Proceedings

5th IEEE European Symposium on Security and Privacy Workshops

EUROS&PW 2020

Proceedings

5th IEEE European Symposium on Security and Privacy Workshops

7–11 September 2020
Virtual Event

Los Alamitos, California
Washington • Tokyo

The papers in this book comprise the proceedings of the meeting mentioned on the cover and title page. They reflect the authors' opinions and, in the interests of timely dissemination, are published as presented and without change. Their inclusion in this publication does not necessarily constitute endorsement by the editors, the IEEE Computer Society, or the Institute of Electrical and Electronics Engineers, Inc.

BMS Part Number CFP20N35-ART
ISBN 978-1-7281-8597-2

Additional copies may be ordered from:

IEEE Computer Society
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314
Tel: + 1 800 272 6657
Fax: + 1 714 821 4641
http://computer.org/cspress
csbooks@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: + 1 732 981 0060
Fax: + 1 732 981 9667
http://shop.ieee.org/store/
customer-service@ieee.org

IEEE Computer Society
Asia/Pacific Office
Watanabe Bldg., 1-4-2
Minami-Aoyama
Minato-ku, Tokyo 107-0062
JAPAN
Tel: + 81 3 3408 3118
Fax: + 81 3 3408 3553
tokyo.ofc@computer.org

Individual paper REPRINTS may be ordered at: <reprints@computer.org>

Editorial production by Lisa O'Conner

IEEE Computer Society
Conference Publishing Services **(CPS)**
http://www.computer.org/cps

Defensive Programming for Smart Home Cybersecurity

Maria Teresa Rossi*, Renan Greca*†, Ludovico Iovino*, Giorgio Giacinto‡ and Antonia Bertolino†

* *Gran Sasso Science Institute – L'Aquila, Italy, Email: {firstname.lastname}@gssi.it*

† *ISTI - CNR – Pisa, Italy, Email: antonia.bertolino@isti.cnr.it*

‡ *University of Cagliari – Cagliari, Italy, Email: giacinto@diee.unica.it*

***Abstract*—Cybersecurity has become a real issue in the development of smart services in the smart home domain, which is formed by a System of Systems where several smart objects are connected to each other and to the Internet. However, these connections expose the devices to possible attackers inside or outside the network, who may exploit software or hardware vulnerabilities to achieve malicious goals. To alleviate this issue, the use of defensive programming assertions can allow the behaviour of smart objects to be monitored and checked for correctness. Furthermore, open source intelligence tools, such as the Shodan search engine, provide features that could be leveraged to detect potential vulnerabilities. In this paper, we propose an approach for the monitoring of Systems of Systems in the smart home domain exploiting the defensive programming paradigm in combination with Shodan APIs.**

