Proceedings

1st International Workshop on Security and Privacy for the Internet-of-Things [IoTSec]

Co-located with the ACM/IEEE International Conference on Internet of Things Design and Implementation (IoTDI), 2018

Orlando, Florida
April 17, 2018

Table of Contents

Message from the chairs

This is the first international workshop on security and privacy for the internet of things and it is co-located with the ACM/IEEE conference on the Internet of Things design and implementation. This edition takes places in Orlando, Florida, USA. The technical program includes thirteen peer reviewed papers and a keynote talk by Prof. David Nicol from the University of Illinois at Urbana-Champaign.

The aim of this workshop is to bring together researchers are practitioners who are working on the growing area of IoT. With the advent of so many IoT-enabled devices and services, it becomes imperative that proper security and privacy solutions are in place. Furthermore, an examination of the legal and ethical issues is also important since many of these devices/services are found in homes, hospitals, public areas, industrial settings, etc. We hope to foster a community that will look at these topics from a variety of perspectives.

Even though this is the first iteration of this workshop, we received a large amount of interest. The final thirteen papers were chosen from 22 initial submissions (which gives us close to 60% acceptance rate). Each paper was received four reviews. We are glad to present this high-quality program to the research community.

IoTSec owes its existence to a variety of people. We would like to thank the organizing committee of IoTDI and in particular Chenyang Lu, Tarek Abdelzaher, Olaf Landseidel, Hui Lei, Lu Su and Iain Bate. We would also like to thank the technical program committee members and the various reviewers who made this program successful and of high quality. Finally, we would like to thank the authors and participants of the workshop without whom this event would not be successful.

We hope that you will enjoy the IoTSec 2018 program and that it will foster many new research directions and collaborations.

Sibin Mohan
University of Illinois at Urbana-Champaign

Elaine Shi
Cornell University

Workshop Organizers and Program Committee

Program Chairs

Sibin Mohan, University of Illinois at Urbana-Champaign
Elaine Shi, Cornell University

Technical Program Committee

Sean Smith, Dartmouth College
Gedare Bloom, Howard University
Gabriela Ciocarlie, SRI International
Miroslav Pajic, Duke University
Bryan Ward, MIT Lincoln Laboratory
Dawn Schrader, Cornell University
Daniel Mosse, University of Pittsburgh
Mu Zhang, Cornell University
Siddharth Garg, NYU
Rakesh Kumar, University of Illinois at Urbana-Champaign
Rakesh Bobba, Oregon State University
Ramya Raghavendra, IBM Research
Man-Ki Yoon, Yale University
Henry Duwe, Iowa State University
Negin Salajageh, Visa Research
Adam Bates, University of Illinois at Urbana-Champaign
Bo Li, University of California, Berkeley
Ing-Ray Chen, Virginia Tech

Web Chair

Monowar Hasan, University of Illinois at Urbana-Champaign

Additional Reviewers

Ioannis Agadakos, Stevens Institute of Technology
Mahsa Saeidi, Oregon State University
Arezoo Rajabi, Oregon State University
Karim Eldefrawy, University of California, Irvine
Henrique Potter, University of Pittsburgh
Akshith Gunasekaran, Oregon State University
Tancrède Lepoint, SRI
Richard Skowyra, Massachusetts Institute of Technology
Akshith Gunasekaran, Oregon State University
Vedanth Narayanan, Oregon State University
Yogita Garud, Oregon State University
Bogdan Copos, University of California, Davis

Keynote: The Role of Modeling and Simulation in IoT Security Research

Abstract

Many of the challenges of security in IOT relate to scale. How do we discover what scaling problems exist? How do we evaluate solutions to the problems we foresee? While measurement of many IoT devices concurrently may be possible, what sense can we make of the measurements when so much of the IoT infrastructure is opaque to us? There is a role for modeling and simulation in assessing security problems and solutions. This talk highlights the opportunities and identifies the associated challenges.

Speaker Biography

David M. Nicol is the Franklin W. Woeltge Professor of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, and Director of the Information Trust Institute (iti.illinois.edu). He is PI for two national centers for infrastructure resilience: the DHS-funded Critical Infrastructure Reliance Institute (ciri.illinois.edu), and the DoE funded Cyber Resilient Energy Delivery Consortium (cred-c.org); he is also PI for the Boeing Trusted Software Center, and co-PI for the NSA-funded Science of Security lablet. Prior to joining UIUC in 2003 he served on the faculties of the computer science departments at Dartmouth College (1996-2003), and before that the College of William and Mary (1987-1996). He has won recognition for excellence in teaching at all three universities. His research interests include trust analysis of networks and software, analytic modeling, and parallelized discrete-event simulation, research which has led to the founding of startup company Network Perception, and election as Fellow of the IEEE and Fellow of the ACM. He is the inaugural recipient of the ACM SIGSIM Outstanding Contributions award, and co-author of the widely used undergraduate textbook “Discrete-Event Systems Simulation”.

Technical Papers

Session I: Attacks and Defenses

1. **An Overview of Vulnerabilities of Voice Controlled Systems**
   Yuan Gong and Christian Poellabauer.

2. **Control-hijacking Vulnerabilities in IoT Firmware: A Brief Survey**
   Abhinav Mohanty, Islam Obaidat, Fadi Yilmaz and Meera Sridhar.

3. **Cognitive Enhancement as an Attack Surface**
   Daniel Sanchez and Bogdan Copos.

4. **SDN-based In-network Honeypot: Preemptively Disrupt and Mislead Attacks in IoT Network**
   Hui Lin.

5. **Hey, You, Keep away from My Device: Remotely Implanting a Virus Expeller to Defeat Mirai on IoT Devices**.
   Chen Cao, Le Guan, Peng Liu, Neng Gao, Jingqiang Lin and Ji Xiang.

Session II: Causality, Redaction and Legal frameworks

6. **Butterfly Effect: Causality from Chaos in the IoT**
   Ioannis Agadakos, Gabriela Ciocarlie, Bogdan Copos, Tancrede Lepoint, Ulf Lindqvist and Michael Locasto.

7. **Toward an Extensible Framework for Redaction**
   Soteris Demetriou, Nathaniel D. Kaufman, Jonah Baim, Adam J. Goldsher and Carl A. Gunter.

8. **Implicit Authentication in Wearables Using Multiple Biometrics**
   Sudip Vhaduri and Christian Poellabauer.

9. **A Multinational Legal Examination of Individual Security Risks Related to the Internet of Things**
   Andrew Weyl and George Williamson.

Session III: IoT Security Architectures

10. **A Comparison of Data Streaming Frameworks for Anomaly Detection in Embedded Systems**
    Giovani Gracioli, Murray Dunne and Sebastian Fischmeister.

11. **A secure IoT architecture for streaming data analysis and anomaly detection**
    Safa Boudabous, Stephan Clémençon, Ons Jelassi and Mariona Caros Roca.

12. **Anomaly-based Intrusion Detection of IoT Device Sensor Data using Provenance Graphs**
    Ebelechukwu Nwafor, Andre Campbell and Gedare Bloom.

13. **SIOTOME: An Edge-ISP Collaborative Architecture for IoT Security**
    Hamed Haddadi, Vassilis Christophides, Renata Cruz Teixeira, Kenjiro Cho, Shigeya Suzuki and Adrian Perrig.

This page intentionally left blank. The technical papers follow on the next page.

Toward an Extensible Framework for Redaction

Soteris Demetriou*, Nathaniel D. Kaufman*, Jonah Baim*, Adam J. Goldsher* and Carl A. Gunter*
*University of Illinois at Urbana-Champaign
{sdemetr2, nkaufma2, baim2, goldshe2, cgunter}@illinois.edu

***Abstract*—**
Data is being created at an increasing rate by sources like the IoT devices, social media, and camera monitors. This data frequently includes sensitive information that parties must redact to adhere to laws and user privacy policies. At the same time, there is steady progress on *recognizers* that find latent information within rich data streams, and thereby create fresh privacy risks. In this work, we advocate the idea of developing a modular, extensible toolkit based on *decognizers* which are information hiding functions derived from recognizers that redact sensitive information. We offer steps towards an abstract conceptual framework and compositional techniques and discuss requirements for such a toolkit.

I. Introduction

With the advent of IoT, digital information is being generated and collected at an unprecedented pace. These rich streams of data entail challenges for dissemination that respects the privacy of individuals. For example, in social media there is a threat of privacy leakage caused by uploaded images because individuals' faces within the image are automatically recognized and shared as the image is disseminated across friends, and friends of friends [1]. Similarly, government data—collected through cameras in police vehicle dashboards (dashcams), and, increasingly, body cameras (bodycams) mounted on the uniforms of police officers—is subjected to public disclosure requests as indicated in the Freedom of Information Act (FOIA).

Controlling access to such rich media streams becomes challenging because they carry incidental information which can be hard to predict *a priori*. As the technology evolves, *latent information* becomes identifiable by an adversary within primitive objects. For example, face recognition technology allows one to infer the presence of targeted people in an otherwise innocuous image or video, while speech recognition advances, allow one to fingerprint a person from an audio stream. A good example of latent information concerns the genome of James Watson, who asked that the value of his ApoE gene be redacted because of its connection to dementia. Subsequently, research in genomics advanced to the point that this information could be inferred from the values of other (exposed) genes [2]. Previous works rely on empirically combining recognition technologies based on current knowledge and expectations. However, some might lead to significant information leakage [3], while others are specific to a media stream [4], [5], [6], [7], [1]. Other works focused on designing frameworks for controlling access to such data by third-party apps [8], [9]. While these might offer practical solutions on their application domains, we observe a lack of theoretical foundations upon which we can confidently built and combine redaction technology in a highly evolving IoT space.

To keep pace with rapid recognition advancements, alternatively one could choose a *close follower* approach: observe when a type of leak happens and take steps to prevent similar future leaks. For example, Google Street View added a licence plate redaction technology to address privacy concerns about identifying vehicles in street view images. In this work, we make a proposal and the first step towards such a *close follower* solution. This will provide the theory and tools to construct redaction functions—which we call decognizers—directly stemming from newly introduced recognizer technology. Towards this end, we develop an abstract conceptual framework to formally describe such transformations from recognizers to decognizers. We further define basic techniques to combine recognizers in ways which allow the construction of correct decognizers (Section II). We envision this to be translated in practice in the form of a modular, extensible, open-sourced toolkit of functions to recognize and redact sensitive data in rich media systems. We illustrate this with a prototype implementation of an extensible redaction toolkit and its application to assorted redaction scenarios (Section III). Finally, we highlight key issues of both the theoretical framework and the toolkit (Section IV).

II. Concept

Let us consider a simple conceptual model of a modular and extensible toolkit for managing the redaction of sensitive information from diverse media types. The key unit of information is a *record*. We denote records with r. Records can be of many types, including documents, images, audio recordings, videos, tables, and so on. To build the conceptual model we view records at two levels. At a *concrete* level they have common representations on computers like PDF or MPEG files; at an *abstract* level they can be represented as a matrix of values V together with a distinguished element $\perp$. We write $V_{\perp}^{mn}$ for the m by n matrix space whose entries are values $v \in V$ or $\perp$. In this case, an abstract document might be a two dimensional array $V_{\perp}^{mn}$ where V is a space of characters, m is the number of lines and n is the width of the text column. An image is a similar array but V is a space of RGB triples. A video might be a three dimensional array consisting of a collection of images. Let us refer to spaces like $V_{\perp}^{mn}$ as *matrix spaces* and denote them with the character M (so that records r are from M). We assume a *concretion* function C that maps an abstract representation (matrix) to a corresponding concrete representation (ASCII document, WAV recording, *etc.*). To

keep things simple let's assume that this can be done so that there is an inverse *abstraction function* A so that $A \circ C$ and $C \circ A$ are identity functions.

Now, we suppose that sensitive parts of a collection of records can be found with a function. A *recognizer* is a function $\Phi : M \to \mathcal{P}$ where $\mathcal{P}$ is the set of sets of indices p in the matrix space M. For example, if M is $V_{\perp}^{mn}$, then an element $P \in \mathcal{P}$ is a set of pairs (i, j) where $1 \leq i \leq m$ and $1 \leq j \leq n$. For example, a function on ASCII characters in a document might be $\Phi : V_{\perp}^{mn} \to \mathcal{P}$ where the output of Φ on an input document is the set of digits considered to be parts of social security numbers (SSNs). The goal is to use this to redact the SSNs by replacing the recognized digits with the distinguished value $\perp$. We use a special term for the corresponding function that replaces recognized matrix values with $\perp$. This is called a *decognizer* and has the type $\Psi : M \to M$. The *simple* decognizer induced by a recognizer Ψ is defined as follows: $\Psi(r)_p$ is equal to $\perp$ if $p \in \Phi(r)$ and it is equal to r_p otherwise. We'll discuss later the idea of (non-simple) decognizers that produce values other than $\perp$; an example is a video redaction scheme that hides faces with blurring or pixelation.

The next steps to building a conceptual toolkit for managing redaction is to develop a library of recognizers and decognizers and with ways to compose and review them. The composition of decognizers needs to be carefully conducted. For example, if Φ is a recognizer for SSNs and Φ' is a recognizer for credit card numbers, one would reasonably expect nice properties, like an assurance that the order in which the induced decognizers are used does not make a difference. To assure the right results we propose a multiary merge function ν that takes a sequence of recognizers as arguments and produces a recognizer that is independent of the order of its arguments. We define $\nu(\Phi, \Phi')(r) = \Phi(r) \cup \Phi'(r)$. Then we define $\Psi = \mu(\Phi, \Phi')$ to be the decognizer induced by $\nu(\Phi, \Phi')$. Similar definitions are used for lists of arguments $\mu(\Phi_1, \ldots, \Phi_n)$.

Putting all of this together, if we have an ASCII file r and want to use recognizers Φ and Φ' to redact SSNs and credit card numbers from it, then the value $r' = C(\mu(\Phi, \Phi')(A(r)))$ has the desired property. In practice it will not be efficient to literally convert concrete into abstract values to apply an abstract recognizer. However, we know that a proposed concrete decognizer Ψ is correct if $\Psi = C \circ \mu(\Phi, \Phi') \circ A$.

III. Illustration

To explore the idea of a modular and extensible toolkit based on our conceptual model, we built a prototype case study, which explores, within a common implementation framework, a collection of recognizers and decognizers for a diversity of media types including audio, text, and video (Figure 1).

Audio. A FOIA request might dictate the disclosure of an audio stream from a conversation between a police officer and a civilian. However, it might be desirable to redact *names* of minors/victims or *phone numbers*. To illustrate this we

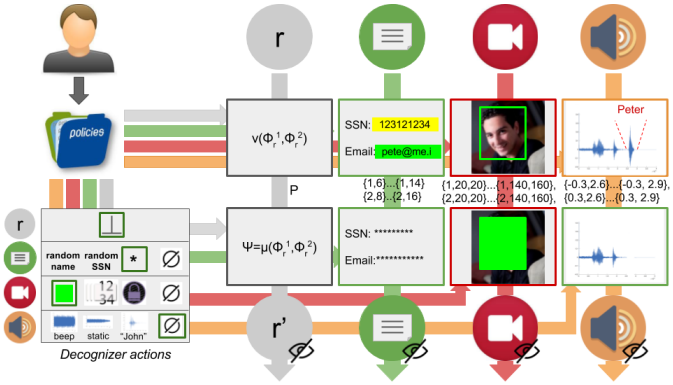


Fig. 1: High-level architecture for a redaction toolkit with simple decognizers applied to an abstract record *r* for document, video and audio streams.

built two respective audio recognizers. We further built a decognizer which, given the original input and the output of the recognizers, replaces the sensitive information with a predefined sound (empty sound or beep sound). The recognizers use Google's speech recognition to transcribe the audio into half second intervals which they process to detect the presence of a target name or phone number using a pre-built library. Information regarding the intervals is fed to the decognizer which replaces the sensitive time intervals of the original audio stream with the pre-defined sound. In particular, our framework composes (merges) the recognizers to assure that the order of their application does not matter.

Text. Another interesting scenario is redaction of sensitive information from text. In fact the majority of FOIA requests currently involve documents, where sensitive information such as email addresses, phone numbers and SSN numbers need to be redacted. To illustrate this scenario, we built three text recognizers: one for email addresses; one for phone numbers; and one for SSN numbers. We further built an induced decognizer which replaces the characters indicated by the recognizers with a special symbol (like an X or *). Note that, if we apply the recognizers serially, this might lead to privacy leakage. Just for purposes of illustration, consider the following scenario: a phone number (10 digits long) contains an SSN number (9 digits long). If we apply the SSN recognizer first which will redact the 9 digits, and then apply the phone recognizer on the result, the latter will fail, resulting in leaking 1 digit of the phone number. According to our analysis (see Section II), the combination of such recognizers needs to provide an assurance that the order of their application is insignificant. Instead, we apply each recognizer on the original input. Then, for each recognizer output, the decognizer is applied to replace the indicated characters, with a special symbol. Finally, we merge the results (either for each line or for each document as a whole) by maximizing the number of special symbols in the final output text. The toolkit further outputs a report, including whether characters where recognized as part of an object, by more than one recognizer. For example, in the case of the SSN number embedded in a phone number, the report will indicate

which characters of the original text were identified as part of both an SSN and a phone number.

Video. Video can be modeled as a sequence of frames. Vision recognizers typically output the pixel coordinates where the objects are detected within a frame. We built three video recognizers: one for faces; one for eye detection; and one for mouth detection. In our implementation we used the opencv library with haar cascades. The recognizers identify the corresponding pixels in the original frame detected as part of a desired sensitive object. We also built an induced decognizer, which given the original input and the output of the combination of the recognizers, it replaces the detected pixels with green pixels (rgb(0,255,0)). The induced decognizer simply maximizes the redaction across recognizers. Thus, it wouldn't matter in which order the recognizers were applied; all pixels corresponding to a recognized mouth, eye or face will be replaced. However, in some cases redaction with simple decognizers can fail (leaking information). For example, if within a frame, part of the face is obstructed by a physical object, then the face recognizer would fail. The toolkit will still redact the mouth and eyes of the person but part of the face will be revealed in some frames. In our implementation, we remember the detected and redacted faces from previous frames. Thus when a mouth or eye is detected within the pixel coordinates of an old face, it automatically redacts the whole historical face region to ensure no information leakage. Video demonstrations can be found on our project's website [10].

