UNIVERSITÀ DEGLI STUDI DI VERONA

Department of Computer Science

Bachelor's degree in Computer Science

SCADA - Threats People Overlook

Supervisor

Isabella Mastroeni

Graduating

Alex Salvetti, VR371475

Academic year 2016/17

EXECUTIVE SUMMARY

SCADA systems are the smart component that governs most of critical infrastructure and their malfunction reflects immediately on our everyday lives.

Initially designed to function reliably and not to be secure, their exposure to public networks is itself a vulnerability and exposes them to Cyber Attacks.

This research outlines how SCADA devices are exposed to Internet, both in quantity and quality. It provides an estimation of worldwide exposure and compares the data of the two previously chosen countries in 2013: Italy and Switzerland.

As exposure alone would not translate into Risk without active attackers, a honeypot has been deployed to mimic an ICS device. Thousands of probes and exploit attempts have been recorded demonstrating that such risk scenario is more valid than ever.

Tools, previous research, methodologies and code developed or used during the research has been thoroughly detailed in this paper.

CONTENTS

1. Introduction **5**

1.1 Research objectives 5

1.2 Thesis organization 5

2. SCADA systems **6**

2.1 What are SCADA systems? 6

2.2 Supervisory 6

2.3 Control 6

2.4 Data Acquisition 7

2.5 Evolution of SCADA Systems on the Internet 7

3. SCADA Exposure **8**

3.1 What is SCADA Exposure? 8

3.2 Research method 8

3.3 Analysis 8

3.4 Conclusion 10

3.5 Report results (2016) 10

3.6 Known limitations 11

4. Shodan-PHP-REST-API **12**

4.1 Introduction 12

4.2 CLI Usage 13

4.3 Tests class - REST API 13

4.3.1 Tests class - Esperimental method 15

4.3.2 Tests class - Exploits REST API 15

5. Honeypot SCADA **16**

5.1 Introduction 16

5.2 Brief Overview of Honeypots 16

5.3 Types of Honeypots 16

5.4 Benefits of Honeypots on SCADA Networks 17

5.5 Conpot 18

5.5.1 Installing Conpot 19

5.6 Monitoring and Alerting on the Honeypot Results 20

5.6.1 Splunk Configuration 20

5.7 Conpot analyzed results 21

5.7.1 A deeper look on results....22
5.8 Conclusion....24
6. Conclusion....25
7. References....26

ACKNOWLEDGEMENTS

First of all, I would like to warmly thank my parents for the support and the help they have given me over these years. In addition, I wish to thank Francesco Ongaro, employer and mentor, for the precious and countless teachings and the great availability during the writing of this thesis.

I'd like to thank all the teachers who have contributed to my university education and in particular to my rapporteur, Prof. Isabella Mastroeni, for allowing me to develop this thesis, leaving me wide autonomy but giving me valuable suggestions for its implementation.

1. INTRODUCTION

1.1 Research objectives

In recent years, the advent of technologies of information and communication technology, ICT, in the supervisory and control systems of industrial processes has led to a significant increase in risk of cyber attacks, increasingly complex and targeted to compromise production processes. In particular, after Stuxnet, the attention has been pointed to information security in such facilities.

This research aims to demonstrate how many of those systems are exposed on the internet and to sensitize the ICS Community to improve the security of National critical infrastructure, enterprise owned systems, smaller private companies and even individuals.

In order to conduct this research a PHP API for Shodan has been implemented to speed up data collection, while analysing more thoroughly the type of attacks SCADA devices are exposed to, as a case study was simulated a Siemens S7-200 device through the use of a honeypot.

1.2 Thesis organization

The thesis is divided into three parts. The first includes chapter 2 and discusses about SCADA systems in general. The second part, chapters 3 and 4, talks about the SCADA Exposure research and the API used to perform it. Finally, the third part includes chapter 5 and focuses exclusively on the case study.

2. SCADA SYSTEMS

2.1 What are SCADA systems?

SCADA systems (“Supervisory Control And Data Acquisition”) are the smart component that governs many parts of critical infrastructure and their malfunction can be reflected immediately on our everyday lives. In recent years, these systems are based more and more on IT technologies. This has led to a lower system insulation to the outside world, exposing it to a series of new threats that range from malware to real cyber attacks.

2.2 Supervisory

Supervision is the function by means of which a SCADA system makes possible the observation of the state and evolution of the states of a controlled process. This function is one of those that characterize those systems, in the sense that a system does not provide access to information about the current status and/or the historical process observed and/or controlled may not be classified as a SCADA system.

2.3 Control

The control function is the ability of a system to make decisions, concerning the evolution of the state of the controlled process, as a function of evolution of the same process. SCADA systems are commonly understood as systems that have, in the function of data acquisition, the entire chain of capture that conveys information from the sensors to processing and storage system.

The control functions are then concentrated in the computer system, which, once made the appropriate processing procedures, uses the data acquisition system in the reverse direction to change the value of parameters of state of the controlled process.

2.4 Data Acquisition

The data acquisition function of a SCADA system is generally considered to be a pure and simple function of exchange of information, between the part of the system that realizes the supervision and control and the controlled process. It is assumed that no decision process takes place between the supervisory and control structures and controlled process. In cases where this condition is not satisfied we have systems that realize something different, compared to a classical SCADA system.

2.5 Evolution of SCADA Systems on the Internet

Many of these industrial systems were developed in isolation, primarily as hardware systems. These systems required an engineer on-site to interact with and operate them. With an advance in technology and a regression in the number of engineers on the market, telecontrol of these machines became the go-to method of interaction, giving birth to the SCADA industry.

Over time, this technique began employing modems to connect. While this technique still originally offered point-to-point connectivity, further attempts at saving costs resulted in the use of communications networks for telecontrol.

