Analyzing the IoT landscape in the Nordics

Carl Magnus Bruhner, Jakob Sjöqvist
Linköping University
Email: {carbr307, jaksj829}@student.liu.se
Supervisor: Andrei Gurtov, andrei.gurtov@liu.se
Project Report for Information Security Course
Linköpings universitet, Sweden

***Abstract*—A fast-growing category of the internet is the Internet of Things (IoT). As the number of IoT devices grows, the payoff of finding vulnerabilities to exploit increases and hence the area of research. The type of IoT devices and their vendors can vary around the world as different service providers either develop their own products or choose different hardware providers. For this reason, the aim with this project is to identify a vulnerable IoT device commonly used in the Nordics and present how vulnerable devices can be identified through the IoT search engine Shodan. Additionally, the most commonly used unprotected network protocols for IoT devices are identified and their use in the Nordics are quantified. Finally, recent research is reviewed, and other similar vulnerabilities presented.**

***Index Terms*—IoT, Shodan, MQTT, RTSP, UPnP, Zigbee, vulnerabilities, network protocols, security**

I. Introduction

Internet of Things (IoT) is an umbrella term for connected devices which, by being able to communicate with other devices in the same network or via the Internet, has created many new opportunities. There are many different types of devices that are included in the term IoT, but typical areas of use are healthcare, smart home systems, industry and production.

The IoT area has developed a lot in recent years and there are many indications that this trend will continue. By enabling communication between devices, many processes have been improved and streamlined, but connecting more devices to networks and the internet also renders some problems. A common and significant problem is security. The security problems are also difficult to isolate and remedy as both the devices as such and the protocols they use to communicate may be vulnerable.

To get maximum value from IoT, many devices are exposed to the internet which imply that not only the administrator or users can access the device but also anyone else with an internet connection. Today, commercial tools are available and can be used to identify vulnerable devices with just a few clicks.

To investigate what the current status of IoT security in the Nordics is, the purpose of this project is to answer the following research questions:

- What IoT devices with known vulnerabilities are popular in the Nordics?
- What kind of vulnerabilities do these IoT devices have, and can vulnerable devices be identified?
- Are there any differences in usage of unprotected IoT network protocols between the Nordic countries?

II. Background

In this section, relevant background to the findings of the project will be introduced. First, general concepts will be presented, followed by common network protocols, services and devices.

A. Internet of Things (IoT)

Internet of Things (IoT) are a term describing the phenomena of the massive number of internet-connected devices making up the ”next generation” of the internet [27]. Cost-effectiveness, ease of use and the ability to use devices for monitoring in many settings and environments are drivers for the usefulness of IoT. However, privacy and security issues are major shortcomings and thus naturally of interest for the security research community [2, 3, 16].

B. Message Queuing Telemetry Transport (MQTT)

Message Queuing Telemetry Transport (MQTT) is a network protocol designed to be lightweight and useful for low-bandwidth networks, such as for Internet of Things (IoT). It is standardized by the Organization for the Advancement of Structured Information Standards (OASIS), with the latest standard being v5.0 published in 2019 [7]. MQTT have TCP/IP port 1883 and port 8883 registered with IANA for non-TLS and TLS communication respectively.

MQTT has a publish-and-subscribe messaging support, where clients can subscribe to different topics being published to. It is also possible to subscribe to any number of subtopics with the help of the wildcard character #.

C. Real Time Streaming Protocol (RTSP)

Real Time Streaming Protocol (RTSP) is a network protocol for data with real-time properties such as audio and video [22]. It uses port 554 (primarily TCP, but UDP as well) by default. RTSP has authentication implemented in the design, but it is not used by default. Among other video streaming services, RTSP is used by various web cameras that streams information over the network [2].

D. Universal Plug and Play (UPnP)

Universal Plug and Play (UPnP) is a technology for seamless peer-to-peer inter-connectivity between different devices in different environments [20]. It is platform independent and support any network technology—including phone and power line. The listed benefits of utilizing UPnP architecture are allowing devices to use zero-configuration and automatic discovery to join networks, announce its presence and get to know the network with other devices automatically. UPnP has a discovery protocol, Simple Service Discovery Protocol (SSDP), that uses the IANA-reserved port 1900 [8]. Additionally, port 7900 is used for multicast event messages.

E. Zigbee

Zigbee is a wireless protocol and standard that is widely used by IoT devices. The Zigbee protocol is developed by the Zigbee Alliance and was designed to be highly reliable, cost-effective, low power consumption, secure and an open global standard [9].

