is not optimized and does not offer full functionality. We use the HTTP protocol for the Nuclear Power Plant profile. Once the user chooses this profile, the webserver is configured to host a HTML page automatically that looks like a Siemens device portal. The webpage designed is not according to the default portal. This is hard to implement due to limited support of Android towards creation of HTML content dynamically. The webpage generated as of now is indeed capable of attracting traffic, but a feel of the default site would make it even better.

Signature generation for Bro IDS is an important feature of HosTaGe ICS. The signatures generated are stored in the device internal memory and have to be manually deployed to the Bro instance. An alternative would be to automate this process.

5.7 Summary

The evaluation section is briefly divided into experimental setup, detection evaluation, Bro signature generation evaluation, Shodan evasion, performance evaluation and limitations. The experimental setup has two parts and the first provides details about the setup made for the evaluation of HosTaGe ICS with Conpot. The second part provides information on the setup made to evaluate the signatures generated by HosTaGe ICS for Bro NIDS.

The detection evaluation subsection is further divided into analysis of individual protocols, comparison of Conpot and HosTaGe ICS, Stuxnet propagation and lastly inferences and evaluation on multistage attacks. The analysis of individual protocols determine the detection capabilities of HostaGe ICS with respect to the Modbus, S7, HTTP and Telnet protocols implemented in our honeypot. Specific attacks on the Modbus protocol provide an insight into the attacker strategies for exploiting the system. The results observed are listed along with the attack data evaluation for Modbus protocol. HosTaGe ICS was evaluated for detection capabilities in comparison to Conpot. The two honeypots were deployed and made open to the Internet. The attacks received on the honeypots are compared and represented graphically. It is observed that HosTaGe ICS performed better on HTTP protocol and is nearly equal to Conpot on the detection of Modbus and S7 protocols. HosTaGe ICS also implemented the Telnet protocol and the attacks received on Telnet are also represented graphically. Detecting the Stuxnet malware on HosTaGe ICS is discussed further by evaluating it by setting up an environment feasible for it to propagate. The evaluation setup is discussed. The malware packets are captured during propagation from the infected host to the network share simulated by HosTaGe ICS. The file injected into the network share is further validated by HosTaGe by computing a hash and checking it with the VirusTotal database. We were successful in detecting the propagation of Stuxnet into our honeypot. During the evaluation period, we received attacks

[11] http://www.dhs.gov/

with a significant pattern. We implemented a module to specifically detect multistage attacks and received significant results. The results obtained are listed accordingly.

HosTaGe ICS signature generation module generates signatures and policies for the Bro NIDS. Evaluation of the signatures generated are discussed where the imposed attacks by us is merged along with large pcap file that are input to Bro for analysis. Bro detected the imposed attacks by us successfully in both live and offline detection modes.

During the evaluation phase, we observed probes from Shodan search engine. We performed a deeper analysis into these probes and found about Shodans *Honeypot or Not?* tool which detected if the end system is an honeypot. The Conpot instance was detected by this tool to be a honeypot but not HosTaGe ICS. HosTaGe ICS managed to send correct replies to these probes which made the tool infer that the host was a real system. Further, we investigated the impact of these probes on the attack detection capabilities of the honeypots. The results are represented through graphs which indicate the decline in the number of attacks for the Conpot instance but not for HosTaGe ICS. However, to prove the impact, we needed more data for representing through curve fitting techniques and distributions. This area will be focused in the future work.

Lastly, we discussed on performance evaluation which focused on power consumption of HostaGe ICS through three modes. The first mode involved running HosTaGe ICS on screen with the multistage attack detection service running in background. Second, we run HosTaGe ICS in the background along with multistage attack detection service and lastly we only run HosTaGe ICS in the background without multistage attack detection service. We observed that HostaGe ICS in full on-screen consumes more power than the other two modes. We also compared the power consumption of HostaGe ICS with another running application like AVG antivirus and concluded that HostaGe ICS consumes quite more power when compared to the antivirus application and the reason would be because of varied occurence of attacks on HosTaGe ICS.

The limitations sections describes the limitations faced during the implementation of HosTaGe ICS and the challenges which could be optimized in the future work.

6 Conclusion

With almost every device being capable of accessible through the Internet, serious threats follow regarding the security of the devices and enterprise networks. ICS form a backbone for basic services offered to humanity. These systems are also highly critical because of their functionality. Such devices have to be protected. The driving research questions behind this thesis were to efficiently detect ICS specific attacks and detect complex malware like Stuxnet by extending HosTaGe capabilities. During the course of the thesis, some significant results observed led to resolving an emerging research question of identifying multistage attacks, a popular strategy followed by attackers to compromise the end systems. To address this question, related work on securing the ICS has been discussed. ICS environments today offer better managing capabilities to its users by implying newer hardware which are compatible with modern Internet infrastructure. This also poses as a serious threat as it opens up heaps of opportunities for attackers to gain illegal access to the infrastructure.

Existing approaches include deploying NIDS that rely on signatures to detect the attacks and honeypots that are capable of simulating the target environment. As IDS rely on signatures, it is sometimes not possible to detect tailored attacks and related honeypots solutions do not provide enough flexibility to analyse these attacks. They provide lesser simulation capabilities. A flexible,user friendly, robust and interactive mechanism is necessary to tackle the discussed issues.

We introduced HosTaGe ICS a mobile based low interactive honeypot for ICS. HosTaGe ICS extends the functionality of HosTaGe, a low interaction honeypot capable of simulating various profiles and protocols. The core idea behind HosTaGe ICS is to be capable of detecting attacks with respect to ICS environment. HosTaGe ICS successfully emulates ICS specific protocols like Modbus and S7 thereby offering a complete simulation of ICS specific devices. The protocols bound together with other protocols like HTTP, Telnet, SMTP and SMB are successfully capable of simulating the master and slave profiles in an ICS environment. The response mechanisms of individual protocols are designed considering previous exploits and malware in the area of ICS. The services were replicated by studying the behavior of popularly deployed Siemens S7 PLCs and studying their vulnerabilities.

