CryPLH: Protecting smart energy systems from targeted attacks with a PLC honeypot

Dániel István Buza, Ferenc Juhász, György Miru, Márk Félegyházi, and Tamás Holczer

Laboratory for Cryptography and System Security (CrySyS Lab),
Department of Networked Systems and Services (HIT),
Budapest University of Technology and Economics (BME)

Abstract. Smart grids consist of suppliers, consumers, and other parts. The main suppliers are normally supervised by industrial control systems. These systems rely on programmable logic controllers (PLCs) to control industrial processes and communicate with the supervisory system. Until recently, industrial operators relied on the assumption that these PLCs are isolated from the online world and hence cannot be the target of attacks. Recent events, such as the infamous Stuxnet attack [15] directed the attention of the security and control system community to the vulnerabilities of control system elements, such as PLCs. In this paper, we design and implement the Crysys PLC honeypot (CryPLH) system to detect targeted attacks against industrial control systems. This PLC honeypot can be implemented as part of a larger security monitoring system. Our honeypot implementation improves upon existing solutions in several aspects: most importantly in level of interaction and ease of configuration. Results of an evaluation show that our honeypot is largely indistinguishable from a real device from the attacker's perspective. As a collateral of our analysis, we were able to identify some security issues in the real PLC device we tested and implemented specific firewall rules to protect the device from targeted attacks.

1 Introduction

For a long time, industrial control system operators relied on security by obscurity, this means they assumed that control devices cannot be accessed from the Internet and hence the operators made minimum effort to protect these devices. This behavior was motivated by the special requirements in control systems, namely high availability, time-critical services and the huge costs of a potential blackout. Security improvements were mainly considered as a potential risk that can disrupt the normal operation of the system.

In recent years, the threat model of industrial control systems has changed. The Stuxnet targeted attack [15] demonstrated that sophisticated malware is able to penetrate into the isolated part of the control system that were traditionally separated from the parts connected to the Internet. Stuxnet in particular contained modules that attacked PLCs in the target system and consequently

caused physical damage in the physical equipment. Recent work [16] demonstrated that attacking PLCs is relatively easy and this makes them an attractive target when attacking industrial control systems.

There have been some efforts to detect attacks against PLCs and to protect them against these attacks [17]. One of the most promising defense mechanisms is the application of honeypots [20] specifically adapted to PLCs. A honeypot is a system that seems to be a PLC based on network behavior, but does not provide any operational functionality in the system. A honeypot practically serves as a trap for attackers. One of the honeypot's aim is to maintain the attacker's interest and thus observe the attack methods against real PLCs. This way, previously unknown attack methods can be revealed that can be analyzed to improve the security of real PLC devices.

There are three kind of honeypots depending on the level of sophistication [19, 20]. Low-interaction honeypots simulate only basic network services (or only a base part of the basic network services). High-interaction honeypots simulate different complex network services. The advantage of the low-interaction ones is that they are easier to design and maintain. They can be more stable but they are easier to discover. The high-interaction honeypots are able to keep the attacker's attention longer. But they are much harder to implement because they have to implement the already known bugs and incorrect activities as well. Hybrid honeypots try to combine low and high interaction parts to get the advantages of both. Hybrid honeypots usually simulate different services on different interaction levels.

There are a few existing PLC honeypot implementations. The Scada HoneyNet Project [21] was started in 2004 by Cisco Critical Infrastructure Assurance Group (CIAG) and was discontinued in 2005. It consists of a set of python scripts, each of them implementing a service of the simulated PLC. The project heavily utilises Honeyd [18, 19], which is a small daemon that creates virtual hosts on a network. The hosts can be configured to run arbitrary services, and their personality can be adapted so that they appear to be running certain operating systems. The Honeyd daemon can be set to simulate a computer that has the OS fingerprint of a PLC and runs the given Cisco scripts on the appropriate ports. With the help of these scripts the Honeyd PLC realises a Telnet, FTP, HTTP and Modbus services. In summary, these scripts seem unfinished, the services are only partially implemented and the realised functionality is nor realistic neither interactive. Also, it is worth to mention that bugs and mistakes are present in the code, for example, if the log file does not exist, the script doesn't create it, instead it throws an unhandled exception. Even an inexperienced attacker would notice in seconds that the simulated PLC is not real, and the information provided by the logs can not be used to uncover the identity or the methods of the attacker, therefore the SCADA HoneyNet Project clearly fails to reach its goals.

