

ICS Devices

Daniel Thorén, János Dani

May 16, 2020

Contents

1	Introduction	3
2	Related work	3
3	Background	3
3.1	ModBus	3
3.2	Tridium Fox	3
3.3	Shodan	4
3.3.1	TopICS Scandinavia	4
3.4	Methods to identify devices	4
4	Method	5
4.1	Tridium Fox	5
4.2	Modbus	5
5	Result	6
5.1	Tridium Fox	6
5.1.1	NSA325v2	6
5.2	Modbus	7
5.2.1	Anybus M-Bus	7
5.2.2	AWU 500	8
5.2.3	CMe3100	8
5.2.4	Corrigo controller	9
5.2.5	CS141	9
5.2.6	DCS-5222L	10
5.2.7	iR3220	10
5.2.8	Netbiter WS100	11
5.2.9	Solar-Log 1200	11
5.2.10	Web relay	13
6	Discussion	14
7	Future work	15
8	Conclusion	15
9	Dictionary	15

10 Appendix	16
10.1 Search queries	16
10.2 Devices	16
10.2.1 Anybus M-Bus	16
10.2.2 AWU 500	18
10.2.3 Solar-Log	19
10.3 Code	20
10.3.1 modbus.py	20
10.3.2 shodanRun.py	21
10.3.3 matches.py	22
11 Sources	27

Abstract

There are many devices that have open ports on the internet many of which are Industrial Control Systems (ICS). ICS devices are used to control numerous things in the industry such as valves, pumps, air drums and more. The software running on these devices are updated seldom, if at all, which might leave them vulnerable to attacks.

The goal of this report is to investigate a few protocols used for such devices as well as some ICS devices using those protocols. The scope of the report is limited to the Nordic region.

1 Introduction

It is well known that there are many ICS devices with open ports to the internet as can be seen in the ICS count table 2. This report will look at some of the most popular protocols for ICS devices and what kind of devices can be found when searching for open ports on the internet. A good tool to use for the search of open ports is Shodan, which continuously crawls the web for open ports on random ips and saves these in a database.

2 Related work

Seppo Tiilikainen investigates the exposure of vulnerable ICS devices in Finland in his master thesis[22]. He has written an extensive chapter on how to identify devices on the internet where he describes different approaches such as port scanning and fingerprinting. This chapter has been used as a starting point for this investigation.

3 Background

There are many different protocols that use standard ports for communication over the internet. Some of these protocols are older than the internet, e.g Mod-Bus, and leaks a lot of information about the specific device. Some of these devices also have a web server which is easily accessible through a regular web browser and can be used to extract information about the device, as Seppo talked about in his thesis[22].

3.1 ModBus

According to modbus.org[19], Modbus is a communication protocol mainly used to communicate with PLC devices. ModBus is a master-slave/client-server communication protocol. It was developed by Modicon, today renamed to Schneider Electric, in 1979.

Since modbus was developed before the internet became wide spread throughout the world it did not incorporate authentication or any other security mechanisms. Unless the company using the protocol has added their own authentication layer the modbus protocol is open to attacks by a third party as described in the article "Analyzing Internet-connected industrial equipment"[12].

3.2 Tridium Fox

Tridium Fox is a framework used to interconnect different devices such as building automation controls, telecommunications, security automation, lightning control and more. The framework is developed by the American company Tridium[14]. According to the "Cyber security and Infrastructure Security Agency" (CISA)[6], the framework has known security issues for versions lower than the versions listed in table 1.

- One of the vulnerabilities enables an attacker to perform a path traversal[5] to gain access to restricted directories. The Tridium Niagara AX software does not restrict access to parent directories which means that a user can move up to the root folder and access a file containing all the authorized user names and passwords.
- The system also suffers from weak credential storage which means that it stores user names with corresponding passwords in clear text. The credentials are stored in the Niagara configuration file called "config.bog" stored in the root of the station folder.
- The system also stores usernames and passwords in cookies using Base64 encoding.
- It also generates predictable session ids which a potential attacker can use to guess session ID or key.

Software	Version
Niagara AX Framework	3.8 and prior
Niagara 4 Framework	4.4 and prior

Table 1: Niagara vulnerability versions (CISA advisory[7])

3.3 Shodan

Shodan is a tool that scrapes the internet for devices with open ports. It collects information about security vulnerabilities, operating system, protocols and much more. Shodan can be accessed through their web page or through an APIs where it is possible to make custom queries and filter the resulting data.

Shodan scans the internet by continuously generating random IP addresses and ports on which it performs a SYN scan. If the scan was successful it then tries to grab the banner of the device and stores them in its database. An overview of the way that the Shodan scanner works can be seen in the Shodan scan diagram 1.

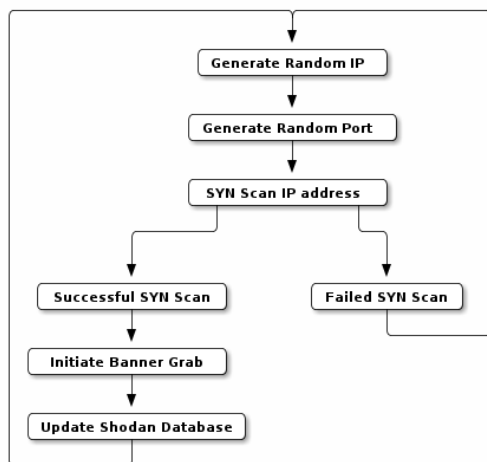


Figure 1: Shodan scan diagram (inspired from the paper "Evaluation of the ability of the Shodan search engine to identify Internet-facing industrial control devices"[16])

Many researchers and security experts have been worried about hackers using Shodan to find industrial devices to target. This group of devices are extra vulnerable to attacks since they are updated

rarely if at all. Because of these known security holes in the firmware of such devices can be exploited for a long time.

The paper "Evaluation of the ability of the Shodan search engine to identify Internet-facing industrial control devices" [16] investigates the possibility to obfuscate the banner information that can be extracted from industrial devices in order to hide their identity. Their results show that the obfuscation made identifying the device nearly impossible thus showing that this is a promising method for increasing the security of industrial devices connected to the internet.

3.3.1 TopICS Scandinavia

By using shodan it is possible to get statistics on ICS devices per country and also per protocol. Table 2 shows how many ICS devices were detected per country in the Nordic region. Table 3 shows the discrepancy between the different protocols and their ports. The search queries used to obtain this information can be found in table 5 the Appendix.

Country	Number of ICS
Sweden	3340
Norway	1253
Finland	942
Denmark	834

Table 2: ICS per country (Shodan statistics[21])

Protocol	Count	ports
Modbus	1436	502
Tridium Fox	1024	1911
OMRON FINS	832	9600
BACnet	578	47808
EtherNetIP	482	44818
General Electric SRTP	310	18245, 18246

Table 3: Protocol Information (Shodan statistics[21])

3.4 Methods to identify devices

The internet is full of different devices and figuring out which device is sitting behind a IP address can be very difficult. There are a number of methods to identify the device behind an ip address the one of which is called port scanning.

