# A Flexible Architecture for Industrial Control System Honeypots

Alexandru Vlad Serbanescu[1,*], Sebastian Obermeier[2] and Der-Yeuan Yu[3]
[1]*KPMG AG, Zurich, Switzerland*
[2]*ABB Corporate Research, Baden, Switzerland*
[3]*Department of Computer Science, ETH, Zurich, Switzerland*

Keywords: Industrial Control System Security, Honeypots, SCADA Security.

Abstract: While frequent reports on targeted attacks for Industrial Control Systems hit the news, the amount of untargeted attacks using standardized industrial protocols is still unclear, especially if devices are mistakenly or even knowingly connected to the Internet. To lay the foundation for a deeper insight into the interest of potential attackers, a large scale honeynet system that captures all interactions using industrial protocols is proposed. Special for the honeynet system architecture is the automated deployment on a cloud infrastructure and its modularisation of the industrial protocols. The centralized-but-redundant data collection allows correlating attacks that happen on multiple devices. A real-world experiment confirms the feasibility of the approach, and results of the observed interactions with the honeynet are presented.

## 1 INTRODUCTION

A large subset of critical infrastructures, from energy production and distribution (e.g. nuclear power plants, gas pipes, the electrical grid) over utilities (e.g. water and wastewater treatment and transportation) to the transport and manufacturing industries, relies on industrial control systems (ICS) for controlling and monitoring the underlying processes and physical functions in real-time. A SCADA system (supervisory control and data acquisition) is a special type of industrial control system, mainly targeted to wide area industrial systems and focused on data acquisition and collection aspects.

Initially air-gapped and often using proprietary protocols, ICS/SCADA systems had their security ensured by means of physical isolation and obscurity, which are currently considered unsafe practices (NIST, 2008). ICS/SCADA systems have since evolved into being integrated into corporate networks (e.g. with resource planning solutions) and widely-deployed, complex systems such as smart grids or advanced metering infrastructures.

As tools are openly available for identifying and compromising ICS/SCADA components accessible on the public Internet, the level of skill and resources required to find and perform attacks against them is reduced (ICS - CERT, 2013). Hence, attacks on critical infrastructures are occurring, with the threat landscape having been recently explored in greater detail (Robinson, 2013) (Morris and Gao, 2013).

However, there is little publicly-available information on the adversarial behaviour and on the amount of interactions between attackers and their targets. Moreover, the efforts made so far have been focused on investigating how classical IT attacks apply to ICS/SCADA systems, with a notable lack of information on attacks carried out using ICS/SCADA-specific protocols. In addition, the influence of having ICS/SCADA components indexed by search engines on the presumably-malicious activities performed using industrial protocols represents almost uncharted territory. Finally, even though some existing tools could be combined and partially used to investigate these matters at a small scale, there is no single solution for achieving this goal, let alone at a larger (i.e. Internet-wide) scale.

### 1.1 Contributions

The most important contributions can be summarised as follows:

- Presentation of a novel ICS/SCADA honeypot architecture that can be globally deployed in an

[*] All related research activities were performed entirely under simultaneous affiliation with ABB Corporate Research and the Department of Computer Science of ETH Zurich. KPMG AG represents the current affiliation only.


Vlad Serbanescu A., Obermeier S. and Yu D..
A Flexible Architecture for Industrial Control System Honeypots.
DOI: 10.5220/0005522500160026
In *Proceedings of the 12th International Conference on Security and Cryptography* (SECRYPT-2015), pages 16-26
ISBN: 978-989-758-117-5

automated manner on heterogeneous deployment platforms, thereby forming a large-scale honeynet. Furthermore, the mimicking functionality together with the security components integrated are modularly-structured and can thus be easily extended.

- Proof-of-concept ICS/SCADA honeynet implementation, including means of automatically deploying it to various on-premises networks or cloud platforms.
- Experimental results revealing information on the adversarial behaviour targeting critical infrastructures, thus advancing the knowledge of the related threat landscape.

## 2 ASSUMPTIONS AND RESEARCH QUESTIONS

We developed our analysis on the following assumptions for investigating the ICS/SCADA-protocols-based adversarial behaviour pertaining to critical infrastructures:

1. Attempts to identify ICS/SCADA systems that belong to critical infrastructures are happening more and more frequently. One of the main techniques to achieve this goal is port scanning. This is a consequence of systems that are part of critical infrastructures attracting the attention of malicious entities, ranging from script kiddies and individual hackers to hacktivist groups and nation states, as listed in (Asgarkhani and Sitnikova, 2014, p.27).
2. Publicly-available tools (e.g. nmap) and information repositories (e.g. the SHODAN search engine[2]) facilitate the aforementioned reconnaissance activities. The latter is used for what is referred to as "indirect intelligence gathering" in (Byres, 2013). The work in (Wilhoit, 2013a, p. 21) shows that many of the attacks "started out with SHODAN queries, but it cannot be accurately said if these were accidental or targeted in nature".
3. If an ICS/SCADA system is identified on the public Internet (e.g. by identifying its gateway device), the classic IT components are most likely the first target for an alleged attacker, not the ICS/SCADA components. Thus, classic and well-documented attack techniques are usually employed, such as brute forcing credentials or exploiting SQL injection vulnerabilities of a web-based GUI. Moreover, attacks tailored to lower level ICS/SCADA equipment are unlikely to be currently seen on the public Internet. This hypothesis is in line with the experimental results in (Buza et al., 2014, p. 191), in which the honeypot collected many general PC targeting attacks, but neither of them was specific to programmable logic controllers (PLCs), which are typically used for the actual control of the industrial processes. However, the rare attacks that do exploit the ICS/SCADA layer are targeted and sophisticated, as shown in (Wilhoit, 2013b) and (Wilhoit, 2013a).

