Multi-stage Attack Detection and Signature Generation with ICS Honeypots

Emmanouil Vasilomanolakis, Shreyas Srinivasa, Carlos Garcia Cordero, Max Mühlhäuser
Center for Advanced Security Research Darmstadt (CASED)
Telecooperation Lab, Technische Universität Darmstadt
{vasilomano, garcia, max}@tk.tu-darmstadt.de, {shreyas.srinivasa}@stud.tu-darmstadt.de

***Abstract*—New attack surfaces are emerging with the rise of Industrial Control System (ICS) devices exposed on the Internet. ICS devices must be protected in a holistic and efficient manner; especially when these are supporting critical infrastructure. Taking this issue into account, cyber-security research is recently being focused on providing early detection and warning mechanisms for ICSs. In this paper we present a novel honeypot capable of detecting multi-stage attacks targeting ICS networks. Upon detecting a multi-stage attack, our honeypot can generate signatures so that misuse Intrusion Detection Systems (IDSs) can subsequently thwart attacks of the same type. Our experimental results indicate that our honeypot and the signatures it generates provide good detection accuracy and that the Bro IDS can successfully use the signatures to prevent future attacks.**

I. Introduction

The dependence of our society to IT networks is constantly increasing. In addition, the emergence and interconnectivity of Industrial Control Systems (ICSs) creates a plethora of security challenges that need to be addressed. Recent highly sophisticated and tailored attacks against these systems, e.g., Stuxnet [5] and Flame [16], are already here to highlight this fact.

Intrusion Detection Systems (IDSs) are nowadays considered a mandatory line of defense for the protection of critical networks [15]. The majority of IDSs in real world networks are passively monitoring and performing signature-based detection. As such, the accuracy of these systems is highly influenced by the quality of the utilized signatures. Security mechanisms that exhibit more active monitoring can act as an additional line of defense by complementing existing systems. In this paper, we introduce a honeypot capable of providing such an active monitoring approach that can generate signatures of detected attacks on-the-fly.

Honeypots are systems whose only value is to be probed, attacked and compromised [11]. Their purpose is to attract malicious users, study their activities and, at the same time, reduce the attack surface. It is important to note that since honeypots do not feature any other purpose, by definition, any interaction with them is considered an attack. Thus, they do not exhibit false positives, i.e., all incoming traffic is considered malicious.

In this paper, we extend our previous work in the area of honeypots [13] with the focus of detecting sophisticated attacks in ICS networks. We specialize in the identification of attacks that originate from the same entity and make use of multiple protocols. We argue that these attacks can realistically describe cases of Advanced Persistent Threats (APTs) and targeted real world attacks. Our honeypot is able to extract signatures from detected attacks which can subsequently be imported into the Bro IDS [7]. We compare our solution with another honeypot from the related work, discuss the possibility of evading the honeypots and evaluate the quality and applicability of the generated signatures.

The remainder of this paper is as follows. In Section II, we present the related work in the areas of honeypots and alert generation. Section III provides an overview of the system as a whole; we give a formal model for the detection mechanisms and describe the signature generation process. Section IV presents and discusses the results of our evaluation. Lastly, Section V concludes this paper.

II. Related Work

In this section we discuss related work in the areas of honeypots and signature generation with an emphasis on ICS networks.

In our previous work [13] we presented *HosTaGe*, an open-source low-interaction mobile honeypot. The idea was to introduce lightweight portable honeypots for mobile devices that aim at detecting malicious devices in wireless networks. *HosTaGe* supports the identification of attacks in all major protocols, e.g., HTTP, SMB, Telnet, FTP, MySQL, SIP and SSH. We enhance our system in a twofold manner: first, we add support for ICS protocols and, second, we focus on detecting attacks triggered from the same entity and manifested in multiple protocols. In addition, we formalize such attacks via state machines and generate corresponding signatures for them to be used by IDSs.