While this is preliminary, since more complicated scenarios exist in reality, it demonstrates that while the *commutative* property of the ν function can ensure non information leakage in the simple scenarios we described in Section II, it might not be enough for more complex cases. Therefore, new properties need to be defined to describe more complex recognizer and decognizer functions, which in turn will allow the development of correct redaction application for more interesting scenarios.

IV. Discussion

Complex Recognizers. In Section II we described "simple" abstract decognizers, which replace a matrix value in the original input as indicated by a recognizer, with a distinguished value. This is meaningful and useful in many applications. For example, on a par with our illustration (see Section III), a redaction service could replace all characters belonging to an SSN number with an asterisk (*); in social media, if we were to perform access control on faces rather than whole images we could replace pixels recognized as being in the protected face with black pixels. However, in some cases, the replacement value is not simple. For example, in the audio case, we replaced time intervals with a given sound (empty or beep). Representing audio in the abstract space is more complicated: audio can be seen as a signal in the time domain or the frequency domain. In the former case we could use the root mean squared (RMS) amplitude value per time sample, while, in the former case, we could use the amplitude and frequency values. This would allow us to perceive audio signals as two-dimensional arrays. The recognizers must recognize the time intervals or frequencies belonging to sensitive sound. The replacement value used by the decognizer can be a constant amplitude value to represent the beep sound or empty sound.

However, for some other applications, we want to go beyond the simple decognizer strategy and the issue is not just representation. A well-known example is hiding faces by *blurring* them. Using a simple *box blur* approach, pixels recognized as part of a sensitive face are replaced with the average value of their neighboring pixels in the original matrix. Obviously, our simple decognizer cannot describe this operation. Things become even more challenging with other redaction operations. For example, one might wish to *encrypt* the recognized values [11]. This would allow redacted faces to be replaced later using a key even without access to the original image. Moreover, previous work has shown that by *substituting* words with synonyms or by replacing characters within words, renders automatic identification of the original input more challenging [12]. A number of works also focus on strategies in *de-identifying* health information [13]. In general, the simple decognizer induced by a recognizer can describe useful scenarios, but this is just the tip of the iceberg. We need an extended formalization of the redaction algebra to describe more complicated functions that create decognizers from recognizers. Ideally such an algebra will support proofs of useful properties about potential information leakage.

Recognizer Quality. Assurance against information leaking is directly correlated with recognizers' detection performance. An *ideal* recognizer would be one that never misses a sensitive object and also all the objects that it detects are sensitive objects. However, most of the recognition technology (speech recognition, human/face detectors) are not ideal and use probabilistic models to make estimates. In a toolkit multiple recognizers that perform a similar function could be submitted by different developers and a user could find support for selecting the most appropriate one. We propose three different schemes for recognizer quality evaluation as follows: (a) manual; (b) assisting; (c) crowdsourced.

In the *manual* scheme, the user runs all candidate recognizers and decognizers on the input records and then manually evaluates their accuracy and selects how to apply them. This is feasible in applications or requests where the input size is tractable and the cost of an error is high. On the other hand, it can guarantee the least information leakage since the user explicitly selects the best redactions. Moreover, through this process, the user has the opportunity to manually redact information missed by all candidate recognizers. Thus it makes the scheme appropriate for sensitive applications like the declassification of documents for review by congressional committees. Many FOIA requests might have this standard as well. Note however, that here the amount of work the user needs to do may be significantly reduced since the framework will automatically find and redact many sensitive instances.

In the *assisting* scheme, the toolkit automates more of the previous process to further alleviate the user from labori-

ous manual evaluations. For example, the tool can provide the administrator with a report and/or cues focusing on the differences between the candidate recognizers. This reduces the administrator effort but it might end up revealing more information than intended. For example, all candidate video recognizers might end up missing the same faces in the same frames. The user might never be given the opportunity to catch and rectify this event.

Alternatively, the framework could utilize *crowdsourcing* for recognizer evaluation. For example, crowdsourcing platforms such as Amazon Mechanical Turk, Microworkers or similar platforms, can be utilized to enroll users. These will be queried to manually evaluate the accuracy of recognizers on predefined inputs. Of course, some care is needed to assure that sensitive information is not leaked to the crowdsource platform participants. This might be done by taking data out of context such as identifying fragments of names or pictures; or it could be done by labeling non-sensitive data and using this to train classifiers that are used on the sensitive data.

Toolkit Utility. Currently, an institution interested in redaction would need to either develop recognizer and redaction algorithms from scratch, or perform a wide scale search for suitable technologies. A toolkit could reduce this effort through a global, open-source repository offering a collection of libraries for recognizer and decognizer algorithms.

A redaction toolkit should support *policies*: that is, given a policy described in a suitable formalism, the correct combination of recognizers and decognizers should be chosen to be applied on the input records. Consider for example the following simple FOIA policy: *replace all SSN numbers from all input documents with character* *. In this case, the toolkit will present the user with all candidate text recognizers that detect SSN numbers and their induced decognizers which replace the characters belonging to SSN numbers with a star symbol. A Facebook policy applied when a person is not authorized to see a particular face in an image could be: *replace face having id='123' with rectangle having color='green'*. This would replace all the pixels belonging to the particular face with green pixels.

Community Value. Last but not least, such a toolkit could offer significant value to the community. We envision this to be analogous to the Weka [14] toolkit for data mining. The envisioned toolkit can be the equivalent for redaction where recognizers and decognizers can be integrated and extended by the community. The toolkit can offer support in combining such recognizers by enforcing correctness checks on their input and output arguments. Furthermore, it would be of value to researchers active in information redaction, dataset anonymity, and also to government and private institutions which are either legally bound to perform redactions (FOIA) or offer a relevant service (Google Street View, Facebook etc.).

V. Related Work

There is a body of literature on solutions for controlled disclosure of specific media types such as images [7], [15], [11], [1], text [3], [16] and video streams [6], [5], [4], [17]. Other works focus on designing frameworks were third-party applications gain controlled access to all or parts of various media objects [8], [9]. All prior techniques can be complementary to our approach. Our system does not focus on developing the recognition technology but instead it uses it as a means to continuously update its sensitive information discovery capabilities. Moreover, we abstract away from a specific application domain and make the first step towards developing the theory associated with the description of recognizers and the development of respective decognizers and their compositions for redacting information from media streams.

VI. Conclusion

To keep pace with the evolution of recognition technology we argued the value of a close follower modular extensible redaction toolkit. We made the first step towards the development of a theory which can be leveraged to express correctness properties in the utilization and composition of recognizers and decognizers. To showcase the application of the proposed theory in practice we developed a prototype performing redaction on a variety of media streams for different application scenarios and identified key points for further development.

References

[1] P. Ilia, I. Polakis, E. Athanasopoulos, F. Maggi, and S. Ioannidis, "Face/off: Preventing privacy leakage from photos in social networks," in *CCS*. ACM, 2015.

[2] S. Sadigh-Eteghad, M. Talebi, and M. Farhoudi, "Association of apolipoprotein e epsilon 4 allele with sporadic late onset alzheimers disease," *A meta-analysis. Neurosciences (Riyadh)*, 2012.

[3] D. Lopresti and A. L. Spitz, "Quantifying information leakage in document redaction," in *HDP Workshop*. ACM, 2004.

[4] E. T. Hassan, R. Hasan, P. Shaffer, D. Crandall, and A. Kapadia, "Cartooning for enhanced privacy in lifelogging and streaming videos," *CVPRW*, 2017.

[5] N. Raval, A. Srivastava, A. Razeen, K. Lebeck, A. Machanavajjhala, and L. P. Cox, "What you mark is what apps see," in *MobiSys*. ACM, 2016.

[6] R. Templeman, M. Korayem, D. J. Crandall, and A. Kapadia, "Placeavoider: Steering first-person cameras away from sensitive spaces." in *NDSS*. ISOC, 2014.

[7] S. Jana, A. Narayanan, and V. Shmatikov, "A scanner darkly: Protecting user privacy from perceptual applications," in *S&P*. IEEE, 2013.

[8] S. Jana, D. Molnar, A. Moshchuk, A. M. Dunn, B. Livshits, H. J. Wang, and E. Ofek, "Enabling fine-grained permissions for augmented reality applications with recognizers." in *USENIX Security*, 2013.

[9] F. Roesner, D. Molnar, A. Moshchuk, T. Kohno, and H. J. Wang, "World-driven access control for continuous sensing," in *CCS*. ACM, 2014.

[10] "Project website," https://goo.gl/7HFzPX, 2017.

[11] P. Aditya, R. Sen, P. Druschel, S. J. Oh, R. Benenson, M. Fritz, B. Schiele, B. Bhattacharjee, and T. W. I-pic, "A platform for privacy-compliant image capture," in *MobiSys*. ACM, 2016.

[12] B. Li and Y. Vorobeychik, "Feature cross-substitution in adversarial classification," in *NIPS*, 2014.

[13] B. A. Malin, K. E. Emam, and C. M. O'keefe, "Biomedical data privacy: problems, perspectives, and recent advances," 2013.

[14] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[15] A. Frome, G. Cheung, A. Abdulkader, M. Zennaro, B. Wu, A. Bissacco, H. Adam, H. Neven, and L. Vincent, "Large-scale privacy protection in Google street view," in *ICCV*. IEEE, 2009.

[16] C. M. Cumby and R. Ghani, "A machine learning based system for semi-automatically redacting documents." in *IAAI*, 2011.

[17] M. Korayem, R. Templeman, D. Chen, D. Crandall, and A. Kapadia, "Enhancing lifelogging privacy by detecting screens," in *CHI*. ACM, 2016.

A Multinational Legal Examination of Individual Security Risks Related to the Internet of Things

Andrew Weyl
University of Wollongong, School of Law
Wollogong, Australia
aweyl@uow.edu.au
Department of Legal Studies
Hodges University
Naples, U.S.A.
aweyl@hodges.edu

Businesses, hospitals, governments and other organizations endlessly collect and store enormous amounts of personal data every day. Individuals rarely consider how their personal data is collected, stored or secured. In fact, few of us realize how our daily activities create a significant amount of personal data. The frequency by which we create personal data is increasing exponentially as we rely more heavily on Internet of Things (IoT) devices connected to the internet to provide numerous benefits. This discussion will examine the legal consequences of collecting personal data gathered from IoT devices from a multinational perspective.

I. PERSONAL DATA

Businesses, hospitals, governments and other organizations endlessly collect and store enormous amounts of personal data every day. Individuals rarely think about how their personal data is collected, stored or secured. In fact, few of us realize how our daily activities create a significant amount of personal data. The amount of data we create is increasing exponentially resulting from the Internet of Things (IoT) where devices connected to the internet provide tack our actions and activities. This discussion will examine the legal consequences of collecting personal data gathered from IoT devices from a multinational perspective.

II. INTERNET OF THINGS

It is not uncommon for someone to set their alarm, program the coffee maker, warm/cool the house, go for a morning run, drive to work, sit at the computer, access a few phone apps throughout the day, head to the gym, grab a dinner out, return home to watch television and head to bed completely unaware that every action has been tracked and stored. Welcome to the Internet of Things, a world where billions of internet connected devices provide convenience in our everyday lives. Every connected device is collecting data about us, our location, habits and purchases. The conveniences provided create a temptation to ignore the security risks.

Internet of things typically means the interconnection via the internet of computing devices embedded in everyday objects, enabling them to send and receive data.[1] It is the concept of connecting any device with the capacity to download information to the internet and possibly to other devices. Included are such things as cellphones, coffee makers, appliances, headphones, lighting systems, wearable devices, home locks, thermostats and any other internet connected device.[2] This also applies to workstations, copy machines, employee badges, business and mechanical components such as the jet engine of an airplane *(Malaysian Airline flight ML370 sent out a signal shortly before disappearing, providing an arc of possible crash locations).*

A. Consequences

What happens to all the data collected regarding our habits, purchases, and location? Recently Strava, a company which collects data from fitness apps and watches, released a worldwide map showing running courses collected from every user of its services. The release of this information was intended to provide opportunities for runners to explore routes they might not have known existed. Rather than planning and mapping out a new course, the release of all data accumulated from runners throughout the world would provide runners the opportunity to use a course someone else had already established. Strava's action to release this data resulted in the unforeseen consequence of revealing the location of secret U.S. military bases in the Middle East.[3] Military personnel were using their GPS smart watches and apps to monitor how far and fast they had run. This running information was uploaded to the internet and collected by Strava. When Strava released the data publically, though well intended, it revealed every customer's past run, not realizing the legal consequences.

Some auto insurance companies also collect customer driving data by installing telematics devices into the insured car to

[1] Jacob Morgan, A simple explanation of the 'Internet of Things'. Forbes, May 13, 2014
[2] Internet of Things, FTC Staff Report. January 2015
[3] Aja Romano, How a fitness app revealed military secrets and the new reality of data collection. Vox, Feb. 1, 2018

provide data regarding driving habits.[4] Good drivers are rewarded with lower premiums with bad drivers paying higher premiums. Several car rental companies have also installed telematics to insure rented cars stay within pre-designated regions and to insure drivers obey posted laws.[5] Automotive manufactures have been collecting data on automobile use for a number of years.[6] Benefits to auto manufactures include the ability to monitor engine function, unlock vehicles, call emergency services, and provide the location of stolen cars. Disadvantages include voiding warranties in the event of excessive use, monitoring individual driving habits, and always knowing where vehicles (and presumptively the owners) are located.[7] Would the outcome of OJ Simpson's criminal trial have been different if Ford had GPS location tracking for the infamous white Bronco?

In an effort to insure employees are working at their productive peak, businesses have experimented with devices that track location (chips in ID badges), productivity (computer use), hygiene (hand sanitizers that connect to employee badges), and even voice tone (call center headphones with Bluetooth voice data monitors).[8]

It is believed that by 2020 there will be over 26 billion connected devices.[9] With billions of devices interconnected with each other and to the internet what can people do to make sure that their information stays secure? Will someone be able to hack into your toaster and thereby get access to your entire network? The IoT opens companies all over the world to more security threats. The more information collected, the greater the potential for data breaches, hacks, and unintentional releases. Also of importance is the issue of privacy and data sharing. Governments have an obligation to protect its citizens from potential harm through effective education regarding potential risks, and through effective legislation regarding security measures.

B. Permission

The first step in providing security for IoT devices is for companies to seek permission to collect, store, use the data generated. Consumers need to be aware when they grant permission for the collection of data they oftentimes lose the right to that data, regardless of how private. When devices ask for permission to access our data many of us obediently provide the requested access without question. Does a fitness tracker need access to your contacts? Does an app that will remotely start your car need access to your call data? Does a smart home lock need access to your phone storage? It is important for consumers to realize they are often times blindly giving access to large amounts of data to companies who have little or no safeguards in place regarding your personal information.

While large tech companies such as Apple and Google provide reasonable protections for data and provide consumers the opportunity to increase their security settings, some companies make no promises to provide any security for your information collected through IoT. Further, personal data has economic value which can, and is, sold to other organizations for profit.[10] Once third party businesses have purchased your data they are not bound to disclosure agreements and are extremely difficult to monitor for security purposes.

When access to data is freely given and that data falls into the hands of a third party there is no longer any privity of contract to protect the original user. Privity of contract is a legal term whereby a party who is not a party to the agreement is not bound to the terms of the agreement.[11] If, for example, Jane Doe feels comfortable giving XYZ Company access to the data on her phone or device XYZ has a legal obligation to honor the terms of any agreement between Doe and XYZ. However, the terms of the agreement often give ownership to XYZ any data collected from Doe's activities. XYZ has a legal obligation to protect this data. XYZ may have a legal right to sell this data to ABC for profit. The obligations imposed upon ABC to protect this data are far less certain. Further, Doe, who willingly provided permission for XYZ to access her data, has no breach of contract claim against ABC if the data is disclosed. The legal implications of collecting, safeguarding, selling, and storing data depend on the applicable jurisdiction. Individual countries provide different protections for the safety of personal data therefore it is important to understand how these rights protect parties.

III. MULTINATIONAL LEGAL AND LEGISLATIVE APPROACHES

When determining how private data collected from devices used in the IoT it is essential to look at legislation passed in the applicable jurisdiction. Various nations deal with data privacy, collection and storage differently. Both the European Union and Australia have recently passed legislation that will take effect in 2018 to address data privacy issues. While the United States has noted the importance of data privacy in the age of IoT, no federal legislation has been adopted.

A. European Data Protection Laws

The European Union has a unified data protection law called the Data Protection Directive. The EU's Data Protection

[4] Cherise Threewit, How do those car insurance tracking devices work? U.S.News & World Report, Oct. 24, 2016

[5] https://www.positionlogic.com/industries-gps-tracking-solutions/gps-tracking-solution-rental-car-services

[6] Rob Stumpf, Car Manufacturers have an alarming ability to farm and sell driver data. The Drive Nov. 23, 2017

[7] Time Cushing, Cars are delivering tons of driving data to manufacturers with minimal security and even less transparency. Tech Dirt, Feb. 17, 2015

[8] Scott Preppat, Regulating the Internet of Things: First steps towards managing discrimination, privacy, security and consent. 93 Tex. Law Rev. 85 (2014)

[9] https://www.gartner.com/newsroom/id/2636073

[10] https://www.economist.com/news/briefing/21721634-how-it-shaping-up-data-giving-rise-new-economy

[11] Oxford Law Dictionary, 3rd Edition, 2017

Directive regulates the processing of personal data within the European Union and is an important component of the EU's privacy and human rights law. However, recognizing the need to modify this law to address globalization and technological developments, the European Union prepared an updated European General Data Protection Regulation which becomes effective mid-2018.[12] The Data Protection Directive asserts that personal information should not be processed at all, but if it is, it must fall within certain categories of transparency, legitimate purpose, and proportionality (fairness, justice, constitutionality).[13] The new law will also expand the data protection regime currently in place to cover all international companies doing business in the EU.[14]

A recent EU decision directly explored whether the definition of "personal information" in European law included dynamic IP addresses that could only be identified when linked with data held by a third party (in this case an ISP).[15] The dispute in that case concerned storage by the German government of the IP addresses of devices that visited government websites. The court found that even though a dynamic IP address is not itself personal information, it can become personal information when linked with other data.[16] Under the new European Data Protection directive, privacy standards have now clarified that no untargeted, indiscriminate collection of data is permissible, even if it is for the purposes of protecting national security or investigating serious crime.[17]

The advantage of the new regulation is that it provides recent, comprehensive regulation of data privacy into a single source. Businesses are not required to determine the rules from 28 EU member states regarding legislation pertaining to safeguards of personal data. But the European Data Protection regulation has its shortcomings. It is said the European Commission did not have the vision to propose a truly modern privacy law.[18] It is, in essence, the old privacy law one level more detailed and complicated. In addition, the search for compromises between the 28 member states and the European Parliament has left the new law in ruins.[19] After almost four years of negotiations, the stakeholders preferred to finish the process instead of taking the time to properly finalize it.[20] When other countries notice the frustration the new law will most likely cause in the EU, they may disqualify it as a potential global standard.

