A Push-button security analysis of IoT systems.

Aarti S Kashyap
Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, Canada
akashyap18@ece.ubc.ca

***Abstract*— The Internet of Things (IoT) is playing an important role in different aspects of our lives. Medical devices, smart cars, drones, and smart grids incorporate IoT devices as key components. Many vulnerabilities and attacks have been discovered against different classes of IoT devices. Therefore, we need techniques to analyze their security, so that we can address their potential vulnerabilities. IoT devices, unlike remote servers, are more user-facing and the attacker may interact with them more extensively through physical access. In this paper, we define a formal set of actions for attackers.We perform manual attacks on the system in order to formalize the system. Existing methods of analysis rely on a predefined set of attacks and therefore have limited effect and do not consider all the scenarios. We define an adversarial model for the analysis of IoT systems based on the access the attacker has and the actions that he can perform. We use openAPS(open Artificial Pancreas System) for our case study and evaluate the analysis results of our model on it and based on the threat model which we use, and, it's analysis, we formulate a formal model for openAPS based on what parts of the orginal openAPS to abstract out.**

I. INTRODUCTION

Emerging Internet of Things (IoT) technologies are a central component of a smart world where service ubiquity and network convergence are the cornerstones. With the emergence of ubiquitous computing, intelligence is getting embedded in environmental and daily life items to support a variety of applications based on person-to-person, person-to-machine, machine-to-person and machine-to-machine interactions. Recently, IoT in healthcare industry gained momentum as an important component of fully connected clinical systems, where patients are responsible for their own healthcare. This involves the construction of a DIY (Do it yourself) rig for an individual patient based on BAN design [20] (e.g, mobility, energy consumption, coverage, connectivity).

*We would like to thank NSERC for the research gift.

A. Problem

There have been discovered many attacks and vulnerabilities against IoT devices such as modern cars, smart meters, and medical devices. [1, 2, 3]. However, most of these discovered attacks were discovered in an ad-hoc or opportunistic manner, and may hence not be comprehensive. For Artificial Pancreas systems, a few security review have been done, however, the analysis is from a high level perspective which talks about availability, confidentiality, Integrity and Authorization [21] Therefore, a systematic mechanism to analyze the security of IoT is required by the developers discover the attacks, improve the design or implementation of the system, and find efficient ways to build security mechanisms to detect the attacks.

B. Existing solutions

Prior work for analysis of attacks against software systems consists of three major approaches 1) attack trees [6] 2) attack patters [4] and 3) attack graphs[7,8]. However, attack patterns are very restricted as they are dependent on the knowledge of pre-existing attacks. Attack trees can be generated based on probabilistic models or decision trees using machine learning techniques however they need initial training period in order to detect attacks apart from the ones known. However, for an IoT safety-critical system, the results are non-deterministic every time. For example in an

artificial pancreas system, the amount of insulin to get the blood glucose level to normal depends on the previous basal inputs, and the predictive algorithm which depends on the many factors. Hence there is no specific data set which makes the output of the system deterministic. Different aspects of the system can be analyzed for security attacks such as tampering of data, however, in a IoT system, there are several ways of tampering the data. Data can be tampered during the training period of the predictive algorithm for basals to be injected. Data can be tampered by inducing noise, when it is being sent to the controller, data can be sent repeatedly by rebooting the system before it receives the acknowledgement flag by rebooting it and finally data can be sent by inducing artificial delays.

C. Our approach

To demonstrate our approach we took openAPS (open source Artificial Pancreas system) as the test-bed. openAPS is the open source software which utilizes FDA approved existing hardware technologies available. Using the existing hardware such as insulin pumps and continuous glucose monitor, it predicts the amount of insulin to be injected to maintain the Blood Group levels. In this paper, we try to analyze the formal approaches which we can use to conduct a security analysis of a hybrid system such as openAPS. In order to use formal analysis we develop a three step approach. In the first step we manually study the system and try attacking the system in different ways assuming the attacker has physical access to the system. Based on the manual attacks we performed we categorize the total number of attacks for this system into five attacks. In the final step we make a grid in order to relate those five categories with each and hence form a model of the system based on these observations. We further use a tool called Why3[22] to model the attacks for the system.