More in detail, Zigbee implements the IEEE 802.15.4 standard which is a wireless standard suited for wireless private area networks (WPAN). Due to the requirement of low power consumption, the transmission range and rate is limited in Zigbee. When Zigbee communicates in the 2.4 GHz spectrum, the maximum (theoretical) transmission rate is around 250 kbps. To improve the transmission range, Zigbee leverage mesh technology; each individual device is not required to be connected with the source device as messages can be forwarded from the source device via intermediate devices to reach the destination node [9].

In terms of security, Zigbee networks are secured by 128-bit symmetric AES encryption [26].

Popular products that use the Zigbee protocol includes Philips Hue smart home system [21]. As of 2020, more than half a billion Zigbee chips has been sold [1].

F. Shodan

Shodan[1] is a search engine for internet-connected devices, including IoT [27]. It can identify over 100 000 consumer IoT devices such as webcams. Beside the general search engine on the website, Shodan offers capabilities such as an API, a map interface, image crawls, vulnerability database and more. The image crawl stream for instance displays images retrieved from IP addresses with open access.

G. Network cameras

Network security cameras, wireless smart cameras and IP cameras are all various forms or synonyms for cameras that are connected to the internet [2]. Being forms of IoT devices, network cameras too share the vulnerabilities known for IoT in general. This would for instance open up the risk (or opportunity) of gaining access to camera feeds due to poor or non-existing access control. Even with access control, default credentials are also a known issue allowing for ”authorized” unauthorized access.

[1]https://www.shodan.io

Fig. 1. Home Assistant hardware [17]

H. Home Assistant

Home Assistant is an open-source software project that aims to simplify home automation by connecting different IoT systems. Having all IoT systems connected and controlled from one application, the user can create scenes that involve a range of protocols and devices from multiple manufacturers [11]. One example of such a scene could be to turn on all lights if the door is unlocked and it is dark outside.

To support multiple protocols and devices, Home Assistant is built using a modular architecture and new modules can be added as a new type of device or protocol is introduced [6]. As of the time of writing this report, Home Assistant has more than 1700 integrations and the library grows continuously as new integrations are developed and released by the community.

Home Assistant is privacy oriented and hence no hosted application is offered. Instead, each user has to install and run the application on their own hardware. Hardware with Home Assistant installed by default is also sold from the official Home Assistant website. In short, Home Assistant can be delivered in three different formats [11]:

- Home Assistant Core *(application to be deployed to any modern operating system)*
- Home Assistant OS *(minimal Linux operating system with pre-configured installation of Home Assistant Core)*
- Hardware *(delivered with Home Assistant OS)*

Figure 1 shows the Home Assistant hardware, and Figure 2 show an example of a Home Assistant dashboard.

III. Method

The project was primarily based on quantitative methods and the main tool used throughout the project was Shodan. As a complement, qualitative methods will be used to perform a more in-depth analysis of known vulnerabilities for identified devices, protocols and applications.

The first phase was about identifying interesting and relevant devices, protocols and software. For a device to be

Fig. 2. Example Home Assistant Dashboard [24]

considered as interesting and relevant it has to be widely used. In order to identify widely used devices, applications and protocols used in home automation or other IoT environments as well as trends from online communities and forums were analyzed, in conjunction with top lists from Shodan and retailers as references.

The compiled list of popular devices was then subject for further investigation by figuring out how these devices can be found using Shodan. The results from Shodan were then used to quantify the occurrence of IoT devices. In addition, all devices were also grouped by their properties (e.g., protocols used, device type, manufacturer) and each category was then subject for a more in-depth analysis where known vulnerabilities for the given category was inspected and analyzed.

A. *Identifying vulnerabilities in Home Assistant*

By making the assumption that many users want to be able to connect to their home automation systems, it may also be assumed that they may expose their Home Assistant installation to the internet to make it possible to access their system via the internet. To identify these devices, a fingerprint was required in order to find the devices via Shodan. By inspecting the source code for the Home Assistant front-end, the HTML title of the authorization page was found to be *Home Assistant* [12]. With this title as the fingerprint, active Home Assistant installations in the Nordic countries was found using the Shodan search query `http.title:"Home Assistant" country:se,no,dk,fi,is`, which returned more than 3800 results.