[12] Official Journal of the European Communities, Directive 95/46/EL Amended; 2016
[13] Nigel Hawthorn, 10 Things you need to know about the EU Data Protection Regulation. Computer World UK, May 6, 2015
[14] https://www.hg.org/data-protection.html
[15] Patrick Bryer v Bundesrepublik Deutschland, Court of Justice of the EU, Oct. 19, 2016
[16] Id.
[17] Id.
[18] Ulrich Wuermeling, The elusive quest for global privacy standards. US News, Mar. 22, 2016
[19] Same as above.
[20] Same as above.

The EU Data Protection Regulation three prong test would likely find Strava did not breach individual data security. Strava likely would succeed regarding the legitimate purpose of the data release and the proportionality. Releasing locations of individual running courses was for the legitimate purpose of enabling other runners to access the same courses. Further, no issues of fairness and justice arise from revealing where its users choose to run. Transparency of the data released would ultimately be Strava's most challenging hurdle, but the company could legitimately argue no personal data had been released. Simply revealing where someone ran does not breach transparency requirements.

B. Australia

On 13 February 2017, the Australian Senate passed the Privacy Amendment (Notifiable Data Breaches) to be enacted into law from 22 February 2018. The Act requires an obligation of notification for data breaches.[21] Despite having updated its privacy laws in March 2014, Australia had been lagging behind other countries in relation to data breach notification obligations.[22]

The new law will complement Australia's existing privacy laws which are set out in the Privacy Act 1988 (Cth) (Privacy Act). The Privacy Act lists various principles, 13 of which apply to organizations and provides compulsory guidelines on how those organisations collect, store, manage and disclose personal information in Australia.[23] Personal information is defined as: "information or an opinion, whether true or not, and whether recorded in a material form or not, about an identified individual, or an individual who is reasonably identifiable."[24] Common examples are an individual's name, signature, address, telephone number, date of birth, medical records, bank account details and commentary or opinion about a person. The law is also intended to cover data breach events.

In possibly Australia's most important privacy case to date, the Federal Court recently dealt a severe blow to Australia's information privacy laws by narrowing the definition of "personal information" holding that Australia's data privacy laws only protect "personal information", which is defined by whether a person is identified or identifiable from the data.[25] Why is this significant? Let's return to the Strava example; data regarding running paths collected and then released to the internet did not list individual users and avoided personal information linked to individual accounts. Therefore, in Australia this information is not protected personal

[21] Dudley Kneller, Australia: New Mandatory Privacy and Breach Notification Laws. Mondaq, May 22, 2017
[22] Sergio Ferreira, 100 Days before privacy law takes effect. Computer World, Nov. 15, 2017
[23] Melinda L McLelland and Emily R Fedeles, Australia's new breach notification law to take effect February 2018. Mondaq, Mar. 16, 2017
[24] Sergio Ferreira, 100 Days before privacy law takes effect. Computer World, Nov. 15, 2017
[25] Jake Goldenfein, Australia's privacy laws gutted in court ruling on what's 'Personal Information'. The Conversation, Jan. 19, 2017

information. However, it is not exceptionally difficult to link general data to private users. Strava releasing running paths did not reveal individual information or military base locations, and yet the connections were made.

Individual privacy protections need to be enhanced in light of recent litigation in Australia and it will be up to the courts to interpret the level of protections granted in the new Privacy Amendment soon to take effect.

C. United States of America

A number of states in the United States have had data breach notification laws in place for some time. California, upon which many States have based their own laws, has had legislation in place since 2003. Currently 48 U.S. states, the District of Columbia, Guam, Puerto Rico and the Virgin Islands have enacted legislation requiring private or governmental agencies to notify individuals of security breaches of information involving personal information. [26] However, the United States does not have any centralized, formal legislation at the federal level regarding this issue, instead there are numerous pieces of legislation which address a particular privacy issues in a specific context such as the United States Privacy Act, the Safe Harbor Act and the Health Insurance Portability and Accountability Act. Although partial regulations exist, there is no all-encompassing law regulating the acquisition, storage, or use of personal data in the U.S.[27] In general terms whoever can be troubled to key in the data, is deemed to own the right to store and use it, even if the data was collected without permission, except to any extent regulated by law.

1) Federal case law

When data is collected the security of that data is of high importance to consumers especially once a breach has occurred. Additionally, what happens when data is collected without permission? Customers of Pfizer sued when their private data was exposed. In the legal proceedings Pfizer argued that the "various flavors of damages" alleged in the complaint were inherently speculative and not recoverable under Louisiana law, which required that damages be established to a "legal certainty" in order to prevail at trial.[28] The court relied on previous cases which held actual damages cannot be established by remote and conjectural estimates of loss. [29] Seeing the inherent problem with requiring actual and not speculative damages the court stated, "Given the rate at which internet technologies evolve, the ability of computer hackers to stay two steps ahead of the latest security measures, and the comparatively slow speed at which the law responds to cyber threats courts are often *ill equipped* to address issues."[30]

Another high profile case, In re Facebook Internet Tracking Litig., found that the Plaintiffs did not establish a "realistic economic harm or loss that is attributable to Facebook's alleged conduct."[31] Facebook was accused of tracking users' internet activity even after they logged out of the social media website. The Plaintiffs brought action in federal court claiming Facebook breached the terms of their contract with customers and breached their duty of good faith and fair dealing. The federal court held a plaintiff must show that the defendant (1) intentionally accesses without authorization a facility through which an electronic communication service is provided; or (2) intentionally exceeds an authorization to access that facility.[32]

While it is argued Strava may have revealed the location of secret military bases, it did not expose individual private data and would likely prevail in a lawsuit filed in US courts. Not only have US courts been reluctant to find in favor of plaintiffs, they tend to find satisfaction general authorizations of collection and use of the data suffice to meet existing legal requirements.

IV. CONCLUSION

In every case where a court has determined that the essence of a claim was actually not unauthorized access, but rather unauthorized disclosure or use of the obtained information, the violator has been found to be not liable.[33]

The IoT provides consumers with unprecedented opportunities but with opportunity comes consequence. The meteoric rise of IoT devices creates new issues regarding privacy. Personal data is collected in ways many of us do not even imagine. Much of the data generated is necessary for the use of IoT devices and provides the benefits inherent in their nature. However, consumers must be aware they are being monitored and choose the level of privacy they wish to enjoy. Governments need to pass effective stringent legislation regarding individual privacy and the methods used to collect, store, and disseminate private information. It is not enough to require companies to have a plan regarding data storage; organizations must actively engage in safeguarding private data. Ultimately it is up to the consumer to insure individual personal data remains safe by restricting authorization of its use.

[26] https://www.dlapiperdataprotection.com/?t=law&c=US
[27] https://www.hg.org/data-protection.html
[28] Ponder v. Pfizer, Inc., 522 F. Supp. 2d 793, 796–97 (M.D. La. 2007)
[29] Id.
[30] Id. At 797–98
[31] In re Facebook Internet Tracking Litig., 263 F. Supp. 3d 836, 841–42 (N.D. Cal. 2017)
[32] Id.
[33] In re Google Inc., No. 13-MD-02430-LHK, 2013 WL 5423918, at *2 (N.D. Cal. Sept. 26, 2013)

An Overview of Vulnerabilities of Voice Controlled Systems

Yuan Gong
Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556
Email: ygong1@nd.edu

Christian Poellabauer
Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556
Email: cpoellab@nd.edu

***Abstract*—Over the last few years, a rapidly increasing number of Internet-of-Things (IoT) systems that adopt voice as the primary user input have emerged. These systems have been shown to be vulnerable to various types of voice spoofing attacks. However, how exactly these techniques differ or relate to each other has not been extensively studied. In this paper, we provide a survey of recent attack and defense techniques for voice controlled systems and propose a classification of these techniques. We also discuss the need for a universal defense strategy that protects a system from various types of attacks.**

I. Introduction

An increasing number of IoT systems rely on voice input as the primary user-machine interface. For example, voice-controlled devices such as Amazon Echo, Google Home, Apple HomePod, and Xiaomi AI allow users to control their smart home appliances, adjust thermostats, activate home security systems, purchase items online, initiate phone calls, and complete many other tasks with ease. In addition, most smartphones are also equipped with smart voice assistants such as Siri, Google Now, and Cortana, which provide a convenient and natural user interface to control smartphone functionality or IoT devices. Voice-driven user interfaces allow hands-free and eyes-free operation where users can interact with a system while focusing their attention elsewhere. A block diagram of a typical voice controlled system (VCS) is shown in Figure 1.

Despite their convenience, VCSs also raise new security concerns. One such concern is their vulnerability to a voice replay attack [1], i.e., an attacker can replay a previously recorded voice to make the IoT system perform a specific malicious action. Such malicious actions include the opening and unlocking of doors, making unauthorized purchases, controlling sensitive home appliances (e.g., security cameras and thermostats), and transmitting sensitive information. While a simple voice replay attack is relatively easy to detect by a user, and therefore presents only a limited threat, recent studies have pointed out more concerning and effective types of attacks, including self-triggered attacks [2], [3], inaudible attacks [4], [5], and human-imperceptible attacks [6], [7], [8]. These attacks are very different from each other in terms of their implementation, which requires different domain knowledge in areas such as the operating system, signal processing, and machine learning. However, how exactly they differ or relate to each other has not been extensively studied. As a consequence, to the best of our knowledge, current defense techniques only aim to defend attacks of one specific category, with the assumption that the defender knows the details of the attacking technology. From a security standpoint, this is deeply unsatisfactory. Therefore, in this paper, we provide a survey of recent attack and defense techniques for voice controlled systems, discuss their relationships, and propose a classification of these techniques. We further discuss a potential universal defense strategy for different types of attacks. We expect that the analysis and discussion in this paper will provide useful insights for future studies and efforts in building secure voice-driven IoT systems.

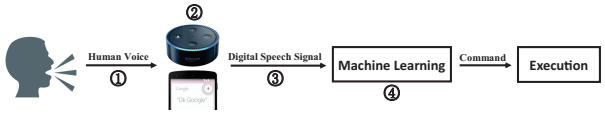


Fig. 1. An illustration of a typical voice controlled system. The device captures the human voice, converts it into a digital speech signal, and feeds it into a machine learning model. The corresponding command is then executed by the connected IoT devices. Potential points of attack in this scenario include: ①: spoofing the system using previously recorded audio, ②: hacking into the operating system to force the voice-driven software to accept commands erroneously, ③: emitting carefully designed illegitimate analog signals that will be converted into legitimate digital speech signals by the hardware, and ④: using carefully crafted speech adversarial examples to fool the machine learning model.

II. Voice-Based Attacks

With the rapidly growing popularity and functionality of voice-driven IoT devices, the potential of voice-based attacks becomes a non-negligible security risk. As discussed in [9], [2], [10], an attack may lead to severe losses, e.g., a burglar could enter a house by tricking a voice-based smart lock or an attacker could make unauthorized purchases and credit card charges using a voice-based system. Such attacks can be very simple and often difficult or even impossible to detect by humans and voice attacks can be hidden by other sounds or embedded into audio and video recordings. Further, it is also very easy to scale up such attacks, e.g., a hidden malicious audio sample in a YouTube video could simultaneously target millions of devices.

Although the implementations of existing attack techniques may be very different, their goals are the same: generating a signal that leads a voice controlled system to execute a specific malicious command that the user cannot detect or recognize.

In the following sections, we first introduce representative state-of-the-art attack approaches according to the type of implementation. We then further discuss the positives and negatives of each approach and how they relate to each other. The attacker performance discussed in this section is evaluated and reported by the original publication. Due to rapid changes of cloud-based systems, the attacker performance is also likely to change over time.

A. Attack Classification Based On Implementation

1) Basic Voice Replay Attack: It is widely known that voice controlled systems are vulnerable to voice replay attacks, i.e., an attacker can replay a previously recorded voice to make a system perform a specific action [1], [13], e.g., as demonstrated previously with the popular Amazon Alexa technology [10]. A shortcoming of the basic voice replay attack is that it is easy to detect and therefore has a limited practical impact. Nevertheless, as shown later in this section, voice replay attacks are the basis of other more advanced and dangerous attacks.

2) Operating System Level Attack: Compared to basic voice replay attacks, an operating system (OS) level attack exploits vulnerabilities of the OS to make the attack self-triggered and more imperceptible. Representative examples of this are the A11y attack [3], GVS-Attack [2], and the approach presented in [9]. In [3], the authors propose a malware that collects a user's voice and then performs a self-replay attack as a background service. In [2], the authors further verify that the built-in microphone and speaker can be used simultaneously and that the use of the speaker does not require user permission on Android devices. They take advantage of this and propose a zero-permission malware, which continuously analyzes the environment and conducts the attack once it finds that no user is nearby. The attack uses the device's built-in speaker to replay a recorded or synthetic speech, which is then accepted as a legitimate command. This self-triggered attack is thus more dangerous and practical. While this attack can still be detected by the user, the authors point out that if the malware has high permissions, it is even possible for it to import an audio file to the microphone without playing it, which can make the attack completely inaudible. In [9], the authors analyze the permission vulnerability to the voice attack in detail and propose an approach to bypass the permission management of the Android system. The authors also find that some malicious actions require a multiple-step command and further propose an interactive attack that can execute more advanced commands.

3) Hardware Level Attack: A hardware level attack replays a synthetic non-speech analog signal instead of human voice. The analog signal is carefully designed according to the characteristics of the hardware (e.g., the analog-digital converter). The signal is inaudible, but can be converted into a legitimate digital speech signal by the hardware. Representative approaches are the Dolphin attack [4] and the IEMI attack [5]. In [4], the authors utilize the non-linearity of a Micro Electro Mechanical Systems (MEMS) microphone over ultrasounds and successfully generate inaudible ultrasound signals that can be accepted as legitimate target commands. Generating such ultrasound signals requires a special device that includes a controller (e.g., another smartphone), an amplifier, and an ultrasonic transducer. The longest attack distance is 175cm. In [5], the authors take advantage of the fact that a wired microphone-capable headphone can be used as a microphone and an FM antenna simultaneously and demonstrate that it is possible to trigger voice commands remotely by emitting a carefully designed inaudible AM-modulated signal. This attack is only effective when the wired headphone is plugged into the device. A limitation of hardware level attacks is that generating the attack signal requires special devices, and also some preconditions must be met (e.g., the victim device needs to be in the attack range and the microphone needs to be plugged in). While the synthetic signal is inaudible to the user, the signal generator might still be noticed by the user.

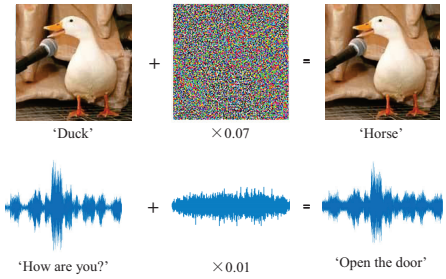


Fig. 2. An illustration of machine learning adversarial examples. Studies have shown that by adding an imperceptibly small, but carefully designed perturbation, an attack can successfully lead the machine learning model to making a wrong prediction. Such attacks have been used in computer vision (upper graphs) [14] and speech recognition (lower graphs) [12], [7], [8].

4) Machine Learning Level Attack: State-of-the-art voice controlled systems are usually equipped with an automatic speech recognition (ASR) algorithm to convert digital speech signal to text. Deep neural network (DNN) based algorithms such as DeepSpeech [15] can achieve excellent performance with around 95% word recognition rate and hence dominate the field. However, recent studies show that machine learning models, especially DNN based models, are vulnerable to attacks by adversarial examples [14]. That is, machine learning models might mis-classify perturbed examples that are only slightly different from correctly classified examples (illustrated in Figure 2). In speech, adversarial samples can sound like normal speech, but will actually be recognized as a completely different malicious command by the machine, e.g., an audio file might sound like "hello", but will be recognized as "open the door" by the ASR system. In recent years, several examples of such attacks have been studied [6], [7], [8], [11], [12].

Cocaine Noodles [11] and Hidden Voice Command [6] are the first efforts to utilize the differences in the way humans and computers recognize speech and to successfully generate adversarial sound examples that are intelligible as a specific command to ASR systems (Google Now and CMU Sphinx), but are not easily understandable by humans. The authors observe that ASR systems rely on acoustic features

TABLE I
REPRESENTATIVE VOICE ATTACK TECHNIQUES

Attack Name	Attack Type	Adversary's Knowledge	Implementation
GVS Attack [2]	Operating System	White box	Continuously analyze the environment and conduct voice replay attack using built-in microphone when opportunities arise.
A11y Attack [3]	Operating System	White box	Collect the voice of a user and conduct self-replay attack as a background service.
Monkey Attack [9]	Operating System	White box	Bypass authority management of the OS and conduct interactive voice replay attack to execute more advanced commands.
Dolphin Attack [4]	Hardware	White box	Emit ultrasound signal that can be converted into a legitimate speech digital signal by the MEMS microphone.
IEMI Attack [5]	Hardware	White box	Emit AM-modulated signal that can be converted into a legitimate speech digital signal by the wired microphone-capable headphone.
Cocaine Noodles [11]	Machine Learning	Black box	Similar to the hidden voice command.
Hidden Voice Command [6]	Machine Learning	Black & White box	Mangle malicious voice commands so that it retains enough acoustic features for the ASR system, but becomes unintelligible to humans.
Houdini [12]	Machine Learning	Black & White box	Produce sound that is almost no different to normal speech, but fails to be recognized by both known or unknown ASR systems.
Speech Adversarial Example [7]	Machine Learning	White box	Produce sound that is over 98% similar to any given speech, but makes the DNN model fail to recognize the gender, identity, and emotion.
Targeted Speech Adversarial Example [8]	Machine Learning	White box	Produce sound that is over 99.9% similar to any given speech, but transcribes as any desired malicious command by the ASR.