D. Challenges

The current formal analysis systems are the systems which have some underlying mathematical model as it's base. For example in order to model a database and find security vulnerabilities the database can be modelled as a series of relational algebraic theories[23]. Similarly, prior work in this field targets systems with well-defined properties (such as communication protocols) [10, 24, 25] in order to find security vulnerabilities. However, IoT devices in general do not have standard implementations which we can translate into a formal model. Hence, in order to first build a formal model, we have to find the right level of abstraction for the system to model it into. If it is too close to the original system, the model will suffer from space state explosion [26], however, if the modelling is too generic then it is bound to suffer from false positives and it will be difficult to map the system to the abstracted model to find the security attacks.

E. Contributions of the paper

We make the following contributions to the paper:

1) Adversarial model for IoT systems: We look into the general attack scenarios for an IoT system where other humans can be in contact with the physical hardware and formulate different set of scenarios possible.

2) Comparison between the previous techniques and the current technique : We explore the previous work done in this area and do a comparison between the practicality of the existing techniques and using the formal analysis model for performing security analysis of the system.

3) openAPS: We perform the attacks which we formulated on openAPS which is a safety-critical IoT system.

4) openAPS vs APS: We analyze how the openAPS is different from current closed loop APS systems and hence why it should require a much more intense security analysis model.

II. ANALYZED SYSTEM

A. Open-source Artificial Pancreas System (openAPS)

OpenAPS is a closed loop APS technology widely available to anyone with compatible medical devices who is willing to build their own system. The basic openAPS setup consists of the following four main components.

OpenAPS differs from other APS currently in clinical trials in two significant ways. First, it is designed to use existing approved medical devices (insulin pumps), commodity hardware (CGM), and

open source software (openAPS). Secondly, it is designed primarily for safety, understandability, and interoperability with existing treatment approaches and existing devices.[15]
The basic openAPS setup consists of the following four main components as shown in Fig.1.

1) A compatible insulin pump : Currently openAPS provides support with Medtronic MiniMed models.OpenAPS allows to remotely set temporary basal rate command with specific range of Medtronic pumps.

2) A CGM : A continuous glucose monitor (CGM) is a device used for monitoring blood glucose on a continual basis by people with either type I or type II diabetes.Before getting started the users are requested to collect 30 days worth of data using the CGM. They can store data with a compatible open source tool Nightscout [16] which allows the openAPS to fetch data for predicting the basil inputs based on previous patterns and the current rate for a personalized touch for the user.

3) A small computer (Intel Edison, or Raspberry Pi for example) and a radio board/stick (i.e. Explorer Board for Edison or Explorer HAT for Pi) : The Explorer HAT sits on top of the Raspberry pi which is used for changing settings for the openAPS software through the display and up and down buttons available on the top as shown in Fig. 2. The user can change the basal inputs from the menu.

4) A battery : A lipo battery with JST connector is required to keep the pi powered up at all times when using the closed loop system.

Fig. 1. openAPS loop.

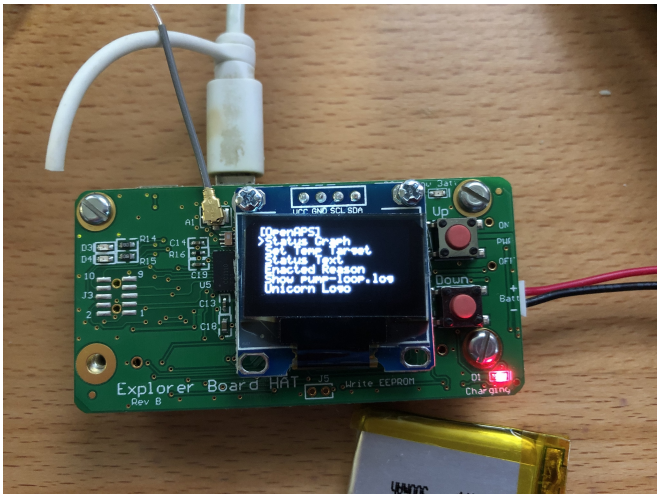

Fig. 2. Explorer HAT

III. RELATED WORK

Below we discuss techniques for performing automated security analysis and their limitations for openAPS.

A. Techniques for building security mechanisms

1) Hardware-based techniques: Hardware based approaches provide security through special hardware modules, such as a Trusted Platform Module (TPM) [12]. The memory and power limitations of embedded systems make the use of TMP-based techniques challenging .