The SCADA Honeynet [7] is maintained by Digital Bond and is freely available from their website. It utilises two virtual machines one of which is a Generation III honeywall (by The HoneyNet Project [10]) extended with Digital Bonds

Quickdraw IDS signatures [7]. The purpose of this unit is to monitor all network activity to identify and log every malicious attack that may occur against the simulated PLC. The other virtual machine simulates a popular PLC that runs five services (FTP, Telnet, HTTP, SNMP, Modbus TCP), the FTP, HTTP and Modbus services are implemented by different Java applications while the Telnet and SNMP services are realised by python scripts. The core of the VM is Honeyd that routes the created virtual host's network traffic (the data streams and datagrams) from the appropriate ports to these applications and scripts. The Digital Bond's SCADA Honeynet is a huge improvement over the Cisco Honeynet Project. With the returned service banners and OS fingerprint it can make scanning and information gathering tools (such as nmap or nessus) believe that it is a real PLC, thus it can be effective against automated attacks and tools. However, the simulated services provide very little interaction, and they might not be able to keep an attacker attracted for long enough to uncover new targeted PLC attacks.

The Conpot project [1] by The Honeynet Project [3] was released in May 2013, and it is available for everyone from their website. Conpot is an ICS honeypot with the goal to collect intelligence about the motives and methods of adversaries targeting industrial control systems. The honeypot realizes two major ICS protocols: Modbus and SNMP. There is also an HTTP service, and a logging system in the honeypot. These services are implemented with Python scripts. The honeypot emulates a Siemens S7-200 CPU type PLC by default, but it can be easily reconfigured through various profiles. The base concept of Conpot by The Honeynet Project is good, but it still have some defects which really need fixing. The honeypot is easy to install and use, but to reach it's full potential, it needs a lot of customization. The HTTP server is not working yet, but it's an important part of a honeypot, because it is a major attack surface where we can examine the behaviour of an attacker. Because of the little interaction the honeypot provides, it's possible that the attacker moves on before it's methods and behavior can be discovered.

The contributions of the paper is the following:

- We implemented a PLC honeypot, which is superior to existing solutions both in usability and indistinguishability.
- The implementation was tested against a real PLC to find differences. The test highlighted only minor problems.
- The honeypot was installed in a public network to gather real attacks successfully.

The remainder of the paper is organized as follows: In Section 2 we introduce our CryPLH implementation. The evaluation and public testing is described in Section 3 and Section 4 respectively, while the summary of the paper is in Section 5.

2 PLC honeypot for security monitoring

In this section, we present our honeypot implementation for security monitoring of industrial control systems. For the reference implementation, we selected a Siemens Simatic 300(1) PLC device. This device has a IM151-8 PN/DP CPU and it uses firmware version 3.2.6. The PLC has four network interfaces, the first one is a serial port that uses PROFIBUS protocol to control the devices attached to it in a real production environment, the second interface is a Fast Ethernet port that is used to manage the PLC, it should be connected to the management network. The third and fourth interface can be used to form a ring topology of the PLCs. Our goal in this reference implementation was to develop a high-interaction honeypot which appears identical to the real device from an attacker's point of view. We developed a system which is complex but easy to configure, so it can be extended to simulate different (but similar) PLC types.

We have setup a test environment to assess the properties of our Honeypot implementation as shown in Figure 1. We used an attacker machine equipped with Backtrack Linux R5 [11] to test and analyze the services offered by the PLC and our PLC honeypot. The PLC and the PLC honeypot were installed in the same subnetwork as the virtual machine that runs the Backtrack R5. The initial nessus [12] and nmap [13] scans showed that the Siemens Simatic 300(1) PLC runs four services these are the http, https, isotsap and snmp. The Siemens PLC's port forwarding guide [14] also confirmed that the PLC is capable of running these services.