1. Introduction

Many complex tasks are today handled by *Systems of Systems* (SoS), which are compositions of independent systems that collaborate towards achieving a common task [1], or *mission*. Such a mission goes beyond the goals and capabilities of the individual systems, or *constituents*: indeed the concept of SoS allows for the rapid development of innovative and broad distributed systems in many critical domains [2]. Examples may include: the communication and control of a smart city environment, the safe and intelligent distribution of energy in smart grids, or the immediate response and coordination of IT networks in face of an emergency situation. All these and many other situations rely on the collaboration among a variety of hardware and software modules, and often humans as well, to provide important services to citizens and society. By allowing interconnected physical objects to sense the environment, share data and act accordingly, the Internet of Things (IoT) [3] plays a central role in the diffusion of SoS. In this paper, we focus on IoT-enabled *smart homes*: a facility, usually recently constructed or reformed, that is equipped with special structured networking capabilities enabling occupants to remotely control or program an array of automated home electronic devices [4]. Considering the context of smart homes, security is a crucial concern in order to keep homeowners, as well as their property, safe from intruders and to prevent unwanted behavior of the connected devices and systems. Indeed, along with enhancing our quality of life, smart homes also increase our risk surface, in particular concerning cybersecurity. If a single constituent system within an SoS is affected by a vulnerability, this can serve as an entry point for the whole network, thus putting the entire SoS at risk of being compromised. Cybersecurity vulnerabilities can have several root causes [5], such as: software, as bugs or design flaws; hardware issues, such as faulty components or sensitivity to certain kinds of interference; unnecessarily open networking ports; inadequate authentication and insufficient access management mechanisms including the use of default or weak passwords; improper patch management; or insufficient audit mechanisms. These vulnerabilities can expose a system to attackers either by allowing them access to the underlying hardware of a specific device, or they can be leveraged to open the access to other systems in the network through default chains of trust. In other words, vulnerabilities create the risk of having the system exposed to any remote attacker over the Internet. As such, it is crucial that vulnerabilities are detected by the system managers as early as possible and measures are taken to properly address them and reduce the risk of compromising the network. Within a smart home environment the connection of all the devices into a network can, on one hand, facilitate communication and control; on the other hand, it can lead to security issues [4]. There are two types of security threats [5], [6]: i) *passive*, when the objective of the attack is only to steal information from the system without affecting it; ii) *active*, when the attack alters the normal functioning of the system. Cyber-attacks are not a new concern to IoT [7], but as IoT devices and connections become a prevalent part of modern life, the safety and security of users must be dealt with upmost importance. The usual CIA triad, Confidentiality, Integrity, and Availability, can be used to categorize IoT attacks. While attacks against availability aim to disrupt the system, attacks against confidentiality and integrity are more subtle, where the attacker breaches the communication channel between two systems in an attempt to intercept messages sent among them, and eventually tamper with them. These kind of attacks can be the stepping stone for attacks with more severe consequences that can bring the SoS in an unsafe state, and undermine its availability. Attacks towards smart homes usually target common devices such as smart refrigerators or other con-

DOI 10.1109/EuroS&PW51379.2020.00086

nected electronic devices in order to change their expected behaviour. Typically, attacks targeting IoT devices may expose data in real-time with the intent of intercepting communications between devices, allowing the attacker to steal data or trigger malfunctions. For instance, *smart meters* provide real-time data on electricity and gas usage, enabling optimizations of energy consumption and distribution, and empowering consumers to make intelligent decisions about their energy usage. If smart meters are used in the context of an SoS, e.g., by activating other devices when the consumption is under a certain level, an attack can contribute to an erroneous activation and thus excessive or unnecessary energy consumption, as minor damage. If this erroneous behaviour is detected immediately, the user can take actions and correct the behaviour of the affected devices. Moreover, in all cases in which the possible erroneous behaviour of a device has been openly disclosed, the device should be promptly patched, or the triggering event should be blocked.

To avoid these types of security issues, we propose an approach for monitoring the behavior of smart home devices and timely detect potential vulnerabilities. The proposed approach is based on defensive programming combined with the adoption of open source intelligence tools [1] on the interconnected devices through the use of Shodan APIs [2].

2. Smart Consumption in Smart Homes

In the literature there are many definitions for the concept of a *Smart home* [8]. Generally, a smart home is a system including several appliances and services connected in a network in such a way that they can be monitored and managed. In Figure 1 a smart home case study is represented through a high level architecture. A set of connected things form the local network, and in particular a smart meter monitors the energy production and consumption of the building and, according to some thresholds of energy produced or consumed, various smart objects in the smart home system could be activated or deactivated. Usually commercial smart meter monitors offer also APIs, in order to provide customization capabilities, and build third party applications interacting with the meter's data. In this example, the smart meter offers an API for sensing electrical production with solar panels and power consumption in real time of the whole house. In this context, we can have a software system developed with customized source code (*Third-Party App*) that continuously *reads* the smart meter API available from a local IP address and controls all the smart objects connected to the LAN. The main application can be seen as an intelligent energy consumption scenario, that is, if the solar panels are producing more than a certain amount of energy and the current energy consumption is very low, the software can decide to activate an energy intensive smart device, such as a washing machine.

