# The design and implementation of a PLC honeypot for detecting cyber attacks against industrial control systems

T. Holczer[1], M. Félegyházi[1], L. Buttyán[1]

[1]Laboratory of Cryptography and System Security, Budapest University of Technology and Economics, Hungary

*E-mail contact of main author: holczer@crysys.hu*

**Abstract.** Programmable logic controllers (PLCs) are used widely 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 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 a PLC honeypot system to detect targeted attacks against industrial control systems. Our honeypot implementation improves upon existing solutions in several aspects: most importantly it is a high interaction easy to configure solution. The usability of this approach is showed in preliminary real life deployments.

**Key Words**: PLC, honeypot, intrusion detection, industrial networks.

## 1. Introduction

Traditional industrial infrastructures are now heavily supported by a computing infrastructure to make control processes in nuclear facilities cheaper, faster and easier to implement. Yet, the implementation of computer-controlled processes opens the door for cyber-attacks. Our paper focuses on protecting industrial control systems in critical nuclear infrastructures against such cyber-attacks.

The structure of industrial process control equipment (e.g., PLCs) is very different from the traditional design of computers, since their function is completely different. The most obvious difference is that these devices have guaranteed response time. This property makes them usable in safety-critical systems. PLCs, for example, have different network interfaces to communicate with each other, controlled or regulated devices and other network devices. If a system has more than one PLC device then they can directly be linked to each other (similar to other networks, the topology of these networks can vary).

The management and maintenance of the PLCs are mostly done over the network. Manufacturers strongly suggest that users should connect these devices only to trusted networks, i.e. to networks that are not connected to the Internet. Until recently, attacks against PLCs were rather rare. Unfortunately connecting these PLC devices directly to unprotected networks became common practice. This can be convenient during maintenance (which can be done from anywhere even from other countries or continents). The same logic led some users to disable the PLC's built-in basic security mechanisms (for example, use well-known default passwords).

As the PLCs are unprotected, they became popular targets for attackers. In most cases, it is not challenging to take over the control of a PLC. As these devices control industrial processes, the failure of them can be critical; it can cause significant environmental and financial damage as well.

To protect industrial control systems in nuclear facilities against possible cyber-attacks, we need to know which techniques are used against PLCs. We propose to mitigate targeted attacks against PLCs with the use of honeypots, well-known from traditional computer systems. A PLC honeypot is a computer system that looks like a PLC based on network behaviour, but actually it is a trap for attackers. PLC honeypots help in maintaining the attacker's interest and allow defenders to observe the attack methods used against real PLCs. Thus, previously unknown methods can be revealed and these lessons can be used to protect the control system against future attacks.

In this paper, we report on the development of a high interaction honeypot, which realizes almost all services of a Siemens ET 200S PLC. Obviously the process controlling parts are omitted, as they are very complex, and an attacker cannot interact with them directly. To develop the PLC honeypot, we studied and understood how a real PLC communicates with a STEP7 management station, and implemented this functionality as a program running on a PC. We also examined and implemented the embedded webserver running on the real PLC working in close connection with the management service. We also implemented the SNMP service of the PLC. We customized the TCP/IP stack of Linux to create a stack almost identical to the PLC's. Finally, we integrated our services to an easy to use package, which can turn any Debian-based Linux PC into a PLC honeypot. The structure of our paper is the following: we review the state of the art in Section 2. We describe the implementation of the PLC honeypot in Section 3. Some operational observations are analysed in Section 4, while we conclude our paper in Section 5.

## 2. State of the art

There have been some efforts to detect attacks against PLCs and to protect them against these attacks [1]. One of the most promising defense mechanisms is the application of honeypots [2] 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. When designing a honeypot the goal is to make the honeypot as indistinguishable from the real device as possible, so the attacker attacks the honeypot instead of the real device.

There are three kind of honeypots depending on the level of sophistication [2][3]. 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 [4] 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 utilizes Honeyd [3], 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 realizes a Telnet, FTP, HTTP and Modbus services. In summary, these scripts seem unfinished, the services are only partially implemented and the realized 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 cannot 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 [5] 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 [6]) extended with Digital Bond’s Quickdraw IDS signatures [5]. 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 realized 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 [7] by The Honeynet Project 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 has some defects which really need fixing. The honeypot is easy to install and use, but to reach its full potential, it needs a lot of customization. The HTTP server is not fully functional yet, but it's an important part of a honeypot, because it is a major attack surface where we can examine the behavior of an attacker. Because of the little interaction the honeypot provides, it's possible that the attacker moves on before its methods and behavior can be discovered.

A good survey of existing industrial honeypots can be found in [8]. They analyze the requirements of honeypots in SCADA systems, and the reason why to use it. They also

created an experimental setup, where low interaction honeypots (Honeyd), and real PLCs as high interaction honeypots were used.