Company-specific protocols began causing conflicts with desired interoperability between devices, and the simultaneous push for firmware and software led to the abandonment of point-to-point connectivity in favor of IP technology.

Security was an afterthought with the evolution of these SCADA systems. These systems were not designed to handle internet-like traffic, and as a result were rife with security vulnerabilities. Despite the knowledge of these vulnerabilities, many infrastructure software vendors maintained that their products were safe, claiming they were “air-gapped” from the internet, meaning that while the systems were all internally connected on a network, none of the system components was exposed to the internet.

3. SCADA EXPOSURE

3.1 What is SCADA Exposure?

SCADA exposure is a research project that I have made in collaboration with ISGroup SRL during my stage and it focuses on providing an estimation of exposed SCADA/ICS devices, outlining conclusions on the world scene and a comparison between two different year releases, 2013 and 2016. Full research and data is available on: scadaexposure.com

3.2 Research method

We used a set of 95 search queries (dorks) against generic search engines such as Google or Bing, as well as more specialized ones such as Shodan, which focus on indexing machines connected to the internet. This represents the largest collection of search queries specialized in finding SCADA devices up to date. We did not only collect them, we improved the queries to get more and more solid results. For example, some dorks were focusing on authentication banners, but in this way one will always and only get authentication protected devices, ignoring unauthenticated ones.

For each dork we performed several interrogations, dividing the results in three sets: Worldwide, to get the global presence of a particular device; CH, to limit the results to Switzerland; IT to limit the results to Italy. These two countries have been chosen as they are neighbouring nations and thus perfect for a comparative analysis.

To perform these interrogations, we developed a Shodan API written in PHP for a faster way to get results, also looking to the future for making new releases every year. This will be described in further detail later on in the paper.

3.3 Analysis

It is known that ICS and SCADA systems are a link between the digital and physical world so the consequences of malfunctioning can be extremely serious. It is also a fact that in complex systems even the failure of non-critical components can cause unplanned collateral damage.

The research is focused on demonstrating that such systems are not "air-gapped" (deployed on a different, separated network) as many suggests, instead such devices are often exposed to random attackers from the Internet. This means that devices, thought to be completely isolated from external attackers, must be re-engineered with modern threat models.

Looking to 2016's research results, using our set of dorks, we discovered a total of 126,533 SCADA devices, categorized in 29 Vendors and 62 Products. Switzerland accounted for 867 ICS devices on a total of 2,864,000 devices (dork country:"CH") connected to Internet and indexed by our data sources. Italy accounted for 3,296 ICS devices on a total of 8,842,000 devices (dork country:"IT").

While Switzerland has 3 times less devices than Italy, we came to the conclusion that Italy has more exposed ICS devices in proportion. This means that the usage of SCADA elements is more pervasive and/or that their security is worse than in Switzerland. Speaking of absolute proportions Italy has 3.8 times more ICS devices than Switzerland, a number that is higher than 3.

Analysis between ICS/SCADA systems:

- 2.6% of world’s SCADA devices are in Italy;
- 0.7% of world’s SCADA devices are in Switzerland;
- Italian and Swiss SCADA devices represent the 3.3% of the global exposure.

Analysis between ICS/SCADA systems and non-SCADA systems:

- 0,04% of Italian exposed devices are SCADA;
- 0,03% of total Swiss exposed devices are SCADA;
- Italy is proportionally more exposed (+26.30%);
- In Italy every 2,680 devices scanned one is SCADA;
- In Switzerland every 3,300 devices scanned one is SCADA;

According to these statistics, by randomly scanning 10 IP per second, you will find one exposed SCADA system every 10 minutes.

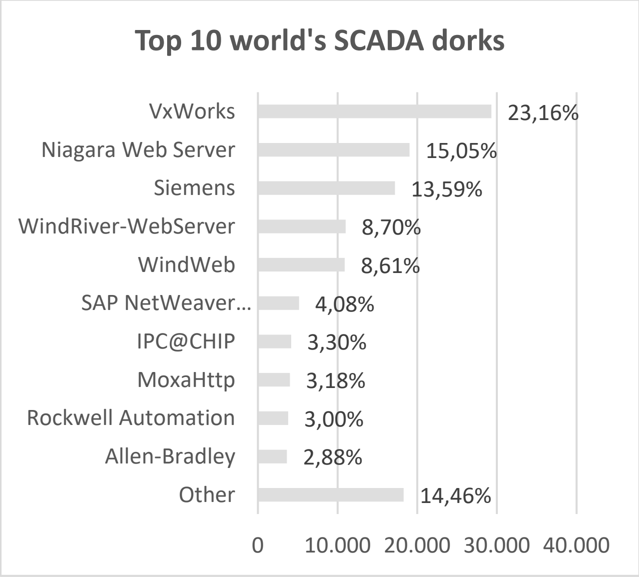
3.4 Conclusion

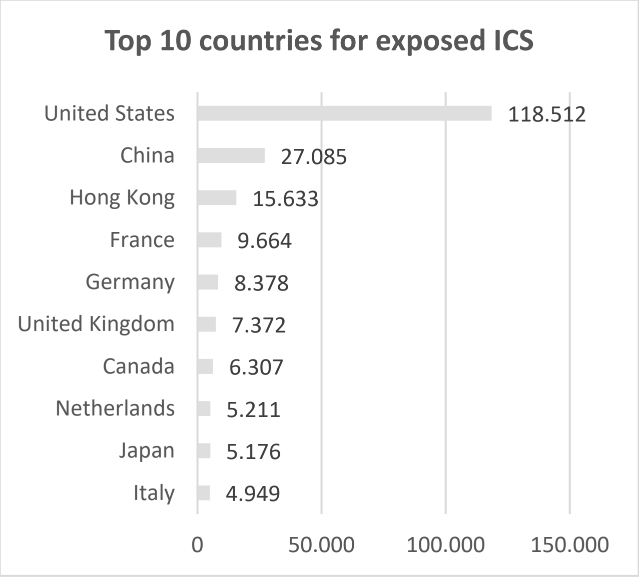
Using the same set of dorks from 2013, we found that those devices are less exposed today. In 2013, a world total of 509,199 SCADA devices were discovered compared to the 126,533 in 2016. We hope that the lesser exposure is real, but it could also mean that some old exposed devices were replaced by newer ICS not observed by our research.