Port scanning is when a request is sent to each port of an IP address in order to find out which ports are open and has services listening. The responses from the device behind the IP address can reveal information about the services running on the ports of the device.

The knowledge of which ports are open and what the respective response looked like can be cross referenced with the responses of known devices. This method of identifying devices is called fingerprinting. According to the article "Improving the National Cyber-security by Finding Vulnerable Industrial Control Systems from the Internet"[22], the open source scanning software Nmap has a fingerprint database consisting of over 2000 entries.

If the database does not contain a specific device one can also look at the responses manually. HTTP headers and bodies sometimes contain information that can be used to find the specific device.

The previously mentioned master thesis on finding vulnerable ICS devices on the internet[22] also details that protocols such as netBIOS and SNMP which are created to be used inside of private networks usually give away specific device information. If these protocols are exposed to the internet through faulty configuration they can sometimes be used to identify the device.

4 Method

Shodans Python3 API was used to search for devices with the default ports of the protocols open to the internet. A script that queries shodan and parses the result was created for each of the investigated protocols, see the Code section in Appendix 10.3. The Shodan queries can be seen in table 4.

Protocol	Query
Modbus	port:"502" country:"no,se,dk,fi"
Tridium	port:"1911,4911" country:"no,se,dk,fi"

Table 4: Queries used

These queries resulted in a list of the devices with various information, such as the IP address, location, protocols and more. These devices were loaded in separate files, each file had 100 entries, which could later be queried, sorted and counted depending on different search parameters.

4.1 Tridium Fox

Tridium fox leaks a lot of information about the device its running on when pinging the port, observe that two ports were used here since the protocol actually uses both 1911 and 4911. This information includes version numbers of the protocol itself, the operating system its running on and more.

The script parses the information gained from Shodan API page by page (i.e file by file) and looks at how many of the devices run versions with known security flaws. This gives an idea of how many devices running Tridium Fox in the Nordic region has potential security vulnerabilities and a possibility at leaking device information.

4.2 Modbus

The Shodan results were parsed and the IP addresses extracted. The script sent an HTTP request to the IP address with Python3s requests[13] API to see if the device hosted a web server. Each request was constrained to a timeout of 10 seconds and a maximum of two tries. If a response with status code 200 was received the HTML document with the response were parsed and matched with different strings. The match that was successful then categorized the device IP with the search string/strings and later printed all the categories with there respective IP addresses and the total number of devices in the category. The uncategorized addresses were later examined manually on random to find similarities between the devices for another categorization round, this was done multiple times until most of the devices with response code 200 were categorized. This method is illustrated in 2

The search strings consisted of company names, device name or other keywords that stood out in the HTML document and could be matched. Most of these were located in the <head> tag in the HTML-document but some could be found inside the body.

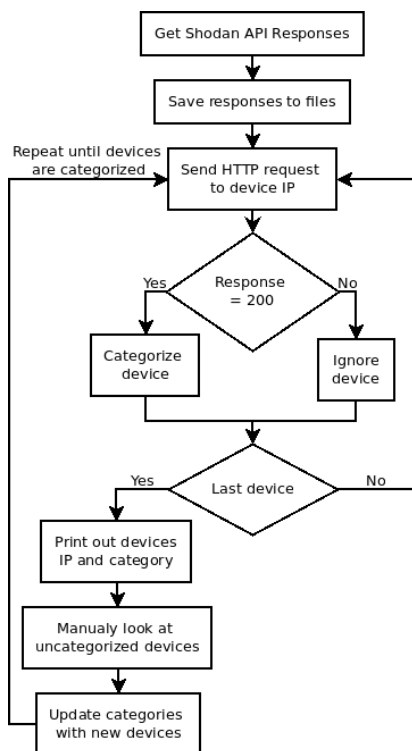


Figure 2: Device categorization method

5.1.1 NSA325v2

A zyxel Network Area Storage (NAS) device for home use was found with an exposed port on the internet. The Nmap scan revealed that the device had port 80 exposed which indicated that it hosted a website. When connecting to the web server manually the NAS login page seen in figure 3 revealed the id of the device in the top left corner.

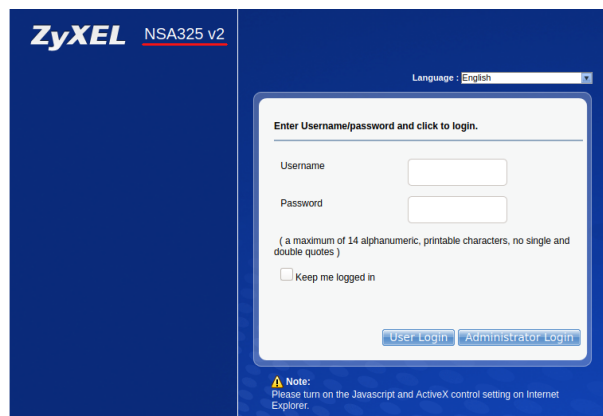


Figure 3: NSA325v2 Website with login pop-up

When searching for the device id, zyxels website revealed a simple home NAS device as can be seen in figure 4.



Figure 4: NSA325v2 physical device

5 Result

A number of devices were found by parsing the json data from Shodan and sending HTTP request to the websites. The devices with exposed web servers were then investigated manually to find specific device data. Most devices that was found were using the modbus protocol since it was much easier to extract data from the websites those devices hosted.

5.1 Tridium Fox

Since the websites hosted on tridium fox devices usually did not reveal any information about the specific device it was hosted on it was very difficult to identify said devices. Only one out of the many IP addresses tested bore any fruit and that was the NSA325v2 NAS device.

5.2 Modbus

Many devices with varying security issues were found for the Modbus protocol during this investigation. The ones that were identified are listed here along with information about how they were found and what they are used for.

5.2.1 Anybus M-Bus

The website hosted on this device displayed a lot of information on the startpage as is shown in figure 5.

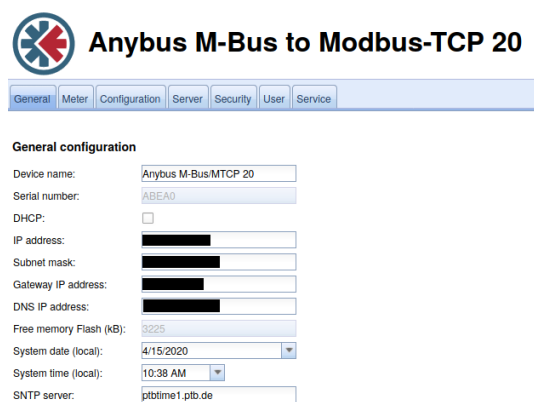


Figure 5: Anybus home page

The HTML document also contained the keyword anybus in the <title> tag which was used to categorize these devices, a total of 3 devices were found with this keyword. When searching for the device name that was displayed on the start page the company website [18] appeared. The webpage described the device as being a Modbus TCP gateway that allows M-Bus devices to communicate on a modbus TCP network. A picture of the device and a use case for it can be seen in figure 32 in the appendix.

The most interesting thing about this device was the login status obtained when entering the website. Figure 6 shows that the login status of the user who entered the website was immediately authenticated as "Logged in as 'web'".