If we consider an exemplary application that senses energy supply/demand and activates devices accordingly, the response returned by the smart meter API can be simplified as in Listing 1[3]. Usually smart appliances provide data in common interchange formats, e.g., JSON, XML, so other applications can easily interact. REST API are provided in these cases, whereby this paradigm offers an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data. The first attribute of the API's JSON response of the smart meter is the result of the GET call, in which two important values are reported: the house's consumption of energy in kW (in this case 2.0) and the instant electricity production of the solar panels (in this moment 0). This would characterize a moment in which the solar panels are not producing, e.g., at night or cloudy day and the consumption is up to 2.0 KW, for instance activating a machine machine can range from 500 watt up to 1300 watt (1.3Kw). Additionally, the timestamp and the electricity price could be retrieved, but these are unnecessary for this case study. In this setting, we consider a simple Java application reading the smart meter API in order to make decisions about the activation of smart devices.

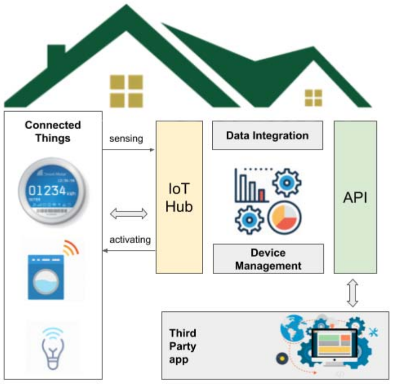


Figure 1. Architecture of the presented case study.

Listing 1. Sample GET response from the the smart meter's API.

```
{
 "electricity": {
  "consuming": {
   "actual": {
    "reading": 2.0,
    "unit": "kW"
   },
  },
  "received": {
    "actual": {
    "reading": 0,
    "unit": "kW"
   },
  }
 },
 "gas": {
  "reading": 0.5,
  "unit": "m3"
 }
}
```

The reported application is a simplified version of a complex application that also considers the daily history of electricity production/consumption and matches the information with a weather station, in order to guarantee

[1] https://osintframework.com

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

[3] This is a simplification of how an API for smart meter can be implemented, inspired by a commercial product: OWL Intuition PV.

that the production of solar energy will be sufficient for the needed amount of time. For sake of simplicity we assume that the *sensing* and *activating* methods are part of a scheduled routine that also detects other useful information, but Listing 2 only contains a snippet of code doing the simple task we are discussing.

Consider this snippet as an example of third-party code that works with the IoT devices. It continuously calls the smart meter API URL and stores the resulting JSON in the `smartmeter` variable. Then this application checks if the energy available is more than 5 kW and, if so, it instantiates a device as a new `Thing` which is activated according to the priority list; otherwise, it continues to monitor the API until this condition is matched.

Listing 2. Example of Third Party app

```
while (true) {
    URL api =
        new URL("https://192.168.1.12/"+
                "smarthome/meter/get");
    Thing smartmeter = new Thing();
    smartmeter.setIpAddress("192.168.1.12");
    String content =
        MyFileUtils.urlToString(api,"utf-8");
    JSONObject toJsonObject =
        new JSONObject(content);
    JSONObject smartmeter =
        toJsonObject.get("electricity");
    Double consumption =
        smartmeter.get("consuming"))
                .get("actual"))
                .get("reading");
    Double received = ...;
    Double delta = 5.0;
    if (received - consumption >= delta) {
        Thing t = new Thing();
        t.activate();
        ...
    }
}
```

If the smart meter has exposed vulnerabilities, it can be the target of malicious attacks which could affect the reading shown in the API response. Optimistically, this could result in the undesired activation or deactivation of devices, affecting the electricity consumption of the smart home. However, such an attack could be escalated to an even worse situation, causing incidents such as power outages or even a voltage overload.