2) Intrusion detection systems (IDS): An intrusion detection system (IDS) is a device or software application that monitors a network or systems for malicious activity or policy violations. Mohan et. al. [18] propose a host-based IDS running on a Hyper-visor, for embedded systems equipped with multicore processors. This IDS runs on a dedicated core, and monitors the controller of the system, which is running on the other cores. However, their approach may only be applied to devices that are equipped with multicore processor and a Timing Trace Module (TTM), a special hardware module for obtaining accurate timing information.

3) Remote attestation: Software verification techniques such as virtualization-based remote attestation suggested by LeMay et. al. [13], Pioneer

[14], and oblivious hashing verify the integrity of software on a third party machine by executing an instance of the program on a remote server.These techniques require the embedded system to be connected to the network at all times and maintain a fast and reliable connection to the server.

B. Techniques for analyzing attacks

1) Attack patterns: Attack patterns capture the common methods for exploiting system vulnerabilities. Each attack pattern encapsulates details about the attack including attack prerequisites, targeted vulnerabilities, attacker goals and resources required. Fernandez et. al. [4] study the steps taken to perform a set of attacks and abstract the steps into attack patterns. They study Denial of Service (DoS) attacks on VoIP networks and show that their patterns can improve the security of the system at design time, and help security investigators trace the attacks. These attacks are similar to signature based intrusion detection techniques.

2) Attack trees: Attack trees are conceptual diagrams showing how an asset, or target, might be attacked.In the field of information technology, they have been used to describe threats on computer systems and possible attacks to realize those threats. Attack trees are top-down hierarchical structures in which lower level activities combine to achieve the higher level goals. The final goal of the attacker is presented at the root. Byres et. al. [6] develop attack trees for power system control networks.

3) Attack graphs: Attack graphs have been mainly used to analyze attacks against networked systems. They take the vulnerability information of each host in a network of hosts, along with the network information, and generate the attack graph. Sheyner et. al. [7] and Jha et. al. [8] propose techniques for automatically generating and analyzing attack graphs for networks. They assume that the vulnerability information for each node is available. Based on this information, they analyze the chains of attacks and their effects on the network.

4) Formal Analysis: Formal techniques have been used to evaluate the security of computer systems using tools such as Z3 [9]. Delaune et. al. analyze the security of PKCS11, an API for cryptographic devices [10]. . Gritzalis et. al. [5] analyze security protocols over open networks and distributed systems. However, hyrid closed loop real-time openAPS (Open-source Artificial Pancreas system) does not have a formal specification. Therefore, designing a formal model for security analysis is challenging.

5) Program Analysis: In recent work, Ivan et. al. [11] find sensor-spoofing attacks against embedded systems via program analysis. They use symbolic execution to find sensor readings that leads to unsafe states in the program. Then, as an attacker, they generate inputs that produce those sensor readings. However, their work is limited to sensor spoofing and does not allow for finding attacks given a general model of the attacker (for instance, when an attacker can tamper with network communications, reboot systems, etc.

6) Artificial Pancreas system security: There has been prior work in the field of Artificial Pancreas system, with respect to formal approaches. Taisa et. al [17] used a data-driven approach to model Artificial Pancreas systems for verification. They model the system using information flow and evaluate the rechability of the Pancreas system for the commercially available products. By rechability, they try to evaluate the bad states that the system can go into based on the inputs inside the predictive algorithm used in Artificial Pancreas system. However, their approach does not take into consideration the security vulnerabilities. Also, nowhere before, the physical access of the hackers is taken into consideration for conducting security analysis.

7) Summary: Existing techniques for analyzing attacks against embedded systems do not provide guarantees for finding all the attacks within a search space. Also, they require a precise model of the attacks and hence, do not consider unknown attacks. For safety-critical systems, we need techniques that do not need a comprehensive and precise database of attacks for analyzing their security.

IV. ANALYSIS

For the analysis of the system we picked openAPS as our test bed. We followed a three step approach in order to analyze the system. We attacked the system based on the actions and capabalities of the attacks. Based on the results, we broke

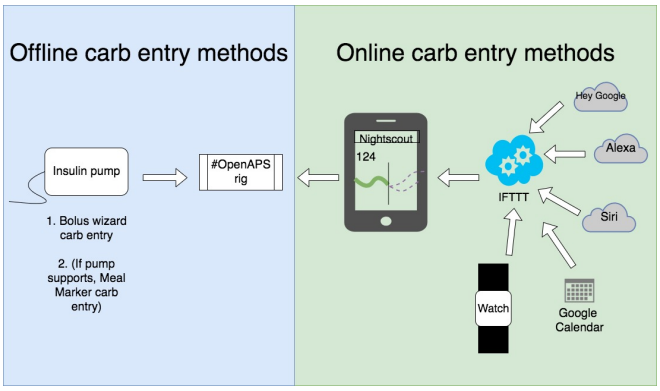

Fig. 3. Carbs entry

down the attacks into more basic attributes causing the attacks. Finally, based on the basic attributes we manually modelled the system in order to automate all the attacks which we tried to perform as attackers.