Based on these assumptions, the goal of our experiments was to answer the following questions:

1. Although attacks at the ICS/SCADA level seem unlikely over the public Internet, can we observe attempts to identify using industrial protocols (e.g. Modbus, IEC-104) ICS/SCADA equipment that is directly (and hence wrongly) accessible from the public Internet?
2. Do ICS/SCADA components that are directly connected to the Internet draw attention, and are there attempts to interact with them in a short amount of time (on the order of days or possibly even hours, as observed for example in (Wilhoit, 2013b))?
3. How fast do information repositories (e.g., SHODAN) interact with and list newly-connected devices?
4. If an ICS/SCADA-specific device is identified, how is it typically interacted with (e.g., port scan, connection attempts, exchange of protocol-level data)?

In addition, exploring the feasibility of utilising a single framework for monitoring the ICS/SCADA-related threat landscape at a large scale has been established as a goal as well.

## 3 RELATED WORK

One of the first attempts of translating honeypots to the ICS/SCADA application domain is represented by the SCADA HoneyNet Project, whose authors tried in 2004 to evaluate the "feasibility of building a software-based framework to simulate a variety of industrial networks such as SCADA, DCS [distributed control system, another specific type of ICS], and PLC architectures"[3]. Their aim was to simu-

[2] SHODAN Computer Search Engine. http://www.shodanhq.com/

[3] V. Pothamsetty and M. Franz. SCADA HoneyNet Project: Building Honeypots for Industrial Networks. http://scadahoneynet.sourceforge.net/

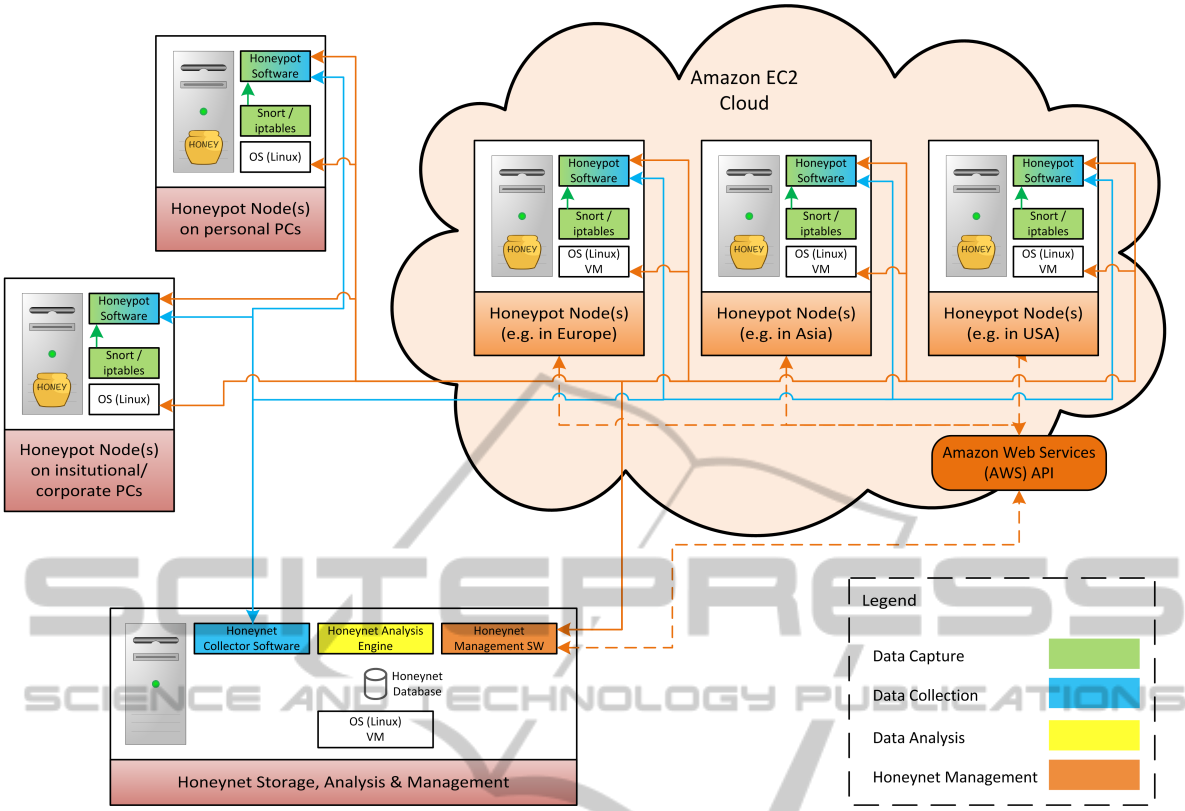


Figure 1: Cloud-based industrial honeynet architecture.

late a PLC using Honeyd[4] (a popular low-interaction honeypot) to mimic a set of protocols and services specific to industrial networks in order to address the lack of (publicly-available) information about SCADA vulnerabilities and attacks and of publicly available SCADA security tools [3]. The main contributions include scripts that simulate protocols such as FTP, Modbus TCP and Telnet, and a web server component. The work on this project was discontinued in 2005. The simulation achieved is limited with respect to the functions and operations implemented, e.g. by comparing the Modbus synchronous simulation script to the interactive, full Modbus functionality, and not designed for automated deployment on cloud infrastructures.