3. Detecting Vulnerabilities in 3rd Party Apps

In IoT-enabled SoS, smart objects are usually connected to sensors, which collect information about the environment, and actuators, which perform actions according to the collected information. If one such smart object is subject to a vulnerability, it could be exploited as an entry point and expose the entire SoS to malicious interference. One way of protecting smart objects from attackers is by using tools to automatically and timely detect vulnerable devices. One such tool is Shodan, which provides a dashboard and APIs listing devices currently publicly exposed over IP to anyone on the Internet. An example of a Shodan entry can be a vulnerability publicly available showing how to execute a command injection to a Smart Camera connected over IP. This would mean that if the user recognise this device as part of its network, he / she should take immediately countermeasures to deal with this issue. Shodan can be used to inspect if one of the systems in the network, subject of our study, has a potential vulnerability, by checking if it is exposing a public interface to the Internet. It is important to observe that, in a SoS context, the API's purpose is to detect devices that have services publicly exposed to the Internet, along with details about these services. Once it is known that a device is reachable by Shodan, further analysis must be performed to determine whether or not a vulnerability exists, so it can finally be patched if needed.

A constant monitoring using Shodan's APIs will help an SoS to adapt and recalculate in case a vulnerability is exposed by one of its constituent systems. The Shodan API provides several features that could be valuable to detect potential vulnerabilities in a smart environment, such as: IP lookup, host search, IP crawl, automatic alert in a given list of IP addresses.

4. Defensive Programming against SoS Vulnerabilities

The approach we propose to deal with the case study presented in section 2 and its potential vulnerabilities is to use a well-known programming paradigm called *Defensive programming* [9]. According to such paradigm, developers must assume that any possible vulnerability will be exploited and thus apply good practices to develop code that is secure against attacks. One such practice consists in preserving code correct execution by using *assertions*. The idea is to add assertions in the code so to check if the code is executing correctly and produced values are within expectations. As known, an assertion is in fact an expression that must be evaluated *true* if the program is running correctly; if not, defensive programming requires that the execution be terminated. Defensive programming also considers how the program can be stopped securely. By combining this paradigm with the functionalities of a vulnerability monitoring systems like Shodan, we can handle SoS vulnerabilities in multiple scenarios. In the case study represented in Figure 1, considering assertions in the development of third party applications may help to check if the interactions among devices involves a *Thing* that appears in the Shodan directory, as vulnerable. If we find that even one of the smart objects is exposed to vulnerabilities the program will notify the system manager and / or alter its behaviour. Of course this is applicable to the smart thing software itself, even if it results easier to be managed with respect to third party apps, making this interleaving connections more complex. As such, defensive programming can be a good practice in the development of third-party applications for SoS but it is not a silver bullet, nor the unique solution. For instance, the code reported in listing 2, in particular every occurrence of the concept `Thing`, should be surrounded by an assertion checking if this device appears as exposed on the Shodan repository. To provide a sample of this approach, we simulated the Shodan API response locally and the result is reported in Listing 3. We had to simulate the response because the Shodan database is (fortunately) not densely populated, and we would not like to expose a real vulnerability of a third-party device for our experiment. This result is the

JSON response of the REST call `/shodan/host/ip`, checking if the IP indicated as parameter appears in the list of exposed devices.

Listing 3. Sample GET response from Shodan's API.

```
{
  "region_code": null,
  "ip": "192.168.1.12",
  "country_name": "Italy", "hostnames": [],
  "data": [
    {
      "product": "Smart Meter Device of the GSSI",
      "os": null,
      "timestamp": "2014-01-12T18:25:41",
      "isp": "Telecom"
    }
  ]
}
```

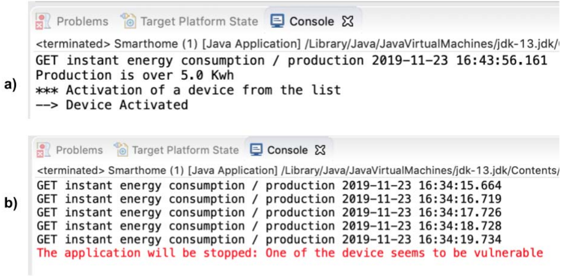
This result highlights that the smart meter used in the application, connected then with the required ip, is exposed and might be affected by a vulnerability, so it is important to alert the manager of the smart home.