The acquired result contained the HTTP headers sent from Home Assistant when it was crawled by Shodan. The HTTP-header was found to contain version information about `httpaio` which is the underlying web server software used by Home Assistant [12]. As `httpaio` is a dependency of Home Assistant, the version number was used to get an indication of which version of Home Assistant being used by analyzing the change log. The version of `httpaio` that was found to be part of the last vulnerable version of Home Assistant was `httpaio` version 3.7.3 [13]. In addition, for a subset of all devices Shodan did also provide the raw Home Assistant version. By combining these two techniques, vulnerable installations could be identified using the Shodan API.

Known vulnerabilities in Home Assistant was found by searching the internet, and more in detail by searching in common vulnerabilities and exposure (CVE) databases.

IV. Results and Discussion

Based on the aforementioned method, the following results have been compiled. They are presented grouped as vulnerable devices, vulnerable protocols and regional data findings.

Vulnerable Devices

In this section we will further explore the vulnerability of Home Assistant and investigate how widespread it is in the Nordic countries by using Shodan.

A. Home Assistant

One major known vulnerability was found for Home Assistant [19]. The vulnerability itself was not in the Home Assistant core but in multiple custom integrations. The vulnerability allowed an unauthenticated attacker to perform directory traversal and view all files accessible to the Home Assistant process. Thus, it was possible for an attacker to steal credentials used by other custom integration, such as API keys used to control the home alarm system. One of the integrations that contained this security issue was the *Home Assistant Community Store (HACS)*[2] which has more than 1700 stars on GitHub and the latest version (1.12.3) has been downloaded more than 18,000 times in eight days. HACS is an add-on to Home Assistant that simplifies the process of installing additional integrations that are developed by enthusiasts but not released in the official integration repository.

Given the directory traversal vulnerability and the popularity of HACS, it is clear that many systems and installations was and still are vulnerable due to users not keeping their systems updated.

To investigate the magnitude and severity of this vulnerability, the source code for the custom integration for the home alarm providers Verisure[3] (available via official integration repository) and Sector Alarm[4] (available via HACS) was analyzed. Both the integrations are wrappers for their respective API and hence all actions that are initiated in Home Assistant needs to be sent via the provider's API service. To communicate with the API, both APIs requires credentials. As described in the documentation for Home Assistant, all credentials and secrets are stored in a `secrets.yaml` file [23].

Given the directory traversal vulnerability in Home Assistant that allowed an unauthorized user to access the file system, it is possible for an attacker to extract all system secrets—including API keys used by Verisure or Sector Alarm—and hence an attacker could control the alarm system.

Using the Shodan API, vulnerable devices were identified in the Nordic countries. Active Home Assistant devices were found by using the Shodan query `http.title:"Home Assistant" country:se,no,dk,fi,is` and devices running version 2021.1.5 of Home Assistant or version 3.7.3 of httpaio or older was considered vulnerable as described in section III. The code used is available in Appendix A.

Table I presents the results, with both vulnerable and total number of devices. Furthermore, the cities with the most vulnerable devices are presented in Table II with vulnerable and total number of devices.

The ports used by Home Assistant are primarily port 443 (1832 devices), port 8123 (1660 devices) and port 80 (106 devices). Among the vulnerable installations port 8123 (224 devices) followed by port 443 (131 devices) and port 80 (10 devices) are the most used. The default port used by Home Assistant is port 8123 [15].

[2]https://github.com/hacs

[3]https://github.com/home-assistant/core/tree/dev/homeassistant/components/verisure

[4]https://github.com/gjohansson-ST/sector

Country	Vulnerable	Total
Sweden	221	2019
Norway	73	785
Finland	50	403
Denmark	32	552
Iceland	7	51

TABLE I
HOME ASSISTANT DEVICES PER COUNTRY

City	Vulnerable	Total
Stockholm (Sweden)	59	508
Oslo (Norway)	73	214
Gothenburg (Sweden)	17	166

TABLE II
HOME ASSISTANT DEVICES PER CITY

UNPROTECTED PROTOCOLS

In this section we will present the result in terms of which protocols we found to imply vulnerability due to the connections being unprotected. The protocols we have found to have issues of being unprotected and used in the context of IoT spans across multiple categories.

B. MQTT

Even though MQTT is considered to be secure, it is not invulnerable to bad security practices. A 2018 article by Martin Hron of Avast [14] concludes that more than 49 000 MQTT servers were publicly visible on the internet and indexed by Shodan. Of these, 32 000 did not have any password protection. By finding unprotected MQTT servers and subscribing to the top-most wildcard topic #, you can receive all messages being published.