When navigating to the "User" tab, the privileges of the "admin", "web" and "ftp" was shown as is depicted in figure 33 in the appendix. The surprising part was that the privileges of the "admin" user and the "web" user were almost identical

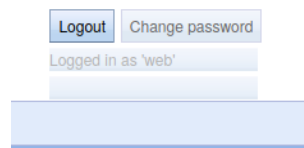


Figure 6: Anybus login status

except for changing password and accessing the device through ftp. The "web" user had full write and read access to all devices and services connected to the device.

This meant that we could manipulate all the settings displayed on the "Configuration", "Server" and "Security" tabs as shown in the figures with the corresponding names⁷⁸.

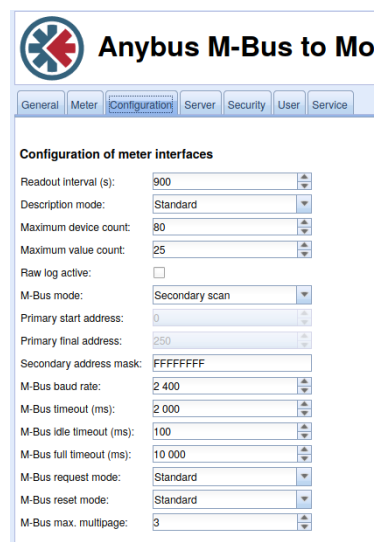


Figure 7: Anybus configuration settings

An even worse observation or rather test that was made was that you could log in as admin with a very easily guessed password, this meant that that you could now change passwords and presumably other important information.

The "Meter" tab on the website revealed the connected devices as shown in figure 34. There were three of them, two of which were some kind of heat/cooling devices and one device which was labeled as "Electricity". The devices could be expanded to expose all the variables that could be set. The values that the Electric device exposed

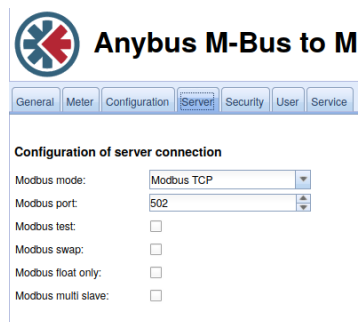


Figure 8: Anybus server settings



Figure 10: AWU 500 physical device

can be seen in figure 34 in the appendix. When right clicking on a value it was possible to edit the values, the prompt showing up when clicking on the edit button is depicted in figure 9.

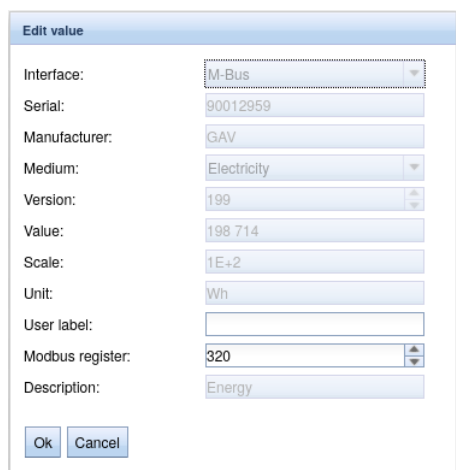


Figure 9: Anybus meter edit

5.2.2 AWU 500

The device name was extracted from the HTML code 11 of the website this device 10 hosted, later the search string "alliance" were used and a total of 12 AWU devices were found.

Searching for this name on the internet gave a Swedish manual[1] in which the following description was found (translated from Swedish to English).

"Alliance Web is a Linux based web server made for surveillance of Modbus devices. It has a graphical web interface supposed to replace the majority

of the other visualization tools in the system."

With other words, this device is intended to sit between other Modbus devices and the internet in such a way that multiple devices in a system can be controlled from a single interface. Alliance have a few example networks on their website [link here] which showed networks of devices that ranged from a few devices to more than 30 devices, see figure 35 in appendix for an example network. For this reason the area which the AWU 500 was found was examined closer without any results, this could be a future work to research.

```
<html> event
<!--LOGIN_PAGE-->
</head>
<body style="background: none; backgr
<title>Alliance - AWU 500</title>
<script language="JavaScript1.2" ty
<style> </style>
<table height="100%" cellspacing="0
</body>
</html>
```

Figure 11: AWU 500 home page HTML-code

5.2.3 CMe3100

This device is used as a M-Bus Gateway which collects data from multiple measurement devices using the M-Bus protocol and then sends the collected data to the receiving system.

This device was found through manually inspecting the HTTP page hosted on the device as can be seen in figure 12. The model number is underscored by a red line. Using the search string "elvaco" a total of two devices were found.

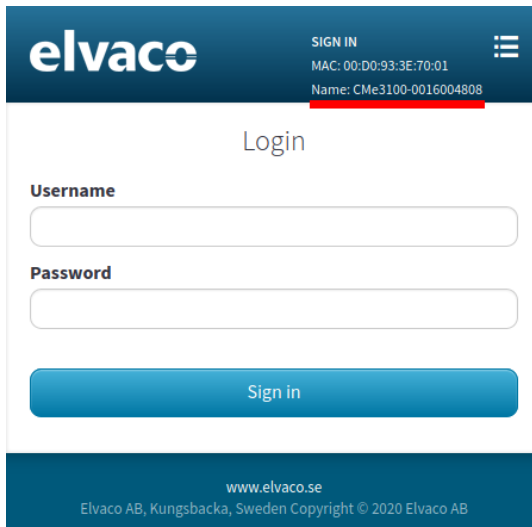


Figure 12: CMe3100 login page/home page

Searching for the model number on the internet revealed a datasheet[8] with the picture depicted in figure 13.

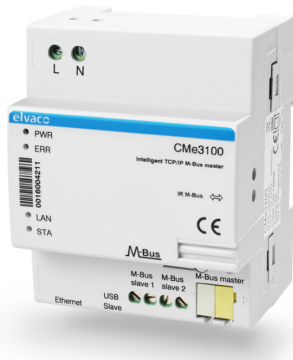


Figure 13: CMe3100 physical device

5.2.4 Corrigo controller

There are many models of this controller and pinpointing the exact model of each such device proved to be very difficult. Thus the devices belonging to this controller family are bunched together. A list of different controllers from this company can be seen on their website[15].

The devices in this family are used for environmental control of different kinds. Figure 14 shows

a sample web page hosted on one such device. The web pages hosted on different devices look almost identical to each other.

The script counting occurrences of devices found 214 devices of this type using the following search parameters: ["regin"]

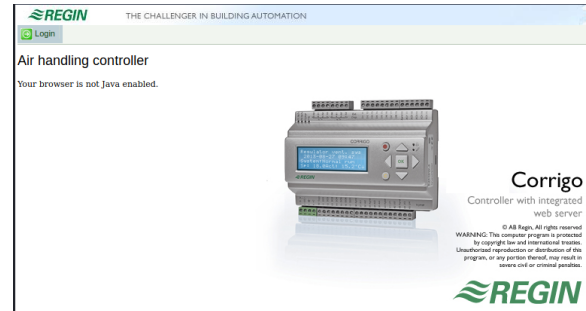


Figure 14: Corrigo controller home page with example device

5.2.5 CS141

This device seems to be a network card which interfaces with the modbus (among other) protocol and was categorized with the keywords ["cs141", "webmanager"] with a total of 2 devices. The website hosted on the device shows the model number "CS141" and the company name "Coromatic" as shown in figure 15.