Our research has highlighted a drop of approximately 75% in global terms, while Italy followed the global trend with a drop of approximately 80%, Switzerland has obtained less decrease compared to Italy with a drop of approximately 60%. This could either signify that Italian industries have outperformed Swiss ones in minimizing the exposure of SCADA systems, or that Italy has improved security by switching to new and upgraded ICS devices not present in our comparison.

In a perfect world none of the found systems should be accessible by untrusted networks. Yet, the security of ICS/SCADA is often not given the treatment it deserves. Both small and medium business, as well as individuals completely rely on the vendors when it comes to the security of the Internet of Things. They are rarely made aware of the high risks of being attacked.
SCADA Exposure research reminds us that the “Security through Obscurity” principle cannot serve as a good basis to achieve effective protection from modern attacks.

3.5 Report results (2016)





3.6 Known limitations

We did not connect to the found devices, it was not within the scope of the research and we do not publish specific IP addresses of targets, attackers already actively exploit such information. Publishing it would not add value to our research.

There is no assertion that systems are still reachable from the public Internet or vulnerable at all. They were indexed at some point by our datasources, and thus already exposed to the Internet. This alone represent a violation of most common security best practices, especially when it comes to ICS devices. Only information that seemed accurate and truthful was included in our results.

Search queries used to identify devices are not perfect and may overlap, resulting in larger subtotals and totals than the unique IP count, this may change with future releases if more resources are found to implement the needed screening and filtering.

It is impossible to know what these systems are connected to, and thus it is impossible to know the actual Risk. That is why to effectively assess Risk, Penetration Tests are used. They are real "simulations" of attacks, in the sense that targets are actually attacked but the test is executed in order to avoid deliberate damage. Again, no attack was performed against these devices.

Used datasources do not spider every possible device, they should be used as a statistical indication only. It can be easily understood that devices that appear in such databases have an increased chance to be attacked.

There are actually much more publicly accessible devices than the ones indexed by out datasources. We believe that this compensates known limitations. We were only looking for statistical evidence to reach educated conclusions.

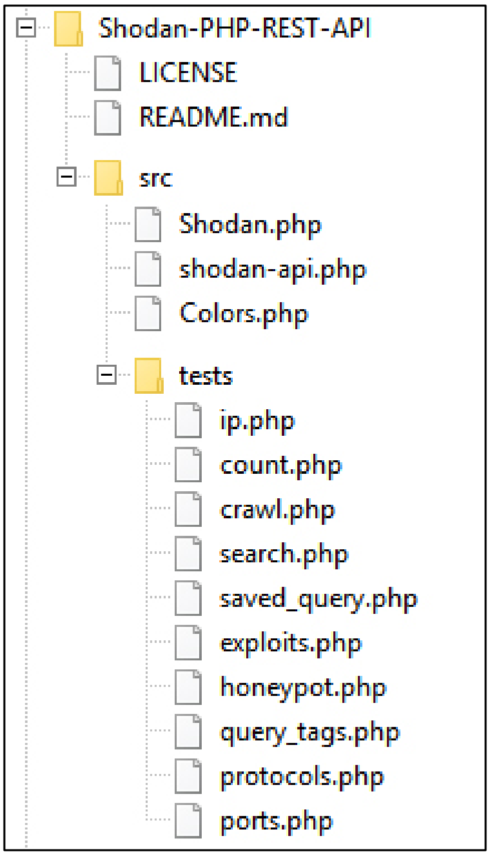
4. Shodan-PHP-REST-API

4.1 Introduction

As mentioned before we used shodan.io for searching exposed SCADA devices, but to make our research more automated and reusable in future we decided to write a PHP API for Shodan.
The API source code is available on github: Shodan-PHP-REST-API.

The project has an API class that is the file Shodan.php, here is where costants, shodan methods and the generation of the HTTP requests are defined.
Shodan-api.php is the CLI interface, allowing us to run differents commands; it also provides an how-to function.
Our API implementation uses 3 different base URLs:

1. Shodan API (api.shodan.io), that is the base URL for all the shodan methods;
2. Streaming API (stream.shodan.io), that is an HTTP-based service that returns a real-time stream of data collected by Shodan;
3. Exploits API (exploits.shodan.io/api), that provides access to several exploit/vulnerability data sources.



```
Shodan-PHP-REST-API
    LICENSE
    README.md
    src
        Shodan.php
        shodan-api.php
        Colors.php
        tests
            ip.php
            count.php
            crawl.php
            search.php
            saved_query.php
            exploits.php
            honeypot.php
            query_tags.php
            protocols.php
            ports.php
```

Figure 1 - Shodan API hierarchy

The class Colors.php was made to print the title of each test with a defined color, thus having better view of the start of the test itself.
Then a tests folder has been made to provide some examples on how to write search queries, we can use the CLI -r flag for running them all or call one with the -t flag.