We formally modeled all the attack strategies to arrive at a generic attack model. This also led to the conclusion that formally modeling the attacks and the malware propagation techniques helps in reverse engineering and anatomy of a malware. Inorder to detect complex malware like Stuxnet, we analyze the propagation of Stuxnet and formally modelling it for better understanding and implementation.

Further, HosTaGe ICS extends the detection capabilities of HosTaGe to detect file injection and multistage attacks. File injection detects any malware injected into the system and verifies it with an online virus database. This feature enables HosTaGe ICS to detect complex malware like Stuxnet, which are capable of devastating the ICS infrastructure. HosTaGe ICS also efficiently detects multistage attacks, a crucial strategy followed by adversaries for attacking a specific target. We extended HosTaGe ICS to be more productive by generating signatures for the attacks detected in specific protocols. These signatures can be directly deployed on Bro IDS to check incoming packets for malicious payload. Also, HostaGe ICS is capable of generating policies for multistage attacks detection. These policies help in reducing false positives and also provide front end protection for Intrusion Prevention System (IPS) by dropping packets from blacklisted attackers. The signatures and policies work effectively on both live and offline analysis using Bro. Through these techniques, we were able to filter genuine attacks and reduce the number of false positives hereby improving the detection efficiency.

The results obtained during the course of the thesis demonstrate that HosTaGe ICS is capable of effectively simulating ICS specific profiles and protocols by attracting huge number of malicious traffic. Further, the implementation of an approach to detect Stuxnet propagation through File Injection proved that HosTaGe ICS is capable of detecting such complex malware. The results achieved also led to the identification of multistage attacks and devise a method to detect them in HosTaGe ICS. The attacks also led to the observation of Shodan probes that fetch information from target systems, to determine if an end system is real or a honeypot. Shodans *Honeypot or Not?* service poses a huge threat for Enterprise that use honeypots as a line of defence to monitor their networks. We describe Shodans strategy to determine honeypots by inspecting and analyzing their packets.

HosTaGe ICS like its predecessor is robust to attacks and offers good interaction capabilities. Ideal features like obfuscation, stealth, atmost simulation, low-complexity, reduced resource consumption, user-friendly, flexible make it a good player amongst other honeypots in the same area. We also argue that HosTaGe ICS being a mobile honeypot offers better detection capabilities than other related honeypot services available for ICS. It also supports addition of more protocols and detection of attack strategies. HosTaGe ICS has been developed considering atmost productivity in live scenarios. Since it is deployed on a mobile device, the administration is fairly simple for all levels of users. Advanced features like service settings are also available for advanced users. ICS are vital for human basic needs, said so there is a need to avoid inhibition and take measures to safeguard them.

6.1 Future Work

HosTaGe ICS opens up opportunities for future work and enhancing the system further. Some of the features were decided to be ignored based on some factors like time, development, library support and lack of information. What follows will be a brief overview of various enhancements for HosTaGe ICS. There are a lot of developments in the field of ICS and the technology is continuously evolving to provide better support and functionality. Newer protocols in the Profibus[55] module could be included to simulate ICS environment better. Simulation of newer industrial devices such as PLC with vulnerabilities could be added. HosTaGe ICS currently supports simulation of a single slave device. This could be enhanced to provide support for multiple slave devices connected together through the Modbus. This provides a more complex environment for the attackers hereby gaining more attention.

Soon after the ICS were first targeted by Stuxnet, newer malware were designed to create similar impact. Malware like Flame, Duqu , Duqu2 and Havex have had impacts similar to that of Stuxnet. Mechanisms to detect such malware could be added to HosTaGe ICS as an extended detection capability. The current signature generation module creates signatures for Bro IDS. This functionality can be extended to generate signatures for the Snort IDS[43]. Snort is a widely deployed enterprise grade intrusion detection monitor.

HostaGe ICS emulates different protocols and runs services in the background which utilize more power from the device. This could be optimized by providing different battery utilization modes for user to choose from. This could not only reduce the power consumption but also provide different specific working modes.

HosTaGe ICS could be made more flexible by adding a plugin where newer protocols developed could be integrated through the app itself. This adds a new advantage of quickly updating the existing protocols and also for adding newer protocols into the collection.

We discussed about Shodan probes on our evaluation section. Shodans *Honeypot or Not?* tool poses as huge threat for honeypot researchers and users as the main intention of deploying them is not achieved. The research could be extended further to assess the impact of Shodan probes and also handling them. Though HosTaGe ICS is currently not affected by this tool, ensuring that the probes do not impact its identity is important.

HosTaGe ICS currently supports detecting portscans, file injections and multistage attacks as a part of attack detection strategies. More strategies like Denial Of Service can be implemented to offer better detection capabilities.