After implementing the services mentioned above, we integrated them onto a virtual machine to create the honeypot (as presented in Figure 2). The VM runs a minimal version of Ubuntu Linux, that only has the necessary services and libraries installed. We also implemented a bash script that can start, restart or stop the honeypot. It is run by *(initd)* on every start up. Every simulator has its own way to fine tune it, however there is a main configuration file of the honeypot. In this file global settings can be set, such as the IP and network interface that the honeypot is being run on. The startup-script reads these values and configures each service simulator before it launches them. The script also sets iptable rules to block all incoming connection that are not destined to one of the simulators. The TCP connections are refused by a TCP reset, just like on the real PLC. The UDP datagrams that are not sent to the SNMP service are simply dropped. The script starts tcpdump to capture the network traffic on the honeypots interface. The final step is to change the behaviour of the TCP/IP stack. The files in /proc/sys/ipv4/ provide an interface for this on Linux. These files contain values, that can be read or written and control the operation of TCP and IP. The IP ttl is set to 30, and the MTU is 1518, also the PLC has a fixed TCP window size, which unfortunately can not be exactly set, because of the limitations of the /proc file system. Some of the TCP related values are compiled into the kernel, so they cannot be changed through these files. Currently this is a limitation of the honeypot which we will address in our future work.

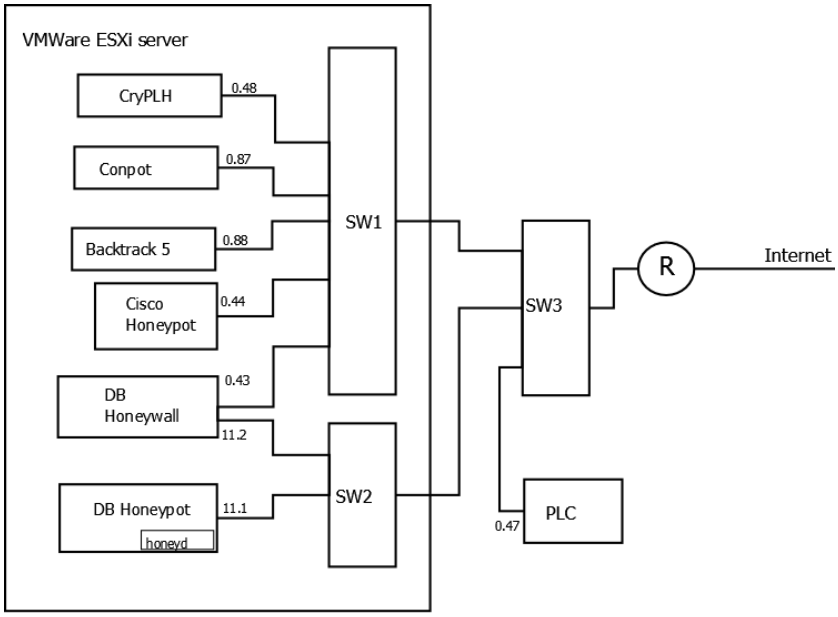

Fig. 1. The laboratory setup of the honeypot testbed

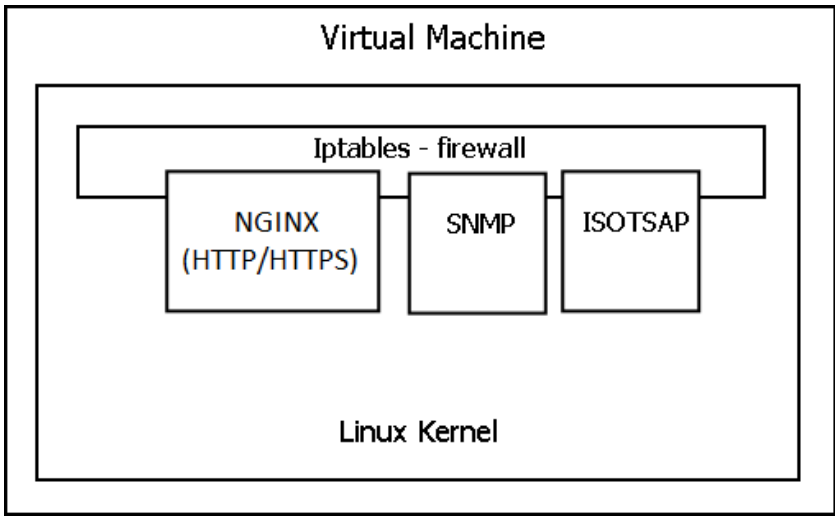

Fig. 2. The abstract model of the integrated honeypot

2.1 Hypertext Transfer Protocol (HTTP) implementation

HTTP is one of the basic protocols on the Internet and it is also implemented on the Siemens Simatic 300(1) PLC device. Secure HTTP (HTTPS) is an extension

of HTTP service. It is not an independent protocol it is layering HTTP over SSL/TLS protocol. The device uses a web server from the MiniWeb project[4] that was developed in the C language to provide a small HTTP server with high efficiency and high portability. It also implements a login procedure to access the device.