In their recent work, Minn et al. [6] proposed *IoTPOT*, a honeypot that emphasizes on IoT devices by emulating the Telnet protocol. Their results show an increase of attacks on Telnet that target Internet of Things (IoT) devices, which also corresponds to our previous work's findings [14]. However, their focus is limited only into Telnet-based attacks (and different CPU architectures). Conpot [9] is another low interaction honeypot that focuses on emulating server side ICSs. Conpot was one of the first honeypots detecting ICS

978-1-5090-0223-8/16/$31.00 © 2016 IEEE

network attacks and is considered the state-of-the-art in this area. Conpot has a few disadvantages however; first, it does not support the emulation of Telnet, and the information that is logged, e.g., for Modbus attacks, is not always sufficient for an in-depth analysis of an attack. Moreover, as we will demonstrate in Section IV, Conpot can be identified as a honeypot by specialized tools. Nevertheless, as this honeypot is, with respect to its ICS support, the closest to ours, we compare these two systems in Section IV.

In the recent years there have been proposals in the area of signature generation via the utilization of honeypots [4], [12], [3]. For instance, HoneyComb [4] is a system that makes use of the *honeyd* honeypot [8] to generate alert signatures. It has the advantage of only using honeypot network traffic and thus reducing false positives. However, as a result of utilizing honeyd, only high level TCP or UDP information can be examined, making it unsuitable for payload-level analysis. Additionally, neither HoneyComb, nor any honeypot, is able to identify multi-stage attacks.

Finally, with regards to generic signature generation, in [10], Sengar et al. proposed a novel intrusion detection mechanism for Voice over IP (VoIP) that utilizes protocol-level state machines. Our approach is inspired by this work and builds on top of it. We move beyond VoIP and focus on attacks that target ICS networks. Our finite state machines are higher level as we focus on the detection of multi-stage (and thus multi-protocol) malicious activity. Furthermore, we utilize this state machine-based detection mechanism in our honeypot only as the initial step for improving intrusion detection. Our aim is to utilize the results gathered from Extended Finite State Machines (EFSMs) to automatically generate signatures and security policies that can be deployed in existing state-of-the-art IDSs, i.e., Bro, rather than proposing a new IDS.

III. *HosTaGe* ICS Honeypot

In this section, we first give an overview of our system, discuss the ICS-specific protocols simulated by *HosTaGe* followed by the formal model used to describe the detection mechanisms we employ. Lastly, we discuss the signature generation of our proposal.

HosTaGe is a lightweight low-interaction honeypot for mobile devices. It is an open source project, written in Java, consisting of more than $15,000$ lines of code. As a low-interaction honeypot, *HosTaGe* simulates protocols in the TCP/IP stack, emulating many typical protocols utilized by adversaries, e.g., HTTPS, FTP, MySQL, SIP, SSH, etc. In order to support the protection of ICS networks, we modified, included and rewrote protocol detection mechanisms to follow the model we describe in this section. Moreover, we argue that a honeypot can be a supplemental security mechanism. Hence, we support such a task by generating signatures that can be automatically used in IDSs. A practical usage scenario can be the following. *HosTaGe* can be placed in the perimeter of a corporate network that, internally, includes ICSs. In the case of an adversary who aims at attacking the internal network, it is highly possible that the honeypot will be triggered during either the reconnaissance phase or the actual attack. If this is the case, *HosTaGe* will not only notify the network administrators but will also produce a signature for the attack and forward it to the internal IDSs.

A. ICS Specific Protocol Emulation

We extend the protocol emulation capabilities of *HosTaGe* by adding (or improving) support for protocols that are utilized in ICSs, i.e., *Modbus, S7, SNMP, HTTP, Telnet, SMB* and *SMTP*. In the following we briefly discuss these protocols and how they are emulated in our honeypot.

Among the various profiles supported by ICS devices, *Modbus* acts as a backbone for device communication. Modbus is a serial communications protocol initially published by Modicon for usage in its Programmable Logic Controllers (PLCs). It is now considered as the standard that connects industrial devices, and in particular it establishes communication between master and slave devices. As Modbus can also be used in a wireless mode for Remote Terminal Unit (RTU)-based communication, it is widely used in various ICS networks, e.g., nuclear power plants, gas and oil distribution, for communication between the PLCs.