4.2 CLI Usage

These are the commands that can be used in the command line while running the API:

Short form	Long form	Variables
-r	--run-tests	
-t	--run-test	STRING
-m	--method	ShodanHost --ip STRING [--history BOOLEAN] [--minify BOOLEAN]
-m	--method	ShodanHostCount --query STRING [--facets STRING]
-m	--method	ShodanHostSearch --query STRING [--facets STRING]
-m	--method	ShodanHostSearchTokens --query STRING
-m	--method	ShodanPorts
-m	--method	ShodanProtocols
-m	--method	ShodanScan --ips STRING
-m	--method	ShodanScanInternet --port INTEGER --protocol STRING
-m	--method	ShodanScan_Id --id STRING
-m	--method	ShodanServices
-m	--method	ShodanQuery [--page INTEGER] [--sort STRING] [--order STRING]
-m	--method	ShodanQuerySearch --query STRING [--page INTEGER]
-m	--method	ShodanQueryTags [--size INTEGER]
-m	--method	LabsHoneyscore --ip STRING
-m	--method	Search --query STRING [--facets STRING] [--page INTEGER]
-m	--method	Count --query STRING [--facets STRING]
-m	--method	ShodanBanners
-m	--method	ShodanAsn --asn STRING
-m	--method	ShodanCountries --countries STRING
-m	--method	ShodanPorts_Stream --ports STRING

4.3 Tests class - REST API

Shodan Host (/tests/ip.php): Return all services that have been found on the given host IP.

```
var_dump($client->ShodanHost(array(
    'ip' => '69.171.230.68', // https://www.facebook.com/
)));
```

Shodan Host Count (/tests/count.php): Returns the total number of results that matched the query and any facet information that was requested.

```
var_dump($client->ShodanHostCount(array(
  'query' => 'Niagara Web Server',
)));
```

Shodan Host Search (/tests/search.php): Search Shodan using the same query syntax as the website and use facets to get summary information for different properties. - This method may use API query credits depending on usage.

```
var_dump($client->ShodanHostSearch(array(
    'query' => 'Niagara Web Server',
)));
```

Shodan Host Search Tokens (/tests/search.php): This method lets you determine which filters are being used by the query string and what parameters were provided to the filters.

```
var_dump($client->ShodanHostSearchTokens(array(
    'query' => 'Niagara Web Server country:"IT"',
)));
```

Shodan Ports (/tests/ports.php): This method returns a list of port numbers that the crawlers are looking for.

```
var_dump($client->ShodanPorts());
```

Shodan Protocols (/tests/protocols.php): This method returns an object containing all the protocols that can be used when launching an Internet scan.

```
var_dump($client->ShodanProtocols());
```

Shodan Scan (/tests/crawl.php): Use this method to request Shodan to crawl a network. - POST METHOD REQUIRE PAID API KEY.

```
var_dump($client->ShodanScan(array(
    'ips' => '69.171.230.0/24',
)));
```

Shodan Scan Internet (/tests/crawl.php): Use this method to request Shodan to crawl the Internet for a specific port. - POST METHOD REQUIRE PAID API KEY AND SHODAN PERMISSION.

```
var_dump($client->ShodanScanInternet(array(
    'port' => '80',
    'protocol' => 'dns-tcp',
)));
```

Shodan Scan Id (/tests/crawl.php): Check the progress of a previously submitted scan request.

```
var_dump($client->ShodanScan_Id(array(
    'id' => 'R2XRT5HH6X67PFAB',
)));
```

Shodan Services (/tests/crawl.php): This method returns an object containing all the services that the Shodan crawlers look at. It can also be used as a quick and practical way to resolve a port number to the name of a service.

```
var_dump($client->ShodanServices());
```

Shodan Query (/tests/saved_query.php): Use this method to obtain a list of search queries that users have saved in Shodan.

```
var_dump($client->ShodanQuery(array(
    'page' => '1',
)));
```

Shodan Query (/tests/saved_query.php): Use this method to search the directory of search queries that users have saved in Shodan.

```
var_dump($client->ShodanQuery(array(
    'query' => 'fax',
)));
```

Shodan Query Tags (/tests/query_tags.php): Use this method to obtain a list of popular tags for the saved search queries in Shodan.

```
var_dump($client->ShodanQueryTags(array(
    'size' => '30',
)));
```

4.3.1 Tests class - Esperimental method

Labs Honeyscore (/tests/honeypot.php): Calculates a honeypot probability score ranging from 0 (not a honeypot) to 1.0 (is a honeypot).

```
var_dump($client->LabsHoneyscore(array(
    'ip' => '54.231.184.227', // http://mushmush.org/
)));
```

4.3.2 Tests class - Exploits REST API

Search Exploits (/tests/exploits.php): Search across a variety of data sources for exploits and use facets to get summary information.

```
var_dump($client->Search(array(
    'query' => 'cve',
)));
```

Count Exploits (/tests/exploits.php): This method behaves identical to the "/search" method with the difference that it doesn't return any results.

```
var_dump($client->Count(array(
    'query' => 'cve',
)));
```

5. HONEYPOT SCADA

5.1 Introduction

The goal is to make a more complete search, so the aim was to study what attackers usually do on SCADA devices by looking at already occurred attacks on industries. However, instead of conducting this type of study, I decided to simulate a SCADA device and observe who would be interested in attacking it. For this study I decided to use a Honeypot called Conpot that is specifically designed to simulate SCADA devices.

5.2 Brief Overview of Honeypots

A honeypot is a computer security mechanism set to detect, deflect, or, in some manner, counteract attempts at unauthorized use of information systems. Generally, they take the form of physical or virtual systems that mimic the actual devices on the network, with heavy monitoring and logging so an attacker’s actions can be studied.

Honeypots provide security researchers with a unique opportunity to study their enemy. By analyzing how real-world attacks are taking place, how often they take place, and what attackers are leaving behind (e.g. rootkits, Trojans, and exploits), a researcher can come up with better ways to defend against such attacks.