List of Figures

List of Tables

Listings

A List of Acronyms

ICS	Industrial Control Systems . . . 1
SCADA	Supervisory Control and Data Acquisition . . . 1
DCS	Distributed Control Systems . . . 7
PLC	Programmable Logic Controllers . . . 1
RTU	Real Time Unit . . . 7
WAN	Wide Area Network . . . 8
EFSM	Extended Finite State Machine . . . 17
PLD	Payload Level Detection . . . 26
SPLD	Single Payload Level Detection . . . 26
MSLD	Multistage Level Detection . . . 26
HMI	Human Machine Interface . . . 7
LAN	Local Area Network . . . 8
IETF	Internet Engineering Task Force . . . 10
RFC	Request For Comments . . . 10
TCP	Transmission Control Protocol . . . 10
ISN	Initial Sequence Number . . . 11
IP	Internet Protocol . . . 10
TLS	Transport Layer Security . . . 13
SSH	Secure Shell . . . 13
HTTPS	Secure Hypertext Transfer Protocol . . . 13
IPSec	Internet Protocol layer security . . . 13
IIS	Internet Information Server . . . 14
FTP	File Transfer Protocol . . . 12
DDOS	Distributed Denial of Service . . . 17
IDS	Intrusion Detection System . . . 1
IPS	Intrusion Prevention System . . . 56
API	Application Programming Interface . . . 35
CIFS	Common Internet File System . . . 35
WinCC	Windows Control Center . . . 30
DLL	Dynamic Link Library . . . 32
SNMP	Simple Network Management Protocol . . . 12
SMTP	Simple Mail Transfer Protocol . . . 12
JSON	Java Synchronous Object Notation . . . 22
OS	Operating System . . . 10
FSM	Finite State Machine . . . 25
RPC	Remote Procedure Call . . . 32
CIDS	Collaborative Intrusion Detection System . . . 17
NIDS	Network Intrusion Detection System . . . 1
TCP	Transmission Control Protocol . . . 10
GUI	Graphical User Interface . . . 21
COTP	Connection Oriented Transport Protocol . . . 11
PDU	Protocol Data Unit . . . 11

VoIP	Voice over Internet Protocol	25
IPv6	Internet Protocol Version 6	13
SMB	Server Message Block	15
SQL	Structured Query Language	15
S7	Siemens S7 protocol	1
HTTP	Hyper Text Transfer Protocol	12
RDBMS	Relational Database Management Systems	18

B Listings

Listing 3: Packets received from Shodan Probes

```
//93.120.27.62 Modbus Packets
2015-07-27 15:55:40,505 New modbus session from 93.120.27.62
    (1651567b-c1df-42da-aa4d-4596ff158256)
2015-07-27 15:55:40,510 New connection from 93.120.27.62:54601.
    (1651567b-c1df-42da-aa4d-4596ff158256)
2015-07-27 15:55:40,521 Modbus traffic from 93.120.27.62: {'function_code': None,
    'slave_id': 0, 'request': '0000000000020011', 'response': '9101'}
    (1651567b-c1df-42da-aa4d-4596ff158256)


//66.240.236.119 S7 Packets
2015-08-27 22:08:27,989 New s7comm session from 66.240.236.119
    (91a441e7-5059-4fb6-8bf4-7ebc5bae4086)
2015-08-27 22:08:27,994 New connection from 66.240.236.119:52994.
    (91a441e7-5059-4fb6-8bf4-7ebc5bae4086)


//71.6.135.131 HTTP Packets
2015-08-16 17:43:26,221 New http session from 71.6.135.131
    (899d404f-c8c3-4a7f-abfa-b7b4c1bb1f4d)
2015-08-16 17:43:26,225 HTTP/1.1 GET request from ('71.6.135.131', 60616): ('/', ['Host:
    130.83.208.166\r\n', 'Accept-Encoding: identity\r\n'], None).
    899d404f-c8c3-4a7f-abfa-b7b4c1bb1f4d
2015-08-16 17:43:26,228 HTTP/1.1 response to ('71.6.135.131', 60616): 302.
    899d404f-c8c3-4a7f-abfa-b7b4c1bb1f4d
2015-08-16 17:43:26,761 HTTP/1.1 GET request from ('71.6.135.131', 60892): ('/robots.txt',
    ['Host: 130.83.208.166\r\n', 'Accept-Encoding: identity\r\n'], None).
    899d404f-c8c3-4a7f-abfa-b7b4c1bb1f4d


//66.240.192.138 HTTP Packets
2015-09-27 16:36:04,341 New http session from 66.240.192.138
    (1651ea2c-fa69-4924-a44b-a4cb10d4e588)
2015-09-27 16:36:04,345 HTTP/1.1 GET request from ('66.240.192.138', 36902): ('/', ['Host:
    130.83.208.166\r\n', 'Accept-Encoding: identity\r\n'], None).
    1651ea2c-fa69-4924-a44b-a4cb10d4e588
2015-09-27 16:36:04,353 HTTP/1.1 response to ('66.240.192.138', 36902): 302.
    1651ea2c-fa69-4924-a44b-a4cb10d4e588
2015-09-27 16:36:04,913 HTTP/1.1 GET request from ('66.240.192.138', 37419): ('/robots.txt',
    ['Host: 130.83.208.166\r\n', 'Accept-Encoding: identity\r\n'], None).
    1651ea2c-fa69-4924-a44b-a4cb10d4e588

//71.6.167.142 S7 Packets
2015-08-12 05:43:35,256 New s7comm session from 71.6.167.142
    (3df36aea-2178-4f31-ae17-68d961bd99ff)
2015-08-12 05:43:35,261 New connection from 71.6.167.142:55412.
    (3df36aea-2178-4f31-ae17-68d961bd99ff)
2015-08-12 05:43:41,437 New connection from 71.6.167.142:56775.
    (3df36aea-2178-4f31-ae17-68d961bd99ff)


//188.138.9.50 S7 Packets
2015-07-19 03:18:56,463 New s7comm session from 188.138.9.50
    (c2f351af-36a5-402b-93cd-28413f036836)
2015-07-19 03:18:56,467 New connection from 188.138.9.50:53854.
    (c2f351af-36a5-402b-93cd-28413f036836)
2015-07-19 03:18:56,666 New connection from 188.138.9.50:53868.
    (c2f351af-36a5-402b-93cd-28413f036836)
```

```
//85.25.103.50 Modbus Packets
2015-08-24 07:02:56,907 New modbus session from 85.25.103.50
    (499b645e-98ce-409c-ba4f-3f42d4240056)
2015-08-24 07:02:56,912 New connection from 85.25.103.50:51086.
    (499b645e-98ce-409c-ba4f-3f42d4240056)
2015-08-24 07:02:56,923 Modbus traffic from 85.25.103.50: {'function_code': None,
    'slave_id': 0, 'request': '0000000000020011', 'response': '9101'}
    (499b645e-98ce-409c-ba4f-3f42d4240056)
```