(e.g., Mel-frequency cepstral coefficients or MFCC) extracted from the audio. Therefore, their attack mangles a malicious voice command signal in such a way that it retains enough acoustic features for the ASR system to accept it while making it difficult for humans to understand. The limitation of the approach in [11], [6] is that the generated audio does not sound like legitimate speech. As a matter of fact, strictly speaking, Cocaine Noodles and Hidden Voice Commands are not really machine learning adversarial examples, because they use sounds that are not similar to legitimate speech [16]. A user might notice that the malicious sound is an abnormal condition and may take counteractions. Further, generating such adversary examples requires a subjective human test step to ensure that it is imperceptible by humans, which makes the approach not fully automated.

In contrast to [11], [6], more recent efforts [12], [7], [8] take advantage of an intriguing property of DNN by generating malicious audio that sounds almost completely like normal speech by adopting a mathematical optimization method. The goal of these techniques is to design a minor perturbation in the speech signal that can fool an ASR system. In [8], the authors propose a method that can produce an audio waveform that is less than 0.1% different from a given audio waveform, but will be transcribed as any desired text by DeepSpeech [15]. In [7], the authors demonstrate that a 2% designed distortion of speech can make state-of-the-art DNN models fail to recognize the gender and identity of the speaker. In [12], the authors show that such attacks are transferable to different and unknown ASR models. Such attacks are dangerous, because users do not expect that normal speech samples, such as "hello", could be translated into a malicious command by an IoT device. Despite their extraordinary performance, these methods have a clear limitation: since the perturbation is very minor, it might not be correctly captured over the air by the victim device and hence fail to attack. In [8], the authors report that their method does not have an over-the-air threat, while in [12], [7], the authors do not report the over-the-air attack performance.

B. The Adversary's Knowledge

One important factor of attacks is the adversary's knowledge. *White-box attacks* assume that the adversary knows all details (e.g., design details and characteristics) of the target, while *black-box attacks* assume that the adversary does not have such information. Table I summarizes several attack schemes discussed in this paper, including their type and the required adversary's knowledge. Hardware level attacks are usually white-box attacks, because the device can be easily dis-assembled and tested, e.g., in [4], the authors first test the frequency response of the MEMS microphone and then take advantage of its nonlinearity to conduct the attack. All OS level attacks are white-box attacks. Note that all discussed schemes [2], [9], [3] are targeting Android devices, which is not surprising since Android is open-source and its inner workings (e.g., authority management, inter-process communication) are well known. In contrast, the OS of Amazon Echo is closed, which prevents it from being attacked. In fact, it is difficult to perform black-box hardware and OS level attacks.

Different from the OS and hardware level attacks, practical machine learning level attacks are usually black-box attacks, because state-of-the-art ASR systems for IoT devices do not release their detailed algorithms and the training sets. These ASR systems run in the cloud, where an adversary may not have access. However, machine learning attacks are able to attack unknown ASR systems, e.g., in [11], [6], [12], the authors successfully attack Google Voice without knowing its details. This is because ASRs use similar (explicit or implicit) acoustic features and models (e.g., network architectures), which makes the machine learning adversarial examples universal. This characteristic makes machine learning level attacks more dangerous.

Another noteworthy point is that attacks can be combined to become even more dangerous, e.g., GVS-Attacks [2] and the approach described in [8] can be combined so that the malware replay machine learning adversarial example sounds like normal speech instead of a malicious command when it finds an opportunity. Further, all attacks can be combined with the one in [9] to become an interactive attack.

III. Defense Strategies

To defend and prevent OS level attacks, voice input and output need to be decoupled since simultaneously using voice input and output has been shown to affect users' security [9], [2], [3]. For example, AuDroid [13] has been proposed to manage the audio channel authority. By using different security levels for different audio channel usage patterns, AuDroid can resist a voice attack using the device's built-in speaker [9], [2]. However, AuDroid uses a speaker verification system to defend external replay attacks, including hardware and machine learning level attacks, which is not effective enough. Therefore, AuDroid is only robust to OS level attacks.

One defense strategy seems promising for hardware and machine learning level attacks is *adversarial training*, i.e., training an extra machine learning model that can classify legitimate samples and adversaries. In [4], the authors use support vector machine (SVM) to build such a classifier that can fully defend against their proposed attack. Similarly, in [6], the authors use logistic regression and achieve 99.8% defense rate. A limitation of adversarial training is that it needs to know the details of the attack technology or it needs to collect a sufficient amount of adversarial examples. In practice, the attackers will not publish their approaches and they can always change the parameters (e.g., the modulation frequency in [5] or the perturbation factor in [7]) to bypass the defense. Thus, adversarial training is weak in preventing unknown attacks.

Other efforts [2], [6], [13] mention the possibility of using speaker verification (SV) systems for defense. However, this is not strong enough, because the SV system itself is vulnerable to machine learning adversarial examples [7] and previously recorded user speech [6], [1].

From the perspective of the defender, a strategy that can resist various (and even unknown) attacks is expected. One observation is that all existing attacks are based on the replay attack: OS level and machine learning level attacks replay a sound and hardware level attacks replay a designed signal. In other words, the sound source is an electronic device (e.g., loudspeaker, signal generator) instead of a live speaker. On the other hand, only the command from a live speaker should be legitimate in practical applications. That is, **if we can determine if the received signal is from a live speaker, we can defend all above mentioned and even unknown attacks**. To the best of our knowledge, such techniques are non-trivial and have not been widely studied. As an approximation, in [10], the authors propose a virtual security button (VSButton) that leverages Wi-Fi technology to detect indoor human motions and voice commands are only accepted when human motion is detected. The limitation of this work is that voice commands are not necessarily accompanied with detectable motion. In [17], the authors propose VAuth, which collects the body-surface vibration of the user via a wearable device and guarantees that the voice command is from the user. The limitation of VAuth is the need for wearable devices (i.e., earbuds, eyeglasses, and necklaces), which may be inconvenient. Finally, in [1], the authors determine if the source of voice commands is a loudspeaker via a magnetometer and reject such commands. The limitation of this work is that this works only up to 10cm, which is less than the usual human-device distance. Further, this approach does not work for unconventional loudspeakers and malicious signal generators. In summary, existing defense techniques are able to address only some vulnerabilities and therefore more powerful defense techniques will be required to protect voice-driven IoT devices.

References

[1] S. Chen, K. Ren, S. Piao, C. Wang, Q. Wang, J. Weng, L. Su, and A. Mohaisen, "You can hear but you cannot steal: Defending against voice impersonation attacks on smartphones," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 183–195.

[2] W. Diao, X. Liu, Z. Zhou, and K. Zhang, "Your voice assistant is mine: How to abuse speakers to steal information and control your phone," in *Proc. of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*. ACM, 2014, pp. 63–74.

[3] Y. Jang, C. Song, S. P. Chung, T. Wang, and W. Lee, "A11y attacks: Exploiting accessibility in operating systems," in *Proc. of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 103–115.

[4] G. Zhang, C. Yan, X. Ji *et al.*, "Dolphinattack: Inaudible voice commands," in *Proc. of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 103–117.

[5] C. Kasmi and J. L. Esteves, "Iemi threats for information security: Remote command injection on modern smartphones," *IEEE Transactions on Electromagnetic Compatibility*, vol. 57, no. 6, pp. 1752–1755, 2015.

[6] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, "Hidden voice commands." in *USENIX Security Symposium*, 2016, pp. 513–530.

[7] Y. Gong and C. Poellabauer, "Crafting adversarial examples for speech paralinguistics applications," *arXiv preprint arXiv:1711.03280*, 2017.

[8] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," *arXiv preprint arXiv:1801.01944*, 2018.

[9] E. Alepis and C. Patsakis, "Monkey says, monkey does: security and privacy on voice assistants," *IEEE Access*, vol. 5, pp. 17 841–17 851, 2017.

[10] X. Lei, G.-H. Tu, A. X. Liu, C.-Y. Li, and T. Xie, "The insecurity of home digital voice assistants-amazon alexa as a case study," *arXiv preprint arXiv:1712.03327*, 2017.

[11] T. Vaidya, Y. Zhang, M. Sherr, and C. Shields, "Cocaine noodles: exploiting the gap between human and machine speech recognition," *Presented at WOOT*, vol. 15, pp. 10–11, 2015.

[12] M. Cisse, Y. Adi, N. Neverova, and J. Keshet, "Houdini: Fooling deep structured prediction models," *arXiv preprint arXiv:1707.05373*, 2017.

[13] G. Petracca, Y. Sun, T. Jaeger, and A. Atamli, "Audroid: Preventing attacks on audio channels in mobile devices," in *Proc. of the 31st Annual Computer Security Applications Conference*. ACM, 2015, pp. 181–190.

[14] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[15] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and Y. N. Andrew, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[16] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 39–57.

[17] H. Feng, K. Fawaz, and K. G. Shin, "Continuous authentication for voice assistants," *arXiv preprint arXiv:1701.04507*, 2017.

Biometric-Based Wearable User Authentication During Sedentary and Non-sedentary Periods

Sudip Vhaduri[1], and Christian Poellabauer[1]
[1]Department of Computer Science and Engineering
University of Notre Dame, IN 46556
{svhaduri,cpoellab}@nd.edu

***Abstract*—The Internet of Things (IoT) is increasingly empowering people with an interconnected world of physical objects ranging from smart buildings to portable smart devices such as wearables. With the recent advances in mobile sensing, wearables have become a rich collection of portable sensors and are able to provide various types of services including health and fitness tracking, financial transactions, and unlocking smart locks and vehicles. Existing explicit authentication approaches (i.e., PINs or pattern locks) suffer from several limitations including limited display size, shoulder surfing, and recall burden. Oftentimes, users completely disable security features out of convenience. Therefore, there is a need for a burden-free (implicit) authentication mechanism for wearable device users based on easily obtainable biometric data. In this paper, we present an implicit wearable device user authentication mechanism using combinations of three types of coarse-grained minute-level biometrics: behavioral (step counts), physiological (heart rate), and hybrid (calorie burn and metabolic equivalent of task). From our analysis of 421 Fitbit users from a two-year long health study, we are able to authenticate subjects with average accuracy values of around 92% and 88% during *sedentary* and *non-sedentary* periods, respectively. Our findings also show that (a) behavioral biometrics do not work well during *sedentary* periods and (b) hybrid biometrics typically perform better than other biometrics.**

I. Introduction

With the rise of the Internet of Things (IoT), we are now able to remotely monitor and control physical objects, such as vehicles, buildings, health sensors, and many other smart devices. One specific example of such smart devices are wearables, with their ever improving sensing capabilities and network connectivity. Wrist-worn smart devices, such as fitness bands or smartwatches, are used for an increasing number of applications, including user identification for third party services [1], creating a vault for sensitive information (i.e., passwords, credit card information) [2], unlocking vehicles [2], accessing phones and other paired devices, managing financial payments [3], health and fitness tracking, and monitoring of other individuals (e.g., child monitoring or fall detection of elderly people).

While providing these new applications, wearables also introduce various new security and privacy challenges. For example, unauthorized access to a wearable device can provide an attacker with access to IoT systems controlled and monitored by the wearable [4], [5]. Wearables often also collect and store significant amounts of personal (and confidential) user data, which need to be protected from theft. As a consequence, it is essential to provide authentication and security mechanisms for these devices. Existing wearable device authentication mechanisms include knowledge-based regular PIN locks or pattern locks [2], which suffer from scalability concerns [6], since in the IoT world users are flooded with passwords/PINs to obtain access to various objects and services. Additionally, knowledge-based approaches require users to explicitly interact with a display (if present), which can be inconvenient to use [6], [5]. One consequence of this is that many users completely omit the authentication process and leave their devices vulnerable to attacks. Finally, knowledge-based approaches also suffer from observation attacks such as shoulder surfing [6]. Therefore, in recent years, biometric-based solutions have been proposed, since they provide opportunities for implicit authentication, i.e., no direct user involvement or attention is required [6], [5]. However, biometric-based authentication also has challenges and shortcomings, specifically in terms of accuracy and usability. For example, behavioral biometric-based approaches (e.g., gait and gesture) often fail to authenticate a user during periods of low physical activity (e.g., during sedentary tasks) [7], [5], and physiological biometric-based approaches (e.g., ECG or EEG signals) require very precise readings from expensive sensors, which are not available on most wearables due to computational and energy constraints [8].

II. Related Work

Compared to mobile device user authentication, wearable device user authentication is a relatively new research area and traditional user authentication approaches are often not suitable for wearable devices, where computational capabilities and energy resources are much more constrained, or where low-cost sensors may be less accurate (noisy data recordings) or collect recordings only infrequently (e.g., once per minute) [8]. For example, most wearable health trackers make occasional heart rate measurements only instead of collecting raw and much more detailed (but also more costly in terms of energy and computational burden) ECG measurements. Recently researchers have proposed authentication techniques based on behavioral biometrics (e.g., gait [7], gesture [9], and activity type [1], [5]) and physiological biometrics (e.g., PPG signals [10]). Almost all of these studies are based on controlled data collections and the accuracy of these techniques has often been verified with limited numbers of subjects and

over short time periods only. All of these user authentication techniques are also context dependent, e.g., behavioral biometric-based approaches do not work during *sedentary* periods, a model developed for one activity type does not work for other types, and heart rate values captured by a PPG sensor are affected by activity types and their intensities. Therefore, there is a need for a generic authentication approach that is able to consider different combinations of easily obtainable coarse-grained biometric data.

III. Approach

In this work, we propose an implicit and reliable wearable device user authentication scheme that relies on coarse-grained minute-level biometrics that are widely available on state-of-the-art wearables. While the combination of multiple biometrics will result in highly accurate user identification, the reliance on coarse-grained readings from sensors that are commonly found on most fitness and health trackers makes the proposed solution easy to deploy and resource efficient. Compared to our previous work [11], [12], in this paper, we investigate how different combinations of four common biometrics perform when authenticating users during both *sedentary* and *non-sedentary* periods. Before we describe the details of the authentication models, we first discuss the dataset, pre-processing steps, feature computation, and feature selection. For the following analysis we use minute-level **heart rate**, **calorie burn**, **step counts**, and **metabolic equivalent of task (MET)** as sensor data.

A. NetHealth Study Dataset

The *NetHealth* mobile crowd sensing (MCS) study [11], [12], [13], [14], [15], [16], [17], [18] began at the University of Notre Dame in 2015. For this study, over 400 individuals were recruited from the freshmen class and the students were instructed to continuously wear a Fitbit Charge HR device that was provided to them. The data being collected by the Fitbit devices include minute-level heart rate, average heart rate, calorie burn, metabolic equivalent of task or MET, physical activity level/intensity (e.g., sedentary, light, fair, and high), step count, sleep status, and self-recorded activity labels. These collected data can be divided into three biometric groups: **behavioral** (e.g., step counts, activity level/intensity), **physiological** (e.g., heart rate), and **hybrid** (e.g., calorie burn, MET) biometrics, where hybrid biometrics are derived from both behavioral and physiological biometrics.

B. Data Pre-Processing and Feature Computation

Since we are using a real-world dataset, we first need to clean the dataset before using it. Then, we need to segment the continuous stream of biometrics, followed by feature computations before we can build our authentication models.

1) Filtering Invalid Activity Data: A Fitbit device collects heart rate data only when the device is actually worn, but the device collects activity data all the time, even if the device is not worn. Therefore, before we can use the activity data for our analysis, we need to remove "invalid" periods, i.e., the device is not worn. For our analysis, we consider data from 421 Fitbit users.

2) Data Segmentation and Feature Computation: For the classification task, we first segment continuous heart rate, calorie burn, MET, and step counts into five-minute non-overlapping windows starting from a change of activity levels. Since the sampling rate is one sample per minute, each window contains five consecutive samples. When we segment the data into windows, we start from the beginning of an activity level and check for the next five minutes if the same activity level continues. With this approach, we set the reference point at the beginning of an activity level, since the biometrics vary across different activity levels.

For each biometric, we compute 31 statistical features in both time and frequency domains: mean (μ), standard deviation (σ), variance (σ^2), coefficient of variation (cov), maximum (max), minimum (min), range (ran), coefficient of range ($coran$), percentiles ($p25$, $p50$, $p75$, and $p95$), inter quartile range (iqr), coefficient of inter quartile range (coi), mean absolute deviation (mad_μ), median absolute deviation (mad_Mdn), mean frequency (f_μ), median frequency (f_Mdn), power (P), number of peaks (np), energy (E), root mean square (rms), peak magnitude to rms ratio ($p2rms$), root sum of squares (rss), signal to noise ratio (snr), skewness (γ), kurtosis (κ), amplitude of the main frequency (a_main) and secondary frequency (a_sec), and main frequency (f_main) and secondary frequency (f_sec) of the *Discrete Fourier Transform* (DFT) signal obtained using the *Fast Fourier Transform* (FFT) function for each window of biometric data. For *non-sedentary* periods, we also consider the activity level as an additional feature. Therefore, we compute a maximum of 124 and 125 features for each window during *sedentary* and *non-sedentary* periods, respectively.

In the rest of this paper, each biometric is referred to by its initial: "C" (calorie burn), "S" (step count), "M" (MET), and "H" (heart rate). Combinations of these letters are used to represent the corresponding combinations of the biometrics, e.g., "CH" represents a combination of calorie burn and heart rate. Therefore, a biometric combination $b \in \{$C, S, M, H, CS, CM, CH, SM, SH, MH, CSM, CSH, CMH, SMH, CSMH$\}$.

C. Feature Selection

To find relevant features, we first use the *Two-sample Kolmogorov-Smirnov* (KS)-test with the null hypothesis H_0: "the two data sets are from the same distribution." For each feature, we calculate the $p\text{-}value$ for data points from each pair of subjects and drop a feature if most of its $p\text{-}values$ are higher than $\alpha = .05$, i.e., the non-discriminating features. We find that during *sedentary* minutes the behavioral biometric (step count) has no significant feature. However, the behavioral biometric contributes to a good number of significant features during *non-sedentary* (i.e., *lightly*, *fairly*, and *highly* active) minutes.