The code reported in Listing 2 could be easily extended with the defensive strategy we propose and, in particular, the API call for getting the electrical information from the smart meter should be refined as reported in Listing 4.

Listing 4. Defensive Programming Example

```
try {
    Shodan shodan = new Shodan(key);
    if (received - consumption >= delta) {
        assert
            shodan.checkIfDeviceIsVulnerable(
                smartmeter.getIp_address()
            ):
            "Device seems to be vulnerable";
        Thing t = new Thing();
        t.activate();
    }
} catch (AssertionError error) {
    // Output expected AssertionErrors.
    System.err.println(
    "The application will be stopped");
    // here the developer can customize the
    // actions to be undertaken
    return;
}
```

This very simple code, refined in the perspective of defensive programming, prevents the activation of the device controlled by parameters of the smart meter in case the smart meter appears in the Shodan repository.



```
<terminated> Smarthome (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.jdk/
GET instant energy consumption / production 2019-11-23 16:43:56.161
Production is over 5.0 Kwh
*** Activation of a device from the list
--> Device Activated
```

```
<terminated> Smarthome (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/
GET instant energy consumption / production 2019-11-23 16:34:15.664
GET instant energy consumption / production 2019-11-23 16:34:16.719
GET instant energy consumption / production 2019-11-23 16:34:17.726
GET instant energy consumption / production 2019-11-23 16:34:18.728
GET instant energy consumption / production 2019-11-23 16:34:19.734
The application will be stopped: One of the device seems to be vulnerable
```

Figure 2. Log of the running application

The code reported in Listing 4 should replace the affected part of the code in Listing 2. The main point deserving attention is the assertion in which the `shodan` library with the `checkIfDeviceIsVulnerable` method to check if the device is listed in the Shodan repository. If the assertion fails, the exception will be treated and the program will be stopped. This behaviour can be further customized to better manage the flow of activation or notification to the developer.

The log reported in Figure 2 reports two flows of execution of the code. Figure 2 a) reports the monitoring in which a new device is activated when there is sufficient electricity available. In Figure 2 b) the condition is not matched, so when the event is triggered the assertion fails, warning the system administration of a potential vulnerability and halting the application. This log simply shows the different flows in the cases where the assertion passes or fails. In the a) part, the log shows that the device, not being listed in the Shodan repository, triggers the activation of another device from the priority list. In the option b) the device is actually listed during the monitoring phase and then an alert (highlighted in red) is raised to warn the system administrator to resolve the issue. Of course this example has been tested in a local environment with an Eclipse installation to simulate the home environment, the API and the flow. The action undertaken when the application is discovered as vulnerable is limited to a console print out. We plan to simulate this in a real scenario with sensing devices and a real case study. It worth noting that while this paper uses Shodan as the repository of the vulnerabilities, the approach we propose is general enough to be extended to other sources. More in general, Open Source Intelligence (OSINT) refers to the collection of information from public sources to gather data of publicly available information sources about individuals and business intelligence purposes. Based on it this approach can be extended to add other sources in order to gather more information about exploits or other info about our devices vulnerabilities.

5. Relevant Application Scenarios

Beyond the simple case discussed in the previous section, we foresee several other possible scenarios in which defensive programming combined with OSINT application can support SoS cybersecurity. In this section we briefly discuss some of such possible scenarios.

Assertions Actions The used assertion simply stops the program to avoid possible unwanted behaviour without planning a possible action of resolution. For this reason, a notification mechanism could be introduced to avoid stopping the program's execution and react accordingly and manually on the affected code. This point can be further automated with self adaptation plans.

Self-adaptation Self-adaptive software checks its own behaviour and evolves autonomously when the evaluation indicates that it is not possible to accomplish what the software is intended to do [10]. Self adapting may be a positive strategy if the Shodan library reveals that one of the device results as affected by vulnerabilities. Instead of stopping execution, the program can self-adapt its behaviour with a reconfiguration plan. For instance, if the assertion fails the program could try to continue to coordinate the activation of other devices without relying on the affected device. An adaptation plan could be a lookup for other smart meters connected to the network or use historical data to make informed decisions (AI could be the cutting edge technology).