Listing 4: Bro Policy for Multistage Attack detected by HosTaGE ICS

```
@load base/frameworks/notice

export{
   redef enum Notice::Type += {
      Multistage
   };
}

//Declaring and Initializing variables
global attack_ip = 130.83.208.167;
global attack_port : vector of port = vector(80/tcp,22/tcp);
global attack_count = 0;
global attack_subnet = 130.83.208.0/24;


//On a connection request
event connection_established(c: connection)
{

print fmt ("Initiating.............");

//Check for attack on different protocols as inferred from HosTaGe ICS
for (i in attack_port){

   //Check if it is the first connection from the attacker
   if(attack_count==0){

   if ((c$id$orig_h in attack_subnet) && (c$id$resp_p==attack_port[0]))
       {
   local net_time1: time = network_time();
   print fmt("The first attack ip is ");
   print c$id$orig_h;
   print fmt("and the first port is");
   print c$id$resp_p;
       ++attack_count;
   print attack_count;
   next;

           }

   }
//On the next connection
   else {

       if ((c$id$orig_h in attack_subnet) && (c$id$resp_p == attack_port[1])){
   local net_time2: time = network_time();
   print("The second ip is");
   print c$id$orig_h;
   print fmt("The second port is");
   print c$id$resp_p;

   //Check for time interval
   if(net_time2-net_time1 <= 15 min){
   print fmt ("MULTISTAGE ATTACK!!!");
   }
   NOTICE([$note = Multistage,
               $conn = c,
               $msg = fmt("Multistage Attack! from %s",c$id$orig_h)]);
   attack_count = 0;
```

```
                }

            }

            }
        }
```

Listing 5: Nmap Modbus Discovery Script

```
ocal bin = require "bin"
local comm = require "comm"
local nmap = require "nmap"
local shortport = require "shortport"
local stdnse = require "stdnse"
local string = require "string"
local table = require "table"

description = [[
Enumerates SCADA Modbus slave ids (sids) and collects their device information.

Modbus is one of the popular SCADA protocols. This script does Modbus device
information disclosure. It tries to find legal sids (slave ids) of Modbus
devices and to get additional information about the vendor and firmware. This
script is improvement of modscan python utility written by Mark Bristow.

Information about MODBUS protocol and security issues:
* MODBUS application protocol specification:
    http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf
* Defcon 16 Modscan presentation:
    https://www.defcon.org/images/defcon-16/dc16-presentations/defcon-16-bristow.pdf
* Modscan utility is hosted at google code: http://code.google.com/p/modscan/
]]

---
-- @usage
-- nmap --script modbus-discover.nse --script-args='modbus-discover.aggressive=true' -p 502 <host>
--
-- @args aggressive - boolean value defines find all or just first sid
--
-- @output
-- PORT   STATE SERVICE
-- 502/tcp open modbus
-- | modbus-discover:
-- |  sid 0x64:
-- |    Slave ID data: \xFA\xFFPM710PowerMeter
-- |    Device identification: Schneider Electric PM710 v03.110
-- |  sid 0x96:
-- |_   error: GATEWAY TARGET DEVICE FAILED TO RESPONSE
--
-- @xmloutput
-- <table key="sid 0x64">
--   <elem key="Slave ID data">\xFA\xFFPM710PowerMeter</elem>
--   <elem key="Device identification">Schneider Electric PM710 v03.110</elem>
-- </table>
-- <table key="sid 0x96">
--   <elem key="error">GATEWAY TARGET DEVICE FAILED TO RESPONSE</elem>
-- </table>

-- Version 0.2 - /12.12.10/ - script cleanup
-- Version 0.3 - /13.12.10/ - several bugfixes

author = "Alexander Rudakov"
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
categories = {"discovery", "intrusive"}


portrule = shortport.port_or_service(502, "modbus")

local form_rsid = function(sid, functionId, data)
  local payload_len = 2
  if ( #data > 0 ) then
```

```
    payload_len = payload_len + #data
  end
  return "\0\0\0\0\0" .. bin.pack('CCC', payload_len, sid, functionId) .. data
end

discover_device_id_recursive = function(host, port, sid, start_id, objects_table)
  local rsid = form_rsid(sid, 0x2B, "\x0E\x01" .. bin.pack('C', start_id))
  local status, result = comm.exchange(host, port, rsid)
  if ( status and (#result >= 8)) then
    local ret_code = string.byte(result, 8)
    if ( ret_code == 0x2B and #result >= 15 ) then
      local more_follows = string.byte(result, 12)
      local next_object_id = string.byte(result, 13)
      local number_of_objects = string.byte(result, 14)
      stdnse.debug1("more = 0x%x, next_id = 0x%x, obj_number = 0x%x", more_follows,
          next_object_id, number_of_objects)
      local offset = 15
      for i = start_id, (number_of_objects - 1) do
        local object_id = string.byte(result, offset)
        local object_len = string.byte(result, offset + 1)
        -- error data format --
        if object_len == nil then break end
        local object_value = string.sub(result, offset + 2, offset + 1 + object_len)
        stdnse.debug1("Object id = 0x%x, value = %s", object_id, object_value)
        table.insert(objects_table, object_id + 1, object_value)
        offset = offset + 2 + object_len
      end
      if ( more_follows == 0xFF and next_object_id ~= 0x00 ) then
        stdnse.debug1("Has more objects")
        return discover_device_id_recursive(host, port, sid, next_object_id, objects_table)
      end
    end
  end
  return objects_table
end

local discover_device_id = function(host, port, sid)
  return discover_device_id_recursive(host, port, sid, 0x0, {})
end

local extract_slave_id = function(response)
  local byte_count = string.byte(response, 9)
  if ( byte_count == nil or byte_count == 0) then return nil end
  local offset, slave_id = bin.unpack("A"..byte_count, response, 10)
  return slave_id
end

modbus_exception_codes = {
  [1] = "ILLEGAL FUNCTION",
  [2] = "ILLEGAL DATA ADDRESS",
  [3] = "ILLEGAL DATA VALUE",
  [4] = "SLAVE DEVICE FAILURE",
  [5] = "ACKNOWLEDGE",
  [6] = "SLAVE DEVICE BUSY",
  [8] = "MEMORY PARITY ERROR",
  [10] = "GATEWAY PATH UNAVAILABLE",
  [11] = "GATEWAY TARGET DEVICE FAILED TO RESPOND"
}

action = function(host, port)
  -- If false, stop after first sid.
  local aggressive = stdnse.get_script_args('modbus-discover.aggressive')

  local opts = {request_timeout=2000}
```

```
  local results = stdnse.output_table()

  for sid = 1, 246 do
    stdnse.debug3("Sending command with sid = %d", sid)
    local rsid = form_rsid(sid, 0x11, "")

    local status, result = comm.exchange(host, port, rsid, opts)
    if ( status and (#result >= 8) ) then
      local ret_code = string.byte(result, 8)
      if ( ret_code == (0x11) or ret_code == (0x11 + 128) ) then
        local sid_table = stdnse.output_table()
        if ret_code == (0x11) then
          local slave_id = extract_slave_id(result)
          sid_table["Slave ID data"] = slave_id or "<unknown>"
        elseif ret_code == (0x11 + 128) then
          local exception_code = string.byte(result, 9)
          local exception_string = modbus_exception_codes[exception_code]
          if ( exception_string == nil ) then
            exception_string = ("Unknown exception (0x%x)"):format(exception_code)
          end
          sid_table["error"] = exception_string
        end

        local device_table = discover_device_id(host, port, sid)
        if ( #device_table > 0 ) then
          sid_table["Device identification"] = table.concat(device_table, " ")
        end
        if ( #sid_table > 0 ) then
          results[("sid 0x%x"):format(sid)] = sid_table
        end
        if ( not aggressive ) then break end
      end
    end
  end

  if ( #results > 0 ) then
    port.state = "open"
    port.version.name = "modbus"
    nmap.set_port_version(host, port)
    return results
  end
end
```