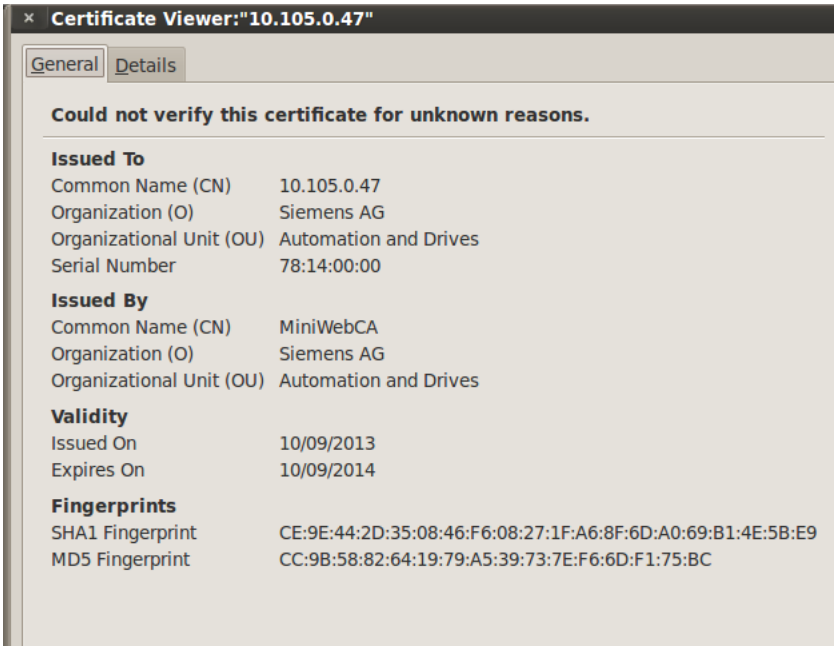

Fig. 3. Details of the PLC's certificate

The device also implements the HTTPS service. There is no difference of the served data and pages. The device uses a self-signed (not trusted) certificate as it is shown on Figure 2.1. The certificate is issued to Siemens AG organization issued by the same organization with MiniWebCA common name. The signature algorithm is SHA-1 with RSA Encryption and it uses an 1024 bits long public key. Observing different PLC devices we found that each device use a unique certificate (instead of using the same certificate on every device).

During the test we found that sometimes the PLC is not responding that is probably an indicator for a bug in the firmware. Also, requesting `Portal.mwsl` without parameters causes a complete crash of the system [2]. This can cause security issues when an attacker can send arbitrary URLs to the web server. To improve the safety of a PLC it is strongly suggested to use a firewall to filter HTTP requests and drop the ones which contain `Portal.mwsl` without parameters. We implemented this function by `iptables` as well using `-m string` option.

We simulated the behavior of the MiniWeb server on our PLC honeypot. We had to manage some changes in the copied files to remove all links to the real device. And we rewrote the login mechanism (there is no name and password check) to simulate that there is a password but the attacker failed to guess it. After the changes we were able to simulate the HTTP service with static pages.