This paper is a follow up paper of our previous work [9], where we introduced our first results on creating a PLC honeypot. In this paper we show, how we could make a high interaction honeypot from our previous low interaction honeypot.

## 3. PLC honeypot implementation

To create a honeypot, the target system must be first decided. We decided to make a honeypot from the Siemens ET/200S PLC. We decided to base our work on this product, because Siemens products are used very widely all around the world. This product belongs to the 300 series which is one of the most widely used series of Siemens'. After uploading an initial configuration, it was clear, that the PLC offers three main services. They are the STEP7 management service on TCP port 102, the HTTP(S) service on TCP port 80 and 443, and the SNMP service on UDP port 161. The HTTP and the STEP7 services are closely connected, because both can read several buffers and variables representing the inner state of the PLC. Thus we decided to implement these two services in one emulated service. The SNMP is mainly responsible for reporting static data (such as device name) and some dynamic data (such as uptime or interface counters). These data are only loosely connected to the other services, so we implemented the SNMP in a separate service. In the next subsections, we describe, how we implemented these services.

### 3.1 Integrated web and management service

The inner state of the PLC can be queried by two protocols: HTTP(S) and STEP7. Through a regular web browser, the state of the PLC can be read (such as running state, error buffers etc). The STEP7 protocol is a closed protocol designed by Siemens to manage and interact with Siemens industrial process control devices. It can read and write every memory position, list or state. The two services must deliver the same response, when the same question is asked, thus we decided to implement them together in C++. We chose C++, because some libraries and code fragments used in the project was written in C and C++.

After some preliminary analysis, it was clear, that the embedded web server uses a special version of the open source web server called Miniweb[1]. We used the same web server to implement the HTTP protocol. The home page of the PLC can also be accessed through HTTPS. We decided to use an SSL terminator called SziluSSL[2], as it offers only the required functionalities we wanted. Other widely used SSL proxies or terminators like nginx offered much more functionality, then we required, and the extra functionalities would be detectable by an attacker.

The implementation of the STEP7 protocol was the toughest part of our project. It is a complex management service and protocol, which can interact and modify every part of the PLC. Unfortunately, there is no publicly available documentation of the protocol. Our

[1] http://dunkels.com/adam/miniweb

[2] https://github.com/szilu/SSL-Proxy

implementation was based on the Snap7 project[3] which was extended by several parts. We implemented the following main functionalities:

- Querying and setting the system clock
- System State List functions (contains read-only information about the PLC)
- Block information
- Data read and write requests
- Downloading blocks and firmware update (in PLC terminology downloading means information sent to the PLC from the programming station)
- Uploading blocks (in PLC terminology uploading means information sent from the PLC to the programming station)
- Password protection
- PLC Control functions
- Monitoring and modification through Variable Table

The implemented service listens on TCP port 80 and 102, and through the SSL proxy, it listens on TCP port 443.

### 3.2 Simple network management protocol

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 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 realized 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. Most of the dynamic data could be retrieved by system calls or simple file reads from the /proc file system.

[3] https://snap7.sourceforge.net

### 3.3 Integration of services

The implemented services were integrated on a virtual machine running Linux operating system. The processes (integrated STEP7-HTTP server, SNMP server, SSL terminator) were run automatically on boot. The behavior of the honeypot was driven by a common configuration file. This configuration file defined which interface to attach, which IP address to use, the address of the management station etc.

To hide all other processes, and mimic the behavior of the PLC, we used the embedded firewall of Linux (iptables). Within the firewall, we could filter out any unneeded traffic, and modify the packet headers where needed (e.g.: the PLC uses an initial TTL value of 32, while Linux uses 64). Also the TCP/IP stack of the Linux machine was modified in run time to use constants and algorithms similar to the PLC.