Listing 6: Metasploit Modbus Detect Script

```
##
# This module requires Metasploit: http://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

require 'msf/core'

class Metasploit3 < Msf::Auxiliary

  include Msf::Exploit::Remote::Tcp
  include Msf::Auxiliary::Scanner

  def initialize
    super(
      'Name'        => 'Modbus Version Scanner',
      'Description' => %q{
          This module detects the Modbus service, tested on a SAIA PCD1.M2 system.
        Modbus is a clear text protocol used in common SCADA systems, developed
        originally as a serial-line (RS232) async protocol, and later transformed to IP,
        which is called ModbusTCP.
      },
      'References' =>
        [
          [ 'URL', 'http://www.saia-pcd.com/en/products/plc/pcd-overview/Pages/pcd1-m2.aspx' ],
          [ 'URL', 'http://en.wikipedia.org/wiki/Modbus:TCP' ]
        ],
      'Author'      => [ 'EsMnemon <esm[at]mnemonic.no>' ],
      'DisclosureDate' => 'Nov 1 2011',
      'License'     => MSF_LICENSE
      )

    register_options(
      [
        Opt::RPORT(502),
        OptInt.new('UNIT_ID', [true, "ModBus Unit Identifier, 1..255, most often 1 ", 1]),
        OptInt.new('TIMEOUT', [true, 'Timeout for the network probe', 10])
      ], self.class)
  end

  def run_host(ip)
    # read input register=func:04, register 1
    sploit="\x21\x00\x00\x00\x00\x06\x01\x04\x00\x01\x00\x00"
    sploit[6] = [datastore['UNIT_ID']].pack("C")
    connect()
    sock.put(sploit)
    data = sock.get_once

    # Theory: When sending a modbus request of some sort, the endpoint will return
    # with at least the same transaction-id, and protocol-id
    if data
      if data[0,4] == "\x21\x00\x00\x00"
        print_good("#{ip}:#{rport} - MODBUS - received correct MODBUS/TCP header (unit-ID:
            #{datastore['UNIT_ID']})")
      else
        print_error("#{ip}:#{rport} - MODBUS - received incorrect data #{data[0,4].inspect} (not
            modbus/tcp?)")
      end
    else
      vprint_status("#{ip}:#{rport} - MODBUS - did not receive data.")
    end
```

```
      disconnect()
    end
  end
```