a) Modbus: 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. Modeling the correct behavior of Modbus requires simulating the responses of the Modbus system for a command issued by a Modbus master device. *HosTaGe* simulates the Modbus communication protocols by implementing the request/reply mechanism. The Siemens PLCs also have registers as memory units, which store sensor data and application logic. The registers have the memory mapped to blocks at the register level. Through the Modbus protocol, the data in registers can be accessed and set. *HosTaGe* supports this request/reply paradigm for the memory mapped registers through dedicated data structures. Furthermore, the honeypot also supports the instruction set of Modbus, which is implemented as function codes. The instruction set involves registers' R/W, as well as the Modbus service diagnostics. Furthermore, in our Modbus implementation we aim on avoiding detection from well-known reconnaissance tools. In more details, the Nmap port scanning tool, with special scripts, is able to perform Modbus-specific detection and reconnaissance. Similarly, the Metasploit exploit toolkit also provides support for the detection and exploitation of Modbus. *HosTaGe* is able to respond correctly in all these cases. In Section IV-C, we discuss the importance of this in the context of honeypot evasion attacks.

b) S7: The *S7* protocol (S7 Communication) is a Siemens proprietary protocol utilized in PLCs of the Siemens S7-300/400 families. It is used for PLC programming, exchanging data between PLCs, accessing PLC data from SCADA systems and for diagnostic purposes. The protocol forms as a base for accessing the registers for R/W operations

[1] For simplicity we include coils in the term registers, even though strictly speaking they exhibit differences.

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.

C. Detection Mechanisms in HosTaGe

HosTaGe can distinguish between three different classes of attacks: *Single-Protocol Level Detection (SPLD), Multi-Stage Level Detection (MSLD)* and *Payload Level Detection (PLD)*.

SPLD attacks refer to those that occur on a single protocol, e.g., 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. For example, we can infer that a multitude of malware is trying to exploit the Telnet protocol (see Section IV) due to the increasing amount of IoT devices such as IP Webcams. As this is the simplest of the three cases, we omit the description of its respective EFSM due to space constrains. In fact, the EFSM shown in Figure 1 can be seen as a generalized example of 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 our honeypot with the EFSM shown in Figure 1, as described in Section III-B. An important factor in MSLD is the time-window (tw) that determines whether an attack should be mapped as the SPLD or the 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. In this paper (see Section IV), we experimented with the tw value of 15 minutes. In practice, this suggests that when $tw = 15$, a transition from the *Attack* to the *Multistage Attack Level 1* state (via the *Different Protocol* predicate) is possible when a new attack occurs within the specified tw.

Fig. 2. EFSM for PLD in the case of Stuxnet propagation

PLD extends the applicability of the EFSM with respect to the input 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. Take Figure 2 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 1. 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 in which the presence of Stuxnet is reported.

D. Signature Generation

As discussed in the previous section, upon detecting an intrusion, *HosTaGe* is able to generate signatures for the Bro IDS. IDSs usually inspect all incoming packets for malicious content and make use of signatures to determine whether traffic is malicious or not. *HosTaGe* captures not only the attack packets, but also all received connection requests at the packet level. In addition, it records the entire connection tear-down that takes place between an adversary and the honeypot. *HosTaGe* utilizes specific modules to handle the above features, i.e., the *listener, logger, connection guard, HosTaGe service, multistage attack* detection, and *file injection* detection modules. The Bro IDS uses the so-called Bro language, an event-driven scripting language that can be used for extending and customizing the IDS's functionality. Bro's signatures rely on packet data to check for the content to be matched for incoming packets. The *signature generation* module in *HosTaGe* is used to generate the signature files [2] for Bro. By specifying the protocols for the signature generation mechanism, the packet information of the connection are derived through the *attack records* module from *HosTaGe* to a template signature file which is used to generate signature files for Bro. The *attack record* module of *HosTaGe* generates signature files for a protocol.

An example of a signature generated by *HosTaGe*, for the Modbus protocol, is shown in the Listing 1. This signature, automatically created from *HosTaGe* and written in the Bro language, is able to detect the well-known Metasploit script for Modbus services identification.

Listing 1. Modbus attack signature generated by *HosTaGe*

IV. Results

In this section we show various experiments that demonstrate *HosTaGe*'s detection capabilities as well as its ability to generate signatures from detected attacks. In addition, we

[2] Utilizing the Bro terminology, *HosTaGe* can generate security policy files.

discuss observed multi-stage attacks and give some insights regarding the detectability of our honeypot.