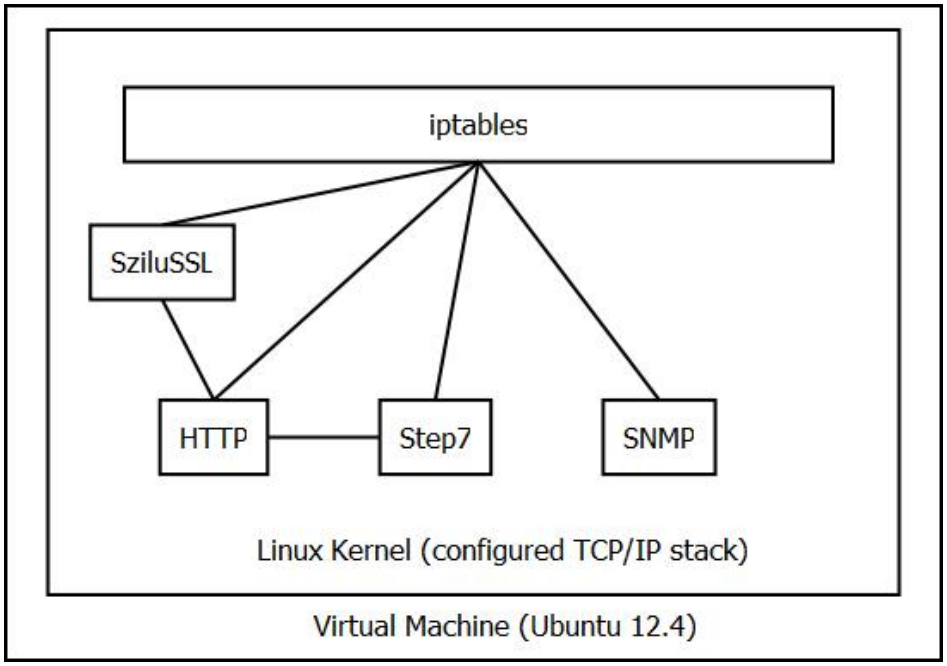


*FIG. 1. Integrated PLC honeypot*

The integrated system was tested both manually and automatically to find any difference between the real PLC and the PLC honeypot. After successfully passing all the tests, it was deployed to capture real traffic. The details of the deployment are described in the next section.

## 4. Real life deployments

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 a 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.

Three 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. The third test was 12 days long, and also only the interesting traffic was analyzed.

### 4.1 First short test

In the short (eight days long) test no traffic accessed neither the STEP7 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 STEP7 port. The TCP streams are most likely originated from Shodan [10], 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.

### 4.3 Second short test

The second short test was similar to the previous tests. There was no STEP7 or SNMP traffic. The attacks against the web server contained attacks against PHPMyAdmin pages and Cisco/LinkSys routers running a special binary CGI executable called `tmUnblock.cgi`.

The distribution of source countries was the following:

| Country | Percentage |
|---|---|
| **China** | 29 |
| **United States of America** | 29 |
| **Russian Federation** | 10 |
| **Germany** | 4 |
| **Italy** | 4 |
| **Netherlands** | 4 |
| **Romania** | 4 |
| **Colombia** | 2 |
| **France** | 2 |
| **Indonesia** | 2 |
| **Liechtenstein** | 2 |
| **Macedonia** | 2 |

| | |
|---|---|
| **Poland** | 2 |
| **Turkey** | 2 |
| **Ukraine** | 2 |

In general we can conclude from the tests that if we put the honeypot to a secure environment, then probably no traffic will access the honeypot. If we put the honeypot into a public network, then a lot of general attacks access the honeypot, but only a very small fraction is specific to industrial control systems.

## 5. Conclusion

Monitoring the critical infrastructure is a very important task as we cannot prevent every malicious attack. In this paper we described the implementation of a PLC honeypot, which can play a crucial role in this task. We described how we implemented and integrated the different services into a working system. The PLC honeypot was deployed in real networks to capture traffic, which was done successfully. In the future we want to complete our honeypot to a fully functional industrial system monitoring framework, which can be used to detect previously unknown attacks in critical systems.

## REFERENCES


[1] Koopman, Philip. "Embedded system security." *Computer* 37.7 (2004): 95-97.

[2] Provos, Niels, and Thorsten Holz. *Virtual honeypots: from botnet tracking to intrusion detection*. Pearson Education, 2007.

[3] Provos, Niels. "Honeyd-a virtual honeypot daemon." In *10th DFN-CERT Workshop, Hamburg, Germany*, vol. 2, p. 4. 2003.

[4] Pothamsetty, Venkat, and Matthew Franz. "Scada honeynet project: Building honeypots for industrial networks." (2008).

[5] DigitalBond. "SCADA Honeynet." http://www.digitalbond.com/tools/scada-honeynet/

[6] Honeyd. "Honeywall project." http://www.honeyd.org/honeywall/

[7] Lukas Rist. "Introducing Conpot" http://honeynet.org/node/1047 (2013)

[8] Disso, Jules Pagna, Kevin Jones, and Steven Bailey. "A Plausible Solution to SCADA Security Honeypot Systems." In *Broadband and Wireless Computing, Communication and Applications (BWCCA), 2013 Eighth International Conference on*, pp. 443-448. IEEE, 2013.

[9] Buza, Dániel István, Ferenc Juhász, György Miru, Márk Félegyházi, and Tamás Holczer. "CryPLH: Protecting smart energy systems from targeted attacks with a PLC honeypot." In *Smart Grid Security*, pp. 181-192. Springer International Publishing, 2014.

[10] Matherly, John C. "SHODAN the computer search engine." *Available at [Online]: http://www. shodanhq. com/help* (2009).