Listing 7: Metasploit script of Writing into Modbus Registers

```
##
# This module requires Metasploit: http://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

require 'msf/core'

class Metasploit3 < Msf::Auxiliary

  include Msf::Exploit::Remote::Tcp

  def initialize(info = {})
    super(update_info(info,
      'Name'        => 'Modbus Client Utility',
      'Description' => %q{
        This module allows reading and writing data to a PLC using the Modbus protocol.
        This module is based on the 'modiconstop.rb' Basecamp module from DigitalBond,
        as well as the mbtget perl script.
      },
      'Author'       =>
        [
          'EsMnemon <esm[at]mnemonic.no>', # original write-only module
          'Arnaud SOULLIE <arnaud.soullie[at]solucom.fr>' # new code that allows read/write
        ],
      'License'      => MSF_LICENSE,
      'Actions'      =>
        [
          ['READ_COIL', { 'Description' => 'Read one bit from a coil' } ],
          ['WRITE_COIL', { 'Description' => 'Write one bit to a coil' } ],
          ['READ_REGISTER', { 'Description' => 'Read one word from a register' } ],
          ['WRITE_REGISTER', { 'Description' => 'Write one word to a register' } ]
        ],
      'DefaultAction' => 'READ_REGISTER'
      ))

    register_options(
      [
        Opt::RPORT(502),
        OptInt.new('DATA', [false, "Data to write (WRITE_COIL and WRITE_REGISTER modes only)"]),
        OptInt.new('DATA_ADDRESS', [true, "Modbus data address"]),
        OptInt.new('UNIT_NUMBER', [false, "Modbus unit number", 1]),
      ], self.class)

  end

  # a wrapper just to be sure we increment the counter
  def send_frame(payload)
    sock.put(payload)
    @modbus_counter += 1
    sock.get_once(-1, sock.def_read_timeout)
  end

  def make_payload(payload)
    packet_data = [@modbus_counter].pack("n")
    packet_data += "\x00\x00\x00" #dunno what these are
    packet_data += [payload.size].pack("c") # size byte
    packet_data += payload

    packet_data
  end
```

```
def make_read_payload
  payload = [datastore['UNIT_NUMBER']].pack("c")
  payload += [@function_code].pack("c")
  payload += [datastore['DATA_ADDRESS']].pack("n")
  payload += [1].pack("n")
  make_payload(payload)
end

def make_write_coil_payload(data)
  payload = [datastore['UNIT_NUMBER']].pack("c")
  payload += [@function_code].pack("c")
  payload += [datastore['DATA_ADDRESS']].pack("n")
  payload += [data].pack("c")
  payload += "\x00"

  packet_data = make_payload(payload)

  packet_data
end

def make_write_register_payload(data)
  payload = [datastore['UNIT_NUMBER']].pack("c")
  payload += [@function_code].pack("c")
  payload += [datastore['DATA_ADDRESS']].pack("n")
  payload += [data].pack("n")

  make_payload(payload)
end

def handle_error(response)
  case response.reverse.unpack("c")[0].to_i
  when 1
    print_error("Error : ILLEGAL FUNCTION")
  when 2
    print_error("Error : ILLEGAL DATA ADDRESS")
  when 3
    print_error("Error : ILLEGAL DATA VALUE")
  when 4
    print_error("Error : SLAVE DEVICE FAILURE")
  when 6
    print_error("Error : SLAVE DEVICE BUSY")
  else
    print_error("Unknown error")
  end
  return
end

def read_coil
  @function_code = 0x1
  print_status("Sending READ COIL...")
  response = send_frame(make_read_payload)
  if response.nil?
    print_error("No answer for the READ COIL")
    return
  elsif response.unpack("C*")[7] == (0x80 | @function_code)
    handle_error(response)
  elsif response.unpack("C*")[7] == @function_code
    value = response[9].unpack("c")[0]
    print_good("Coil value at address #{datastore['DATA_ADDRESS']} : #{value}")
  else
    print_error("Unknown answer")
  end
end
```

```
def read_register
  @function_code = 3
  print_status("Sending READ REGISTER...")
  response = send_frame(make_read_payload)
  if response.nil?
    print_error("No answer for the READ REGISTER")
  elsif response.unpack("C*")[7] == (0x80 | @function_code)
    handle_error(response)
  elsif response.unpack("C*")[7] == @function_code
    value = response[9..10].unpack("n")[0]
    print_good("Register value at address #{datastore['DATA_ADDRESS']} : #{value}")
  else
    print_error("Unknown answer")
  end
end

def write_coil
  @function_code = 5
  if datastore['DATA'] == 0
    data = 0
  elsif datastore['DATA'] == 1
    data = 255
  else
    print_error("Data value must be 0 or 1 in WRITE_COIL mode")
    return
  end
  print_status("Sending WRITE COIL...")
  response = send_frame(make_write_coil_payload(data))
  if response.nil?
    print_error("No answer for the WRITE COIL")
  elsif response.unpack("C*")[7] == (0x80 | @function_code)
    handle_error(response)
  elsif response.unpack("C*")[7] == @function_code
    print_good("Value #{datastore['DATA']} successfully written at coil address
        #{datastore['DATA_ADDRESS']}")
  else
    print_error("Unknown answer")
  end
end

def write_register
  @function_code = 6
  if datastore['DATA'] < 0 || datastore['DATA'] > 65535
    print_error("Data to write must be an integer between 0 and 65535 in WRITE_REGISTER mode")
    return
  end
  print_status("Sending WRITE REGISTER...")
  response = send_frame(make_write_register_payload(datastore['DATA']))
  if response.nil?
    print_error("No answer for the WRITE REGISTER")
  elsif response.unpack("C*")[7] == (0x80 | @function_code)
    handle_error(response)
  elsif response.unpack("C*")[7] == @function_code
    print_good("Value #{datastore['DATA']} successfully written at registry address
        #{datastore['DATA_ADDRESS']}")
  else
    print_error("Unknown answer")
  end
end

def run
  @modbus_counter = 0x0000 # used for modbus frames
  connect
  case action.name
```

```
      when "READ_COIL"
        read_coil
      when "READ_REGISTER"
        read_register
      when "WRITE_COIL"
        write_coil
      when "WRITE_REGISTER"
        write_register
      else
        print_error("Invalid ACTION")
      end
      disconnect
    end
  end
```

References

[1] Vendor-supplied backdoor passwords - a continuing vulnerability, 2003.

[2] Dillon Beresford. Exploiting siemens simatic s7 plcs. *Black Hat USA*, 2011.