A. Honeypot Comparison

In this section, the *HosTaGe* and Conpot honeypots will be compared. Both honeypots were deployed in a controlled environment with no firewalls in between the honeypots and the Internet. The honeypots had similar IP addresses in the sense that they were in the same $/24$ subnet. Note that the IP addresses were dynamic, hence, offering more generic observations for the two honeypots. The evaluation period for our tests was 12 weeks for all protocols except for $S7$ that was limited to eight weeks. We should note that the purpose of this experiment is to compare the two honeypots and to identify automated attacks that target ICS networks; thus, no advertisement of the honeypots was made. However, as we later show, both honeypots were probed by a certain search engine (Shodan).

Fig. 3. Comparison of attacks on *HosTaGe* and Conpot for HTTP, Modbus, $S7$ and Telnet. Note, that Conpot does not support the Telnet protocol.

The results of our analysis are given in Figure 3. We compare the results gathered from the two honeypots for the HTTP, Modbus and $S7$ protocols. In addition, we present the results gathered for Telnet (even though Conpot does not support it) as Telnet is considered an important attack factor for ICSs networks [6]. Finally, note that the $S7$ protocol comparison presented in the figure is for a shorter period of time (i.e., 8 weeks) as $S7$'s implementation in *HosTaGe* was completed in a later stage. As one can observe from the results, *HosTaGe* exhibits good detection accuracy in comparison to Conpot. In fact, in most cases *HosTaGe* detected more attacks than Conpot (e.g., HTTP, $S7$), while for the case of Modbus the number of detected attacks is almost equivalent.

Lastly, Figure 4 depicts the malicious unique IP addresses (and also their intersection) that were detected for each honeypot (for the HTTP and Modbus protocols). In a glance, the histogram shows that *HosTaGe* was able to consistently detect more unique IP addresses (of attackers) than Conpot.

Fig. 4. Comparison of unique and common malicious IP addresses targeting *HosTaGe* and Conpot

B. Multi-Stage attacks

During the observed period, *HosTaGe* also automatically detected three multi-stage ICS attacks (within a time-window of $tw = 15$ minutes). First, an attack from Iran was detected, which based on the Geo-IP information, appeared to be close to where Stuxnet was initially detected, i.e., the Tehran Nuclear Research Center. The attack included a portscan and sent an HTTP GET request to the honeypot. Subsequently, we detected an Nmap script that was trying to validate whether a Modbus service is indeed running on the host. The second interesting multi-stage attack was originated from the Zhejiang Nuclear Industry in China. It started with a portscan, continued with an HTTP attack and finished with Modbus protocol requests that queried specific ICS services. The last attack was from an area close to a power plant in Colombia. The attack stages consisted of a web-connection through HTTP, a general purpose scan of the network and finally an attempted Telnet connection to the honeypot.

C. Honeypot Evasion

One essential requirement for honeypots is their ability to remain undetected; hiding the fact that this is indeed a honeypot and not what is being advertised. In this section we discuss how, in the aforementioned experimental setup, *HosTaGe* managed to remain undetected (while Conpot did not) from the Shodan[3] search engine. Shodan is a specialized search engine that crawls the Internet and attempts to identify connected devices, e.g., IP web-cameras, ICSs, etc. [1]. Through the continuous process of crawling and indexing, the system creates an up-to-date database of systems and services exposed on the Internet. Recently, Shodan implemented detection mechanisms to identify honeypots. It performs a series of probes, and checks, and subsequently creates a score for each probed device. Based on this score value, Shodan determines whether a specific system is an honeypot or not.

[3] https://www.shodan.io

A few weeks after our initial experiments, both *HosTaGe* and Conpot received probes from IPs that belong to Shodan. The protocols probed were S7, Modbus, SSH and HTTP, among others. For the SSH and HTTP protocols the messages and requests were of low complexity; the HTTP protocol involved a GET request and the SSH an attempt to establish a connection. The S7 and Modbus attacks were more complicated. Our analysis indicates that the probes are most likely the result of modified Nmap/Metasploit scripts for ICSs identification. These probes were able to identify Conpot as a honeypot but not *HosTaGe*. The S7 attack involved an audit of 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. On the one hand, Conpot could not respond meaningfully in all the aforementioned requests (either due to static serial numbers or Modbus protocol simulation errors) and thus it was classified as a honeypot. On the other hand, *HosTaGe* managed to respond successfully and hence remain undetected.

D. Signature Generation

The last part of our evaluation focuses on the signature generation mechanism and its effectiveness. We study the applicability of the generated signatures.