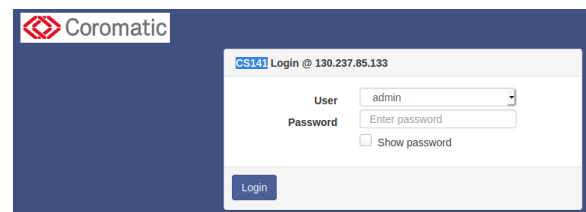


Figure 15: CS141 login page/home page

When searching for the device name an image of the device 16 in question appears. When reading about the device on the manufacturers website[9] it is described as a "Ethernet Adapter for the control and the management of UPS Facilities". With other words, it is a device used to monitor and manage UPS (Uninterrupted Power Supply) devices remotely. The company name was found

on the hosted website confirms this, they seem to be a Swedish distributor of UPS hardware.



Figure 16: CS141 physical device

5.2.6 DCS-5222L

The device name was obtained by manually looking at the web page hosted by the device. The Id was printed on the login screen as can be seen in figure 17.

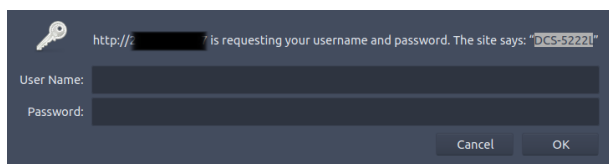


Figure 17: DCS-5222L login pop up on home page

The device is a D-Link web camera as shown in figure 18.



Figure 18: DCS-5222L physical device

5.2.7 iR3220

Finding this device involved multiple steps. First the website hosted on the device had the word "EcoGuard". Searching for this word on the internet revealed a company[2] with that name which specialized in collecting measurement data about building

complexes and displaying the information on a platform they call "CURVES". This further supports that this is the correct company since the website hosted on the device also says that it is running in "CURVES mode". Both of the keywords are underscored in figure 19.

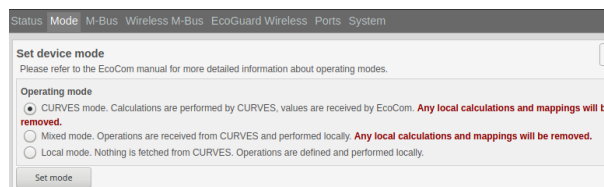


Figure 19: EcoCom home page

By continuing to search for the devices that EcoGuard uses, a manual[3] in Swedish was found. The manual explains in detail how to set up a monitoring network using the device. The device itself should be connected to the internet and is supposed to send data to the company's servers. Sensors of various kinds are then connected to the device as can be seen in figure 20 from the manual.

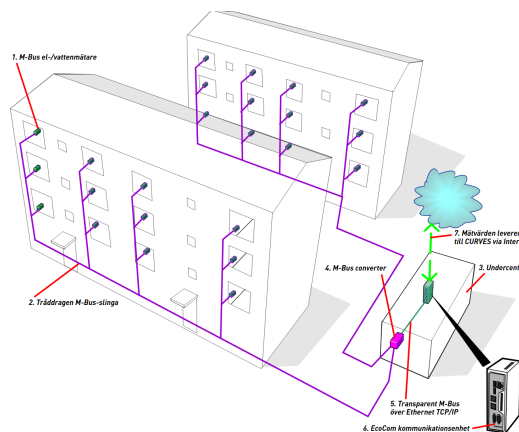



Figure 20: EcoGuard network illustration from manual

Another document[4] issued by the company EcoGuard detailed the specifications of their device as can be seen in figure 21.

The images of the device depicted in figure 22 that appeared both in the manual and the other document revealed the following text "Baltos iR3220" on the device.


 Gör det mätbara användbart®

EcoCom, rev 1-2016 support@ecoguard.se, 019 - 25 21 00, Elementvägen 14, 792 27 Örebro

TEKNISKA DATA

EcoGuard - EcoCom

System

Hardware:	Sitara AM3352 Cortex-A8 @ 600MHz 256MB DDR3 256MB NAND Flash Mass Storage: 1 x SD-Slot Serial port 1: RS232 Modbus, M-Bus, EXOline Serial port 2: RS485 2-wire Modbus, M-Bus, EXOline
------------------	--

Figure 21: EcoGuard specifications



Figure 22: iR3220 physical device

By searching for this product number the web site of the company "Vision Systems" revealed the producer of the device. The device on their website [11] had the same specifications and model number as the aforementioned documents and websites. Thus we can be almost certain that the device found through shodan is the device in question.

The webpage on Vision Systems website detailing the specifications of the device shows the same processor, ports etc as the specifications provided by EcoGuard. This further strengthens the connection between the device and the company EcoGuard.

The script found 113 devices of this kind using the keyword ecom. A devices website must have all of these keywords to match.

5.2.8 Netbiter WS100

This device seems to be some kind of gateway for modbus devices, a total of 3 devices were found and categorized using the keyword "netbiter". According to the manual which could be found on the company website[17], one should put a sim card in to it for internet connectivity which implies that it uses the telephone network.

The device id was displayed on the website hosted on the device together with the name of the company, see figure 23.

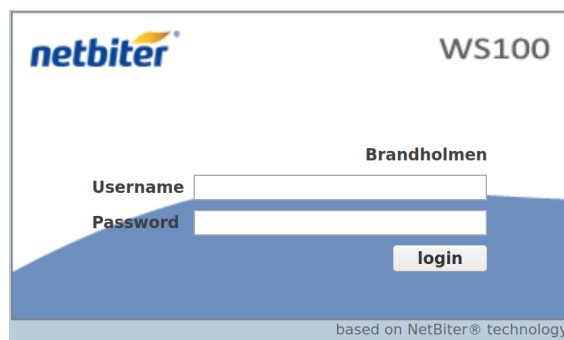


Figure 23: WS100 login page/home page

The companies website revealed an image of the device as can be seen in figure 24.

5.2.9 Solar-Log 1200

A total of 20 devices were found using the keyword "solar", these devices hosted a web server with a quite complex control panel. When entering the



Figure 24: WS100 physical device

website hosted on this device the user is presented with graphs showing the current energy production of what seems to be solar panels. The starting page is shown in figure 25. The device monitors the solar panels and generates statistics regarding the effectiveness of the panels. It also showed tables with cost calculations.

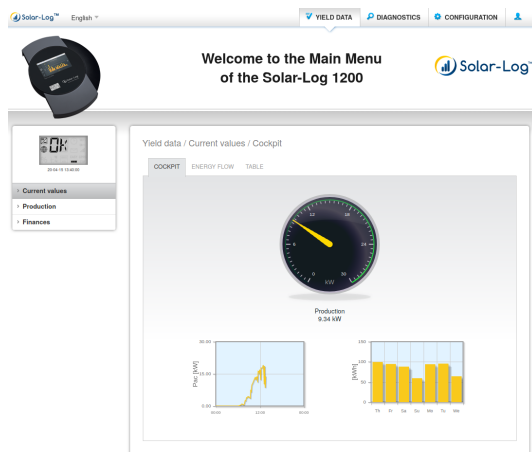


Figure 25: Solar-Log home page

When entering the "about" page the information displayed in the About view was shown as depicted in figure 26.