A. Threat Model

1) Access Model: In this paper, we try to look at every possible way of attacking openAPS. This includes both physical and network attacks against the openAPS system. Since the user of the closed loop artificial pancreas system will be carrying the system around with them unlike the usual systems which are usually not accessible to the attacker. Since openAPS also gives an option of linking the system to the internet to get the data stored in Nightscout as explained above. The attacker has access to the communication interfaces of openAPS (Wifi, LAN and the serial interface between the components).

As we can see in Fig. 3. there is also an offline method of utilizing the loop. Hence the attacker can easily gain access to the offline system if in the vicinity (Bluetooth or other protocols which the offline loop is utilizing).

The attacker also has access to the rig of the entire system. The Explorer Hat as shown in Fig. 2 contains the basic functionality for entering information into the openAPS system such as current basal rates which can be changed manually from the board. It also allows to set the temporary targets, reboot the system, cancelling reboots,

2) Action Model: The online carb entry methods as shown in Fig. 3 open numerous other attack possibilities such as spoofing by using a crafted HTTP Request containing the session id information from another user and hence can result in wrong predictions by the algorithm. When it gets connected to the internet it can becomes vulnerable to attacks such as repudiation, Denial of service, Elevation of privileges[27]. The question here is which of these attacks can eventually cause the most amount of damage. In general, not all these attacks will be considered grave, however, when it comes to a safety-critical system such as openAPS, the attacks can be fatal as the prediction algorithm in openAPS, uses the data from Nightscout to predict the basal inputs for the user. These are realistic assumptions since, for making such changes the attacker does require root access. Based on the access model, we assume that the attacker drop, replay the messages being sent to the openAPS. The openAPS consists of CGM and the insulin pump which exchange the information with the rig which is the Raspberry pi zero for us, the attacker can tamper with the insulin pump and the CGM also along with the Rpi Zero itself. Dropping or replaying the data can lead to excess of insulin getting injected into the system which can be fatal. Since openAPS is a DIY (Do it yourself) system, usually parents set the closed loop for the kids. This means that there are multiple users in a particular household. This can lead to multiple issues, with parents also monitoring the behaviour when not with their kids can lead to some usable security problems. For instance, data tampering or mutations if in case someone else gets hold of the system from which the parent is monitoring.

Since the code is open source and freely available the attackers can use input based attacks based on the information flow. They can reverse engineer the codes and based on observation induce noise in the loop to cause wrong predictions.

B. Approach

After conducting the manual attacks based on the threat model described above, automating the entire process becomes an important part of analyzing a IoT based system. The reason it's a push button security analysis technique is because, instead of following the prior work where the focus is on specific attacks based on invariant [28,29] when we model the entire system and are able to

	Access	Actions
A1	Physical access to the device	1)Tampering with the Explorer HAT by changing the temporary basals. 2)Removing the power-supply of the rig. 3)Observing visible indicators. 4)Tampering the insulin pump. 5)Tampering the CGM.
A2	Physical access to the internal /external communication interfaces	1) Inducing noise in the transmission between CGM and its sensor. 2) Tampering data by inducing noise in the communication between CGM and rig and insulin pump and rig . 3) Tampering data by interfering with the Wifi/Bluetooth/Zigbee system .
A3	Access to the internet	1) Web Attacks such as spoofing 2) Storage system attack on Nightscout .
A4	Access to external materials	1) Material causes damage to the CGM which calculates the BG level using sensors from the skin contact different from the actual BG calculation. 2) Thermal affects such as increasing or decreasing the temperatures of the environment.

Actions and capabilities of the attacker

find any bad state that can exits. It's a unification of all existing invariant. However, modelling such a system can be difficult.