Our first goal is to generate multi-stage attack signatures. To achieve this we manually inject attacks into a *HosTaGe* instance in a controlled environment. Attacks are additionally captured by the network packet capture tool Wireshark and saved for future reference (see below). Simultaneously, *HosTaGe* detects attacks and constructs signatures from both the protocol and payload-level interaction.

For determining 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[4], malware focused traffic[5], and honeypot captured traffic[6]

First, we determine whether the generated signatures detect false positives. We import the signatures into the Bro IDS and replay the network traffic of the (unmodified) test datasets. With respect to our definitions in Section III-C, we utilize a time-window of $tw = 15$ (minutes) for our tests. No multi-stage attacks were detected by Bro, as expected. Afterwards, 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 the injected attacks without generating any false positives.

V. Conclusion

The increasing number of cyber-attacks, along with the need to protect critical infrastructure, such as ICSs, highlights the need for tailored security mechanisms. In this paper we presented a honeypot capable of detecting attacks on ICS networks. We formalize the detection mechanisms with EFSMs and discuss the ability to automatically generate signatures from observed attacks. Our experiments show that our proposal can be easily deployed in practice and that it is able to support existing intrusion detection infrastructure. As future work, we plan to further examine both honeypot evasion mechanisms as well as honeypot-side techniques to avoid such attacks. Lastly, we aim at further assessing Shodan's impact on the effectiveness of a honeypot (after its identification), as malware might make use of this information to avoid contacting well known honeypots.

References

[1] 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.

[2] Marcelo Fantinato and Mario Jino. Applying extended finite state machines in software testing of interactive systems. In *Interactive Systems: Design, Specification, and Verification (DSV-IS)*, volume 2844, pages 34–45. Springer Berlin Heidelberg, 2003.

[3] Hyang-Ah Kim and Brad Karp. Autograph: Toward automated, distributed worm signature detection. In *USENIX security symposium*, volume 286. San Diego, CA, 2004.

[4] Christian Kreibich and Jon Crowcroft. Honeycomb – Creating Intrusion Detection Signatures Using Honeypots. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 51 – 56, 2004.

[5] R Langner. Stuxnet: Dissecting a cyberwarfare weapon. *Security & Privacy, IEEE*, (June):49–51, 2011.

[6] Yin Pa Minn, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, and Christian Rossow. IoTPOT : Analysing the Rise of IoT Compromises. In *9th USENIX Workshop on Offensive Technologies (WOOT)*. USENIX Association, 2015.

[7] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23-24):2435–2463, dec 1999.

[8] Niels Provos. Honeyd : A Virtual Honeypot Daemon. In *DFN-CERT workshop*, 2003.

[9] Lukas Rist, Daniel Haslinger, John Smith, Johny Vestergaard, and Andrea Pasquale. Conpot honeypot, 2013.

[10] Hemant Sengar, Duminda Wijesekera, Haining Wang, and Sushil Jajodia. VoIP intrusion detection through interacting protocol state machines. In *International Conference on Dependable Systems and Networks (DSN)*, pages 393–402. IEEE, 2006.

[11] Lance Spitzner. Honeypots : Catching the Insider Threat. In *Computer Security Applications Conference*, pages 170–179. IEEE, 2003.

[12] 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.

[13] 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 *Security and privacy in smartphones & mobile devices*, pages 43–48. ACM, 2013.

[14] Emmanouil Vasilomanolakis, Shankar Karuppayah, Panayotis Kikiras, and Max Mühlhäuser. A honeypot-driven cyber incident monitor: lessons learned and steps ahead. In *International Conference on Security of Information and Networks*. ACM, 2015.

[15] Emmanouil Vasilomanolakis, Shankar Karuppayah, Max Mühlhäuser, and Mathias Fischer. Taxonomy and Survey of Collaborative Intrusion Detection. *ACM Computing Surveys*, 47(4):33, 2015.

[16] Nikos Virvilis and Dimitris Gritzalis. The Big Four - What We Did Wrong in Advanced Persistent Threat Detection? In *International Conference on Availability, Reliability and Security*, pages 248–254. IEEE, 2013.

[4] Small and Big flows datasets: http://tcpreplay.appneta.com/wiki/captures.html

[5] CTU-13 Dataset: https://stratosphereips.org/category/dataset.html

[6] HoneyBot Dataset: http://www.netresec.com/?page=PcapFiles