5.3 Types of Honeypots

Honeypots can be classified based on their deployment (use/action) and based on their level of involvement. Based on deployment, honeypots may be classified as:

1. Production honeypots
2. Research honeypots

Production honeypots are easy to use, capture only limited information, and are used primarily by companies or corporations. Production honeypots are placed inside the production network with other production servers by an organization to improve their overall state of security.

Normally, production honeypots are low-interaction honeypots, which are easier to deploy. They give less information about the attacks or attackers than research honeypots.

Research honeypots are run to gather information about the motives and tactics of the attackers targeting different networks. These honeypots do not add direct value to a specific organization; instead, they are used to research the threats that organizations face and to learn how to better protect against those threats.

Based on design criteria, honeypots can be classified as:

1. Pure honeypots
2. High-interaction honeypots
3. Low-interaction honeypots

Pure honeypots are full-fledged production systems. The activities of the attacker are monitored by using a casual tap that has been installed on the honeypot's link to the network. No other software needs to be installed. Even though a pure honeypot is useful, stealthiness of the defense mechanisms can be ensured by a more controlled mechanism.

High-interaction honeypots imitate the activities of the production systems that host a variety of services and, therefore, an attacker may be allowed a lot of services to waste his time. In general, high-interaction honeypots provide more security by being difficult to detect, but they are expensive to maintain.

Low-interaction honeypots simulate only the services frequently requested by attackers. Since they consume relatively few resources, multiple virtual machines can easily be hosted on one physical system, the virtual systems have a short response time, and less code is required, reducing the complexity of the virtual system's security.

5.4 Benefits of Honeypots on SCADA Networks

There are several benefits to running a honeypot on a SCADA network as opposed to using other security measures. A honeypot does not modify existing SCADA network configurations including firewall and unified threat manager (UTM), or require the insertion of additional inline devices.

Installing inline devices would cause downtime and, depending on the architecture, might be a single point of failure.

The honeypot is simply plugged into the network as any other system would be and set up to run services that look like other devices on your SCADA network. Because the honeypot is not inline and not actually blocking malicious traffic, like an IPS would, this reduces the likelihood of network downtime caused by false positives.

Another advantage of a honeypot is that it can be configured to look like specific devices on a typical SCADA network. That is, it does not have to look like a generic IIS server on Windows, an OpenSSH service on Linux, or even a Telnet service on a Cisco router. You can configure the honeypot to look like a heating, ventilation and cooling (HVAC) system, building access control system (BACS) or industrial control system (ICS). This allows for the monitoring of attacks designed specifically to target the current infrastructure.

For all the upsides of running a honeypot on a SCADA network, there is one major downside: The system has to be monitored and alerts acted upon. Unlike an IPS (Intrusion Prevention Systems), a honeypot will not automatically block attacks. Network security staff time may have to be allocated for monitoring, investigation, and remediation.

5.5 Conpot

As mentioned before, this research uses a low-interaction honeypot called Conpot, which, by default, simulates a Siemens SIMATIC S7-200 PLC, including the Modbus TCP, SNMP, and HTTP protocols.

Conpot is affiliated with the Honeynet Project and can be optionally configured to report back attack data to that project, in order to provide researchers with an idea about how widespread attacks are.

Conpot is a low interactive server side Industrial Control Systems honeypot designed to be easy to deploy, modify and extend. It provides a range of common industrial control protocols giving the basics to build your own system, capable to emulate complex infrastructures to convince an adversary that he just found a huge industrial complex.

The response times of the services can be artificially delayed to mimic the behaviour of a system under constant load. As it provides complete stacks of the protocols, Conpot can be accessed with productive HMI's or extended with real hardware.

5.5.1 Installing Conpot

For the purpose of this research, a common notebook with enough power to run the honeypot was used. Even if it could have been running on a low power device such as Raspberry pi, on the same machine was also been used Splunk. Splunk is a platform for Operational Intelligence to search, monitor, analyze and visualize machine data. Having Splunk on the same machine running at the same time with Conpot may result a little bit of overloading for a Raspberry pi.

Installing Conpot on Ubuntu 16.10 is very straightforward, though Ubuntu will require a number of dependencies. The following dependencies were installed before installing Conpot:

```
$ sudo apt-get install libsmi2ldbl snmp-mibs-downloader python-dev libevent-dev
libxslt1-dev libxml2-dev
```

Then the development version of Conpot was installed:

```
$ cd /opt
$ git clone git@github.com:glastopf/conpot.git
$ cd conpot
$ python setup.py install
```