Online Testing Our approach can be leveraged for promoting the application of proactive online testing activities, for the purposes of security assessment [11]. Online testing consists of continuing the execution of test cases on the system in production, and is recently gaining attention, motivated by the continuous evolution and complexity of modern pervasive systems. This is especially true in dynamic contexts such as smart homes, and more in general for any SoS. We envisage a potential application of an online testing strategy that by periodically checking the inserted assertions (i.e., considering a test suite aiming at covering the included assertions), would allow to timely detect, and react against, possible vulnerabilities, before these are experienced by the SoS end-users. Alternatively, we could instead inject security attacks on purpose, similarly to Netflix Chaos Monkey concept [12], so to test if the system is ready to react to attacks.

Security Testability It is well known since the early 90's that proper placement of assertions improves a program's testability [13]. By including *ad hoc* assertions for dealing with potential vulnerabilities, we are improving the potential of disclosing possible attacks and hence naturally increase a system testability for security. Of course, such statement is strictly dependent on the quality of the assertion themselves, therefore it is important in future work to study the effectiveness and extent of the assertions we produce, and where we embed them.

6. Related Works

In the literature, the issue of security in smart city systems is already a hot topic and different approaches to solve it have been proposed. With regard to the smart city domain, in [14] the authors discuss various privacy and security solutions, recommendations, and standards for smart cities and their services. The challenges highlighted here are given by the exploitation of ICT to offer high quality smart city services to the people. Consequently, IoT contexts present several security, privacy and ethical challenges as exposed in [15]. The issues presented here are due to the complexity of IoT systems and the heterogeneity of their components. The authors highlight also the security issues related to the connections between IoT objects. Thus, they propose a threat model for all the smart city areas, including smart home. Moreover, in [16] the authors present the results of a security assessment of IoT devices extending a public dataset of Shodan queries. From this study, they extracted query terms used to access to a class of IoT devices, showing how easy is to detect vulnerabilities in IoT devices. These queries can serve as a starting point for further application of Shodan in cybesecurity applications. Focusing on the smart home context, in [17] is described the context of smart buildings in which there is the need of managing the overflowing of energy generation and it is raised up the need of the integration of cybersecurity testing. In particular, here they highlight the importance of knowing what type of attack is happening to be able to implement a digital forensic readiness. In [18] are exposed the risks of having cloud-based IoT technologies in the different smart city contexts. In particular, for the smart buildings context they propose a robust Intrusion Detection System (IDS) based on machine learning algorithms that detects any abnormality in the traffic of data in the cloud. [19] proposes an approach that calculate threat factors based on different features (e.g., system architecture, networks, operating systems, database schemas, encryption techniques, security policies, business operations, and corporate data). The calculated factors are responsible for telling how a smart city system can face cyber-threats in terms of robustness. The work in [20] proposes an architecture for Smart Homes that facilitates security analysis of the IoT systems in the house, giving a global view of the devices and networks composing the smart home. This approach wants to sketch the different systems of a smart home system in order to know what can be the crucial vulnerabilities. In [21], the authors highlight the challenges of preserving the security of SoS domains due to their characteristics (e.g., operational independence of constituent systems, absence of central authority). Here, one of the main security challenge is to manage cascading attacks, thus, the authors propose a DSML to represent SoS security architecture enabling the discovery, analysis and resolution of cascading attacks. With regard to the defensive programming paradigm, in the literature we can find how its exploitation can be used to deal with cyber-security issues. In [22] a defensive programming approach is exposed to a sensor architecture in order to identify sensor malfunctions upon their occurrence. This continuous checking approach is implemented as a model containing a set of rules that promptly alerts about a malfunction and contributes in diagnosing the problem. Regarding the IoT context, in [23] the authors expose security and privacy issues in the deployment of smart camera sensors. Here, it is explained how can be exploited the defensive programming paradigm to ensure trustworthiness in a smart camera sensors network. In [24] an iterative process for security management in systems characterised by an high inter-connectivity (e.g. Smart Grids) is proposed. Here, four security layers are implemented aiming to increase the security of decentralized control algorithms for critical infrastructures. In the first of these layers, some recommended best practices can be found, including defensive programming, to help in preserving from defeats the infrastructure single nodes in order to prevent infections of the other nodes in the infrastructure. Another approach for cybersecurity in a smart city environment is trust management [25], in which devices store information about each other and maintain a level of trust based on past interactions. This approach can help identify faulty or misbehaving members of the network, although it doesn't stop an external attacker from taking control of a previously trusted, but vulnerable, device.