From this it was easy to get the model number and thus figure out which device was behind the IP address. By googling on the model number a device description and picture, as seen in figure 27, was shown on the company website[10].

The most interesting thing about the website hosted on the device was the security notification that popped up when pressing the red triangle in the top right corner. The notice said that no user password was configured and that the website was

About this Solar-Log™	
Model	Solar-Log 1200
Serial number	276714371
Firmware version	4.1.2 Build 107 - 10.07.2018
Plant data	
Plant size	24940 Wp
Detected devices	
Inverters	RS485-A: 1 x Sungrow
Data transfers	
Portal transfer	20-04-15 13:34:10 - OK
Export (FTP)	Deactivated
E-mail	Deactivated

Figure 26: Solar-Log about page



Figure 27: Illustration of Solar-Log device

not protected from hackers because of this. It then went on asking if we wanted to set the user password now. The popup is displayed in figure 28. This is a major security risk considering that anyone can connect to this device and set the password to whatever thus locking the owners out of their own system.

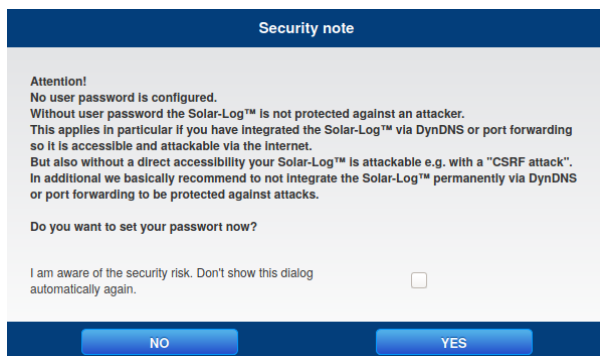


Figure 28: Solar-Log security notification

When investigating the available settings on the page, the access control settings were found as seen in figure 29. It turned out that an unidentified user had full access to all settings in the system including setting a user password, pin code and enabling access control enforcement. There were a large number of settings on this device all of which can be modified by an unauthenticated user. An overview of the available settings can be seen in figure 30. A unidentified user was even able to update the firmware on the device to any uploaded file as can be seen in figure 36 in the appendix.

5.2.10 Web relay

The website hosted on this device only showed a web interface where the word "Webrelay Quad" was listed. The entire website can be seen in figure 31. It looked like the device could control four relays of some kind though it was very unclear what the relays were controlling. The HTML code revealed nothing about the device and it required no authentication to access and change relays. A total 2 devices were found using the keywords ["webrelay", "quad"].

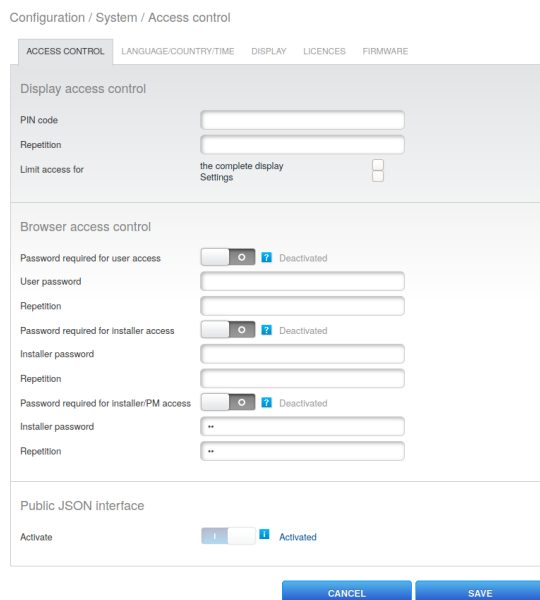


Figure 29: Solar-Log access control

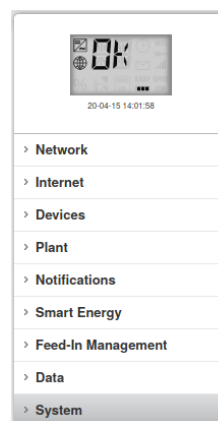


Figure 30: Solar-Log settings

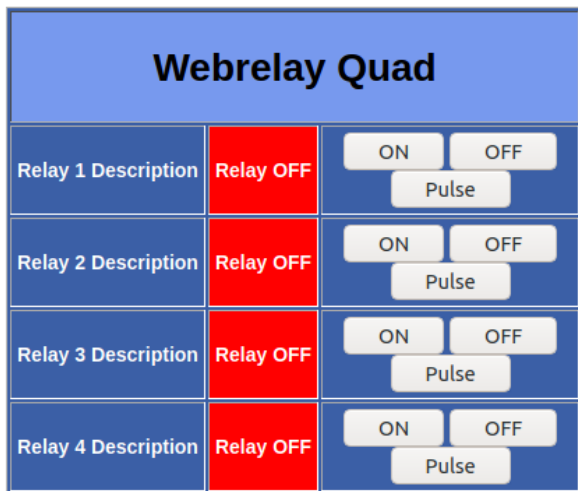


Figure 31: Webrelay Quad home page

6 Discussion

The method used to find exposed devices with a web server running proved very efficient. The manual labor of looking at the websites in order to identify the id of the devices was significantly decreased. By grouping the devices together with keywords found in the HTML code manual searching was minimized further. After looking at a couple of websites of the same kind, unique strings could be identified and used in the program to collect all such websites into one specific group. This decreased the amount of websites that had to be inspected manually significantly.

One potential problem with grouping the websites is that some websites that actually differed from the ones manually parsed might have been overlooked. Since the grouping mechanism only parsed the raw HTML code and looked for keywords related to the company and device id there might have been variations to said device that was missed in the process.

Since the grouping mechanism might be inaccurate the number of devices of each type might not be completely accurate. Especially since only websites hosted on such devices were explored. Other devices with the same device id might be active without exposing their web server to the internet while exposing other ports.

Another aspect to consider regarding the grouping of devices is the way in which the HTTP requests were handled. If the script did not get a response within after a second attempt or were a little to slow to respond it assumed that there was no web server hosted on that IP address. Because of this the amount of devices varied slightly from time to time when running to script.

With other words, the numbers given in this report regarding the amount of devices of each type is only an estimate using the outlined method.

A possible way to gain further information from the devices might be querie the Telnet protocol to see if that port is exposed to the internet. This was not used when collecting information for this report.

Another avenue of information gathering that was considered was to group devices by their geographical location. By doing so it might be possible to discover what kind of facility the devices were stationed in and thus gleam further information regarding what they were used for. For example, it might reveal a power plant, production facility or a hospital which have many ICS devices that work together in a larger system.