[3] Roland Bodenheim, Jonathan Butts, Stephen Dunlap, and Barry Mullins. Evaluation of the ability of the shodan search engine to identify internet-facing industrial control devices. *International Journal of Critical Infrastructure Protection*, 7(2):114–123, 2014.

[4] Roland C Bodenheim. Impact of the shodan computer search engine on internet-facing industrial control system devices. Technical report, DTIC Document, 2014.

[5] Stuart A Boyer. *SCADA: supervisory control and data acquisition*. International Society of Automation, 2009.

[6] Anatolii Grigorevich Butkovskiy. Distributed control systems. 1969.

[7] Eric J Byres, Matthew Franz, and Darrin Miller. The use of attack trees in assessing vulnerabilities in scada systems. In *Proceedings of the International Infrastructure Survivability Workshop*. Citeseer, 2004.

[8] Thomas M Chen and Saeed Abu-Nimeh. Lessons from stuxnet. *Computer*, 44(4):91–93, 2011.

[9] DELL. Dell security annual threat report 2015. 2015.

[10] Jules Pagna Disso, Kevin Jones, and Steven Bailey. A plausible solution to scada security honeypot systems. In *BWCCA*, pages 443–448. IEEE, 2013.

[11] Zakir Durumeric, James Kasten, David Adrian, J Alex Halderman, Michael Bailey, Frank Li, Nicolas Weaver, Johanna Amann, Jethro Beekman, Mathias Payer, et al. The matter of heartbleed. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, pages 475–488. ACM, 2014.

[12] Zakir Durumeric, Eric Wustrow, and J Alex Halderman. Zmap: Fast internet-wide scanning and its security applications. In *Usenix Security*, pages 605–620, 2013.

[13] Carlos Garcia Cordero Emmanouil Vasilomanolakis, Shreyas Srinivasa and Max Mühlhäuser. Multi-stage attack detection and signature generation with ics honeypots. *Passive and Active Measurement Conference*, 2016.

[14] Nicolas Falliere, Liam O Murchu, and Eric Chien. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response*, 5, 2011.

[15] Marcelo Fantinato and Mario Jino. Applying extended finite state machines in software testing of interactive systems. In JoaquimA. Jorge, Nuno Jardim Nunes, and JoÃčo FalcÃčo e Cunha, editors, *Interactive Systems. Design, Specification, and Verification*, volume 2844 of *Lecture Notes in Computer Science*, pages 34–45. Springer Berlin Heidelberg, 2003.

[16] A. Galante, A. Kokos, and S. Zanero. Bluebat: Towards practical bluetooth honeypots. In *Communications, 2009. ICC '09. IEEE International Conference on*, pages 1–6, June 2009.

[17] Dao gang Peng, Hao Zhang, Li Yang, and Hui Li. Design and realization of modbus protocol based on embedded linux system. In *Embedded Software and Systems Symposia, 2008. ICESS Symposia '08. International Conference on*, pages 275–280, July 2008.

[18] Pedro Garcia-Teodoro, J Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1):18–28, 2009.

[19] D Goldman. Shodan: The scariest search engine on the internet. *CNN Money*, 2013.

[20] Thorsten Holz and N Provos. Virtual honeypots: from botnet tracking to intrusion detection, 2008.

[21] Nicholas Ianelli and Aaron Hackworth. Botnets as a vehicle for online crime. *FORENSIC COMPUTER SCIENCE IJoFCS*, page 19, 2005.

[22] Vinay M Igure, Sean A Laughter, and Ronald D Williams. Security issues in scada networks. *Computers & Security*, 25(7):498–506, 2006.

[23] V Jacobsen, Craig Leres, and Steven McCanne. Tcpdump/libpcap, 2005.

[24] Kristopher Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Technical report, DTIC Document, 1999.

[25] Amit Kleinmann and Avishai Wool. Accurate modeling of the siemens s7 scada protocol for intrusion detection and digital forensic. *Journal of Digital Forensics, Security and Law*, 9(2):37–50, 2014.

[26] AV Kolesnichenko, Pieter-Tjerk de Boer, AKI Remke, Emmanuele Zambon, and BRHM Haverkort. Is quantitative analysis of stuxnet possible? 2011.

[27] Christian Kreibich and Jon Crowcroft. Honeycomb: creating intrusion detection signatures using honeypots. *ACM SIGCOMM Computer Communication Review*, 34(1):51–56, 2004.

[28] Ralph Langner. Stuxnet: Dissecting a cyberwarfare weapon. pages 49–51, May 2011.

[29] Jon Larimer. Beyond autorun: Exploiting vulnerabilities with removable storage. *Black Hat*, 2011.

[30] Feng-Li Lian, James R. Moyne, and D.M. Tilbury. Performance evaluation of control networks: Ethernet, controlnet, and devicenet. *Control Systems, IEEE*, 21(1):66–83, Feb 2001.

[31] Steffen Liebergeld, Matthias Lange, and Ravishankar Borgaonkar. Cellpot: A concept for next generation cellular network honeypots. 2014.

[32] Steffen Liebergeld, Matthias Lange, and Collin Mulliner. Nomadic honeypots: A novel concept for smartphone honeypots, 2013.

[33] Gordon Fyodor Lyon. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure, 2009.

[34] Aleksandr Matrosov, Eugene Rodionov, David Harley, and Juraj Malcho. Stuxnet under the microscope. *ESET LLC (September 2010)*, 2010.

[35] David Maynor. *Metasploit toolkit for penetration testing, exploit development, and vulnerability research*. Elsevier, 2011.

[36] IDA Modbus. Modbus application protocol specification v1. 1a. *North Grafton, Massachusetts (www. modbus. org/specs. php)*, 2004.