And finally Conpot was launched for starting logging all the attacks:

```
$ conpot --template default


                       _
   ___ ___ ___ ___ ___| |_
  |  _| . |   | . | . |  _|
  |___|___|_|_|  _|___|_|
              |_|

  Version 0.5.1
  MushMush Foundation

2016-10-31 00:01:07,875 Starting Conpot using template:
/usr/local/lib/python2.7/dist-packages/conpot/templates/default
2016-10-31 00:01:07,875 Starting Conpot using configuration found in:
/usr/local/lib/python2.7/dist-packages/conpot/conpot.cfg
2016-10-31 00:01:08,983 Fetched ---.---.---.--- as external ip.
2016-10-31 00:01:08,985 add_slave() takes exactly 2 arguments (4 given)
2016-10-31 00:01:08,986 Found and enabled ('modbus', <class
'conpot.protocols.modbus.modbus_server.ModbusServer'>) protocol.
2016-10-31 00:01:08,988 Conpot S7Comm initialized
2016-10-31 00:01:08,988 Found and enabled ('s7comm', <class
'conpot.protocols.s7comm.s7_server.S7Server'>) protocol.
2016-10-31 00:01:08,990 Found and enabled ('http', <class
'conpot.protocols.http.web_server.HTTPServer'>) protocol.
2016-10-31 00:01:08,991 Found and enabled ('snmp', <class
'conpot.protocols.snmp.snmp_server.SNMPServer'>) protocol.
2016-10-31 00:01:08,993 Conpot Bacnet initialized using the
/usr/local/lib/python2.7/dist-
packages/conpot/templates/default/bacnet/bacnet.xml template.
2016-10-31 00:01:08,993 Found and enabled ('bacnet', <class
'conpot.protocols.bacnet.bacnet_server.BacnetServer'>) protocol.
```

```
2016-10-31 00:01:08,994 IPMI BMC initialized.
2016-10-31 00:01:08,994 Conpot IPMI initialized using
/usr/local/lib/python2.7/dist-packages/conpot/templates/default/ipmi/ipmi.xml
template
2016-10-31 00:01:08,995 Found and enabled ('ipmi', <class
'conpot.protocols.ipmi.ipmi_server.IpmiServer'>) protocol.
2016-10-31 00:01:08,995 No proxy template found. Service will remain
unconfigured/stopped.
2016-10-31 00:01:08,995 Modbus server started on: ('0.0.0.0', 502)
2016-10-31 00:01:08,996 S7Comm server started on: ('0.0.0.0', 102)
2016-10-31 00:01:08,996 HTTP server started on: ('0.0.0.0', 80)
2016-10-31 00:01:09,209 SNMP server started on: ('0.0.0.0', 161)
2016-10-31 00:01:09,211 Bacnet server started on: ('0.0.0.0', 47808)
2016-10-31 00:01:09,211 IPMI server started on: ('0.0.0.0', 623)
2016-10-31 00:01:13,998 Privileges dropped, running as "nobody:nogroup"
```

5.6 Monitoring and Alerting on the Honeypot Results

The honeypot was operating on my home network 24/7 but I wanted to be able to see if someone was attacking the honeypot even when I was not home. So if an attacker’s activity generates logs on the honeypot, I will not know about it unless logged directly into the honeypot itself.

Conpot can be configured to send logs to a remote Syslog server, which solves part of the problem as the logs can then be sent to a remote system. However, there still needs to be a good way to view and search through the logs, as well as send alerts when there is activity. A useful tool for doing this is Splunk Enterprise, which will index logs for easier searching and alerting.

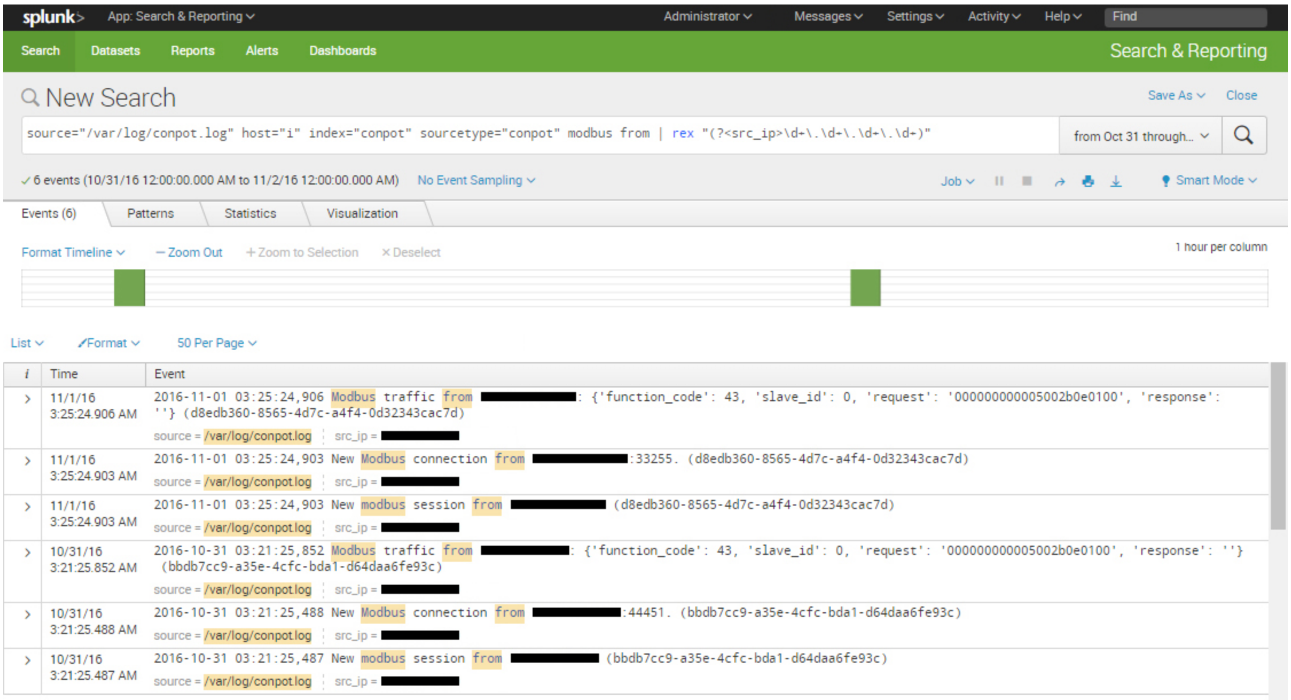
5.6.1 Splunk Configuration

Splunk’s configuration is based on indexes, which are repositories for Splunk data. While all logs can be sent to the same index, having multiple indexes makes searching easier and more granular. Therefore, the first thing to be done as a Splunk administrator is to create an index by going to the Settings menu and choosing Data ➡ Indexes. Then add a new index called “conpot” and save it with the default values.

After the index is created, it is time to add a data source. This is done from the Splunk home screen by clicking on the “Add Data” button. Choose the data type as “A file or directory of files” and on then choose to consume any file on the Splunk server. Browse the server for the location of /var/log/conpot.log.