Next, we apply the Coefficient of Variation (COV)-approach on features obtained from the KS-test. The feature that varies more (i.e., higher cov values) across subjects has a higher

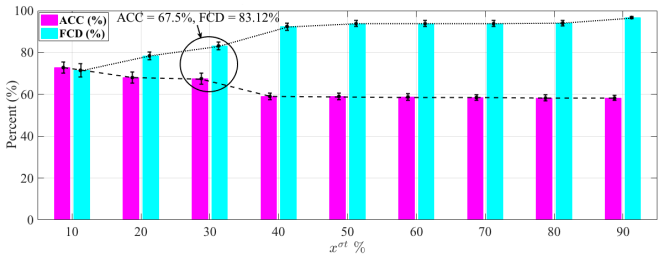


Fig. 1. Average ACC and FCD across different values of parameter $x^{\sigma t}$ during *non-sedentary* periods.

chance of capturing subject varying information, i.e., it can be an influential feature and can better distinguish the subject compared to less influential features that do not vary much. Compared to our previous standard deviation-based approach, the COV-approach is a better measure when comparing different features since cov is a measure of relative variability, i.e., $cov = \sigma/\mu$. For each biometric combination and its associated feature set, we compute the cov of all features in the set and then we find the maximum of the cov values of all features in the set. Next, we compute a set of thresholds using $x^{\sigma t} \in \{10, 20, ..., 90\}$ percent of that maximum cov value. Finally, for each threshold, we pick only those features that have cov values higher than the threshold.

Finding a proper threshold $x^{\sigma t}$ can be tricky; if it is chosen too small, this may lead to a feature set containing redundant and less important features, which may lead to overfitting. In contrast, if the threshold is chosen too high, this may lead to a very small feature set and poor accuracy. In Section IV-B1, we present the optimal values of $x^{\sigma t}$.

A sample feature set obtained using the COV-approach during *non-sedentary* periods with $b = CM$ and $x^{\sigma t} = 30\%$ consists of 27 features: "C" (μ, σ, max, min, ran, $p25$, $p50$, $p75$, $p95$, iqr, mad_μ, mad_Mdn, rms, rss, a_main, a_sec) and "M" (μ, max, $p25$, $p50$, $p75$, $p95$, P, E, rms, rss, a_main).

IV. User Authentication

In this section, we analyze the performance of different feature sets using the binary *Quadratic Support Vector Machine* (q-svm), the best classifier to authenticate wearable device users as found in our previous work [12]. Before analyzing the performance, we prepare our training-testing datasets. When preparing the datasets, the number of windows that we consider is at least 10 times the number of features in the set. This helps to avoid overfitting. For each feature set, we further balance the dataset by randomly selecting the same number of windows per activity level per subject. Next, we split the entire dataset into 75%–25% for training and testing.

A. Performance Measures

To evaluate the performance of different feature sets we use *Accuracy* (ACC) (in %) as the primary measure, which is the fraction of predictions that are correct, i.e., $(TP+TN)/(TP+TN+FP+FN) \times 100\%$, where terminologies have their

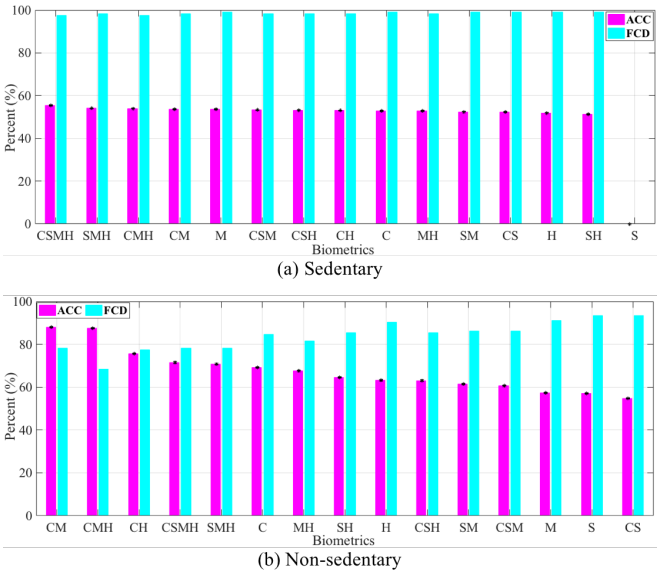


Fig. 2. Bar graphs of authentication Average ACC (in %) and FCD (in %) variations across different biometrics of the COV-approach. The bars inside each subplot are sorted based on average ACC and FCD values. The best biometrics obtained from the two subplots are (a) $b = CSMH$ and (b) $b = CM$.

usual meaning in machine learning. We also consider *Feature Count Decrease* (FCD) (in %) as an additional performance measure. This is a measure of improvement in feature count that a feature selection approach can achieve defined as $FCD = (n^T - n)/n^T \times 100\%$, where n^T and n are the maximum number of features in the initial feature set that we start with (i.e., 124 for *sedentary* and 125 for *non-sedentary* periods with $b = CSMH$) and the number of features in a feature set, respectively. If two feature sets achieve the same accuracy, then the set with higher FCD, i.e., lower feature count, is better since it will lower the computational load, while achieving the same accuracy as the other set.

B. User Authentication Models

When building authentication models for a feature set with N subjects (each having $|W|$ random windows), we train and test N binary q-svm classifiers. Each of these N classification models is used to authenticate a subject from the other $N-1$ subjects. Each subject is identified by an anonymous subject ID. We perform wearable device user authentication separately for *sedentary* and *non-sedentary* periods. First, we find the optimal sets of parameters for different feature selection approaches (Section IV-B1). Next, for each feature selection approach, using its optimal parameter set, we then compare the performance of different biometrics to find the best biometric combination (Section IV-B2).

1) Finding Optimal Parameter Sets: To find the optimal parameter set for each feature selection approach, we first compute the average of all ACC and FCD values obtained for all possible combinations of subjects and biometrics. Then graphically we determine the optimal parameter setting.

TABLE I
AUTHENTICATION ACCURACY SUMMARY ($l = 0$ IS *sedentary* AND $l = 1$ IS *non-sedentary*

l	App--roach	mean (SD) ACC	mean (SD) FCD	Best biometric's mean ACC $(b, n, N, \|W\|)$
0	COV	53.12 (1.03)	98.62 (0.59)	55.46 (CMH,3,415,475)
	KS	76.26 (12.46)	64.06 (15.24)	91.71 (CM,53,412,544)
1	COV	68.24 (10.03)	83.24 (6.67)	88.00 (CM,27,332,331)
	KS	73.89 (9.80)	70.97 (13.26)	88.40 (CM,30,332,331)

Figure 1 shows an example of optimal parameter selection for the COV-approach during *non-sedentary* periods. In this figure we observe that with the increase of $x^{\sigma t}$ the average ACC decreases, but FCD increases. Therefore, we try to find an optimal value of $x^{\sigma t}$ at which both ACC and FCD achieve higher values. We pick $x^{\sigma t} = 30\%$ as our optimal value since after this $x^{\sigma t}$ ACC drops and reaches saturation. Similarly, FCD reaches saturation after $x^{\sigma t} = 30\%$ (Figure 1). We obtain the $x^{\sigma t}$ threshold value for *sedentary* periods using the same approach.

2) Comparing Biometrics of Each Feature Selection Approach: First, we investigate how classifier performance varies across different biometrics for the same feature selection approach. Figure 2 shows the ACC and FCD variation across different biometrics and their associated feature sets obtained from the COV-approach. In Figure 2 (a) we observe that during *sedentary* periods all biometrics except the behavioral biometric (i.e., step counts) perform similarly. During *non-sedentary* periods $b = CM$ has the best performance compared to the other 14 biometrics (Figure 2 (b)). Table I summarizes the user authentication performance, where the average ACC and FCD values are computed from all possible 15 biometric combinations under a specific feature selection approach. Similarly, the last column in the table also represents an average ACC, but it is computed for a particular biometric combination under a specific feature selection approach. For example, we obtain an average ACC = 55.46 for $b = CMH$ under the COV-approach during *sedentary* periods. On average the KS-approach achieves a better ACC compared to the COV-approach. However, the KS-approach has a poor average FCD compared to the COV-approach. In the last column (i.e., "Best biometric's mean ACC" column) in Table I we observe that the two hybrid biometrics (calorie burn (C) and MET (M)) together perform better than other biometrics. During *non-sedentary* periods the KS- and COV-approaches have similar performances. However, during *sedentary* periods there is a big difference between KS- and COV-approaches.

V. CONCLUSIONS

To our best knowledge, our work is the first to use three different types of less informative coarse-grained processed biometric data (i.e., behavioral, physiological, and hybrid) to authenticate the wearable device users implicitly during both *sedentary* and *non-sedentary* periods.

Our findings from the different combinations of the four biometrics (Section IV-B2) show that when behavioral biometrics (step counts) fail to authenticate a user during *sedentary* periods, our multi-modal biometric-based approach can still authenticate the users with a good average accuracy (around 92% with *Genuine Acceptance Rate (GAR)* = .98, obtained from a set of 412 subjects). Similarly, for *non-sedentary* periods we achieve an average accuracy of 88% with *GAR* = .99 using only 27 features (based on a set of 332 subjects). In general, we find that the hybrid biometrics (calorie burn and MET) achieve better performance compared to other biometrics. These accuracy values can further be improved by considering various spatio-temporal factors that can impact person-dependent biometrics. However, to make the authentication approach generic, we build models with relatively smaller feature sets.

REFERENCES

[1] A. Bianchi and I. Oakley, "Wearable authentication: Trends and opportunities," *it-Information Technology*, vol. 58, no. 5, pp. 255–262, 2016.

[2] T. Nguyen and N. Memon, "Smartwatches locking methods: A comparative study," in *Symposium on Usable Privacy and Security*, 2017.

[3] S. Seneviratne, Y. Hu, T. Nguyen, G. Lan *et al.*, "A survey of wearable devices and challenges," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2573–2620, 2017.

[4] M. Shahzad and M. P. Singh, "Continuous authentication and authorization for the internet of things," *IEEE Internet Computing*, vol. 21, no. 2, pp. 86–90, 2017.

[5] Y. Zeng, A. Pande, J. Zhu, and P. Mohapatra, "Wearia: Wearable device implicit authentication based on activity information," in *Proc. A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2017.

[6] J. Unar, W. C. Seng, and A. Abbasi, "A review of biometric technology along with trends and prospects," *Pattern recognition*, vol. 47, no. 8, pp. 2673–2688, 2014.

[7] G. Cola, M. Avvenuti, F. Musso, and A. Vecchio, "Gait-based authentication using a wrist-worn device," in *Proc. Mobile and Ubiquitous Systems: Computing, Networking and Services*. ACM, 2016.

[8] J. Blasco, T. M. Chen, J. Tapiador, and P. Peris-Lopez, "A survey of wearable biometric recognition systems," *ACM Computing Surveys (CSUR)*, vol. 49, no. 3, p. 43, 2016.

[9] S. Davidson, D. Smith, C. Yang, and S. Cheah, "Smartwatch user identification as a means of authentication," *Department of Computer Science and Engineering Std*, 2016.

[10] N. Karimian, M. Tehranipoor, and D. Forte, "Non-fiducial ppg-based authentication for healthcare application," in *Proc. Biomedical & Health Informatics (BHI)*. IEEE, 2017.

[11] S. Vhaduri and C. Poellabauer, "Towards reliable wearable-user identification," in *Proc. Healthcare Informatics (ICHI)*. IEEE, 2017.

[12] ——, "Wearable device user authentication using physiological and behavioral metrics," in *Proc. Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2017.

[13] S. Vhaduri, C. Poellabauer, A. Striegel, O. Lizardo, and D. Hachen, "Discovering places of interest using sensor data from smartphones and wearables," in *Ubiquitous Intelligence and Computing (UIC), 2017 IEEE International Conference on*. IEEE, 2017.

[14] S. Vhaduri and C. Poellabauer, "Hierarchical cooperative discovery of personal places from location traces," *IEEE Transactions on Mobile Computing*, 2017, doi: http://ieeexplore.ieee.org/document/8119982/.

[15] ——, "Impact of different pre-sleep phone use patterns on sleep quality," in *Proc. Wearable and Implantable Body Sensor Networks (BSN)*. IEEE, 2018.

[16] S. Vhaduri, A. Munch, and C. Poellabauer, "Assessing health trends of college students using smartphones," in *Proc. Healthcare Innovation Point-Of-Care Technologies Conference (HI-POCT)*. IEEE, 2016.

[17] S. Vhaduri and C. Poellabauer, "Human factors in the design of longitudinal smartphone-based wellness surveys," in *Proc. Healthcare Informatics (ICHI), 2016 IEEE International Conference on*. IEEE, 2016.

[18] ——, "Cooperative discovery of personal places from location traces," in *Proc. Computer Communication and Networks (ICCCN)*. IEEE, 2016.

Hey, You, Keep away from My Device: Remotely Implanting a Virus Expeller to Defeat Mirai on IoT Devices

Chen Cao*, Le Guan*, Peng Liu*, Neng Gao†, Jingqiang Lin†, and Ji Xiang†
* The Pennsylvania State University
† State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences
caochen11@mails.ucas.ac.cn, {lug14, pliu}@ist.psu.edu, {gaoneng, linjingqiang, xiangji}@iie.ac.cn

***Abstract*—Mirai is a botnet which targets out-of-dated Internet-of-Things (IoT) devices. The disruptive Distributed Denial of Service (DDoS) attack in 2016 has hit major Internet companies, causing intermittent service for millions of Internet users. Since the affected devices typically do not support firmware update, it becomes challenging to fix the holes for vulnerable devices in the wild.**

Both industry and academia have made great efforts in amending the situation. However, none of these proposals is simple to deploy and at the same time effective in solving the problem. In this work, we design a collaborative defense strategy to tackle the Mirai botnet. Our key idea is to take advantage of human involvement in the least aggressive way. In particular, at a negotiated time slot, a customer is required to reboot the compromised device, then a "white" Mirai operated by the manufacturer breaks into the clean-state IoT devices immediately. The "white" Mirai expels other malicious Mirai variants, blocks vulnerable ports, and establishes a heartbeat connection with the server operated by the manufacturer. Once the heartbeat is lost, the server re-implants the "white" Mirai instantly. We have implemented a prototype of the proposed system, and evaluation results show that our system can defeat Mirai attacks effectively.

I. Introduction

On October 21st, 2016, Dyn, an infrastructure vendor, which serves Internet's top giants, such as Netflix and Twitter, was attacked by the record Distributed Denial of Service (DDoS) attack [1]. The attack was later found originated from a botnet malware – Mirai. It was the same botnet malware that attacked a security researcher's blog website and had the record 620 Gbps stream on September 21st [2]. Based on the released code of the original Mirai, new Mirai variants are emerging, which have the ability to exploit zero-day vulnerabilities [3], [4]. Mirai variants mainly target digital video recorders (DVRs) and IP cameras, which are mainly old and low-end products which have no firmware update capability. In addition, since the firmware is read-only, the Mirai code can only stay in the DRAM of the device; rebooting the device also wipes the Mirai code. That being said, once infected, the only recourse is to wait for the device to be rebooted since there is no path to remediation through any type of reconfiguration. Given the large amount of vulnerable devices and the fact that there is no easy way to patch them, Mirai based attacks have become a time bomb which no one can defuse.

Under the pressure from media, some manufacturers claimed they were to recall vulnerable products. For example, Hangzhou Xiongmai Technology planned to recall 4.3 million Internet-connected camera products from the U.S market [5]. Although the company invested a huge time and energy to amend the situation, it only mitigates later attacks, simply because device users have no incentive to cooperate in the recall process. They are not willing to spend time on packing these devices and sending them back, because the Mirai malware does not effect the normal operations of the compromised devices. As a result, there are still a lot of vulnerable devices remaining in the wild.

The manufacturer could also contact customers to perform remote diagnoses so as to minimize disruption to users. If the device was found compromised, the customer agent could explain the bad consequence of leaving the device in the wild, and offer to send the customer an updated device. In return, the manufacturer might provide some bonus for their cooperation. However, a manufacturer may have already had millions of products sold around the world [5]. It needs huge amount of resources for the customer service to contact every user.

Academia is also active in solving the Mirai problem. In [6], the authors proposed to deploy a "white" Mirai-like system. The government is required to record vulnerable products and force the related manufacturers to apply the "white" Mirai approach. This solution inherits a bunch of code from Mirai. Notably, like the malicious Mirai, the "white" Mirai actively scans neighbor vulnerable devices and infects them. The infected devices then become immune to any other similar attack, as a result of blocked ports. There is a combat between the "white" and real Mirai. Only the winner takes control of the majority of the devices. In fact, Mirai has already infected millions of devices, implying that it has a better chance to win the game. In this sense, the solution is non-deterministic. Last but not least, the solution has legal concerns. Although this "white" Mirai system is pushed by the government, it remains controversial as to whether it is illegal to compromise a device without the consent of the user or the manufacturer.

Status quo: Neither industry nor academia has the effective and feasible solution to defeat Mirai. This is evidenced by the emerging Mirai variants such as satori [3] and persirai [4] (Section II-B).

TABLE I
COMPARISON OF DIFFERENT SOLUTIONS

	Simplicity	Effectiveness	Manageability
Recall	✗	✓	✓
Customer Service	✗	✓	✗
White Mirai	✓	✗	✗
Our solution	✓	✓	✓

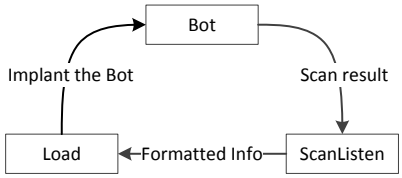


Fig. 1. Mirai's infection process

In this paper, we propose a new solution to tackle Mirai issues. The key idea of our solution is to take advantage of human involvement in the least aggressive way. The idea of kicking in human is based on the observation that rebooting the device wipes the Mirai code in memory. At a high-level, our solution resembles the "white" Mirai work – both solutions utilize Mirai code. In addition to that, we require the customer to cooperate with the manufacturer by rebooting the device at a negotiated time slot. At this time slot, Mirai has very little chance to kick in, while the "white" Mirai could break into the device immediately. In particular, the manufacturer builds an implanter server which remotely implants a virus expeller into a vulnerable device to expel Mirai. Once implanted into a device, the virus expeller blocks all the vulnerable ports exploitable by Mirai. The implanter server is also responsible for discovering the vulnerable devices and keeping the virus expeller alive.