Building on the HoneyNet Project, Digital Bond, Inc. (a control system security consulting and research company) has developed its own version of a SCADA Honeynet[5], which uses two virtual machines as its foundation instead of the Honeyd honeypot. The simulation of a popular PLC and of its exposed services (FTP, Telnet, Modbus TCP, HTTP, SNMP) is carried out by one virtual machine, while the other one concentrates on monitoring all network activity and statistics[5] and includes tools for this purpose. The approach contains in-house-developed IDS signatures for detecting malicious activity targeting the simulated PLC. The latest version of this project dates back to 2011. A recent example of simulating a control system using this SCADA Honeynet version can be found in (Wade, 2011). The SCADA Honeynet developed by Digital Bond, Inc. also supports the use of a real, hardware PLC instead of a simulated one. The overall architecture proposed, although reflecting substantial improvement over the SCADA HoneyNet Project, is unsuitable for wide-scale deployment and operation (for example because of the lack of management, operation and per-node individualization facilities), and makes functionality and protocol-coverage extensions difficult. Therefore, the approach developed in this paper also focuses on modularization of the protocol simulation functionality.

The most recent implementation of a SCADA honeypot (and also in active development) is Conpot, a “low interactive server side [...] honeypot designed to be easy to deploy, modify and extend”[6]. In its default configuration, it simulates a Siemens

[4]N. Provos. Honeyd Virtual Honeypot. http://www.honeyd.org/

[5]Digital Bond Inc. SCADA Honeynet. http://www.digitalbond.com/tools/scada-honeynet/

[6]The Honeynet Project. CONPOT ICS/SCADA Honeypot. http://www.conpot.org

SIMATIC PLC exposing HTTP, Modbus TCP and SNMP protocols, and also supports the use of production equipment instead of the mimicked components (e.g. human-machine interface, PLC). A design and implementation case-study for a honeypot based on Conpot can be found in (Scott, 2014). According to (Buza et al., 2014, p. 183), Conpot is easy to install and use, but requires a lot of customization effort. The customization and configuration effort represents a common disadvantage of most, if not all publicly-available ICS/SCADA honeypots surveyed in this paper.

In contrast to the approach chosen in this paper, none of the aforementioned ICS/SCADA honeypot/honeynet seamlessly integrates a port scanning detection mechanism – not only from a functional point of view, but also from a collected data aggregation and analysis perspective.

Concerning the use of publicly-available resources for identifying ICS/SCADA systems – in the case of SHODAN, a search engine crawling the Internet and indexing devices according to the service banners returned – recent research efforts such as (Bodenheim et al., 2014) and (Patton et al., 2014) highlight their considerable potential as indirect reconnaissance tools. Specifically, in (Bodenheim et al., 2014) it is argued that SHODAN represents a ”powerful reconnaissance tool for targeting (ICS systems)”, indexing four newly-connected, Internet-facing PLCs in approximately 19 days, whereas in (Patton et al., 2014) roughly 1400 SCADA devices from a SHODAN database subset are found to be using default login credentials. It is also reported in (Wilhoit, 2013a) that, while using a proprietary ICS honeypot, many of the attacks observed started out with SHODAN queries. However, (Wilhoit, 2013a) cannot accurately assert if these were accidental or targeted in nature.

Therefore, being indexed in public information repositories appears to increase the exposure of ICS/SCADA systems to Internet-originating attacks and, hence, the associated risks. However, there is currently no information available on the influence of such listings on the adversarial behaviour using ICS-specific protocols itself.

# 4 APPROACH

The selected approach for investigating the adversarial behaviour pertaining to critical infrastructures is an ICS/SCADA honeynet, which exposes multiple industrial protocols to the Internet. It is deployed on a global scale using the Amazon EC2 cloud environment.

A honeynet-based approach has been chosen because it enables a software-only simulation of ICS/SCADA devices from various manufacturers in large numbers and adaptable configurations. Thus, no deployment or operational constraints other than the software-related ones are faced.

In order to answer the questions concerning the attack landscape, only the presence of ICS/SCADA devices that support various specific protocols has been simulated. It was not necessary to precisely mimic the devices individually since the research objectives currently established are not restricted to a particular device class/type; they are concerned with the overall associated threat landscape. Moreover, with only minimal information available on the particular adversarial behaviour considered, a less-detailed-but-broad initial investigation has been considered more valuable than a deep-but-narrow one, as it provides a good overview of the threat landscape and, hence, support for directing and optimising subsequent research efforts.

As capturing port scanning activities is crucial for relating the amount of targeted interactions to the overall number of connections, we have designed each honeypot to seamlessly integrate Snort, an open-source network intrusion detection system (Beale et al., 2007). Snort is not only specialized on port scan detection; it can also detect attempts to fingerprint the operating system or probes/attacks using various protocols.

## 4.1 Simulated ICS/SCADA Protocols

We have simulated protocols that are utilised for remote SCADA/ICS communication as these protocols could be routed over the Internet and can thus attract malicious interest.

**Modbus** is a de facto standard for industrial communication, ubiquitously used in equipment ranging from small network-enabled sensors over automation controllers to SCADA system components (Wilamowski and Irwin, 2011).

**IEC 60870-5-104** is frequently used in the electrical generation and distribution industry, especially in Europe (where it is the most commonly-used protocol), but also internationally (where it competes with DNP3 in English-speaking countries).