C. Manual Modelling

Based on the actions and the capabilities, instead of actually modelling it based on network or physical attacks, we selected the five features of the system on which all the types of attacks actually depend. The combination of these features eventually leads to a bad state. The five features which we decided to model were : 1) Data 2) Predictive algorithm 3) Hardware 4) Communication and 5) Time. We model the system where the fault in the algorithm can be due to data, communication or hardware. Similarly the fault in data can be due to hardware error or communication. Instead of treating these individually like other approaches unifying them together and considering how change in one parameter can actually lead to some other fault and hence instead of evaluating everything individually the way intrusion detection systems or other approaches do, this modelling can help evaluate any type of IoT system.

V. RESULTS

In this section we present how based on the threat model, we decided to model the system for security analysis. We address the following research questions:

A. RQ1

(Practicality): How applicable are the attacks discovered by this technique on openAPS?

B. RQ2

(Performance): How long does it take for this methodology to discover the attacks?

C. Testbed

Our test bed consisted of a Raspberry pi zero with an Explorer HAT. The Rpi runs Linux which was our rig for the closed loop openAPS. We used a desktop computer in order to gain access to the rig in order to access it.
We simulate the environment as the insulin pumps and the CGMs that openAPS supports are limited and very expensive. The predictive algorithm expects the users to collect 30 days data and upload to Nightscout. The bolus needs to be entered manually by the users. This is before meals for maintaining the Blood group (BG)

levels. OpenAPS gives a lot of options to the users in order to utilize it.

D. Network based attacks

The fact that there are two ways to access the rig (offline and online mode) paves paths for different set of security attacks. In order to simulate the attacks we perform three sets of tests.

1) Computer and rig are on the same wifi network : We were able to tamper the data when we try to access the rig from the computer from a different network by tampering with the data. We were also able to delay the data which was to be predicted. In order to access the rig from the same wifi network we ssh into the rig from our system.

2) Computer and rig are on different wifi network : Similarly in order to access the rig which is on a different network we needed a UART port on the rig with the USB port. We used "sudo screen /dev/tty.usbserial-* 115200 " in order to gain access to the rig. Once again when we have access to the rig there are endless number of possible attacks which can be performed for instance, an attacker can directly change the predictive algorithm such that no matter what data it gets the dose it predicts is fatal.

3) Phone and rig are on the same wifi network : In this case we use an android app openAPS android in order to gain access to the rig granted they are in the same network.

4) Practicality : Based on these three manual attacks which we performed, we ask ourselves if this attack is applicable in real life. In order to gain access to the rig, we need to be root. However, if someone has gained root access to the system then we are already in grave danger. Considering the fairness constraints, we assumed that it is difficult to guess the password. So here we derive that these observations that it is not necessary to model these attacks in our modelling system.

5) Performance : Here the moment some one gains root access, all we can do is to make sure that user is immediately aware about this aspect, and immediately removes the pumps and the sensors from the body before any further damage can be caused. However, the performance of this attack will work well especially if the user is immediately intimated about this.

E. Hardware based attacks

Unlike remote areas or cloud systems, where the system is not accessible, in a system like openAPS, the attacker has the ease to access the hardware. Assuming the threat model we described above, we try to manually make changes to the hardware and observe the results.

1) Rebooting : An attacker may reboot and restart execution at any point by cutting of the power to openAPS. Even if there is backup battery available the attacker can easily disable it. We study the effect of rebooting the system. The rebooting of the system leads to different scenarios, in the first case we observe that the data gets erased. Due to rebooting of the system, the data which was required to be transferred gets delayed. This can lead to dangerous results in case of a safety-critical system such as openAPS.

2) Practicality : This is one of the major contributions to the paper from our aspect since most of the existing techniques fail to take the physical environment into consideration when developing intrusion detection systems.

3) Performance : We can see that once if the system is modelled, there are three ways it can go. Either it runs till the end returns a counterexample depending on the input, or it can give us wrong output which will be false positives and false negatives.

F. Formal Modal

Based on the attacks and the threat model which we proposed, we categorized the attacks to fall into the following categories:

1) Data: Tampering the data in any way possible.

2) Predictive Algorithm : Tampering the algorithm in any way possible be it data, hardware or network based.

3) Hardware access : The hardware damage includes other categories such as damaging the rig which eventually reaches the data loss or tampering of the predictive algorithm.

4) Communication Protocols : Tampering of Bluetooth, wifi, LAN or any other means of communication channel in offline or online mode.

5) Time: Causing delay in prediction, injection, data transfer can be considered a security attack.