7. Conclusions

Cybersecurity is a real issue in the development of smart services in the smart home domain. Different smart objects communicate automatically and autonomously increasing their exposure to vulnerabilities that can affect the entire SoS. This ripple effect can endanger everyday life activities including operations that can even include safety issues. In this paper we introduced an approach for the continuous monitoring of SoS that combines defensive programming techniques with the usage of Shodan APIs. We illustrated a possible application of the approach in a smart home environment that can of course be extended to more complex scenarios and domains, e.g., smart cities. For this case study, the presented approach would avoid failures that could lead to energy wastage, appliances breakdowns or disclosure of personal data. In the smart city domain there are other contexts in which this approach could be applied. More in general, whenever we have a SoS implying the collaboration among different smart objects connected to a network, the approach can help timely detect potentially exposed vulnerabilities. For instance, in a smart mobility environment in which the traffic is monitored and managed using IoT sensors connected in a network, we can use the same approach of checking their IP addresses. Moreover, in this context vulnerabilities exposure could produce environmental issues in terms of air pollution. Nowadays several researchers advocate a paradigm of security-by-design [26], according to which a system must be built to be secure since its very inception. In the case of SoS, such a paradigm appears hardly applicable, because of the inherent dynamism and autonomic nature of such systems. We see our proposal of adopting defensive programming and of leveraging OSINT technology as a preliminary attempt to get closer to secure-by-design SoS. This work is directed to SoS developers and researchers, and we hope can provide guidelines or inspiration to be taken under consideration. In future work, we aim at developing a complete framework for more applications and further case studies. Moreover we plan to investigate model based design and development for secure systems. Using Model Driven Engineering (MDE) helps to shift the focus from code-centric techniques to code generation. In this context it would be interesting to investigate whether designing systems with models may support the automation of some steps. For instance we could investigate the automatic generation of assertions in our applications with specific protected code areas that can be further customized by developers.

References

[1] M. W. Maier, "Architecting principles for systems-of-systems," *Systems Engineering*, vol. 1, no. 4, pp. 267–284, 1998.

[2] C. B. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, and J. Peleska, "Systems of systems engineering: basic concepts, model-based techniques, and research directions," *ACM Computing Surveys*, vol. 48, no. 2, pp. 1–41, 2015.

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[4] R. J. Robles, T.-h. Kim, D. Cook, and S. Das, "A review on security in smart home development," *International Journal of Advanced Science and Technology*, vol. 15, 2010.

[5] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.

[6] N. Komninos, E. Philippou, and A. Pitsillides, "Survey in smart grid and smart home security: Issues, challenges and countermeasures," *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 1933–1954, 2014.

[7] M. Abomhara *et al.*, "Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks," *Journal of Cyber Security and Mobility*, vol. 4, no. 1, pp. 65–88, 2015.

[8] Li Jiang, Da-You Liu, and Bo Yang, "Smart home research," in *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*, vol. 2, 2004, pp. 659–663.

[9] J. K. Teto, R. Bearden, and D. C.-T. Lo, "The impact of defensive programming on i/o cybersecurity attacks," in *Proceedings of the SouthEast Conference*, 2017, pp. 102–111.