The methods used in this project worked well for the modbus protocol since the websites hosted on such devices varied a lot. Many of the websites revealed information about the company using the device or the company making the device as well as the device model number. Some had pictures of the device while others had a detailed description of its function. There were many sites that were more secure and thus did not leak any information about the device behind it.

The methods did not work that well for Tridium Fox however mostly because almost all of the websites hosted on such devices were the same. Tridium Fox is not just a protocol but a framework in which the web server is most likely bundled. They did not reveal anything about the nature of the device in the HTML code nor in the actual website. Only one device with a different website was found and is presented in the results section. Using Telnet and other possible methods might give more information but this was not tested during this investigation.

7 Future work

A possible way to gain further information from the devices might be query the Telnet protocol to see if that port is exposed to the internet. This might be worth looking into for future projects of this kind.

Another interesting aspect that might be worth looking in to in a future project is the geographical location of the devices. By grouping them together by their actual location one might be able to find facilities that use multiple ICS devices. By finding out what type of facility they were located in, one might be able to get more information regarding what the devices were used for.

Since this report only investigates one protocol thoroughly it would be interesting to explore other protocols as well.

8 Conclusion

The investigation revealed a lot of exposed devices many of which suffered major security holes. Only one protocol was properly investigated and more devices would probably be found if using more methods or continuing using the ones outlined in this report.

The lack of knowledge about security in the industry is scary to say the least. Since the amount of connected ICS devices grows every year more and more critical infrastructure is exposed on the internet. This poses a severe security threat to companies, individuals and whole countries.

9 Dictionary

- **ICS:** Industrial Control System
- **PLC:** Programmable Logic Controller [20] is a industrial digital computer adapted for assembly lines, robotic devices and the like.
- **SCADA:** Supervisory Control and Data Acquisition
- **PLC:** Programmable Logic Controller
- **DCS:** Distributed Control System
- **RTU:** Remote Terminal

- **CISA:** Cyber security and Infrastructure Security Agency

10 Appendix

10.1 Search queries

Table 5 contains the search queries used to find information on Shodan.

Protocol	Search querie
Modbus	port:"502"
Tridium Fox	port:"1911, 4911"
EtherNetIP	port:"44818"
BACnet	port:"47808"
OMRON FINS	port:"9600" response code
General Electric SRTP	port:"18245, 18246" product:"general electric"

Table 5: Queries used for protocol data

10.2 Devices

10.2.1 Anybus M-Bus

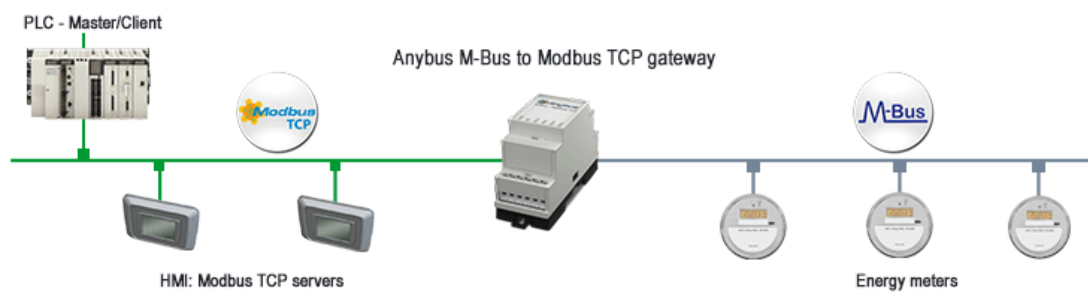


Figure 32: Anybus basic device network

User

Name	Overwrite password	Change password	Sessions	Maximum sessions	Read General	Write General	Read Meter	Write Meter	Read Config	Write Config	Read Server	Write Server	Read Security	Write Security	Read Service	Write Service	Write User	FTP
admin	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0	1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
web	<input type="checkbox"/>	<input type="checkbox"/>	1	-1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ftp	<input type="checkbox"/>	<input checked="" type="checkbox"/>	0	-1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 33: Anybus privileges

Connected meters

Interface	S	Serial	MAN	Medium	Version	Link	Value	Scale	Unit	Cycle	User label	Description	Register
+	M-Bus	00065703	BMT	Combined heat / cooling	7	0				0			10
+	M-Bus	00071737	BMT	Combined heat / cooling	7	0				0			160
-	M-Bus	* 90012959	GAV	Electricity	199	0				0			310
							198 714	1E+2	Wh			Energy	320
							7	1E-1	kVARh			Reactive energy	330
							10 237	1E-1	W			Power	340
							-3 183	1E-4	kVAR			Reactive power	350
							10 731	1E-4	Res			Reserved	360
							955	1E-3				No VIF	370
							4 042	1E-1	V			Volts	380
							2 334	1E-1	V			Volts	390
							1 809	1E-3	A			Ampere	400
							1 776	1E-3	A			Ampere	410
							1 837	1E-3	A			Ampere	420
							3 291	1E-1	W			Power	430
							3 255	1E-1	W			Power	440
							3 546	1E-1	W			Power	450
							-1 041	1E-4	kVAR			Reactive power	460
							-1 226	1E-4	kVAR			Reactive power	470
							-911	1E-4	kVAR			Reactive power	480
							3 452	1E-4	Res			Reserved	490
							3 478	1E-4	Res			Reserved	500
							3 661	1E-4	Res			Reserved	510
							953	1E-3				No VIF	520
							936	1E-3				No VIF	530
							969	1E-3				No VIF	540
							3 997	1E-1	V			Volts	550
							4 064	1E-1	V			Volts	560