According to our analysis most of the security attacks performed in any manner are falling into one of these five categories. All these attacks are dependent on each other and do not mean that if can capture these five characteristic of the system in a formal model, we can unify to find all existing attacks in one push-button model.

VI. DISCUSSION

A. Applicability to other IoT devices

While we have focused on openAPS in this paper, the same ideas can be extended to other classes of embedded devices in IoT systems. The main requirement for doing so is to 1) identify the viable attacker actions specific to that system, and 2) based on the attacker actions, define an abstraction of the system that we can use to build an implementation independent formal model. For example, AUTOSAR (AUTomotive Open System ARchitecture) is an open software architecture that provides the basic infrastructure for developing vehicular software. Similar platforms have been proposed for other classes of embedded systems, e.g., medical devices [30]. Given the open and standardized architecture of these systems, we believe we should be able to take a similar approach for them. Extending this approach to these systems is a direction for future work.

B. Complexity of the attacks

Once we have done manual modelling of the system, its time to automate it. However, the data can be changed in several ways and exploring every state can lead to state space explosion. Hence a certain range of based on the safety standards should be calculated. A certain amount of change in data or tampering in data does not necessarily lead to an unsafe state. Based on the safety standards of the system, the bad state and the good state should be clearly defined. We initially tried modelling the entire system based on the tool, however things became clear that the system needs to be modelled manually initially. After the manual modelling has been achieved, we need to keep making it compact up to a point when it stops exploding.

C. Limitations

There are three limitations to using this methodology to perform an analysis

1) Abstraction level : Which is the correct level of abstraction? The abstraction level that we have proposed in this model is good for our threat model, however it does not take into consideration the code level security vulnerabilities such as buffer-overflow.

2) Mapping the real system to the abstracted system: There is always a chance of getting false positives as the abstracted system is not always properly mapped to the real system. There is high chance that we might not catch any real attacks because of the inconsistencies between the original system and the model.

3) Difficulty Level : It is very difficult to build a formal model for such systems as the current formal verification methodologies are limited. The model checkers provide support for a number of mathematical theories, everything cannot be represented in a mathematical form. Hence, developers prefer using testing than using formal models.

D. Future work

In the future we intend to use Why3[30] to model these attacks and compare the time they take to verify the IoT System. We also intend to use Ontological based approaches to model the system as a schema to define the properties of each category of attack which we built. We intend to compare the formal methods approach with the Artificial Intelligence approach in order to observe which will work better in this context. We intend to use Protege [31] for modelling the system and using the queries in order to know if the inputs will provide us with bad results.

VII. CONCLUSION

IoT devices have gained wide adoption, and their security is an emerging concern. However, existing security techniques do not address their limitations and requirements. In this paper we manually build an analysis model based on the treat model which we propose. Using the threat model we categorize the attacks into five basic categories which are dependent on each other. Instead of considering the final outcome, we consider

the basic entities, the change in whom leads to the attacks. We find these attacks by installing openAPS on a RPi Zero.

REFERENCES

[1] S. Brinkhaus, D. Carluccio, U. Greveler, D B. Justus, and C. Wegener. Smart hacking for privacy. In 28th Chaos Communication Congress, Berlin, Germany, DEC. 2011.

[2] N. Lewson. Smart meter crypto flaw worse than thought, 2010. http://rdist.root.org/2010/01/11/ smart-meter-crypto-flaw-worse-than-thought.

[3] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. Experimental security analysis of a modern automobile. In Proceedings of the 2010 IEEE Symposium on Security and Privacy, SP 10, pages 447462, Washington, DC,

[4] Eduardo Fernandez, Juan Pelaez, and Maria Larrondo-Petrie. 2007. Attack patterns: A new forensic and design tool. In Advances in digital forensics III. Springer, 345357

[5] S. Gritzalis, D. Spinellis, and P. Georgiadis Security protocols over open networks and distributed systems: formal methods for their analysis, design, and verification

[6] Eric J Byres, Matthew Franz, and Darrin Miller. 2004. The use of attack trees in assessing vulnerabilities in SCADA systems. In Proceedings of the International Infrastructure Survivability Workshop. Citeseer.

[7] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M Wing. 2002. Automated generation and analysis of attack graphs. In Security and privacy, 2002. Proceedings. 2002 IEEE Symposium on. IEEE, 273284

[8] Somesh Jha, Oleg Sheyner, and Jeannette Wing. 2002. Two formal analyses of attack graphs. In Computer Security Foundations Workshop, 2002. Proceedings. 15th IEEE. IEEE, 4963.