Searching for port 1883 data in the Nordics on Shodan gives a result of 2443 hits. Of these, 1940 are results containing MQTT in the response data and 900 of them can be verified to be accepted connections (i.e., `MQTT Connection Code: 0`). No result at all was available on port 8883, the port used for encrypted MQTT.

Separating the results between each of the Nordic countries, Table III shows the number of endpoints that accepted the MQTT connection (i.e., response code `0`), alongside the total number of responses containing `MQTT`.

Country	Open	Total
Sweden	397	923
Finland	289	604
Norway	132	242
Denmark	73	156
Iceland	9	15

TABLE III
ACCESSIBLE MQTT ENDPOINTS PER COUNTRY

Besides connection code 0 (Connection accepted), the following MQTT connection codes were observed on port 1883 in the Nordic countries:

- 1 (Connection Refused, unacceptable protocol version)
- 2 (Connection Refused, identifier rejected)
- 3 (Connection Refused, server unavailable)
- 4 (Connection Refused, bad user name or password)
- 5 (Connection Refused, not authorized)

C. RTSP

Like MQTT, RTSP can provide vulnerabilities in terms of sharing unintended information through bad security practices. For instance, scanning a network for the RTSP protocol (i.e., looking at port 554) allows for detecting RTSP video stream. If these feeds are not requiring authentication, anyone that finds the address and uses an RTSP client can access the stream—potentially allowing for gaining access to security cameras, home cameras or similar [2].

Searching for port 554 data in the Nordic countries on Shodan gives a result of 16 160 hits. Of these, 15 544 are valid RTSP responses and of those, 7453 are actual established connections (i.e., 200 OK).

For each Nordic country, Table IV shows the number of endpoints that were open to establish RTSP connections (i.e., `RTSP/1.0 200 OK`) through port 554, as well as the total number of valid RTSP responses.

Country	Open	Total
Sweden	2860	4921
Denmark	2082	5911
Norway	1175	2361
Finland	1150	1907
Iceland	186	444

TABLE IV
ACCESSIBLE RTSP ENDPOINTS PER COUNTRY

Interestingly, Denmark seems to have around 1000 more systems reachable through RTSP than Sweden, but in total around 800 less systems which are open to establish connections (i.e., 200 OK).

Besides 200 OK, the following RTSP/1.0 status codes were observed on port 554 in the Nordic countries:

- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed
- 453 Not Enough Bandwidth
- 454 Session Not Found
- 503 Service Unavailable
- 551 Option Not Supported

D. UPnP

UPnP, with the defined port numbers 1900 and 7900, is also suitable to explore with Shodan. Searching for port 1900 in the Nordics gives 4990 results, of which 4957 includes UPnP in the response. Interestingly, all (!) of these UPnP responses is contained in `HTTP/1.1 200 OK` responses, confirming the requests without errors. No responses were given on the multicast event message port 7900. Table V shows the result for each of the Nordic countries.

Country	Open
Sweden	2491
Norway	1516
Denmark	779
Finland	136
Iceland	35

TABLE V
ACCESSIBLE UPNP ENDPOINTS PER COUNTRY

E. Zigbee

Due to Zigbee being a wireless network standard, it is not possible to use Shodan to find information about Zigbee devices and thus, it is outside the scope for this project to quantify vulnerabilities related to Zigbee.

Even though the Zigbee protocol was designed with security in mind a number of security related issues have been found by various researchers. Wara and Yu [26] present one method to perform a replay attack exploiting vulnerabilities in the electronics manufacturer Philip's implementation of the Zigbee protocol for their Hue light bulbs. More in detail, the attack proposed is about capturing on/off-events that are stored and replayed later. As a countermeasure to replay attacks sequence numbers are used but the authors find the sequence number to be reset when the electrical power is lost (i.e., physical switch for the light bulb is used) [26].

F. A note on IPv6 on Shodan

Looking at the difference between IPv4 and IPv6 in the Nordic countries, we see that Shodan has indexed 4 779 369 hits on IPv4 and only 4321 hits on IPv6—many orders of magnitude less in comparison. Table VI shows the available devices through IPv4 and IPv6 divided per country.