Source types tell Splunk how to deal with certain types of data formats. For instance, there are source types for Syslogs and Apache logs. Splunk does not come with a source type that matches the format of Conpot logs, so it is best to select to “Start a new source type.” In Conpot, each line of a log is a discrete event, so choose “Every line is one event.”

So now name the new source type as “conpot” and finally, set “conpot” as the destination index. When finished, a simple search on the “index=conpot” or “sourcetype=conpot” will show the Conpot log entries so far, though the search can be further refined. The following screen capture shows a search for Modbus connection and with a regular expression for extract the attacker's ip address.



```
source="/var/log/conpot.log" host="i" index="conpot" sourcetype="conpot" modbus from | rex "(?<src_ip>\d+\.\d+\.\d+\.\d+)"
```

✓ 6 events (10/31/16 12:00:00.000 AM to 11/2/16 12:00:00.000 AM)

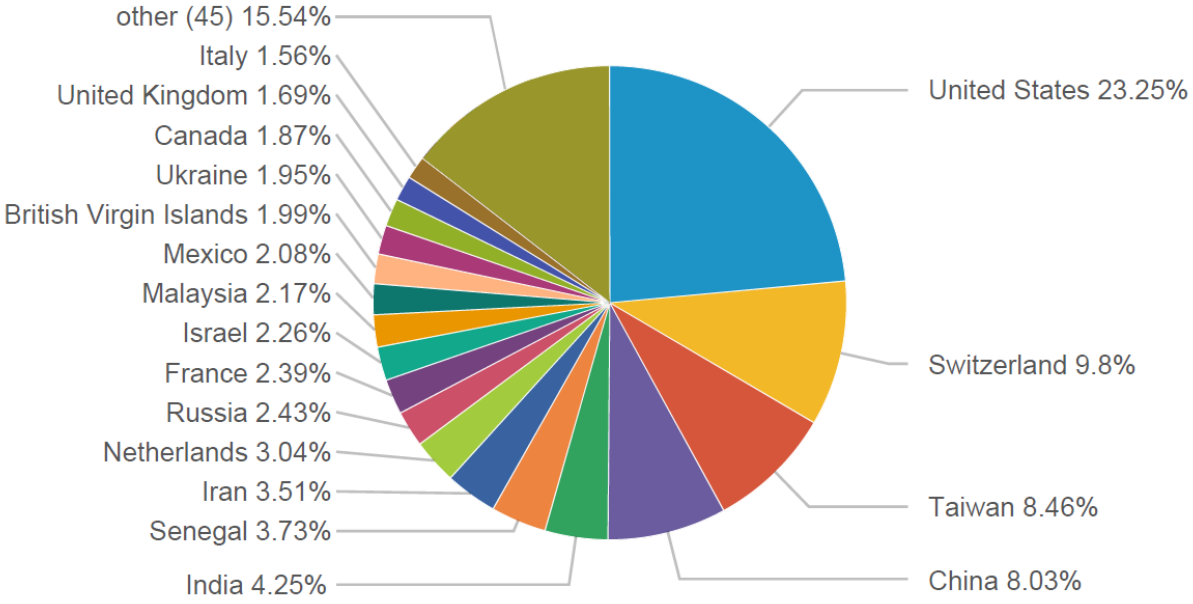
i	Time	Event
>	11/1/16 3:25:24.906 AM	2016-11-01 03:25:24,906 Modbus traffic from [redacted]: {'function_code': 43, 'slave_id': 0, 'request': '000000000005002b0e0100', 'response': ''} (d8edb360-8565-4d7c-a4f4-0d32343cac7d) source = /var/log/conpot.log src_ip = [redacted]
>	11/1/16 3:25:24.903 AM	2016-11-01 03:25:24,903 New Modbus connection from [redacted]:33255. (d8edb360-8565-4d7c-a4f4-0d32343cac7d) source = /var/log/conpot.log src_ip = [redacted]
>	11/1/16 3:25:24.903 AM	2016-11-01 03:25:24,903 New modbus session from [redacted] (d8edb360-8565-4d7c-a4f4-0d32343cac7d) source = /var/log/conpot.log src_ip = [redacted]
>	10/31/16 3:21:25.852 AM	2016-10-31 03:21:25,852 Modbus traffic from [redacted]: {'function_code': 43, 'slave_id': 0, 'request': '000000000005002b0e0100', 'response': ''} (bbdb7cc9-a35e-4cfc-bda1-d64daa6fe93c) source = /var/log/conpot.log src_ip = [redacted]
>	10/31/16 3:21:25.488 AM	2016-10-31 03:21:25,488 New Modbus connection from [redacted]:44451. (bbdb7cc9-a35e-4cfc-bda1-d64daa6fe93c) source = /var/log/conpot.log src_ip = [redacted]
>	10/31/16 3:21:25.487 AM	2016-10-31 03:21:25,487 New modbus session from [redacted] (bbdb7cc9-a35e-4cfc-bda1-d64daa6fe93c) source = /var/log/conpot.log src_ip = [redacted]

5.7 Conpot analysed results

The honeypot was left running through the entire month of November 2016, and as mentioned before it was launched with the default template that simulates a Siemens SIMATIC S7-200 PLC.
This summary will analyse how many connections the Honeypot got and from where they came, and the different types of attacks it was able to record.

The total connections amounts to **2,305**; this, obviously, removing the number of tests I have done to see if everything was working properly. It means that, on average, I have received 77 connections per day, and considering that the IP address of the project was nott known by anyone apart me, I think that is quite an impressive value.

Speaking about which countries are the “key players” connecting during this research, this image made with Splunk shows the results:



5.7.1 A deeper look on results

As we can imagine, not all the connections shown before are to consider alarming, so now we will look for possible dangerous attacks.