# 5 ARCHITECTURE

The honeynet system primarily consists of multiple honeypot nodes deployed in a scattered manner on the public Internet and which communicate with one or

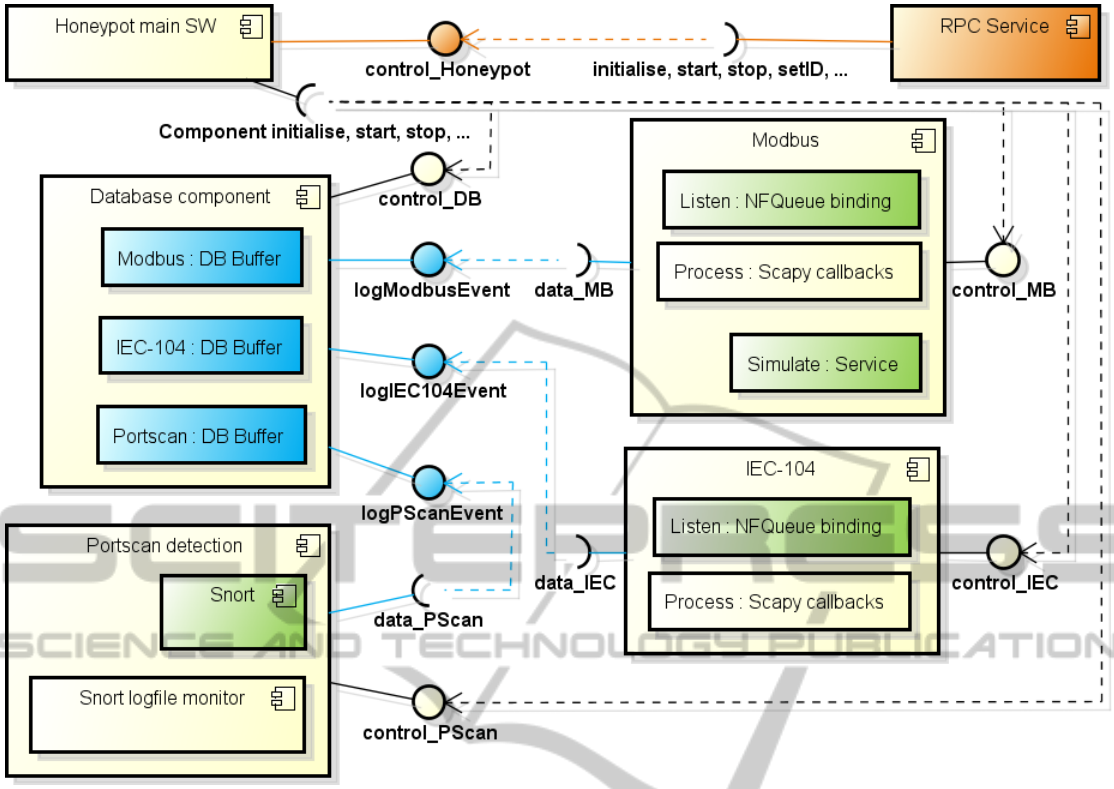


Figure 2: Internal honeypot architecture.

more data storage, analysis and management (SAM) nodes.

The purpose of each honeypot node is to mimic the presence of an ICS/SCADA system component and to monitor for, carry out and record interactions using communication protocols specific to the ICS/SCADA application domain. The nodes can simultaneously simulate different services, according to their individual configuration, thus being able to interact with the environment using different combinations of ICS/SCADA protocols. If a node is interacted with (e.g. port scanned) using a protocol that is not being simulated at that moment, the attempt is also logged. In addition, the honeypot nodes apply local processing to the gathered data before sending it to the SAM node(s) in order to minimise the load on the latter one(s). The information to be captured and stored consists of data at OSI Layer 3 and above, both sent and received.

The internal architecture of the honeynet system used to gather the data analysed in the next section is depicted in Figure 1, with Amazon EC2 as the deployment platform. The modular architecture allows for scalability, easy addition of functionalities and straightforward expansion of the protocols simulated. From a high level perspective, the functionality of the honeynet is organised into *data capture*, *data collection*, *data analysis* and *honeynet management*, hence its structure follows the same logic.

*Data capture* defines the recording of all interactions that occur with the surrounding environment in order to have a raw source of data that can be thereafter analysed and transformed into actionable information/knowledge. Hence, the honeynet processes all interactions, regardless of their characteristics (e.g. completeness, conformance with the corresponding standards) and captures and stores all data situated at OSI Layer 3 and above. This functionality is accomplished locally on each honeypot node at both network and application level.

*Data collection* satisfies the essential need of having all information captured available at one/multiple locations in order to enable a real-time, complete and system-wide image of the honeynet and to be thus able to detect events and discover correlations. According to (Deng and Shukla, 2013), SCADA systems require real-time threat monitoring and early warning systems to identify cyber attacks, whereas effective intrusion detection requires correlation of multiple events that are temporally and/or spatially separated. Therefore, a central data collector has been