Table I compares our solution with three aforementioned solutions. In this table, simplicity means the amount of work that the customers and the manufacturer must be engaged in to carry out the emendation. Effectiveness means how well the solution can defeat Mirai. Manageability means the capability that a manufacturer can keep track of the deployment of the emendation.

In summary, our main contributions are:

1) We propose a simple, effective and manageable solution to defeat Mirai.
2) We implement a proof-of-concept prototype and show that it can successfully shield vulnerable devices from Mirai's infection.

II. EXPLAINING MIRAI

This section details Mirai's design and implementation, which are helpful in understanding the proposed defense. We also describe several Mirai variants.

A. Mirai Analysis

As a botnet malware, Mirai uses the Client-Server (CS) model to connect the bot with a Command and Control (C&C) server. However, compared with traditional botnet malwares, Mirai is special in its infection method. That is, it implants bots through a dedicated server, not peers.

Figure 1 illustrates Mirai's infection process, and the information flow among the three modules, `Bot`, `ScanListen` and `Load`. `Bot` is a program running in the vulnerable device. It scans other devices to find ones having the same vulnerability. If it finds one, that vulnerable device's information will be uploaded to `ScanListen`, which runs in a pre-known server. The information includes login credentials, device IP address, and vulnerable ports, etc. After receiving the information, `ScanListen` formats it and sends it to `Load`, which runs in the same server. `Load` then uses the information to infect the target device to implant a bot. The following summaries main features of the `Bot` module. After being implanted into the vulnerable device, it performs the followings:

1) **Prevents the device from rebooting**. `Bot` only exits in the memory. If the device reboots, it disappears. Therefore, it prevent the device from rebooting in face of a system failure by writing a request command "0x80045704" for example[1].
2) **Hides process.** `Bot` uses a random string to represent its process name.
3) **Prevents second infection.** `Bot` opens port 48101 and binds itself to it. If another `Bot` attempts to bind to this port, it would be detected and the `Bot` would kill that process. In this way, there is only one `Bot` running on the target device.
4) **Rebinds ports.** `Bot` rebinds port 23(TELNET), 22(SSH), 80(HTTP) to block other botnet's infection.
5) **Expels other malwares.** `Bot` scans the system to find the fingerprint of other malwares. With root privilege, it is able to kill other malware processes.
6) **Scans other devices.** `Bot` scans other devices to discover vulnerable ones. When scanning other devices, with hard-coded 62 pairs of back-doored username and password, `Bot` uses brute-force to guess the username and password of other devices.
7) **Performs DDoS attack.** `Bot` connects to the C&C server and waits for the attacking commands.

B. Mirai Variants

Since Mirai code was disclosed [7], several Mirai variants have been identified, including the most recent satori [3] and persirai [4]. Although these variants exploit new vulnerabilities or add more passwords into their hard-coded database, they share many similarities with the original Mirai. We abstract the design of these Mirai variants as *Scan-Load-Bot*. In particular, these malware scan device's ports to verify whether the target device is vulnerable and load the bot into the vulnerable device if possible.

III. DESIGN

The key idea of our solution is to use a "tit for tat" strategy to defeat Mirai, more generally, the Scan-Load-Bot malware.

[1]This command disables the "watchdog" on the device.

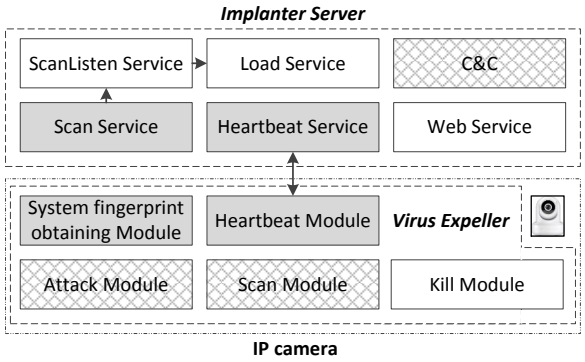


Fig. 2. The Mirai-like system to defeat Mirai. (Gray modules are newly added and grid modules are deleted.)

That is, the manufacturer implants "white" Mirai into the vulnerable devices to expel other virus. We call our "white" Mirai as Mirai-like system.

A. System Architecture

Figure 2 depicts the system architecture of our Mirai-like system and its differences with the original Mirai-family. Two most important components include a virus expeller, which runs on the vulnerable device to expel other Mirai-family bots, and an implanter server, which runs on the manufacture-maintained server to scan and implant the virus expeller into the vulnerable devices. The gridded modules are those removed from the original Mirai. As our system derives from Mirai, we manually remove the code that performs DDoS attacks. The gray modules, i.e. scan service, heartbeat service, heartbeat module and fingerprint module, are newly added to the system.

The Virus Expeller. The virus expeller inherits code from Mirai bot but distinguishes itself from malicious bot by removing the attack module and scan module. Besides, as depicted in figure 2, the virus expeller adds fingerprint obtaining module and heartbeat module. Fingerprint obtaining module is responsible for collecting device's information, such as its unique ID, and kill the virus expeller process if the information does not match. This module is used to avoid the legal problem. If the virus expeller is implanted into a vulnerable device which is produced by another manufacturer, without its acknowledgment, legal problems arise. Therefore, the fingerprint obtaining module can make sure the infected device belongs to the exact manufacturer. Moreover, with the cooperation from the customer, we assume the manufacturer gets the consent from the customer to modify the device. The heartbeat module is a client program for the heartbeat service and reports aliveness periodically.

Once implanted into a vulnerable device, the virus expeller expels Mirai-family bot consecutively. In other words, the virus expeller's core part is a blocking module. It rebinds the ports for remote access as soon as it is implanted into a device.

As Mirai-like system deploys virus expeller in the resource-constraint device, the resource used by the virus expeller should be minimized. Otherwise, the perform of the product can be severely impacted. For this reason, instead of performing the scan on the device, this function is moved to the implanter server.

The Implanter Server. The implanter server runs five services, i.e. Scan service, Load service, ScanListen service, Heartbeat service and HTTP services.

Scan service inherits code from Mirai's scan module. It scans the target IP address list to discover the vulnerable devices. However, in order to implant the virus expeller only to the products of a specific manufacturer, the scan module keeps this specific manufacturer's information and ignores the unmatched devices.

The Load service and ScanListen service are the same with other Mirai-family load and scanlisten modules. They are responsible for collecting the vulnerable devices' information from Scan service and implanting the virus expellers into these devices. Load service is also used for downloading the executable payloads from an HTTP service on the server to the target vulnerable device.

Heartbeat service monitors each virus expeller's aliveness and re-implants them if some accidents happen, such as unplanned power-on and power-off. This would clear the virus expeller.

B. Detailed Design

The proposed system has two phases in the operation of a Mirai-vulnerable device. In the deployment phase, the customer cooperate with the manufacturer to implant our code into the device. In the holding phase, the device works normally, and our code keeps itself active within the lifetime of the device.

Deployment Phase. If a Mirai-family bot has already existed in a device, no virus expeller can be implanted into this device. Because Mirai-family bot should have rebinded the vulnerable ports for remote access. In order to move the needle, the customer must cooperate with the manufacturer to clear existing Mirai-family bot and implant our virus expeller. In this phase, the customer provides its IP address list to the manufacturer, and agrees on a random time with the manufacturer. At that time slot, the customer reboots the vulnerable devices and the malicious bots can be cleared. Then, the manufacturer utilizes the Scan service to quickly scan the IP address list to locate vulnerable devices and implants the virus expellers before other Mirai-family bots can infect them. As the Mirai-family malware does not know the time negotiated by the customer and the manufacturer, our solution can effectively implant the virus expeller and block spread of Mirai-family malware. Furthermore, according to the experiment done in [8], it costs around 98 seconds for Mirai to infect a device after being connected to the Internet. As a result, there is little chance for Mirai to infect these devices at the very moment that the customer reboots them.

Holding Phase. Once implanted into a vulnerable device, the Mirai-like system should make sure our virus expeller is kept

active in the device. We design a heartbeat module for this purpose. In this phase, the heartbeat module in a virus expeller sends back a heartbeat to the Heartbeat service in the remote server periodically. At the same time, the heartbeat service has a timer for each virus expeller and refresh the timer once receiving a heartbeat, and continuously checks all the timers. If one timer exceeds a specific value without receiving any heartbeat messages, it means the related virus expeller has lost the connection to the server. In this way, the manufacturer could know whether a virus expeller is running on a specific device. If this device is accidentally rebooted, the heartbeat is lost immediately. Then Heartbeat service can invoke Scan service to scan the IP address list and can re-implant the virus expeller into the device.

IV. Evaluation

We evaluate the proposed system using a Dahua DH-3004 digital video recorder, which is vulnerable to Mirai. The implanter server runs in a Debian 8 Linux powered by an Intel i7-4790 processors with 2GB memory. Although we use the Dahua device in our evaluation, our solution does not depend on any specifics of the Dahua device. It can be easily applied to protect other Mirai-vulnerable devices.

We conducted three experiments to verify three essential functions of this system in the two operational phases. First, whether the proposed system can implant the virus expeller into a target device. In particular, we tested if the following process can be completed: Scan $\rightarrow$ ScanListen $\rightarrow$ Load $\rightarrow$ Virus Expeller (in the target device) $\rightarrow$ Heartbeat. If the virus expeller is implanted into a device, Heartbeat service can receive the messages sent by it. Using wireshark running on a PC which monitors the network packages, we measured the time taken to implant the virus expeller since Scan service starts to scan this device. On average, it takes about 10 seconds, which is much faster than the wild Mirai does. After that, Heartbeat service periodically receive a message sent from the virus expeller.

Second, after being implanted into the target device, the virus expeller should resist to other Mirai-family's infection. We run the second experiment by implanting the virus expeller into this target device firstly and run Scan service to scan this specific device to see whether Load service can receive any information. As Scan service, Scanlisten service and Load service inherit code from Mirai, if our Mirai-like system can scan the device and return related information to Scanlisten service, this means the target device can be infected by Mirai-family even there is a virus expeller inside. From Load service's log message, we can directly find the virus expeller is implanted into this device and no more message is generated after that. Besides, we used telnet to connect to this device manually. As expect, we could not connect to this device. Therefore, after being implanted into a device, the virus expeller can successfully defeat other Mirai-family bots.

Third, whether the implanter server can re-implant the virus expeller into this specific device if the device is accidentally rebooted. We run this experiment by rebooting the device and monitor the log of Heartbeat service. From the log, we can see a list of operations. First, the virus expeller stops sending back the heartbeat message. Second, the heartbeat service invokes scan service to scan the target IP address list. Third, the scan service finds the target device and the virus expeller is implanted by the load service.

On our prototype, we ran three experiments and proved that our system can successfully accomplish the three designed function, effectively defeating Mirai-family malware.

V. Conclusion

We have presented a new mechanism to aid manufacturer in amending the Mirai-vulnerable devices on the market. Different from recalling or customer services, in which customers are unwilling to cooperate, our solution needs the minimal involvement of customers. Different from the "white" Mirai proposed in [6], our solution does not need brute-force scan, and can deterministically patch the compromised device.

A proof-of-concept prototype has been implemented. Experimental results showed that the proposed solution is both simple and effective. Given the great number of Mirai-vulnerable devices in the wild, our solution provides a promising solution to mitigate the threats from Mirai-family malware, until all the vulnerable devices are retired. Since our solution requires close cooperation with the manufacturers, we plan to contact the involved manufacturers to further carry out our solution in real world.

References

[1] "Ddos attack that disrupted internet was largest of its kind in history, experts say," https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet.

[2] "Hacked cameras, dvrs powered todays massive internet outage," https://krebsonsecurity.com/2016/10/hacked-cameras-dvrs-powered-todays-massive-internet-outage/.

[3] "Warning: Satori, a mirai branch is spreading in worm style on port 37215 and 52869," http://blog.netlab.360.com/warning-satori-a-new-mirai-variant-is-spreading-in-worm-style-on-port-37215-and-52869-en/.

[4] "Persirai: New internet of things (iot) botnet targets ip cameras," https://blog.trendmicro.com/trendlabs-security-intelligence/persirai-new-internet-things-iot-botnet-targets-ip-cameras/.

[5] "Chinese firm recalls camera products linked to massive ddos attack," http://www.pcworld.com/article/3133962/chinese-firm-recalls-camera-products-linked-to-massive-ddos-attack.html.

[6] J. A. Jerkins, "Motivating a market or regulatory solution to iot insecurity with the mirai botnet code," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan 2017, pp. 1–5.

[7] "Mirai source code," https://github.com/jgamblin/Mirai-Source-Code.

[8] "Mirai infection experiment," https://twitter.com/ErrataRob/status/799556482719162368.

A Comparison of Data Streaming Frameworks for Anomaly Detection in Embedded Systems

Murray Dunne, Giovani Gracioli, and Sebastian Fischmeister
University of Waterloo, Canada
{mdunne,g2gracio,sfischme}@uwaterloo.ca

***Abstract*—As IoT devices are integrated into our daily lives, verification and security become of increasing concern. Using anomaly detection methods, we can identify damaged and compromised devices by examining traces of their activity. Collecting these traces with minimal overhead is a core requirement of any anomaly detection system. We evaluate four publish-subscribe broker systems on their viability for trace collection in the context of IoT devices. Our comparison considers ordering and delivery guarantees, client language support, data structure support, intended use case, and maturity. We run each system on original Raspberry Pis and collect network performance statistics, measuring their capability to collected traces in a resource-constrained embedded systems environment. We conclude with recommendations for designing an anomaly detection system for IoT devices.**

I. Introduction

Embedded systems, such as Internet-of-Things (IoT) and autonomous vehicles, are present in our daily lives. Such systems interact with the environment through several sensors and actuators, usually controlling the operation of critical processes. Moreover, these systems generate a huge volume of data, which makes the task of verifying the system specification difficult.

In this context, trace-based anomaly detection can monitor the system behavior and prevent or/and recover from failures [1]. Anomaly detection aims at detecting execution patterns that do not conform with the expected system behavior. It can be done online (during run-time) or offline (by analyzing recorded traces). Online anomaly detection usually receives a stream of data as input and incrementally adapts anomaly scores for the analyzed system, thus providing early detection of an anomaly (when compared to the offline approach).

Trace-based online anomaly detection requires a data streaming infrastructure with minimal performance overhead. Examples of general-purpose streaming frameworks are Zmq, Mqtt, ActiveMQ, Apache Spark, Redis, NATS, Apache Kafka, and RabbitMQ. Other data streaming frameworks, such as ROS, Polysync, Qnx PPS, OpenDDS, RTI Connext, and OpenSplice are designed as a for building an entire product, rather than as an ancillary monitoring application.

Figure 1 shows an overview of a general data streaming framework infrastructure for anomaly detection in embedded systems. Sources are different embedded systems, such as a smart building or an autonomous vehicle, and generate streams of data at run-time. Data from these systems is sent to a data streaming framework. Processors connect to the framework, receive and process data. The processor output is either written back to the framework (to be consumed by another processor) or indicates an anomaly. Finally, sinks receive data and can perform an action, such as storing the data into a file [2].

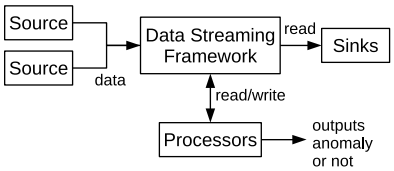


Fig. 1: Overview of general data streaming framework organization.

Several data streaming frameworks have been proposed recently. However, to the best of our knowledge, a comparison among them targeting embedded systems has not been made yet. Two metrics are important for data streaming framework performance: (i) low run-time latency (the time difference between the instant a data is generated by sources and the instant it is received by processors or sinks); and (ii) throughput, the rate of data transmission a framework can support.

In this paper, we present a comparison of existing data streaming frameworks focusing on embedded systems. We compare Redis [3], Kafka [4], NATS Streaming Server [5], and RabbitMQ [6] in terms of characteristics and performance (latency and throughput). We choose them because they have been receiving wide attention by the scientific community and represent different classes of streaming framework (*i.e.*, implemented with different languages, targeted at different platforms, and supporting different features). The performance comparison is carried out using a standard embedded system platform (Raspberry Pi). Our results indicate that Redis and RabbitMQ are suitable frameworks for embedded systems in both features and performance.

The rest of this paper is organized as follows. Section II overviews the main features of the analyzed frameworks. Section III presents the performance evaluation. Section IV discusses related works and Section V concludes the paper.

II. Overview of Data Streaming Frameworks

Redis. REmote DIctionary Server (Redis) is an open-source in-memory key-value database. It supports several data structures, including list, sets, maps, and bitmaps. Redis can be used

as a data streaming framework because it provides a publish-subscribe interface. Redis clients publish data into channels using the REdis Serialization Protocol (RESP). Instances that subscribe to channels receive data in the same order it was published. Redis also supports the integration with on-disk databases. Moreover, it has a small memory consumption; in a 64-bit system, 1 million keys (hash values), representing an object with five fields, use around 160 MB of memory [3]. This small memory consumption is adequate for embedded systems, which usually have limited memory. Redis provides a replication mechanism based on master-slave, in which slave server instances are exact copies of master servers. To reduce the network Round Trip Time (RTT) latency over transmitted messages, Redis implements Pipelining, making it possible to send multiple commands to the server without waiting for individual replies [3]. These replies are instead batched together into a single response.

Kafka. Apache Kafka is a distributed streaming platform written in Java. Kafka runs as a cluster of one or more servers. The cluster stores streams of records (key, value, and a timestamp) in topics. A topic is a category or a name in which records are published. For each topic, the Kafka cluster maintains a structured commit log, formed by partitions. Each partition within a topic is an ordered sequence of records that is continually appended to the structured commit log. Log partitions can be distributed over the cluster servers, providing fault tolerance. Clients subscribe to topics to receive/write real-time streams using a binary protocol over TCP. Kafka provides APIs for sources, processors, and sinks. Moreover, Kafka provides persistent storage by writing topic records to the disk. As Kafka is written in Java, it requires a Java virtual machine (JVM). This may not be appropriate for resource-constrained embedded systems due to JVM memory requirements [4].