Figure 34: Anybus Meter information

10.2.2 AWU 500

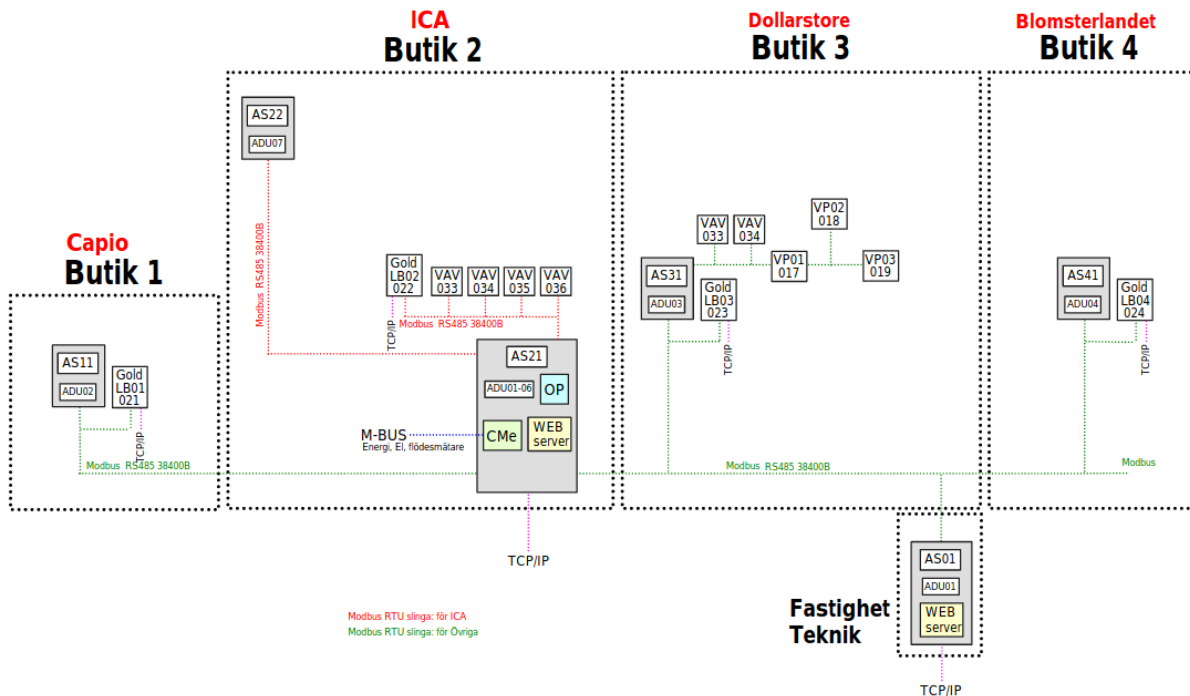


Figure 35: Alliance example network with AWU devices

10.2.3 Solar-Log

Configuration / System / Firmware

ACCESS CONTROL LANGUAGE/COUNTRY/TIME DISPLAY LICENCES **FIRMWARE**

Status

Installed version

Important notice

As new firmware-versions may change existing functions and/or require changes to the configuration, it is necessary to be informed about the changes to ensure the reliable operation of the device.

Important notices are usually especially mentioned in the beginning of the release notes.

If several firmware-versions were available between the currently used version and the version that is to be installed, then all notices of the versions in between apply.

Current firmware versions and related notices can be found on our homepage under downloads.

Update firmware manually

Update firmware from hard disk No file selected.

Check for update via Internet

Check USB drive for update

Check Firmware version automatically

Check Firmware version automatically ? Activated

Figure 36: Solar-Log firmware update

10.3 Code

This section contains the scripts used to collect and parse the data from shodan.

10.3.1 modbus.py

```
1  #! /usr/bin/env python3
2
3  from matches import Matches
4
5  matches = Matches("modbus", 15)
6
7  word_map = [("regin",), ("ecocom",), ("elvaco",), ("wago", "ethernet"),\
8              ("komfovent",), ("stiebel",), ("wdc",), ("ouman", "eh", "net"),\
9              ("solar",), ("webrelay", "quad"), ("cs141", "webmanager"),\
10             ("teltonika",), ("alliance",), ("netbiter",), ("anybus",), ("pcoweb",)]
11
12  matches.print_word_map(word_map)
```

10.3.2 shodanRun.py

```
1  #!/usr/bin/env python3
2
3  from shodan import Shodan as sho
4  import json
5
6  api = sho("vTfeGFDi1hhDvwiwk2SLIcfI1achlY0o")
7
8  matches = []
9  PROTOCOL = "modbus"
10 FOLDER = PROTOCOL + "_results"
11 PAGES = 14
12
13 def write_result(file_name, results):
14     try:
15         with open(file_name, 'a') as f:
16             f.write(json.dumps(results))
17     except Exception as e:
18         print("Could not open file: {}\nError:{}".format(file_name, e))
19
20 def get_shodan_result(query, page):
21     try:
22         results = api.search(query, page)
23
24         file_name = "{}/{}-{}.json".format(FOLDER, PROTOCOL, page)
25
26         write_result(file_name, results)
27     except shodan.APIError as e:
28         print('Error: {}'.format(e))
29
30
31 def main():
32     #query = 'port:"1911, 4911" country:"no,se,dk,fi"'
33     query = 'port:"502" country:"no,se,dk,fi"'
34     for page in range(PAGES+1):
35         print("Getting page {}".format(page))
36         get_shodan_result(query, page)
37
38 main()
```

10.3.3 matches.py

```
1 import json
2 import requests
3
4 class Matches:
5     matches = []
6     loading_string = ""
7     result_file = ""
8
9     def __init__(self, protocoll, pages, file_name = "result.txt"):
10        self.result_file = file_name
11        for i in range(pages):
12            print("Loaded page [{}].format(i))
13            file_name = "{}-{}.json".format(protocoll,i)
14            matches = []
15            with open(file_name) as f:
16                for line in f:
17                    matches = json.loads(line)['matches']
18            self.matches += matches
19        self.loading_string = "[{}].format(" "*pages)
20
21 #####
22 # WRITE TO RES FILE #
23 #####
24 def write_to_file(data):
25     f = open(self.result_file, 'a')
26     f.write(data)
27     f.close()
28
29 #####
30 # PRINT FUNCTIONS #
31 #####
32 def print_matches(self, pred):
33     for match in self.matches:
34         print("{}".format(pred(match)))
35
36 #####
37 # MAP FUNCTIONS #
38 #####
39 def get_map(self, get_key, get_value):
40     match_map = {}
41     for match in self.matches:
42         try:
43             key = get_key(match)
44             value = get_value(match)
45         except:
46             continue
47         if key in match_map:
48             match_map[key].append(value)
```

```

49         else:
50             match_map[key] = [value]
51     return match_map
52
53 def print_map(self, match_dict, print_key, print_value, min_key=0):
54     for key in match_dict:
55         if len(match_dict[key]) > min_key:
56             total = 0
57             print_key(key)
58             for value in match_dict[key]:
59                 print_value(value)
60                 total += 1
61             print("Total devices: {}\n".format(total))
62
63 def write_map(self, match_dict, format_key, format_value):
64     for key in match_dict:
65         if len(match_dict[key]) > min_key:
66             total = 0
67             self.write_to_file(format_key(key))
68             for value in match_dict[key]:
69                 self.write_to_file(format_value(value))
70                 total += 1
71             print("Total devices: {}\n".format(total))
72
73 #####
74 # IP FUNCTIONS #
75 #####
76 def get_ip(match):
77     try:
78         return match['ip_str']
79     except:
80         return "No ip"
81
82 def get_url(self, match):
83     try:
84         return "http://{0}".format(match['ip_str'])
85     except:
86         return "No URL"
87
88 #####
89 # OS FUNCTIONS #
90 #####
91 def get_os(self, match):
92     try:
93         return match["os"]
94     except:
95         return "No os"
96
97 def print_os(self):
98     pred = lambda match : print("{} : {}".format(self.get_url(match), self.get_os(match)))

```

```

99         self.print_matches(pred)
100
101     #####
102     # HTTP FUNCTIONS #
103     #####
104     def multi_match(self, html, words):
105         html = html.lower()
106         for word in words:
107             res = html.find(word)
108             # if not word.lower() in html:
109             #     return False
110             if res == -1:
111                 return False
112         return True
113
114     def get_word_match(self, match, words):
115         url = self.get_url(match)
116         try:
117             req = requests.get(url, timeout=10)
118             if req.status_code == 200:
119                 if self.multi_match(req.text, words):
120                     return match
121             else:
122                 return None
123         else:
124             return "Error"
125     except Exception as e:
126         print("Exception {} occured\n".format(e))
127         return "Error"
128
129     def get_word_matches(self, words):
130         correct_matches = []
131         for match in self.matches:
132             correct_match = self.get_word_match(match, words)
133             if correct_match:
134                 correct_matches.append(correct_match)
135         return correct_matches
136
137     def print_word_matches(self, words=[""]):
138         print("Words to match: ", words)
139         correct_matches = self.get_word_matches(words)
140         for match in correct_matches:
141             print("{}\n".format(self.get_url(match)))
142
143     def get_word_map(self, word_map):
144         result_map = {"None" : [], "No response" : []}
145         categorized = False
146         for match in self.matches:
147             categorized = False
148             for words in word_map:

```



```

149         correct_match = self.get_word_match(match, words)
150         if not correct_match:
151             continue
152         elif correct_match == "Error":
153             result_map["No response"].append(match)
154             categorized = True
155             break
156         else:
157             if words in result_map:
158                 result_map[words].append(correct_match)
159             else:
160                 result_map[words] = [correct_match]
161
162             categorized = True
163             break
164
165         if not categorized:
166             result_map["None"].append(match)
167
168     return result_map
169
170 def print_word_map(self, word_map):
171     word_map = self.get_word_map(word_map)
172
173     print_key = lambda key : print("Words: {}".format(key))
174     print_value = lambda value : print("  {}".format(self.get_url(value)))
175
176     self.print_map(word_map, print_key, print_value)
177
178 def write_word_map(self, word_map):
179     word_map = self.get_word_map(word_map)
180
181     format_key = lambda key : "Words: {}".format(key)
182     format_value = lambda value : "  {}".format(self.get_url(value))
183
184     self.write_map(word_map, format_key, format_value)
185
186     #####
187     # LOCATION FUNCTIONS #
188     #####
189 def get_longitude(self, match):
190     return str(match["location"]["longitude"])
191
192 def get_latitude(self, match):
193     return str(match["location"]["latitude"])
194
195 def get_ip_locations(self):
196     key = lambda match : (self.get_latitude(match), self.get_longitude(match))
197     value = lambda match : match["ip_str"]
198     return self.get_map(key, value)

```

```
199
200 def print_ip_locations(self, min_value=0):
201     print_key = lambda key : print("Location: {0:.10}, {1:.10}".format(key[0], key[1]))
202     print_value = lambda value : print("  http://{0}".format(value))
203     self.print_map(self.get_ip_locations(), print_key, print_value, min_value)
```

11 Sources

References

- [1] CR Fastighetsteknik AB. *AWU 500*. 2010. URL: <http://www.systemalliance.se/wp-content/uploads/2018/05/AWU500-2010-11-16.pdf> (visited on 04/04/2020).
- [2] EcoGuard AB. *About EcoGuard*. ? URL: <https://www.ecoguard.se/om-ecoguard/om-ecoguard> (visited on 04/06/2020).
- [3] EcoGuard AB. *EcoCom manual*. 2016. URL: https://uploads-ssl.webflow.com/59ca716f6560aa0001b1e80c/5ae171ecfe9177569edd7469_EcoCom_Baltos_refManual_160407_webb.pdf (visited on 04/06/2020).
- [4] EcoGuard AB. *EcoCom specifications*. 2016. URL: https://assets.website-files.com/59ca716f6560aa0001b1e80c/5a4e4ab9056b6a0001015873_ECOGUARD_ECOCOM_rev2.pdf (visited on 04/06/2020).
- [5] Common Weakness Enumeration (CWE). *Path Traversal*. 2020. URL: <https://cwe.mitre.org/data/definitions/22.html> (visited on 04/01/2020).
- [6] Cybersecurity and Infrastructure Security Agency (CISA). *About CISA*. ? URL: <https://cwe.mitre.org/data/definitions/22.html> (visited on 04/01/2020).
- [7] Cybersecurity and Infrastructure Security Agency (CISA). *CISA advisory*. 2018. URL: <https://www.us-cert.gov/ics/advisories/ICSA-18-191-03> (visited on 04/01/2020).
- [8] Elvaco. *CMe3100 Datasheet*. ? URL: <https://www.elvaco.se/Image/GetDocument/en/19/cme3100-data-sheet-swedish.pdf> (visited on 04/14/2020).
- [9] GENEREX. *Manufacturer website CS141*. ? URL: <https://www.generex.de/content/view/25/54/> (visited on 04/15/2020).
- [10] Solare Datensysteme GmbH. *Solar-Log 1200*. ? URL: <https://www.solar-log-america.com/products-accessories/monitoring-hardware/solar-log-1200/> (visited on 04/15/2020).
- [11] VS Vision Systems GmbH. *iR3220 on producers website*. ? URL: <http://www.visionssysteme.de/produkte/baltos-ir-3220.html> (visited on 04/06/2020).
- [12] A. Gurtov M. Khodari A. Hansson. *Analyzing Internet-connected industrial equipment*. 2018. URL: <https://ieeexplore-ieee-org.e.bibl.liu.se/document/8372775> (visited on 04/15/2020).
- [13] *HTTP request library for python*. ? URL: <https://2.python-requests.org/en/master/#> (visited on 04/15/2020).
- [14] Tridium Inc. *Tridium website*. ? URL: <https://www.tridium.com/products-services/niagara4> (visited on 04/01/2020).
- [15] Regin Control UK Limited. *Corrigo controller website*. ? URL: <https://www.regincontrols.com/en-GB/product1/exigoardo---controllers-for-heating-24-v/3278/30474/#topcats> (visited on 04/04/2020).
- [16] R. Bodenheimer J. Butts S. Dunlap B. Mullins. *Evaluation of the ability of the Shodan*. 2014. URL: <https://www.sciencedirect.com/science/article/pii/S1874548214000213> (visited on 04/14/2020).
- [17] HMS Networks. *WS100 company website*. ? URL: <https://www.netbiter.com/support/file-doc-downloads/netbiter-ws100> (visited on 04/15/2020).
- [18] HMS Industrial Networks. *Anybus M-Bus device*. ? URL: <https://www.anybus.com/products/gateway-index/specific-gateways/factory-to-building/detail/anybus-m-bus-to-modbus-tcp-gateway> (visited on 04/15/2020).

- [19] Modbus Organization. *Modbus Official Site*. ? URL: <http://www.modbus.org/faq.php> (visited on 03/26/2020).
- [20] Andrew Parr. *Industrial Control Handbook*. 1986. URL: https://books.google.se/books?id=zLwtngK3T1UC&printsec=frontcover&hl=pl&source=gbs_ge_summary_r&redir_esc=y#v=onepage&q&f=false (visited on 04/06/2020).
- [21] Shodan. *Shodan statistics*. 2013. URL: <https://www.shodan.io/report/pGY7P8qw> (visited on 03/31/2020).
- [22] Seppo Tiilikainen. *Improving the National Cyber-security by Finding Vulnerable Industrial Control Systems from the Internet*. 2014. URL: https://aaltodoc.aalto.fi/bitstream/handle/123456789/12918/master_Tiilikainen_Seppo_2014.pdf?sequence=1 (visited on 04/05/2020).