Country	IPv4	IPv6
Sweden	4 779 369	255
Finland	1 055 234	133
Norway	613 690	252
Denmark	570 097	3669
Iceland	64 750	12

TABLE VI
ACCESSIBLE ENDPOINTS THROUGH IPV6 PER COUNTRY

Denmark has a disproportionate large amount of IPv6 hits in comparison with the other countries. Looking at global Shodan data, we see that there are only a little over 1 million hits on IPv6 (compared to 410 million on IPv4). Of these, Denmark is placed at 5th place which seems unreasonable.

As of October 2015, Shodan was gathering millions of results per month over IPv6 compared to hundreds of millions of results over IPv4 [18]. The results of our search queries suggests that Shodan still have not made much progress in terms of indexing IPv6. None of the protocols that we searched for (MQTT, RTSP and UPnP) had any reported results in the Nordic countries. This is also the case for Home Assistant.

V. Related Work

A case study of Intelligent Onvif YY HD IP cameras showed that these cameras—which could be representative—have many security vulnerabilities [2]. The study showed that information about the camera could easily be retrieved; both in terms of connectivity but also credentials such as email address and the password as MD5 hash—or even plain text from application information. It was also possible to retrieve the RTSP links to gain access to the live video without any authentication.

A recent study of UPnP-based IoT devices analyzed security vulnerabilities of such devices and how they could be exploited [16]. The study focuses on malicious use of the UPnP interactions advertisement, discovery, action, event subscriptions and control messages. Based on this, a solution to secure these devices by combining a capability-based access control (CapBAC) and attribute-based access control (ABAC) is presented and compared in terms of detection and prevention of the aforementioned vulnerabilities compared to other solutions.

An additional IoT security study aims at explaining attacks such as Denial of Service (DoS), Man in The Middle (MiTM) and Supervisory Control and Data Acquisition (SCADA) attacks from an IoT perspective, and how they can be prevented to strengthen IoT security [3]. The study is however limited to a theoretical perspective and does not aim to give any practical guiding until the recommendations have been implemented, integrated and tested more thoroughly.

Narrowing the scope to the Nordics, a 2020 study looks at IoT as well as "Industrial IoT", IIoT [5]. The paper presents the state of the art in terms of IoT vulnerability scanning, with a systematic literature review as foundation. With this background, a study is conducted on the current situation of IoT security in the Nordics concluding that additional work needs to be done in terms of IoT security.

A study related to this project used Shodan to make vulnerability scanning of IoT devices in Jordan [4]. The primary purpose was to warn the community about security issues and how vulnerabilities could be exploited. The paper presents scans of common unencrypted internet protocols such as HTTP, FTP, RDP and SMB through their respective standard ports. They also quantified how many IP cameras that could be reached as well as some common industrial control systems (ICS).

In the domain of internet-wide port scanning (IWPS), a recent journal article explores the effect of these scans in terms of congestion, especially for Wireless networks (WLAN) [10]. They present a novel scan rate optimization method to allow for reaching as high scan rates as possible without imposing congestion into the target network, concluding their method to be more efficient than the conventional method. A recent (May 2021) journal article, sharing some of the authors, expanded on these concepts but specifically for IPv6. They present an optimization to provide security maximization while ensuring quality of service (QoS), however concluding that more work needs to be done in this area [25].

VI. Conclusion

In this paper we have investigated a set of devices, applications and protocol from a security perspective. Furthermore, a number of applications and protocols have been subject to analysis and known vulnerabilities described. One conclusion that can be drawn is that all types of devices, applications and protocols have vulnerabilities, but the severity varies greatly. Another conclusion that can be drawn is the importance of keeping systems patched and updated. Using a vulnerable version of Home Assistant could pose a huge security risk as an attacker potentially could steal API credentials. This, in combination with the fact that Home Assistant aims to be the system that glues the pieces together (implying that it may hold a great number of credentials), suggests that the result of being subject to an attack could be a disaster.

The quantification of vulnerable Home Assistant installations aligns with our expectations as it follows the population relation. However, one interesting finding is that the most popular port used among all Home Assistant installations and vulnerable Home Assistant installations differ. For vulnerable installation the default port, 8123, is the most popular in contrast to port 443 being the most used port overall.

Future work could involve analyzing more devices available in the Nordics, as well as some additional protocols. Also, investigating what kind of data that is being exposed through various protocols might be of interest as well.

References

[1] *2020 and Beyond: A Record Year for Zigbee as the Leading Networking Technology for Smart Home and Building*. Zigbee Alliance. URL: https://zigbeealliance.org/news_and_articles/zigbee-momentum/.