NATS Streaming Server. NATS is an open-source data streaming server written in Go. NATS streaming server embeds a NATS server. Thus, the streaming server is not a server, but a client to a NATS server. Clients also communicate with the streaming server through the NATS server. All the communication uses a NATS streaming protocol based on protocol buffers. NATS streaming server provides a publish-subscribe interface based on channels. Clients send and receive messages to/from channels. Messages can be stored in memory or disk files. NATS provides a message logging mechanism to save all messages produced in a channel, allowing historical message replay by subject. Clients may specify a playback start position in the stream of messages stored for the subscribed subjects channel. NATS streaming server does not support clustering of servers. However, it supports fault tolerance by allowing the initialization of a group of servers. Within the group, only one server answers to clients requests, while the others monitor the main server. When the main server fails, another one takes control and acts as the main server [5].

RabbitMQ. RabbitMQ is an open source message brokering server maintained by Pivotal software. It implements the Advanced Message Queuing Protocol (AMQP) (ISO/IEC 19464:2014), a standardized protocol for message brokering services [6]. AMQP defines a two-stage architecture where messages are first transmitted to an exchange which forwards them to different queues depending on the exchange selected. Exchanges exist for broadcasting copies to multiple queues, addressing queues by name, or pattern matching. Messages in AMQP queues are acknowledged upon receipt by the server, and clients must acknowledge a message before it is removed from the queue. Queues must be declared before use and may be saved to disk, so their contents are not lost on restart. There is no inbuilt mechanism for replaying a message history, but RabbitMQ may be configured to store logs of message activity. RabbitMQ supports clustering where queues exist on only one node at a time, but are reachable from all nodes. A configuration option enables replication of entire queues. Nodes that store their data entirely in memory are available.

Table I summarizes the discussed features for each framework. The four framework are mature; they all provide clients in several languages, support disk storage, publish-subscribe interface, and message ordering. They differ in how they organize and process data internally, the languages in which they are written, and communication protocol.

III. Performance Evaluation

A. Experiment Description

The evaluation compares the four frameworks on publisher-to-subscriber latency and throughput. We compare messages differing in size and frequency. To mimic an IoT installation we use two first version Raspberry Pis (single 700MHz ARM11 core, 512M RAM) for the subscriber and the publisher. The server is a Raspberry Pi Version 2 (quad-core ARM Cortex-A7, 1G RAM). Figure 2 details the experimental setup.

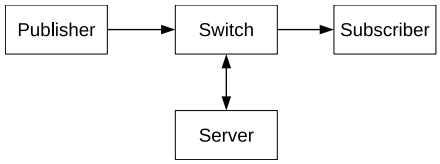


Fig. 2: Overview of the experimental setup.

We consider message sizes of 256 bytes, 1 KiB, 100 KiB, and 1 MiB. The 256 byte messages are analogous to command or status update packets from simple IoT devices such as thermostats or light bulbs. The larger messages represent outputs from more complex devices, such as cameras, lidar, and smart sensors for Industry 4.0. We generate messages with a CSPRNG to eliminate effects from any internal compression. We publish messages at 30Hz, 60Hz, and 100Hz. The lower frequencies resemble routine status updates from passive IoT devices (which are often much slower than 30Hz). The 60Hz rate is a common camera FPS measure, and the 100Hz rate may be used for high-frequency sensors.

We consider three factors: (i) choice of framework; (ii) message size; and (iii) message frequency. All frameworks are configured with persistence disabled because we are targeting online anomaly detection. The clients and server are synchronized using PTP, a network level time synchronization protocol

TABLE I: Features comparison among the analyzed frameworks.

Feature / Framework	Redis	Kafka	NATS	RabbitMQ
Supported data structures	strings, hashes, lists sets, bitmaps	Structured commit log	Queue	Queue
Message ordering	Yes	Yes	Yes	Yes
Client-side languages	About 49 different languages	About 17 different languages	C#,Go,Java,Node.js,Python	About 30 languages
Storage	In-memory dataset and saving in disk	Disk	In-memory or disk	In-memory (saving logs in files)
Written in	C	Scala/Java	Go	C
Message publication	Pub-Sub	Pub-Sub	Pub-Sub	Pub-Sub
Replication	Master-slave	Replicated cluster	Fault Tolerance/Partitioning	Clustering
Protocol	REdis Serialization Protocol (RESP)	Binary protocol over TCP	Based on Google protocol buffer	AMQP

capable of microsecond accuracy [7]. Latency is measured by including the timestamp at which a message is sent within the message itself. The subscriber then notes the timestamp at which it receives the message and subtracts to find the latency. Throughput is measured by multiplying the current message size by the time it takes all the messages in a single run to arrive, then dividing by the total time for that trial.

One sample of our experiment consists of one idealized minute of message transmission. That is, at 100 Hz we expect 6000 messages to be transmitted in a minute. If transmission takes longer than a minute, the experiment waits for all messages to be transmitted. For each configuration ($4 \times 3 \times 4 = 48$), the experiment is run five times. Due to the large startup time of these services, experiments on each framework were run in order: Redis, Kafka, NATS, and then RabbitMQ. Therefore, this is not a fully randomized experiment.

B. Results and Discussion

Consider the latency and throughput results in Figures 3 and 4. In our setup, both Kafka and NATS were unable to transmit large messages. The subscribers hung after a handful of messages were received for all 1 MiB messages to Kafka and NATS, and also for the 100 KiB messages at 100 Hz to Kafka. At high frequencies, Kafka dropped 15% of messages (note that Kafka was configured to keep messages 1 ms, so they would not persist to disk) and NATS dropped 3%. Redis also dropped 0.16% of the 1 MiB messages at 60 Hz and 100 Hz. RabbitMQ dropped no messages.

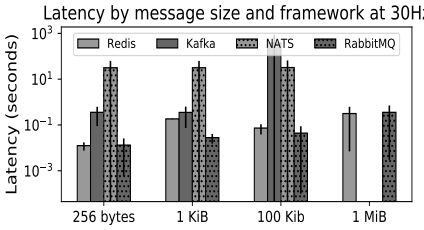
Redis and RabbitMQ behave comparably in all analyzed scenarios for both metrics. At small message sizes (256 bytes and 1 KiB) throughput is comparable across all four systems. This points to the networking capability of the platform as the primary bottleneck for small messages. Even at high message sizes, the throughput values remain comparable for all frameworks except NATS. This is likely due to the NATS Streaming server connecting to the main NATS server, adding a level of indirection. This indirection also impacts the latency of messages sent through NATS. The primary differences between the frameworks are observed in the latency results. RabbitMQ and Redis maintain lower and more consistent latencies across all message sizes; Kafka is significantly slower but still consistent, but NATS is more than an order of magnitude slower and more variable than the other frameworks for all message sizes. We also observe an increase in the variance of message latencies as the data size increases. Especially for Kafka and NATS at 100 KiB and 1 MiB. This is likely due to longer waiting times in buffers.

As an application of a data streaming framework in embedded systems, consider the LIDAR sensor on an autonomous vehicle. The autonomous vehicle could publish the current power consumption of the sensor into the framework and a detector would continuously analyze its state. When the power state goes to off and other sensors are still on, then an anomaly is reported. Alternately, the vehicle could publish the gear pattern from the autonomy software into the data streaming framework. Detectors would then monitor this stream for driving irregularities. The data streaming server would not run on the same platform as the autonomy software; it would run on a smaller, low power system solely tasked with motoring the vehicle. The choice of embedded system to run the framework is constrained by the operating system requirements, the amount of data collected, and the complexity of the anomaly detector. A cloud server may be needed as the number of clients and detectors grows.

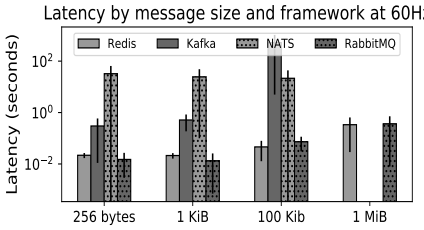
IV. Related Work

There are comparisons between data streaming frameworks available online, but they often lack scientific rigor (*i.e.*, do not entirely describe the experimental environment) and do not target embedded systems. For instance, Yigal discusses the throughput in Kafka and Redis but does not execute both on the same platform [8]. Treat compares the throughput and latency in Kafka and NATS, using a high-performance server. They find that Kafka and NATS present a similar performance in both metrics [9].

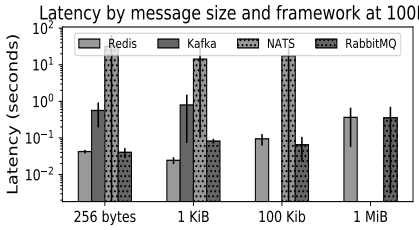
Data streaming frameworks have been used in several works to detect errors and anomalies in different systems. Lopez et al. discuss the characteristics and compare three stream processing platform (Apache Spark, Flink, and Storm) in terms of throughput using a threat detection application [10]. Solaimani et al. used Apache Spark to detect anomaly for multi-source VMware-based cloud data center [11]. Subramaniam et al. proposed a framework to online detect anomalies (outliers detection) in wireless sensors networks [12]. However, the authors only implemented the framework in a simulator. Du et al. proposed a network anomaly detector based on Apache



(a) Latency at 30Hz.

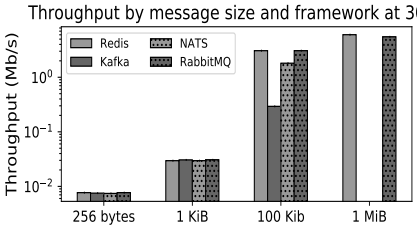


(b) Latency at 60Hz.

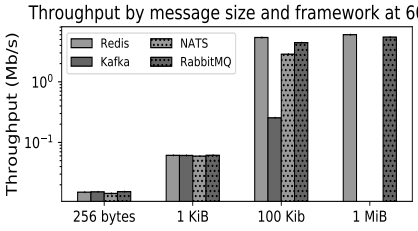


(c) Latency at 100Hz.

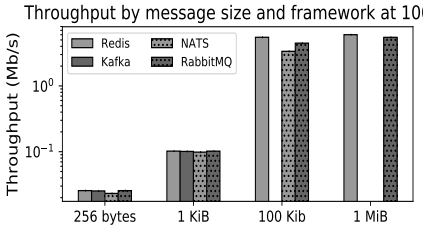
Fig. 3: Experimental latency results.



(a) Throughput at 30Hz.



(b) Throughput at 60Hz.



(c) Throughput at 100Hz.

Fig. 4: Experimental throughput results.

Storm [13]. Shi et al. implemented an online fault diagnosis system based on Apache Spark for power grid equipment [14].

V. Conclusion

As security becomes a growing concern in IoT systems, we turn to anomaly detection techniques to monitor the correctness of devices. Collecting traces for such systems requires an efficient data collection framework that will run on embedded devices. We compared Redis, Kafka, NATS, and RabbitMQ as publish-subscribe brokers on original Raspberry Pis.

Both Redis and RabbitMQ performed nearly identically. They are both C programs designed with the specific goal of lightweight message transmission, making them suitable for embedded systems. Clients for both Redis and RabbitMQ are widely available and require little more than an open network socket. We would recommend either of these systems for supporting trace-based anomaly detection in embedded systems. However, we would not recommend Kafka or NATS. Both are designed for web-based usage, focusing on delivery, concurrency, and fault-tolerant guarantees, rather than raw performance. These guarantees may not be required for an anomaly detection system for IoT devices.

As future work, we plan to integrate and evaluate several embedded system anomaly detectors, such as SiPTA [15] and arrival curves [16] in a data streaming infrastructure based on Redis targeting IoT and embedded systems. Other future work could consider additional low-level publish-subscribe brokers, and use a true real-time operating system.

References

[1] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection for discrete sequences: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):823–839, May 2012.

[2] Gianpaolo Cugola and Alessandro Margara. Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.*, 44(3):15:1–15:62, June 2012.

[3] Redis website, Jan 2018. Available online: https://redis.io/.

[4] Apache kafka website, Jan 2018. Available online: https://kafka.apache.org/.

[5] Nats website, Jan 2018. Available online: https://nats.io/.

[6] Rabbitmq website, Jan 2018. Available online: http://www.rabbitmq.com/.

[7] J. Han and D. K. Jeong. A practical implementation of ieee 1588-2008 transparent clock for distributed measurement and control systems. *IEEE Trans. on Inst. and Meas.*, 59(2):433–439, Feb 2010.

[8] Asaf Yigal. Kafka vs. redis: Log aggregation capabilities and performance, Nov 2016. Available online: https://logz.io/blog/kafka-vs-redis/.

[9] Tyler Treat. Benchmarking nats streaming and apache kafka, Dec 2016. Available online: https://dzone.com/articles/benchmarking-nats-streaming-and-apache-kafka.

[10] M. A. Lopez, A. G. P. Lobato, and O. C. M. B. Duarte. A performance comparison of open-source stream processing platforms. In *2016 IEEE GLOBECOM*, pages 1–6, Dec 2016.

[11] M. Solaimani, M. Iftekhar, L. Khan, B. Thuraisingham, J. Ingram, and Sadi E. Seker. Online anomaly detection for multi-source vmware using a distributed streaming framework. *Softw. Pract. Exper.*, 46(11):1479–1497, November 2016.

[12] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online outlier detection in sensor data using non-parametric models. In *Proc. of the 32Nd VLDB*, pages 187–198, 2006.

[13] Y. Du, J. Liu, F. Liu, and L. Chen. A real-time anomalies detection system based on streaming technology. In *2014 Sixth IHMSC*, volume 2, pages 275–279, Aug 2014.

[14] W. Shi, Y. Zhu, T. Huang, G. Sheng, Y. Lian, G. Wang, and Y. Chen. An integrated data preprocessing framework based on apache spark for fault diagnosis of power grid equipment. *Journal of Signal Processing Systems*, 86(2):221–236, Mar 2017.

[15] Mohammad Mehdi Zeinali Zadeh, Mahmoud Salem, Neeraj Kumar, Greta Cutulenco, and Sebastian Fischmeister. Sipta: Signal processing for trace-based anomaly detection. In *Proc. of the EMSOFT*, pages 1–6, New Dehli, India, Oct. 2014.

[16] M. Salem, M. Crowley, and S. Fischmeister. Anomaly detection using inter-arrival curves for real-time systems. In *ECRTS*, France, 2016.

A secure IoT architecture for streaming data analysis and anomaly detection

Safa Boudabous, Stephan Clémençon, Ons Jelassi, Mariona Carós Roca,
Image, Data & Signal Department (IDS)
Télécom ParisTech, LTCI, Université Paris Saclay
Paris, France
firstname.lastname@telecom-paristech.fr

***Abstract*—Discovery of repeating temporal patterns and prediction based on time stamped data generated by IoT services raise important methodological issues. A typical predictive problem, addressed in this paper, consists in the early detection of change points or anomalies, that may be caused by a malicious use of the system for instance. Although online anomaly detection is now the subject of much attention in the data science literature, motivated by crucial industrial applications such as predictive maintenance and health monitoring of complex infrastructures, the rapidly changing environment inherent to most IoT applications makes this task even more challenging. Beyond the crucial control of the false alarm rate, the quasi real-time analysis must take into account the efficiency of computing resources and the possible security risks in data transfers over the network. We propose here an architecture for analyzing IoT datastreams data and a dedicated method for on-line anomaly/novelty detection, based on nonparametric (mean discrepancy) test statistics and multiple hypothesis testing techniques. Numerical results based on experiments involving synthetic datastreams and real energy consumption datastreams provides empirical evidence of the relevance of the methodology proposed.**

***Keywords*-time series; IoT data; secure Machine Learning architecture; Anomaly detection; change point detection.**

I. Introduction

With an expected total number of 25 billion of connected smart devices by 2020, Internet of Things (IoT in abbreviated form) is expected to enable the development of a wide variety of applications, ranging from domotics to heathcare through predictive maintenance of energy networks for instance. The exponential growth of IoT undoubtedly offers exciting opportunities but also presents great scientific challenges, in the development of accurate massive datastreams analytics and in the design of secure and efficient system architectures to implement them. In particular, a minimal latency when processing streaming data in a secure architecture is a major requirement. Part of the data must be analyzed locally, by the smart devices with high constraints of resources or in middlewares closest to them, in order to preserve privacy, to guarantee personalization or because of bandwidth limits, while other treatments like statistical learning based on the information collected by several sensors at the population level can be performed in a central equipment. Change and anomaly detection in time series has been the subject of a good deal of attention in the statistics and machine-learning literature, see *e.g.* [1] and the references therein. However, most documented techniques are hardly scalable (when considering multivariate high frequency data series) and do not meet the low-latency standards for quasi-real time analysis, as required in the detection of cyberattacks for instance. In the dynamically changing environment inherent to most IoT applications, the sensory data streams collected from connected objects are often influenced by the underlying hourly, daily, and weekly rhythms of human activity. This is reflected by the occurence of a non-stationary behavior in the statistical distribution of the data streams. Due to this complex environment setting, the baseline hypothesis, made by most (sequential) anomaly detection methods, that the distribution is normal or stationary, is not fulfilled in general. Hence, the crucial need for an efficient approach that allows early and accurate abnormal behavior detection in a computationally efficient way.

In this paper, we present a mathematical model for automated online anomaly detection in time stamped datastreams. The distributed processing of certain data analysis task is investigated. The background is presented in Section 2, together with related works. In Section 3, we develop our framework for on-line anomaly detection based on streaming data, while Section 4 describes how the method we promote can be integrated in a global architecture for IoT data collection, processing/learning and alarm generation, combining distributed components and a client-server model. Next, Section 5 presents our evaluation on synthetic and real-world datasets. Some concluding remarks are eventually collected in Section 6.