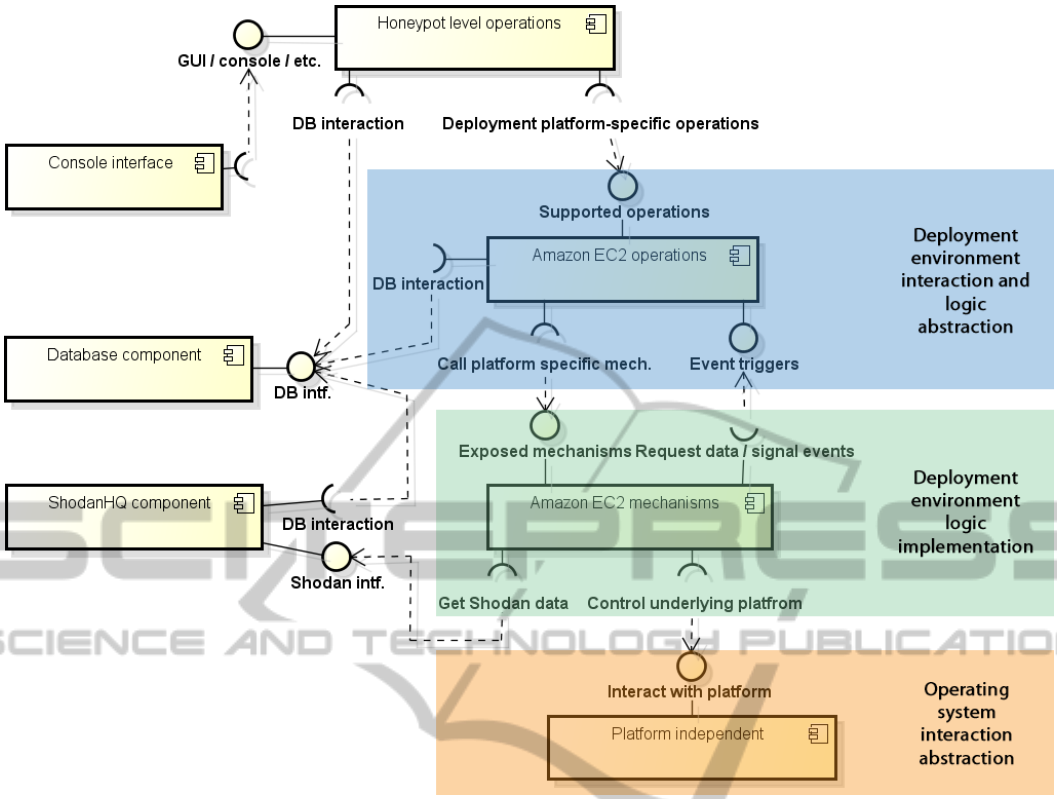


Figure 3: Structure of the honeypot management component.

implemented to allow such correlations between multiple events.

*Data analysis* embodies the key functionality of the honeynet, namely to extract information and to draw conclusions from the data collected. It is performed only on the SAM nodes in order to reduce the computational load on the honeypot nodes. Data analysis is carried out using third-party tools (i.e. Matlab) and custom Python scripts.

*Honeynet management* provides mechanisms for easy configuration, monitoring and deployment (e.g. automated deployment in the Amazon EC2 cloud) and covers all aspects of the honeypot life cycle, from honeypot node bootstrapping over secure remote management and honeynet-wide configuration/code updates to honeypot node shutdown. There are two associated data flows depicted in Figure 1 because the honeynet SAM nodes need not only to control the honeypot software itself, but to also programmatically manage the hosting environment(s) employed using the API(s) exposed (e.g. provisioning a virtual machine on the Amazon EC2 cloud).

The structure of the honeypot software itself is shown in Figure 2, using the same color coding as in Figure 1 for the data capture, data collection and honeypot management functionalities. One of the main goals pursued with this modular structuring has been to enable simple functional extensions, therefore having a minimal number of component interdependencies with uniform underlying logic has been assigned a high priority.

*Honeypot main SW* represents the entry point, which starts in a virtual terminal environment so that its functioning is not dependent on the state of the SSH session and to also allow operator inspection/intervention at any time. The state of all other components (except for the RPC service) is controlled from here, with global commands subsequently calling the required methods in the corresponding components. For example, the command for initialising the honeypot starts with initialising the database component, followed by all protocol simulation services and finishing with the portscan detection module.

The *RPC service* is automatically started in a separate thread for receiving commands and responding to queries from the honeynet controller, which are all tunnelled through an SSH tunnel. The most important control commands to be received using the RPC service are for initialising, starting and stopping the

honeypot software, respectively for setting its unique identifier and for querying its state.

The key activity performed by the honeypot nodes – simulating services and capturing interactions – is realised with *individual components for every service/protocol simulated*. Each such component gathers data using network- and application-level mechanisms and can respond to the messages received, provided that a reply generation function is also implemented. After processing each incoming message, the information thus extracted is sent to the database component.

The *portscan detection* component employs Snort for the actual detection. Hence, the honeypot could also be extended with IPS-specific tasks done by Snort. All events detected are handed over to the database component.

The *database component* handles the data to be sent to the honeynet central database(s). Every simulated protocol has a specific buffer associated for storing the data gathered, which is in turn flushed when full or at regular intervals (whichever occurs first). The port scanning activity detected is also handled by such a buffer.

## 5.1 Data Capture

The capture of data is realised by the sensing points of the honeynet, namely the honeypot nodes, at two different levels on the OSI stack.

**Network level:** For each protocol simulated, iptables entries for both incoming and outgoing traffic on the associated ports are automatically added in order to redirect the raw traffic using netfilter queues to the honeypot software residing at the application level. Here, each packet received is processed by a callback function that either stores it in a temporary buffer if the packet is still needed by other methods or sends it to the associated buffer in the database component directly. In the case of the former, after all methods have finished utilising the captured packet the last one sends the data to the buffer in the database component. This approach has been chosen as it enables using the raw capture and examination of the entire packet (using a packet manipulation tool such as Scapy). In addition, persistently storing the packet dump in the honeynet database becomes also possible – this allows a subsequent detailed inspection of the captured packets using packet analysers.

**Application level:** If there are services implemented for a protocol (e.g. a Modbus server for the Modbus component), each simulated service can integrate the raw data received through the aforementioned netfilter queue redirection and use it internally. For example, the Modbus component makes use of an open-source implementation for simulating a server and by adding hooks at appropriate places specific events can be logged.

An advantage of this data capture mechanism is that tasks such as handling TCP connections can be offloaded from the honeypot software onto the operating system through the use of standard techniques (e.g. creating a TCP socket for listening on a port), thus improving performance.

With regard to the detection of port scans, the output of the port scan detection module is redirected to a file, which is continuously monitored by the honeypot software for file system events. In this way, any port scanning occurrence detected by Snort is immediately processed by the honeynet.

## 5.2 Data Collection

After the data has been captured and processed by the protocol-specific or portscan detection components, it is sent to the associated buffer in the database component. These buffers inherit a base buffer that implements the logic for flushing their content to the database(s) on the SAM nodes and they specialise the structure of the data buffered, the SQL commands employed and the logging API exposed to the data-capture-responsible components. Thus, under normal operation all databases hold identical data. To assure this property under non-ideal conditions, database synchronization mechanisms can be easily added on the SAM nodes with no modification to the honeypot software itself. The logical topology of the honeynet with regard to data collection is hub(s) and spoke, with the SAM node(s) being the hub(s). Multithreaded connection pools are created for each destination database.