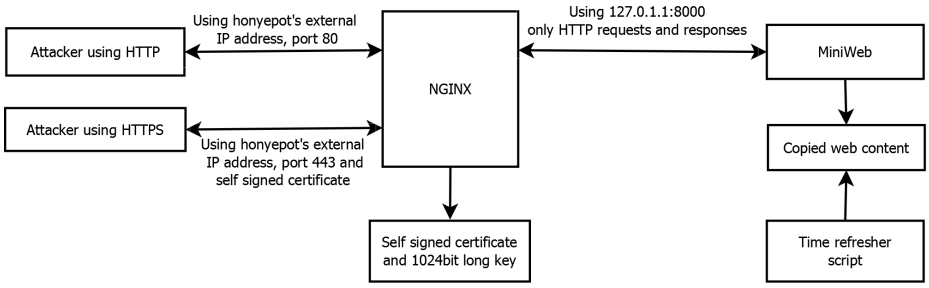

Fig. 4. The HTTP/HTTPS service environment

We also simulate the HTTPS service. We used OpenSSL [6] and generated an 1024 bit long key and a self signed certificate. We added the same meta-information (e.g. location, common name, organization) to the certificate that the device adds to it's self signed certificate. To simulate HTTPS we used nginx [5]. With suitable configuration it tunnels HTTPS traffic to the miniweb server over HTTP. An another advantage of using nginx is that we can separate the HTTP traffic into two parts: traffic between the honeypot and the outside network and the traffic inside the honeypot (between nginx and miniweb). Thus the system becomes more configurable. Figure 2.1 illustrates the complete developed structure.

In spite of the substantial implementation effort, the honeypot device has some features that allow an attacker to detect it. Visually there is no difference between the websites on the real device and the honeypot. But on the honeypot's site it is impossible to log in because we pretend that there is a valid username and password combination but the attacker's guess was wrong. On the honeypot's page there is no effect of selecting other languages (it only uses English language). We have to note that on the real device's site it is also impossible to reach any other offered languages because it does not have enough memory to contain the language files. We do not simulate any changes of the visual presentation of the PLC on the site (e.g. the LEDs state never change). We can say that we pretend that the environment of the simulated device never changes. It can be believable that a device's environment does not have changes which change the PLC's running state. We do not simulate the already known bug by requesting `Portal.mwsl` without any parameters however a script can be easily added to stop all communication for a while after visiting the page. If needed,

these additional features can be simulated at the cost of making the honeypot implementation more complex.

2.2 Implementing the Simple Network Management Protocol (SNMP)

The SNMP service is commonly installed on intermediary and end network devices such as PLCs, because it provides a simple and easy way to monitor and manage the device. In a typical SNMP conversation there are two participants, a manager, that queries the requests and a managed device, that replies to these. The managed device runs an SNMP software that is called the Agent. The Agent interprets the queries and returns the requested data to the manager. The whole hierarchy and the metadata (variable names, permissions and types) are described in Management Information Databases (MIBs) which use ASN.1 notation.

After the thorough exploration of the Siemens PLC SNMP database and implementation, we created the SNMP service on the honeypot device. After carefully analyzing existing SNMP implementations, we decided to implement a custom SNMP Agent that provides more flexibility. The realised Agent, just like the original, listens on the UDP port 161, and accepts SNMP requests and replies to them. Instead of using real MIBs it parses an XML file that contains the list of records that are present on the real PLC. All these records have an OID and a type attribute. They either contain the static data (e.g. the system descriptor, or interface description) that they represent or they contain a special mark and string that tells the interpreter how the dynamic data (e.g. the system uptime, or the number of received IP packets) should be created or retrieved.

There are variables that an attacker could directly or indirectly alter. Such variables are for example the number of received packets on the connected interface or the number of received ICMP echo requests. The value of these records can not be simply generated because an attacker can try to modify these and check if the returned numbers vary accordingly. To avoid the detection of the honeypot, these values are read from the /proc file system which contains information gathered by the OS on Unix systems. The script receives the name of the interface and this name is used when reading device related information from the /proc/net/dev file. The IP, TCP, UDP and ICMP related data is acquired form the /proc/net/snmp. The purpose of this file is to provide information about these protocols for different SNMP Agents. The TCP current establishments value is read from the /proc/net/sockstat file. As it was mentioned before, the script keeps track of the SNMP related events. It uses the data gathered this way to serve requests for SNMP records.

We tested our SNMP implementation with snmpwalk. The result of the test corresponded to the expectations. Later, the Agent was tested against different Get and GetNext requests, the response format was always identical to the original PLC's responses. It is important to note, that there is a known limitation of the SNMP Agent. If a specific group of records are queried from the local network, the returned data is not valid. However it is not a real limitation,

because we assume that the honeypot will be put on a public address, where local network is not defined.