[37] Collin Mulliner, Steffen Liebergeld, and Matthias Lange. Poster: Honeydroid-creating a smartphone honeypot. *IEEE Symposium on Security and Privacy (S&P)*, 2011.

[38] Liam O Murchu. Stuxnet-infecting industrial control systems. In *Virus Bulletin 2010 Conference, Vancouver, BC. http://www. virusbtn. com/pdf/conference_slides/2010/OMurchu-VB2010. pdf (accessed Dec 15, 2010)*, 2010.

[39] Vern Paxson. Bro: a System for Detecting Network Intruders in Real-Time. *Computer Networks*, 31(23-24):2435–2463, 1999.

[40] Y Pouffary and A Young. Iso transport service on top of tcp (itot). Technical report, 1997.

[41] Niels Provos. Honeyd-a virtual honeypot daemon. In *10th DFN-CERT Workshop, Hamburg, Germany*, volume 2, page 4, 2003.

[42] Niels Provos. A virtual honeypot framework. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 1–1, Berkeley, CA, USA, 2004. USENIX Association.

[43] Martin Roesch et al. Snort: Lightweight intrusion detection for networks. In *LISA*, volume 99, pages 229–238, 1999.

[44] Marshall T Rose and Dwight E Cass. Iso transport service on top of the tcp version: 3. 1987.

[45] S. Rubin, S. Jha, and B.P. Miller. Automatic generation and analysis of nids attacks. In *Computer Security Applications Conference, 2004. 20th Annual*, pages 28–38, Dec 2004.

[46] Bruce Schneier. Attack trees. *Dr. Dobbs journal*, 24(12):21–29, 1999.

[47] Hemant Sengar, Duminda Wijesekera, Haining Wang, and Sushil Jajodia. Voip intrusion detection through interacting protocol state machines. In *In DSN*, pages 393–402. IEEE Computer Society, 2006.

[48] Lance Spitzner. Honeypots: Catching the insider threat. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pages 170–179. IEEE, 2003.

[49] Keith Stouffer, Joe Falco, and Karen Scarfone. Guide to industrial control systems (ics) security. *NIST special publication*, pages 800–82, 2011.

[50] Andy Swales. Open modbus/tcp specification. *Schneider Electric*, 29, 1999.

[51] Urjita Thakar, Sudarshan Varma, and AK Ramani. Honeyanalyzer–analysis and extraction of intrusion detection patterns & signatures using honeypot. In *Proceedings of the Second International Conference on Innovations in Information Technology*, 2005.

[52] J.-P. Thomesse. Fieldbus technology in industrial automation. *Proceedings of the IEEE*, 93(6):1073–1101, June 2005.

[53] Tim Thornburgh. Social engineering: The "dark art". In *Proceedings of the 1st Annual Conference on Information Security Curriculum Development*, InfoSecCD '04, pages 133–135, New York, NY, USA, 2004. ACM.

[54] Virus Total. Virustotal-free online virus, malware and url scanner, 2012.

[55] Eduardo Tovar and Francisco Vasques. Real-time fieldbus communications using profibus networks. *Industrial Electronics, IEEE Transactions on*, 46(6):1241–1251, 1999.

[56] Emmanouil Vasilomanolakis, Shankar Karuppayah, Mathias Fischer, Max Mühlhäuser, Mihai Plasoianu, Lars Pandikow, and Wulf Pfeiffer. This network is infected: Hostage - a low-interaction honeypot for mobile devices. In *Proceedings of the Third ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, pages 43–48. ACM, 2013.

[57] Emmanouil Vasilomanolakis, Shankar Karuppayah, Max Mühlhäuser, and Mathias Fischer. Hostage: A mobile honeypot for collaborative defense. In *Proceedings of the 7th International Conference on Security of Information and Networks*, pages 330:330–330:333. ACM, 2014.

[58] Emmanouil Vasilomanolakis, Shreyas Srinivasa, and Max Mühlhäuser. Did you really hack a nuclear power plant? an industrial control mobile honeypot. In *IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2015. to appear.

[59] Nikos Virvilis and Dimitris Gritzalis. The big four-what we did wrong in advanced persistent threat detection? In *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*, pages 248–254. IEEE, 2013.

[60] F Wagner. Moore or mealy model. *States works, Technical notes httpM/stateworkscom*, 2005.

[61] Matthias Wählisch, Sebastian Trapp, Christian Keil, Jochen Schönfelder, Jochen Schiller, et al. First insights from a mobile honeypot. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 305–306. ACM, 2012.

[62] John W. Webb and Ronald A. Reis. *Programmable Logic Controllers: Principles and Applications*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 4th edition, 1998.

[63] Kyle Wilhoit. Who is really attacking your ics equipment? *Trend Micro*, 2013.

[64] Michael Winn, Mason Rice, Stephen Dunlap, Juan Lopez, and Barry Mullins. Constructing cost-effective and targetable industrial control system honeypots for production networks. *International Journal of Critical Infrastructure Protection*, 2015.

[65] Handong Wu, Stephen Schwab, and Robert Lom Peckham. Signature based network intrusion detection system and method, September 9 2008. US Patent 7,424,744.

[66] Z Yang. Powertutor-a power monitor for android-based mobile platforms. *EECS, University of Michigan, retrieved September*, 2, 2012.

[67] Nong Ye, Yebin Zhang, and Connie M Borror. Robustness of the markov-chain model for cyber-attack detection. *Reliability, IEEE Transactions on*, 53(1):116–123, 2004.

[68] Kim Zetter. How digital detectives deciphered stuxnet, the most menacing malware in history. *Wired Magazine*, 11:1–8, 2011.

[69] Chenfeng Vincent Zhou, Christopher Leckie, and Shanika Karunasekera. A survey of coordinated attacks and collaborative intrusion detection. *Computers & Security*, 29(1):124–140, 2010.