[10] R. Laddaga, "Creating robust software through self-adaptation," *IEEE Intelligent Systems*, vol. 14, no. 3, pp. 26–29, 1999.

[11] A. Bertolino, G. De Angelis, S. Kellomaki, and A. Polini, "Enhancing service federation trustworthiness through online testing," *Computer*, vol. 45, no. 1, pp. 66–72, 2011.

[12] C. Bennett and A. Tseitlin, "Chaos monkey released into the wild," *Netflix Tech Blog*, vol. 30, 2012.

[13] J. M. Voas and K. W. Miller, "Software testability: The new verification," *IEEE software*, vol. 12, no. 3, pp. 17–28, 1995.

[14] R. Khatoun and S. Zeadally, "Cybersecurity and privacy solutions in smart cities," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 51–59, 2017.

[15] A. Seeam, O. S. Ogbeh, S. Guness, and X. Bellekens, "Threat modeling and security issues for the internet of things," in *2019 Conference on Next Generation Computing Applications (NextComp)*, 2019, pp. 1–8.

[16] A. Albataineh and I. Alsmadi, "Iot and the risk of internet exposure: Risk assessment using shodan queries," in *2019 IEEE 20th WoWMoM International Symposium*, 2019, pp. 1–5.

[17] E. Bajramovic, K. Waedt, A. Ciriello, and D. Gupta, "Forensic readiness of smart buildings: Preconditions for subsequent cybersecurity tests," in *2016 IEEE International Smart Cities Conference (ISC2)*, Sep. 2016, pp. 1–6.

[18] N. Sengupta, "Designing cyber security system for smart cities," in *Smart Cities Symposium 2018*, April 2018, pp. 1–6.

[19] P. Wang, A. Ali, and W. Kelly, "Data security and threat modeling for smart city infrastructure," in *2015 International Conference on Cyber Security of Smart Cities, Industrial Control System and Communications (SSIC)*, Aug 2015, pp. 1–6.

[20] K. Ghirardello, C. Maple, D. Ng, and P. Kearney, "Cyber security of smart homes: Development of a reference architecture for attack surface analysis," in *Living in the Internet of Things: Cybersecurity of the IoT - 2018*, March 2018, pp. 1–10.

[21] J. El Hachem, Z. Y. Pang, V. Chiprianov, A. Babar, and P. Aniorte, "Model driven software security architecture of systems-of-systems," in *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*, 2016, pp. 89–96.

[22] A. Carteron, C. Consel, and N. Volanschi, "Improving the Reliability of Pervasive Computing Applications By Continuous Checking of Sensor Readings," in *IEEE International Conference on Ubiquitous Intelligence and Computing*, Toulouse, France, Jul. 2016.

[23] M. Loughlin and A. Adnane, "Privacy and trust in smart camera sensor networks," in *2015 10th International Conference on Availability, Reliability and Security*, Aug 2015, pp. 244–248.

[24] M. Stübs, "Towards emergent security in low-latency smart grids with distributed control," in *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, Oct 2018, pp. 1–6.

[25] M. L. Loper and B. Swenson, "Machine to machine trust in smart cities," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 1887–1889.

[26] J. Geismann, C. Gerking, and E. Bodden, "Towards ensuring security by design in cyber-physical systems engineering processes," in *Proceedings of the 2018 International Conference on Software and System Process*, 2018, pp. 123–127.

2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)

EuroSPW 2020

Table of Contents

CACOE: Cyberange Technologies Workshop

CyberCert: Cybersecurity Certification

EuroUSEC: European Workshop on Usable Security

HotSpot: Hot Topics in the Principles of Security and Trust

IWPE: 6th International Workshop on Privacy Engineering

S&B: Security & Privacy on the Blockchain

WACCO: 2nd Workshop on Attackers and Cyber-Crime Operations

WTMC: Workshop on Traffic Measurements for Cybersecurity

SAD: Workshop on Software Attacks and Defenses

SecWeb: SecWeb 2020

SILM: SILM Workshop on Security of Software/Hardware Interfaces