2.3 Step 7 protocol implementation

Siemens SIMATIC STEP7 [9] is an engineering software for configuring and programming Siemens type controllers. We can setup and program several automation systems, for example SIMATIC HMI panels, or Siemens PLCs. The communication is going through TCP port 102 of the PLC and uses the ISO-TSAP protocol. We found that the PLC is protected from unauthorized access. This means that we can not download the firmware to the PLC until we entered the correct password. We decided that we are going to simulate this behaviour on the PLC, but without a correct password. So whenever someone tries to access the PLC, the response always will be that the entered password is incorrect. On one hand, this is a fairly simple solution, but it is also believable: a system which has such a great responsibility in controlling facilities or power plants should not be accessible to anyone. On the other hand, if we would like to simulate the programmability, the response to various inputs could be problematic.

We implemented a simulation of the STEP 7 protocol on the PLC honeypot with a python script. STEP 7 queries the PLC for specific parameters and we included these parameters in our response packets to simulate the behavior of the real device.

3 Evaluation

We performed the needed tests for the single services, and presented the results in the related sections. After integrating the services into a PLC honeypot implementation, we successfully tested that the communication with the modules worked properly. Then, we checked the difference between the honeypot and the real PLC. We found that the PLC honeypot is mostly indistinguishable from the real device, yet there are a few characteristics the attacker can use to spot the PLC honeypot. Although we tried to mimic the PLCs networking stack, due to operating system limitations, we were not able to emulate a fully identical TCP/IP stack. Thus, based on the `nmap O` operating system scan, one can distinguish between the two devices with some probability. If the attacker is close to its target (same LAN segment), then in the nmaps operating system guess report there is an entry for Linux when scanning the honeypot (with several other guesses), but Linux is not guessed when the PLC is scanned. If the attacker is a few hops away from its target (which is the more likely scenario), then the honeypot and the real PLC are indistinguishable using the oeprating system fingerprint. Reliably fooling nmap with the proper TCP/IP parameters is difficult if the attacker is on the same LAN segment. One solution can be writing a TCP proxy, which re-frames the outgoing packets, so it would seem it is coming from a real PLC. Another solution is to rewrite an existing TCP/IP stack to be fully identical to the one used in the PLC. We left this effort as a future work.

4 Public testing

The honeypot was installed on a public network to gather real traffic and possibly attacks. The public IP address of the device was set within an university's IP range. This is not ideal as targeted attacks against industrial control systems do not scan educational IP ranges, however we found some interesting traffic. In the future we want to install our implementation to more appropriate places, where more interesting intelligence information can be collected.

Two tests are described in the followings. The first took eight days and all the logs were analyzed thoroughly. The second test took approximately one month and only the interesting traffic were filtered and analyzed.

4.1 Short test

In the short (eight days long) test no traffic accessed neither the TSAP nor the SNMP port. Several pings and port scans were carried out on the honeypot, and also several attempts were made to gain access via ssh to the machine (most of the ssh scans are originated from China with 6000 as source port). These attempts were all rejected by the firewall.

The web server was accessed several times, but the attackers were mainly looking for vulnerable PHPMyAdmin pages or vulnerabilities in CCTV camera firmwares. The most interesting attacks against the web server were scans for open proxies, where the attacker tried to download scientific journal papers using the university's subscription. As the PLC honeypot does not offer such services, these attempts were unsuccessful. The whole content of the web server was crawled by robots several times as well.

In general no PLC specific attack accessed our honeypot during this test period.

4.2 Long test

In the long test (1 month long), we observed similar attack patterns like in the short test. The only real difference was some traffic to the TSAP port. The TCP streams are most likely originated from Shodan [8], which is a search engine for embedded devices. Shodan made some connections to the TSAP port, and sent some queries to the honeypot. Unfortunately Shodan does not reveal the results of these scans publicly.

5 Summary

The aim of this study was to create a high interaction honeypot called CryPLH, that appears to be a Siemens PLC form an attackers point of view. It needs to be able to log all the action an attacker takes, while trying to exploit the PLC. So that later by analysing these log files new targeted attacks can be uncovered,

possibly before they reach the real equipments. In order to achieve this all the existing PLC honeypots were examined, and a real PLC was thoroughly audited.