[9] Leonardo De Moura and Nikolaj Bjrner. 2008. Z3: An efficient SMT solver. In International conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, 337340

[10] Stphanie Delaune, Steve Kremer, and Graham Steel. 2010. Formal security analysis of PKCS 11 and proprietary extensions. Journal of Computer Security

[11] Ivan Pustogarov, Thomas Ristenpart, and Vitaly Shmatikov. Using Program Analysis to Synthesize Sensor Spoofing Attacks. In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. 757770

[12] Dries Schellekens, Brecht Wyseur, and Bart Preneel. Remote attestation on legacy operating systems with trusted platform modules. Sci. Comput. Program., 74:1322, December 2008.

[13] Michael LeMay, George Gross, Carl A. Gunter, and Sanjam Garg. Unified architecture for large-scale attested metering. In Proceedings of HICCS07, Washington, DC, USA, 2007. IEEE Computer Society.

[14] Arvind Seshadri, Mark Luk, Elaine Shi, Adrian Perrig, Leendert van Doorn, and Pradeep Khosla. Pioneer: verifying code integrity and enforcing untampered code execution on legacy systems. In Proceedings of the twentieth ACM symposium on Operating systems principles, SOSP 05, pages 116, New York, NY, USA, 2005. ACM

[15] OpenAPS https://openaps.org/reference-design/

[16] Nightscout http://www.nightscout.info/

[17] Taisa Kushner, David Bortz, David M. Maahs and Sriram Sankaranarayanan. A Data-Driven Approach to Artificial Pancreas Verification and Synthesis. In 2018 9th ACM/IEEE International Conference on Cyber-Physical Systems.

[18] Sibin Mohan, Jaesik Choi, Man-Ki Yoon, Lui Sha, and Jung-Eun Kim. Securecore: A multicore-based intrusion detection architecture for real-time embedded systems. In Proceedings of the 2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS), Washington, DC, USA, 2013. IEEE Computer Society.

[19] Hacking Medical Devices for Fun and Insulin: Breaking the Human. https://media.blackhat.com/bh-us-11/Radclie/BH US 11 Radclie Hacking Medical Devices WP.pdf.

[20] Yared Berhanu, Habtamu Abie and Mohamed Hamdi. A Testbed for Adaptive Security for IoT in eHealth. In 2013 ASPI '13, September 08 2013, Zurich, Switzerland.

[21] Nathanael Paul, Tadayoshi Kohno and David C. Klonof. A Review of the Security of Insulin Pump Infusion Systems. In Journal of Diabetes Science and Technology

[22] Why3. http://why3.lri.fr/

[23] Yuepeng Wang, Isil Dillig. Verifying Equivalence of Database-Driven Applications. In Programming Language Design and Implementation 2018.

[24] Petr Matousek, Jaroslav Rab, Ondrej Rysavy, and Miroslav Sveda. A formal model for network-wide security analysis. In Engineering of Computer Based Systems, 2008. ECBS 2008. 15th Annual IEEE International Conference and Workshop on the, pages 171181. IEEE, 2008.

[25] Marino Miculan and Caterina Urban. Formal analysis of facebook connect single sign-on authentication protocol. In SOFSEM, volume 11, pages 2228. Citeseer, 2011

[26] J.R.Burch, E.M.Clarke, K.L.McMillan, D.L.Dill and L.J.Hwang.Symbolic model checking: 1020 States and beyond. In Information and Computation Volume 98, Issue 2, June 1992, Pages 142-170

[27] Justin Cris. 2007. Web Based Attacks. In SANS Institute InfoSec Reading Room.

[28] Maryam Raiyat Aliabadi, Amita Kamath, Julien Samson, and Karthik Pattabiraman. ARTINALI: Dynamic Invariant Detection for Cyber-Physical System Security. ACM SIGSOFT Symposium on Foundations of Software Engineering (FSE), 2017

[29] Ekta Aggarwal, Mehdi Karimibiuki, Karthik Pattabiraman and Andr Ivanov. CORGIDS: A Correlation-based Generic Intrusion Detection System. 2018. In ACM International Workshop on Cyber-Physical Systems Security and Privacy (CPS-SPC)

[30] Information and technology standards, advanced metering infrastructure, government of ontario canada. http://www.decc.gov.uk/assets/decc/ Consultations/smart-meter-imp-prospectus/ 1478-design-requirements.pdf.