## 5.3 Honeynet Management

The management of the honeypot nodes is performed using a honeynet controller component, which is entirely separated form the honeypot software. This structure of the honeynet management software is depicted in Figure 3.

Starting from the top, the *honeypot level operations* component exposes an interface for controlling the honeynet from a high-level, system-wide perspective, i.e. for executing commands such as “launch honeypot node on deployment environment X”. It fulfils the role of the gateway redirecting user commands to the underlying component that can execute them according to the deployment environment they are destined for. In addition, it handles queries that

request honeypot-level information, such as learning the state of the honeypot nodes or inspecting captured packets.

The next layer (blue) contains one component for each deployment environment used by the honeynet in order to handle the commands distributed by the previously mentioned gateway. High-level commands such as "add honeypot node" are broken down into the sequence of main steps to be carried out using the corresponding deployment-environment-specific mechanisms, which are implemented in the green layer below, and/or of queries to be send to the honeynet database(s). In other words, the components belonging to this level incorporate the *logic* behind all operations that the honeynet management software supports. In addition, they also expose interfaces for carrying out actions at the request of the underlying mechanisms targeting the same deployment environment: for example, the illustrated *Amazon EC2 operations* component will add data to the database when the underlying component has finished updating and restarting a honeypot node running on the Amazon EC2 cloud platform.

*Amazon EC2 mechanisms*, together with all other components from the same layer (green), expand the steps exposed to the layer above into the actual operations that have to be executed on the target deployment environment, including possible interactions with the API provided by the latter or with other resources (e.g. the ShodanHQ component). For example, the "launchHoneypot" method will: use the Amazon EC2 API to launch a new instance, request a unique identifier from the overlying component, check whether the services to be simulated by the new honeypot node are already listed on SHODAN (and take appropriate actions if they are), provision the new instance, initialise and start the honeypot software. All these steps have to be translated to actual commands issued for the operating system used on the instance obtained from the deployment environment – this is done in the lowest layer of the honeypot management software, as subsequently introduced.

The (deployment) *platform-independent* component implements the entire set of operations to be carried out on the actual hosting environment, i.e. the underlying operating system. This includes, among others, installing the required software, deploying or updating the honeypot software, pushing the specified configuration files, controlling the honeypot software, and retrieving the log files. The role of this component is to hide the operating-system specific implementation details and provide a unified set of commands to all other components that depend on it. The platform-independent component has been implemented for Ubuntu Linux.

Table 1: Deployment scheme on Amazon EC2.

| Amazon EC2 Region | Honeypots | DBs |
|---|---|---|
| Asia Pacific (Tokyo) | 2 | – |
| Asia Pacific (Singapore) | 2 | – |
| Asia Pacific (Sydney) | 2 | – |
| EU (Ireland) | 3 | 1 |
| South America (Sao Paulo) | 1 | – |
| US East (N. Virginia) | 3 | – |
| US West (N. California) | 2 | – |
| US West (Oregon) | 3 | 1 |
| **Total** | **18** | **2** |

Using this deployment scheme, all Amazon EC2 regions available at the start of the experiment were covered and all their underlying availability zones contained one honeypot instance, with the exception of South America (Sao Paulo) and US East (N. Virginia).

The functionality of the honeynet as a whole is augmented by including access to external sources of information. SHODAN has been chosen for this proof-of-concept implementation. A corresponding component is present – the *ShodanHQ component* in this case – in order to hide the external API and to implement more advanced functionalities (compared to the API-provided ones) in cooperation with other components. In addition, a *database component* is also present, which enables the interaction of the other components with the honeynet database(s).

This approach of structuring the honeynet management software offers the following advantages:

- GUIs or other user interfaces can interact with the management system in a technically-transparent manner. In addition, deployment environments can be added without being forced to change the existing user interfaces or the honeynet management software itself – they only have to add support for the new ones.
- Newly-added deployment environments can utilise already implemented components, mechanisms and/or sources of information. Conversely, existing components can be rapidly and easily integrated and benefit from new information sources and/or new auxiliary modules.
- The logic is separated from its actual realisation, thus keeping change requirements from propagating back and forth between them. Moreover, schema alterations in the honeynet database(s) only affect the former, whereas updates of the external components (e.g. ShodanHQ component) influence only the latter.