[2] P A Abdalla and C Varol. "Testing IoT Security: The Case Study of an IP Camera". In: *2020 8th International Symposium on Digital Forensics and Security (ISDFS)*. 2020, pp. 1–5. DOI: 10.1109/ISDFS49300.2020.9116392.

[3] E Ahmed et al. "Internet of Things (IoT): Vulnerabilities, Security Concerns and Things to Consider". In: *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. 2020, pp. 1–6. DOI: 10.1109/ICCCNT49239.2020.9225283.

[4] H Al-Alami, A Hadi, and H Al-Bahadili. "Vulnerability scanning of IoT devices in Jordan using Shodan". In: *2017 2nd International Conference on the Applications of Information Technology in Developing Renewable Energy Processes Systems (IT-DREPS)*. IEEE, Dec. 2017, pp. 1–6. DOI: 10.1109/IT-DREPS.2017.8277814.

[5] A Amro. "IoT Vulnerability Scanning: A State of the Art". In: *Computer Security*. Ed. by S Katsikas et al. Cham: Springer International Publishing, 2020, pp. 84–99. ISBN: 978-3-030-64330-0. DOI: 10.1007/978-3-030-64330-0_6.

[6] *Architecture — Home Assistant Developer Docs*. Home Assistant. URL: https://developers.home-assistant.io/docs/architecture_index/.

[7] A Banks et al., eds. *MQTT Version 5.0*. OASIS, 2019. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html.

[8] A Donoho et al. *UPnP Device Architecture version 2.0*. Apr. 2020. URL: https://openconnectivity.org/upnp-specs/UPnP-arch-DeviceArchitecture-v2.0-20200417.pdf.

[9] D Gislason. *Zigbee Wireless Networking*. Newnes, 2008. ISBN: 9780750685979.

[10] H Hashida, Y Kawamoto, and N Kato. "Efficient Delay-Based Internet-Wide Scanning Method for IoT Devices in Wireless LAN". In: *IEEE Internet of Things Journal* 7.2 (Feb. 2020), pp. 1364–1374. ISSN: 2327-4662. DOI: 10.1109/JIOT.2019.2954539.

[11] *Home Assistant*. Home Assistant. URL: https://www.home-assistant.io/.

[12] *home-assistant/core: Open source home automation that puts local control and privacy first*. Home Assistant. URL: https://github.com/home-assistant/core.

[13] *home-assistant/core/requirements.txt*. Home Assistant. 2021. URL: https://github.com/home-assistant/core/blob / e1427c45f2fd5dc777a55fe129e9f7ac0743a7cb / requirements.txt.

[14] M Hron. *Are smart homes vulnerable to hacking?* Avast, 2018. URL: https://blog.avast.com/mqtt-vulnerabilities-hacking-smart-homes.

[15] *HTTP - Home Assistant*. Home Assistant. URL: https://www.home-assistant.io/integrations/http/.

[16] G Kayas et al. "An Overview of UPnP-based IoT Security: Threats, Vulnerabilities, and Prospective Solutions". In: *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. 2020, pp. 452–460. DOI: 10.1109/IEMCON51383.2020.9284885.

[17] m punkt nu Sverige AB. *ODROID-N2+ Home Assistant Blue bundle Limited edition*. URL: https://en.m.nu/controllers-z-wave/odroid-n2-home-assistant-blue-bundle-limited-edition (visited on 05/15/2021).

[18] J Matherly. *Complete Guide to Shodan*. Leanpub, 2016.

[19] *NVD - CVE-2021-3152*. NIST - National Institute of Standard and Technology. URL: https://nvd.nist.gov/vuln/detail/CVE-2021-3152.

[20] Open Connectivity Foundation. *About UPnP*. URL: https://openconnectivity.org/developer/specifications/upnp-resources/upnp (visited on 04/09/2021).

[21] *Philips hue Bridge V2 - Zigbee Alliance*. Zigbee Alliance. URL: https://zigbeealliance.org/zigbee_products/philips-hue-bridge-v2-51/.

[22] A Rao, R Lanphier, and H Schulzrinne. *Real Time Streaming Protocol (RTSP)*. RFC 2326. Apr. 1998. DOI: 10.17487/RFC2326. URL: https://rfc-editor.org/rfc/rfc2326.txt.

[23] *Storing secrets - Home Assistant*. Home Assistant. URL: https://www.home-assistant.io/docs/configuration/secrets/.