After the exploration of the device, all the discovered services (HTTP, HTTPS, ISO-TSAP and SNMP) were further inspected. Than the simulators of these were implemented and integrated into a Linux based virtual machine. The resulting VM is the honeypot. Although, currently the honeypot has a few limitations which needs to be addressed in the future, still it preforms its task better than any of its predecessors: it is highly configurable, it implements all the services the original PLC implements, and it is interactive.

The most important current issue is to add more functionality to the TSAP implementation, because probably the most interesting traffic would come to that port. Another less urgent issue is the incorrect TCP window size, which cannot be set to the exact required value because of the Linux kernels limitations. In the future a kernel patch or a TCP proxy (that re-frames all the outgoing TCP PDUs and sends them with raw sockets) needs to be written to overcome this problem. The other known issue is related to the SNMP routing records, but this is a less significant problem, because it only exists if the attacker is on the same LAN as the honeypot. The honeypot was tested by independent professionals and no other issue was discovered.

After the initial tests, the honeypot was installed on a public network, to gather information about ongoing attacks. The honeypot collected many general PC targeting attacks, but none of the attacks were PLC specific.

In our future we want to deploy our honeypot within an industrial control system's IP range and gather more intelligence information. By analyzing the collected logs, we will aim at identifying new threats against industrial control systems. We also want to make our honeypot implementation to be publicly available, after correcting the small problems mentioned above.

Acknowledgement

We would like to acknowledge the help and the provided Siemens PLC device to Óbuda University and the company evopro. This work is partially funded by the EIT ICTLabs through activity ASES 13030.

References

1. The conpot project. http://www.conpot.org. Last accessed: 2013-08-04.
2. Crash per webinterface. http://www.sps-forum.de/simatic/52478-s7-1200-crash-per-webinterface.html. Last accessed: 2013-10-16.
3. Introducing conpot. http://honeynet.org/node/1047. Last accessed: 2013-08-04.
4. Miniweb project webpage. http://miniweb.sourceforge.net/. Last accessed: 2013-10-07.
5. Nginx site. http://wiki.nginx.org. Last accessed: 2013-10-16.
6. Openssl: The open source toolkit for ssl/tls. http://www.openssl.org. Last accessed: 2013-10-16.

7. Scada honeynet. http://www.digitalbond.com/tools/scada-honeynet/. Last accessed: 2013-06-17.
8. Shodan - expose online devices. http://www.shodanhq.com/. Last accessed: 2014-03-01.
9. Simatic step 7 engineering software - software for simatic controllers - siemens. http://www.automation.siemens.com/mcms/simatic-controller-software/en/step7/Pages/Default.aspx. Last accessed: 2013-10-18.
10. Honeywall project site. http://www.honeyd.org/honeywall/, 2009. Last accessed: 2013-06-17.
11. Backtrack linux - penetration testing distribution. http://www.backtrack-linux.org/, 2013. Last accessed: 2013-10-23.
12. Nessus vulnerability scanner. http://www.tenable.com/products/nessus, 2013. Last accessed: 2013-10-10.
13. Nmap - free security scanner for network exploration & security audits. http://nmap.org/, 2013. Last accessed: 2013-10-23.
14. Siemens product support. http://support.automation.siemens.com /WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=8970169&caller=view, 2013". Last accessed: 2013-10-13.
15. Boldizsár Bencsáth, Gábor Pék, Levente Buttyán, and Mark Felegyhazi. The cousins of Stuxnet: Duqu, Flame, and Gauss. *Future Internet*, 4(4):971–1003, 2012.
16. K Gorzelak, T Grudziecki, P Jacewicz, P Jaroszewski, Ł Juszczyk, P Kijewski, and A Belasovs. Proactive detection of network security incidents. 2012.
17. Philip Koopman. Embedded system security. *Computer*, 37(7):95–97, 2004.
18. Provos N. Developments of the honeyd virtual honeypot. http://www.honeyd.org/, 2007. Last accessed: 2013-06-16.
19. Niels Provos. Honeyd-a virtual honeypot daemon. In *10th DFN-CERT Workshop, Hamburg, Germany*, volume 2, 2003.
20. Niels Provos and Thorsten Holz. *Virtual honeypots: from botnet tracking to intrusion detection*. Pearson Education, 2007.
21. Pothamsetty V. and Franz M. Scada honeynet project: Building honeypots for industrial networks. 2005.