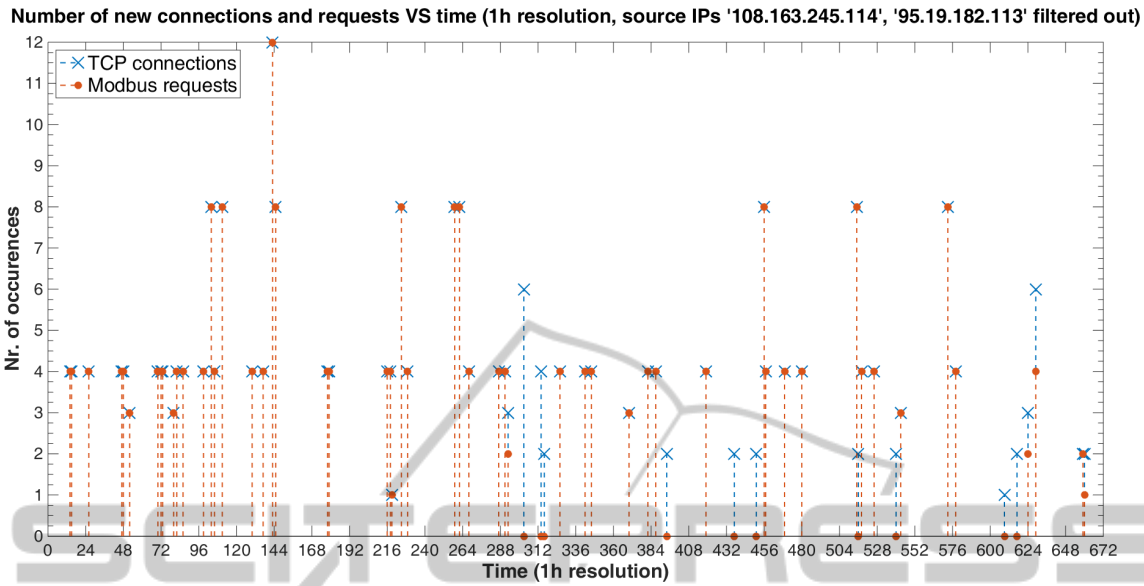


Figure 4: Series of Modbus connections and requests.



- The components that define the operations to be carried out on each operating system can be reused for multiple environments offering the same operating system to the user. Moreover, honeynet management software modifications due to changes in the operating system are restricted to a single component, without affecting the upstream components.

### 5.4 Implementation

Both honeypot node and honeynet management software are implemented in *Python*, with the *Twisted* framework being used for asynchronous operations, network interactions and protocol simulation. The honeynet databases run on *PostgreSQL* DBMS, with which the honeypot nodes communicate through *psycopg2*. The virtual terminal environment employed for running the honeypot software is provided by *screen*.

*Scapy* offers the required support for packet inspection and manipulation tasks and also plays a role in the enumerated protocol simulation, with *nfqueue-bindings* alongside *iptables* being responsible for redirecting raw traffic to/from it. The Modbus server is simulated using *PyModbus*, whereas *Snort* handles the detection of port scanning attempts. In addition, information about the interaction peers is gathered using *dnspython* for reverse DNS lookups and *GeoIP* for their geographical origin.

*Fabric* is used for running commands remotely on the honeypot nodes, while *RPyC* and *Plumbum* enable the secure use of remote procedure calls. *GitHub* has been used for code storage, while code deployment is achieved using the *git* tool. The interaction with the Amazon EC2 and SHODAN APIs is carried out through the *boto* and *shodan* python libraries, respectively.

## 6 EXPERIMENTAL EVALUATION

We have evaluated the framework and implementation through experiments carried out using the Amazon EC2 cloud environment. The protocols exposed are Modbus and IEC 60870-5-104 (IEC 104), and the duration of the experiment was 28 days. During this time, no changes were made to the running honeypots. The deployment scheme for the honeypot nodes is depicted in Table 1.

In order to avoid loss of experimental data, two nodes were used for hosting a database each and for controlling the honeynet, as described in Section 5.2. This also shows the feature of the honeynet to simultaneously log to multiple databases.

As a general note, the interactions are categorized using their type (new connection, request), the protocol employed and their relation to SHODAN (peer belonging to the scanning infrastructure of SHODAN or not). The latter characteristic is determined by performing a reverse DNS lookup of the peer's IP address and checking whether the associated PTR record is a subdomain belonging to SHODAN or not. The information obtained using this method in the case of non-

Table 2: Summary of data collected on Amazon EC2.

| Peer IP addr. | CC | DNS PTR | Modbus | | IEC-104 | | Pscans. |
|---|---|---|---|---|---|---|---|
| | | | Conns. | Reqs. | Conns. | Reqs. | |
| 66.240.192.138 | US | census8.shodan.io. | 48 | 46 | 13 | 13 | 1 |
| 66.240.236.119 | US | census6.shodan.io. | 45 | 43 | 8 | 8 | 2 |
| 71.6.135.131 | US | census7.shodan.io. | 49 | 41 | 6 | 6 | 5 |
| 71.6.165.200 | US | census12.shodan.io. | 44 | 44 | 7 | 7 | 1 |
| 82.221.105.6 | IS | census10.shodan.io. | 8 | 8 | - | - | 2 |
| 82.221.105.7 | IS | census11.shodan.io. | 3 | 3 | - | - | - |
| 85.25.43.94 | DE | rim.census.shodan.io. | 14 | 12 | 7 | 7 | 3 |
| 85.25.103.50 | DE | pacific.census.shodan.io. | 10 | 8 | - | - | 1 |
| 93.120.27.62 | RO | m247.ro.shodan.io. | 4 | 4 | 1 | 1 | 1 |
| 95.19.182.113 | ES | [...].dynamic.jazztel.es. | 5 | 58 | - | - | - |
| 104.54.177.50 | US | [...].rcsntx.sbcglobal.net. | 6 | - | - | - | - |
| 108.163.245.114 | US | server.albrari.com. | 813 | 738 | - | - | 7 |
| 118.192.48.6 | CN | - | 4 | 3 | - | - | - |
| 198.20.69.98 | DE | census2.shodan.io. | 2 | - | 3 | 3 | 4 |
| 198.20.70.114 | US | census3.shodan.io. | 16 | 16 | 6 | 6 | - |
| 188.138.9.50 | US | atlantic.census.shodan.io. | 8 | 8 | 2 | 2 | 3 |
| 202.108.211.62 | CN | - | 13 | 8 | - | - | 4 |
| **Totals (SHODAN):** | | | 251 | 233 | 53 | 53 | 23 |
| **Totals (non-SHODAN):** | | | 841 | 807 | 0 | 0 | 11 |

SHODAN peers can reveal useful details, such as ties to different organizations, presumed owner's identity or the ISP used.