Our emulated SCADA device has 6 services running on different ports open to the internet, that can be classified in terms of severity level:

Protocol	Severity	Risk definition
SNMP	Medium	Provides an attacker the possibility to do reconnaissance on a system, and then for finding an exploit to use on the machine.
HTTP	High	It is the service someone might use to configure the mimicked S7-200.
Modbus	Critical	Using the function codes to modify a master or slave program, an attacker can take advantage of the security flaws in the design of Modbus protocol.
S7Comm	Critical	The attacker can control the CPU’s internal operational state, change logical operations in the PLC’s memory and or shutdown processes connected to the PLC.
Bacnet	Critical	Many attacks can be executed on this protocol that controls building automation system (BAS).
IPMI	Critical	Provides an attacker endless possibility for control the SCADA device, this vulnerability resides on the protocol itself because is meant to manage both the physical and software aspect of the server, remotely and locally.

Now that we gave them a severity level we can see how many of those attacks the honeypot has received, ordered by number from highest to lowest.

1) The highest number of connections are **HTTP**, which we have to split in different types:
   a) HTTP GET request (962): are the number of times the page has been requested by IPs but this number has been altered by GoogleBot that alone requested 342 times the honeypot page.
   b) HTTP POST request (11): are the number of times the page received some input from IPs.

2) In second place we have **SNMP** with a number of 71 sessions, those as mentioned before has medium level of severity as the only thing an eventual attacker can do is do reconnaissance on the SCADA device.

3) Third place belongs to **BACNET** with 37 sessions, that are to be considered more severely due to the control this protocol gives and how it is exposed to many possible threats:
   a) **Password Attacks**: brute force or protocol attacks to gain access to a workstation.
   b) **Data Confidentiality**: data is not protected and is available to the outsider who gains access to the network.
   c) **Data Integrity**: device properties can be modified, data changed or erased, router tables modified, etc.
   d) **Denial of Service**: BACnet DoS attacks can shut down the BAS by overloading the network with useless valid and invalid packets.
   e) **Spoofing Attacks**: forging the source address so that an attack looks like it was initiated by another machine.

4) Fourth place goes to **MODBUS** with 34 traffic connections, but as it is Critical severity level we have to consider this value as someone trying to modify parameters with the aim of taking control on the SCADA device.

5) Fifth place goes to **IPMI** with 28 traffic connections, but as it is Critical severity level we have to consider these connections as possible takeover of the machine, both in hardware and software manner, due to IPMI protocol capabilities.

6) And last but not least **S7comm** with 9 connections that it is another Critical severity level protocol that gives many possibilities of control the CPU state of the targeted machine due to S7-PLC Authtentication and Memory Protection Flaws:

   - If an attacker has captured packets containing the authenticated server session, they can re-authenticate using the same packet and bypass that level of protection without ever needing any physical access to the engineering workstation or the PLC.
   - It is possible to read and write data to the PLC's memory even when the passoword protection is enabled, sensitive information from the PLC can be retrieved through memory dumping and it is also possible to disable the password protection feature on the PLC by flipping the security bit back to an OFF state.

5.8 Conclusion

Considering the data acquired during the month while the honeypot was run, we can come to a conclusion that SCADA devices are under possible attack every day.
SCADA networks typically contain business-critical and mission-critical devices. Consequently, anything that might cause support or downtime issues, such as an antivirus, IDS, or firewall, is often avoided. A low-interaction honeypot can be an effective means of detecting hostile scanning and other activity on a SCADA network without modifying the existing network and system configurations.

The next step to make a more complete research would be using a network monitoring program such as Wireshark, acquire all the packets regarding the services used by the honeypot and deeply study the attacks performed on the honeypot.
For the purpose of this research we will not make any step further than just giving the quantity and quality of those attacks.

6. CONCLUSION

The cybersecurity of ICS is closely tied with the physical safety of populations. Yet, the security of ICS is often not given the treatment it deserves. Small and medium businesses, as well as individuals, are completely reliant on vendors when it comes to the security of the Internet of Things. Consumers do not go beyond the simple basic steps from device manuals, obtaining ready-to-work and easily accessible, but also vulnerable devices. On the other hand, in the enterprise and government sectors, companies understand the high risks associated with the incorrect configuration of an ICS environment.

However, because of that, system owners often consider ICS devices as “black-boxes”, and dread making changes in their environments, including the enhancement of cybersecurity measures.

Nowadays, ICS owners should be aware of modern vulnerabilities and threats, and actively improve the security of their ICS environments based on this knowledge. Here, active vendor support is crucial for the prompt identification and remediation of vulnerabilities in ICS products, as well as for sharing workarounds to protect systems before patches are released.

7. REFERENCES

- ISGroup SRL. (2016) *SCADA Exposure research*: http://scadaexposure.com
- ISGroup SRL. (2016) *Shodan-PHP-REST-API*: https://github.com/ScadaExposure/Shodan-PHP-REST-API
- Wikipedia. (2017) *BACnet*: https://en.wikipedia.org/wiki/BACnet
- David G. Holmberg. (2003) *Enemies at The Gates, Securing the BACnet Building*: http://www.bacnet.org/Bibliography/BACnet-Today-03/20839Holmberg.pdf
- Wikipedia. (2017) *IPMI*: https://en.wikipedia.org/wiki/Intelligent_Platform_Management_Interface
- L. Rist, D. Haslinger, J. Smith, J. Vestergaard. (2016) *Conpot*: http://conpot.org/
- Splunk Inc. (2016) *Splunk Enterprise*: https://www.splunk.com
- Wikipedia. (2017) *SCADA*: https://en.wikipedia.org/wiki/SCADA
- Wikipedia. (2017) *Honeypot*: https://en.wikipedia.org/wiki/Honeypot_(computing)

Credits: Alex Salvetti, alex.salvetti@hotmail.it