[24] Ben Tomlin. *home-assistant-config/ha-main.png at master · benct/home-assistant-config*. 2019. URL: https://github.com/benct/home-assistant-config/blob/master/screenshots/ha-main.png (visited on 05/15/2021).

[25] S Verma, Y Kawamoto, and N Kato. "A Network-Aware Internet-Wide Scan for Security Maximization of IPv6-Enabled WLAN IoT Devices". In: *IEEE Internet of Things Journal* 8.10 (May 2021), pp. 8411–8422. ISSN: 2327-4662. DOI: 10.1109/JIOT.2020.3045733.

[26] M S Wara and Q Yu. "New Replay Attacks on ZigBee Devices for Internet-of-Things (IoT) Applications". In: *2020 IEEE International Conference on Embedded Software and Systems (ICESS)*. 2020, pp. 1–6. DOI: 10.1109/ICESS49830.2020.9301593.

[27] R Williams et al. "Identifying vulnerabilities of consumer Internet of Things (IoT) devices: A scalable approach". In: *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*. 2017, pp. 179–181. DOI: 10.1109/ISI.2017.8004904.

APPENDIX A
CODE FOR VULNERABILITY SEARCHES ON SHODAN

The following Python code was created to help identify vulnerable Home Assistant devices:

find-vulnerable.py

```python
import json
import os
import re
import time

from shodan import Shodan


def is_vulnerable(shodan_result):
    if 'home_assistant' in shodan_result and 'version' in shodan_result['home_assistant']:
        # Match on Home Assistant version
        match = re.search(r'(\d|\.)*', shodan_result['home_assistant']['version'])
        reference_version = 202115  # HA version 2021.1.5
    else:
        # Match on httpaio version
        match = re.search(r'(?<=aiohttp/)((\d|\.)*)', shodan_result['data'])
        reference_version = 373  # httpaio version 3.7.3

    if match:
        version = match.group(0)
        version = version.replace('.', '')
        version = int(version)
        if version < reference_version:
            return True
    return False


def vulnerable_devices_per_country(countries):
    result = {
        'sweden': 0,
        'norway': 0,
        'denmark': 0,
        'finland': 0,
        'iceland': 0
    }
    for country in countries.keys():
        for city in countries[country].keys():
            result[country] = result[country] + countries[country][city]

    return result


def fetch_and_store_shodan_result():
    try:
        api = Shodan(os.environ.get('SHODAN_API_KEY'))
        query = 'http.title:"Home Assistant" country:se,no,dk,fi,is'
        current_page = 0
        total_pages = 39  # Derived from Shodan website (total matches / 100)

        while current_page <= total_pages:
            print('Current page: ' + str(current_page))
            results = api.search(query, page=current_page)

            f = open('nordic-' + str(current_page) + '.json', "w")
            f.write(json.dumps(results['matches'], sort_keys=True, indent=4))
            current_page = current_page + 1

            time.sleep(10)  # Wait to not overwhelm Shodan API
    except Exception as e:
        print('Error: ', e)
```

```
if __name__ == '__main__':
    fetch_and_store_shodan_result()

    countries = {
        'sweden': {},
        'norway': {},
        'denmark': {},
        'finland': {},
        'iceland': {}
    }
    ports = {}
    ip_version = {
        'v4': 0,
        'v6': 0
    }
    for i in range(0, 34):
        f = open('./nordic-' + str(i) + '.json')
        results = json.load(f)

        for result in results:
            try:
                if is_vulnerable(result):
                    country = result['location']['country_name'].lower()
                    city = result['location']['city']
                    port = result['port']

                    if city in countries[country]:
                        countries[country][city] = countries[country][city] + 1
                    else:
                        countries[country][city] = 1

                    if port in ports:
                        ports[port] = ports[port] + 1
                    else:
                        ports[port] = 1

                    if re.search(r'^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\$',
                                 result['ip_str']):
                        # IPv4
                        ip_version['v4'] = ip_version['v4'] + 1
                    else:
                        # IPv6
                        ip_version['v6'] = ip_version['v6'] + 1
            except Exception as e:
                print('Error', e)
    per_country = vulnerable_devices_per_country(countries)

    summary = {'per_country': per_country, 'ip_version': ip_version, 'port': ports,
               'countries': countries}

    f = open('result.json', "w")
    f.write(json.dumps(summary, sort_keys=True, indent=4))
```