### 6.1 Results

A summary of the experimental data collected is shown in Table 2. The IP address, associated GeoIP source country information (if available) and corresponding DNS PTR record (if any) for each peer that has interacted with the honeynet are listed in the first three columns. The next four columns provide a breakdown of the interaction by protocol and/or type, while the last column contains the number of portscans detected by the Snort module *from the current peer*, i.e., it must not be used to assess the placement attractiveness of the honeypots/honeynet.

The experiments show that SHODAN frequently scans large parts of the Amazon EC2 cloud infrastructure and also lists our honeypot nodes that it discovers.

Figure 4 illustrates the number of new Modbus connections and Modbus TCP requests observed throughout the experiments. For a better visualisation, we have filtered out two source IP addresses that established excessive amounts of new connections (813 occurrences for the first one alone) and issued a disproportionately-large number of requests (738 and 58 for the first and second addresses, respectively).

This experiment reveals that the interactions are happening frequently and regularly. Moreover, it confirms that the deployed honeypots are of interest and have hence attracted determined/interested peers.

The experiments have also shown the feasibility of our approach and have validated the automated deployment concept. In the future, we plan to run more experiments to get a deeper insight into the industrial protocol landscape.

## 7 SUMMARY AND CONCLUSIONS

We have presented an architecture for a honeypot that enables building a large-scale honeynet. This honeynet can be automatically deployed on a cloud infrastructure, and allows for a centralized and redundant collection of data. Particular attention has been given to the modular character of the industrial protocol implementation, which enables a straightforward expansion of the industrial protocols that a honeypot can expose. A special feature is also the seamless integration of Snort, an open source intrusion detection system, which is used to detect port scans and operating system fingerprinting in a sophis-

ticated way. The real-world experiments conducted on a cloud infrastructure show the feasibility of the approach proposed. The honeynet not only was regularly probed and indexed by SHODAN (a search engine that searches for specific types of devices), but it has also attracted multiple seemingly-independent peers to interact with it.

The results of the experiments conducted motivate the fundamental need of having proper security mechanisms in place within industrial control systems that are connected to the Internet, as these systems will be swiftly listed in relevant information repositories, thus possibly becoming further exposed to both classical IT and ICS-specific attacks. Moreover, they indicate that potentially malicious activities targeting industrial control systems and using ICS-specific protocols may be currently conducted on the public Internet.

As future work, we plan to extend the number of protocols simulated and also to evaluate the influence of search engine listings on the adversarial behaviour observed. To summarize, we consider the proposed architecture for an ICS honeynet system an important foundation for obtaining a deeper insight into the interest of potential attackers for industrial devices.

## REFERENCES


Asgarkhani, M. and Sitnikova, E. (2014). A strategic approach to managing security in SCADA systems. In *Proceedings of the 13th European Conference on Cyber warefare and Security*, pages 23–32. Academic Conferences and Publishing International Limited.

Beale, J., Baker, A., Esler, J., Kohlenberg, T., and Northcutt, S. (2007). *Snort: IDS and IPS Toolkit*. Jay Beale's open source security series. Syngress.

Bodenheim, R., Butts, J., Dunlap, S., and Mullins, B. (2014). 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.

Buza, D. I., Juhász, F., Miru, G., Félegyházi, M., and Holczer, T. (2014). CryPLH: Protecting smart energy systems from targeted attacks with a PLC honeypot. In *Smart Grid Security*, Lecture Notes in Computer Science, pages 181–192. Springer International Publishing. http://dx.doi.org/10.1007/978-3-319-10329-7_12.

Byres, E. (2013). Project SHINE: 1,000,000 Internet-Connected SCADA and ICS Systems and Counting. Web page.

Deng, Y. and Shukla, S. (2013). A distributed real-time event correlation architecture for SCADA security. In *Critical Infrastructure Protection VII*, volume 417 of *IFIP Advances in Information and Communication Technology*, pages 81–93. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-642-45330-4_6.

ICS - CERT (2013). Increasing threat to industrial control systems (update A). https://ics-cert.us-cert.gov/alerts/ICS-ALERT-12-046-01A.

Morris, T. H. and Gao, W. (2013). Industrial control system cyber attacks. In *Proceedings of the 1st International Symposium for ICS & SCADA Cyber Security Research*. http://ewic.bcs.org/content/ConWebDoc/51165.

NIST (2008). Guide to General Server Security – Recommendations of the National Institute of Standards and Technology. http://csrc.nist.gov/publications/nistpubs/800-123/SP800-123.pdf.

Patton, M., Gross, E., Chinn, R., Forbis, S., Walker, L., and Chen, H. (2014). Uninvited Connections: A Study of Vulnerable Devices on the Internet of Things (IoT). In *Intelligence and Security Informatics Conference (JISIC), 2014 IEEE Joint*, pages 232–235.

Robinson, M. (2013). The SCADA threat landscape. In *Proceedings of the 1st International Symposium for ICS & SCADA Cyber Security Research*. http://ewic.bcs.org/content/ConWebDoc/51166.

Scott, C. (2014). Designing and implementing a honeypot for a SCADA network. Technical report, The SANS Institute.

Wade, S. M. (2011). SCADA honeynets: The attractiveness of honeypots as critical infrastructure security tools for the detection and analysis of advanced threats. Master's thesis, Iowa State University, Ames, Iowa. http://lib.dr.iastate.edu/cgi/viewcontent.cgi?article=3130&context=etd.

Wilamowski, B. M. and Irwin, J. D. (2011). *The Industrial Electronics Handbook – Industrial Communications Systems*, volume 2 of *The Industrial Electronics Handbook*. CRC Press, Taylor & Francis Group, 2 edition.

Wilhoit, K. (2013a). The SCADA that didnt cry wolf – whos really attacking your ICS equipment? – part deux! Black Hat US 2013.

Wilhoit, K. (2013b). Whos really attacking your ICS equipment? Black Hat Europe 2013.