II. Related works

The oldest techniques for anomaly detection in datastreams are based on statistical inference. Anomalies being assumed to be rare, they are assumed to lie in the complementary sets of high probability regions of the probabilistic model. The statistical procedure then consists in computing such regions of the feature space (referred to as *tolerance regions* or *minimum volume sets*) from a robust distribution estimate, based on data composed of 'normal' observations in their vast majority, alarms being raised as soon as data fall outside the region thus built. Such anomaly detection

algorithms were first applied for statistical process control (SPC), providing a monitoring tool so as to control the quality of a product during a continuous manufacturing process. These approaches assume that the distribution of the process prior to a change is fully known. More recently, in [2], a generic analytics engine which provides robust alerts upon changes and anomalies in sensory data streams has been proposed. The engine combines different nonparametric change detection algorithms. Depending on the type and the nature of the considered time series, the authors proposed to use test statistics-based methods such as the two-sample Kolmogorov-Smirnov test in order to detect deviation from stationarity in univariate sensory data, to implement an algorithm based on k-nearest neighbor estimation so as to detect changes in multiple sensor data streams and the algorithm 1-class SVM for change detection in non-periodic non-stationary multi-sensory streams. More generally (see [3]), each adaptive streaming algorithm must define four main components:

1) memory management that explicits the data storage primitives and the forgetting mechanism to dynamically update data samples,
2) change detector that defines a mechanism to identify change point in the stream,
3) learning component that defines a method to update the learned model when new instances arrive,
4) loss estimator that allows generalization error quantification of the prediction model based on the environment feedback.

III. Anomaly Detection Method

A. Solution Pipeline

In this paper, we propose a (nonparametric) statistical approach to anomaly detection that encompasses the dynamic nature of data streams generated by IoT applications. Detection of abnormal behaviors in daily user energy consumption shall be our running example throughout the article. Such data streams are characterized by a periodic daily consumption pattern that reflects the living habit of the user. These data are also impacted by exogenous variables and events, exhibiting many change points, even in the normal state. Hence the need for adaptive learning algorithms able to detect sudden unusual changes in the data stream.

Our algorithm is a two-stage process, as depicted by Fig. 1. The first step ('cold start') aims at learning the normal behavior of the system and compute a set of statistics providing an up-to-date summary of the observed process. The second step consists in identifying anomalies in data streams on near real-time and incrementally adapt the learned model to significant changes in the data in order to control the false alarm rate in particular. The learning step can be performed either offline or online and takes a period of K weeks ($K \geq 2$) to construct/update the model. The method maintains three important informations about the observed data: a representative sample of the data, a set of descriptive statistics (the mean, the standard deviation, the minimal and the maximal value) and a model for the distribution of changepoints in the normal state of the system. This provides a one-week model of the data. Initially, the sample stores the first observed week of data and then we compute statistics measuring deviations between the stored values and the new observed instances. The detection step is a sequential process with two components: The first component consists of an iterative statistical inference method that uses a set of nonparametric test statistics to detect abnormal sequences in the streaming data. It starts by collecting a sample of N instances (N is a tuning parameter, picked larger than 20 in our running example) and then, each new observed instance is added to the sample and several test statistics (measuring each a specific possible deviation from the normal behavior) are computed in order to detect the occurrence of various types of anomalies in the data. Depending on the results of the multiple tests, an alarm is raised if a significant deviation is observed, using the false discovery rate (FDR) method. Thus, if an anomaly is detected, we use the test results and the stored descriptive statistics to define the type and occurrence time of the anomaly and we update the sample in order to gather new observations. If no anomaly is detected, we keep repeating the same process. Note that, although the size of the current data sample is variable, it stays bounded in practice by a maximum value M limited by the memory required by the process and the size of the representative sample of the normal behavior. The method relies on a multiple hypothesis procedure in order to control the false discovery rate. The second component focuses on detecting statistically significant deviations in the distribution of the changepoints. Using a Poisson process (we can refine the approach by considering straightforward nonparametric extensions, i.e. non-homogeneous Poisson processes), we compare the current count of changepoints over a given time interval T to the distribution of changepoints counts of the normal behavior in the same interval. We mainly detect changes in variance and constant readings. Those two cases of deviations affects considerably the distribution of the changepoints in the data. During the detection and whenever no anomaly is detected, the learned model is still updated by refreshing the three informations stored in the learning step.

Lastly, we point out another technique used to control the false discovery rate and to adapt the normal behavior to changes which is the user feedback. The user can confirm that a detected anomaly corresponds to an abnormal behavior, not to a change of use. In fact, the accuracy of the data is affected by concept drift in data streams even if an updating mechanism is set up. Thus, the interaction with user allows adjusting the stored sample of the normal behavior of the distribution of the observed process.

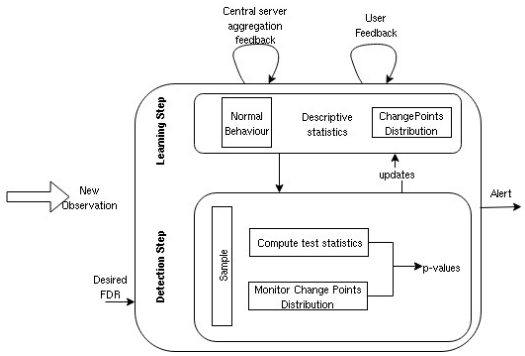


Figure 1. Proposed Solution Architecture

B. Homogeneity Tests Statistics for Anomaly Detection

As the system may deviate from its normal behavior in many ways, anomalies can be of various types. In order to accurately detect them in the datastreams, we need to select a variety of relevant test statistics to compute. As said above, the distribution of the streaming data in the normal state is generally very complex, exhibiting many change points. Nonparametric tests are thus preferred in order to avoid wrong model assumptions, jeopardizing the procedure and yielding a too high false alarm rate. For that, we consider four classical nonparametric test statistics: Wilcoxon Signed Rank test, Kolmogorov-Smirnov test, Fligner test and Mann-Kendall test. The three first ones are sample tests that will be applied on the sample representing the normal behavior and the observed sample. The last one needs only the current values to compute its score. When Wilcoxon Signed Rank statistic tests detects when a possible change occurs in the sequence, Kolmogorov test focuses on general changes in the data distribution, Fligner test looks for changes in the variability of the data and Mann-Kendall test is used to detect a change in the monotonic trend in the data. The combination of those tests allows detect a large variety of anomalies. As hypothesis testing follows here a P-value based approach, each test will be analyzed through the related computed P-value. When no anomaly is detected, the value of the P-value of a given test remains high, between 0.3 and 1.0. Otherwise, it decreases considerably and depending on the sensitivity of the test, it may attend low values. Our approach relies on the Benjamini-Hochberg procedure in order to combine the results of the various testing procedures and control the false discovery rate (FDR). Of course, the power of the global test of anomaly occurence increases as a larger number of test statistics are incorporated in the procedure.

C. Estimation of ChangePoint Distribution

1) ChangePoint Modeling: For changepoint detection, we use the ChangePoint Model method (CPM) proposed by Ross in [4]. It provides a statistical framework for changepoint detection that provides a set of functions to cover different runtime environments: one can choose between offline and online setting and can also select a parametric or a nonparametric testing approach. We are interested in the online nonparametric changepoint modeling that allows to identify changepoint in data sequentially even if the parameters or the distributional form of the observations are unknown. The sequential processing consists of applying sequentially the offline approach. The CPM method considers the problem of changepoint detection as a hypothesis testing one. Hence, under the null hypothesis, it assumes that no change point exists and the observations are independent and identically distributed according to some distribution F_0. If a change point exists at some time τ, then the observations have a distribution F_0 prior to this point, and a distribution F_1 afterwards, where $F_0 \neq F_1$. Since the moment where the changepoint occurs is not known in advance, the selected test statistics will be computed for each instance in the sequence and the maximum value will be used to make a decision. If the maximum value exceeds some appropriately choosen threshold, we conclude that a change point has occurred and the best estimate of the change point location will be immediately following the value which maximizes the test statistics (because the higher the value of the test statistic, the smaller the P-value). In our solution, we used the CPM framework associated with the Mann-Whitney test.

2) Changes in Poisson Distribution: The second component aims at monitoring the distribution of changepoints. We assume that the changepoints in the considered data stream follow a Poisson distribution that offers a great flexibility when it comes to model the law of a counting process and detect possible intensity changes. Initially, in the learning step, we store the average count of changepoints at a given timeslot I for each day of the week, in order to account for seasonality features. Thereafter, in the detection step, we will count the number of change points that occurred. The baseline idea of this function consists of testing how likely to get the observed number of change for a specific timeslot I in a given day of the week. We define a Poisson distribution where the parameter λ is estimated by the average count of changepoints at the same interval I as thatcomputed in the learning step. We use the survival function in order to get a P-value that, when compared to the preselected significance level, allows to decide whether an anomaly occurs or not.

IV. Global architecture

Despite the promising future of IoT, security and privacy still pose serious threats. Our algorithm can be implemented in distributed devices, in a middleware or in a centralized server. In many fields such as smart homes, sensors need to make decisions locally or communicate with external sources based on the data collected to improve decision-making. We propose to update the CPM package in the server with parameters learned in similar devices. An anomaly threshold learned in a device is broadcasted to other

devices having the same behavior. The data processing in our architecture (Fig. 1) involves two components: a local modeling from sensor data and a centralized correlation. Each local source calculates data statistics based on its local data, which is not sent over the network preserving privacy, learns its nominal function, adjusts threshold values based on user feedback, alerts in the event of anomaly detection and raises only statistics at the central site to obtain a global consolidated data modeling view and an inventory of detected anomalies in similar sensors. Our central server applies a clustering procedure to identify sensors exhibiting similar behavior. The consolidated model is, then, shared between all sensors in the same cluster, while detailed statistics are kept in the sensor, so as to preserve privacy.

V. Performance evaluation

A. Datasets Description

We used both a synthetic dataset and a real dataset to assess the performance of our algorithm.

Synthetic Dataset: We implemented a function that allows to generate different random simple and compound time series. The simple time series are formed by a random combination of n periodic signals ($1 \leq n \leq 5$). Whereas, the compound time series have n underlying intensity patterns. The resulted synthetic data represents a noisy periodic signal. We generated different types of anomalies:

- Mean shift: adding a constant to the selected period of time series to increase its amplitude.
- Variance shift: incorporating a white noise or adding a high-frequency oscillating signal.
- Constant readings: multiplying the time series period of time by zero and adding a little noise.
- Outliers: adding values that are much higher than the normal data values in specific points.
- Trend: applying a monotonic increasing linear function to the selected period.

Real Dataset: Data generated by 8 sensors deployed in a building in order to monitor the hourly electricity consumption. It covers a period of one year from January 28^{th} until December 30^{th}, 2015 in 8731 records. The data streams are not annotated, we visually inspected the time series and identified 5 types of long-duration and short-duration anomalies. We used this classification for result evaluation.

B. Experiments and Results

In order to evaluate the quality of the process, we used the area under the ROC curve criterion. For the synthetic data streams, we tested two experiment settings: injecting only the four first types of anomalies without adding a trend and the trend anomaly type. We plotted several ROC curves to evaluate the global performance of the algorithms and the performance by considering one type at a time.

In the synthetic dataset, the algorithm detects well the different anomaly types (see Fig. 2). Higher true positive rates for mean shift and variance shift detection are obtained when trend anomalies are not considered (Fig. 2, 3) .

The algorithm gives satisfying results when applied to the real dataset. We considered only the anomalies we annotated manually. Hence, some anomalies particularly outliers were difficult to detect or assimilated to concept drift.

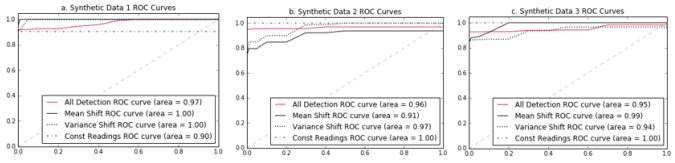


Figure 2. Synthetic Dataset Experiments Results: ROC Curves

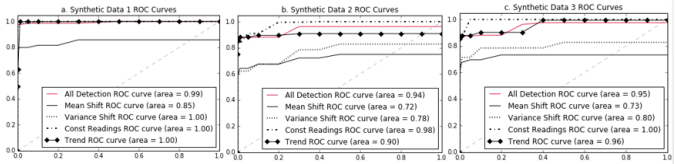


Figure 3. Synthetic Dataset Experiments Results (All Anomalies' Types): ROC Curves

VI. Conclusion

We presented a novel two-stage machine learning method for anomaly detection (AD) in IoT sensors datastreams. The first step consists in learning the normal behavior of the timestamped data, while the second one identifies deviations and update the AD engine. The detection process is based on two tasks: detecting anomalous sequences in the time series and monitoring the distribution of the changepoints by using changepoints modeling, hypothesis testing and false discovery rate control. Parameters and thresholds are updated with a global view calculated on sensors clusters in a central server. We applied this method to synthetic data and to real energy consumption data streams.

Acknowledgment

The authors would like to thank Teralab, DcBrain and the industrial chair "Machine Learning for Big Data".

References

[1] R. Killick and I. Eckley, "changepoint: An r package for changepoint analysis," *J. Statist. Soft.*, 2014.

[2] A. A. Gilad Wallach, Lev Faivishevsky, "Change and anomaly detection framework for internet of things data streams," *ICML Anomaly Detection Workshop*, 2016.

[3] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[4] G. J. Ross *et al.*, "Parametric and nonparametric sequential change detection in r: The cpm package," *J. Statist. Soft.*, 2013.

Anomaly-based Intrusion Detection of IoT Device Sensor Data using Provenance Graphs

Ebelechukwu Nwafor, Andre Campbell and Gedare Bloom
Department of Electrical Engineering and Computer Science
Howard University, Washington, DC 20059
Email: ebelechukwu.nwafor@bison.howard.edu

***Abstract*—The Internet of Things (IoT) has revolutionized the way humans interact with devices. Unfortunately, this technological revolution has been met with privacy and security challenges. One major concern is the notion of malicious intrusions. A common way of detecting a malicious intrusion is by treating an attack as an anomaly and using anomaly detection techniques to detect the source of intrusion. In a given IoT device, provenance, which denotes causality between system events, offers immense benefit for intrusion detection. Provenance provides a comprehensive history of activity performed on a system's data, which indirectly ensures device trust. In this paper, we propose an approach to intrusion detection of IoT devices that leverages sensor data flow as seen through provenance graphs. In this approach, an observed provenance graph is compared to an application's known provenance graph. This comparison involves the dimensionality reduction of a provenance graph to a vector space representation and similarity metric. We evaluated our approach on an IoT application which simulates a climate control system.**

***Index Terms*—Anomaly detection, Intrusion detection, Internet of Things, Data Provenance**

I. Introduction

IoT devices have become an essential part of our daily lives in commercial, industrial, and infrastructure systems. These devices offer benefits to consumers by interacting with the physical environment through sensors and actuators, which allows device automation thereby improving efficiency. Unfortunately, the proliferation of IoT devices has led to an increase in the number of remotely exploitable vulnerabilities leading to new attack vectors that could have disastrous financial and physical implications. In 2015, security researchers demonstrated a vulnerability on the Jeep vehicle which allowed remote control of the automotive system over the Internet [1]. In 2016, researchers discovered a vulnerability that allows Internet connected smart thermostats to be subject to remote ransomware attacks in which an attacker gains sole control of the thermostat until a fee is paid [2]. These are a few examples of some of the potential malicious vulnerabilities that could have devastating, long-lasting impact on an IoT system.

This material is based upon work supported by the National Science Foundation under Grant No. CNS-1646317 and the U.S. Department of Homeland Security under Grant Award Number 2017-ST-062-000003. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

Intrusion detection [3] is the process of discovering malicious exploits in a system. One way of detecting an intrusion is by the use of anomaly detection techniques. An anomaly, also referred to as an outlier, is data that deviates from the normal system behavior. Anomalies could be indicative of a system fault or that a system has been compromised by a malicious event. Due to the sensitive nature of safety critical systems, detecting malicious attacks is of utmost importance.

We propose an approach to identifying anomalous sensor events using provenance graphs. This approach involves the use of a similarity metric to compare observed provenance graphs with provenance graphs derived from an application's normal execution. The result is an anomaly score which is compared with a previously set threshold to classify an observed provenance graph as either anomalous or benign. We evaluate the effectiveness of our approach with a sample IoT application that simulates a climate control system.

II. Graph Similarity Anomaly Detection

A. Provenance Graphs

Provenance denotes the origin of an object and all other activities that occurred on that object. Data provenance, also known as data lineage, can be defined as the history of all activities performed on a data object from its creation to its current state. Provenance ensures data trust [4]. It establishes causality between entities contained in a data object through information flow tracking thereby it allows for the verification of a data source. Provenance is represented by a directed acyclic graph (DAG) in which vertices represent data entities and the edges correspond to the interaction between them.

Most provenance collection frameworks developed to track provenance use system level event sequences in an operating system [5]–[7]. For IoT devices, containing limited or no operating system functionality, it is essential to use a provenance collection framework that places less emphasis on an operating system and more emphasis on application level information flow tracking. For our provenance graph generation, we use PAIoT [8], a provenance collection framework which tracks the information flow of sensor-based events in an IoT device. In PAIoT, sensor data containing observation information is represented as a provenance graph. Sensor events are instrumented in the application source code. Each sensor data generated is defined as a tuple (t, e, a, s_1, r_1) where t represents the time a sensor reading was generated, e

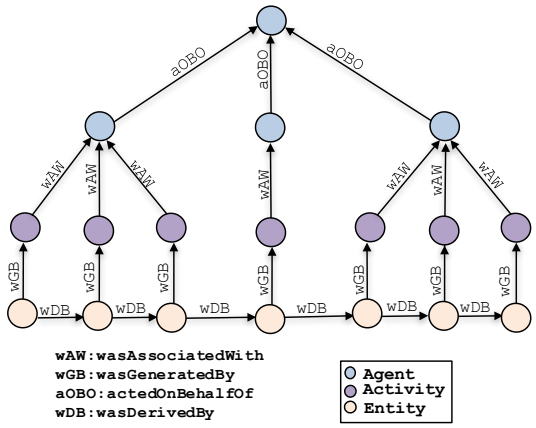


Fig. 1: Components of a provenance graph where nodes represent types (agent, activity, and entity) and edges represent relationships between types

represents the sensor data, a represents activity performed on sensor data, s_1 represents a sensor identifier, and r_1 represents the device information. Tuple information is constructed as a provenance graph and stored in a graph database (Neo4j) where further processing and query analytics can be performed on provenance data.

A provenance graph is a directed acyclic graph, $p = (V, E)$ where V is a set of vertices $V = \{v_1, ..., v_n\}$ such that $v_i = (type, value)$ and E is a set of edges $E = \{e_1, ..., e_n\}$ where $e_i = (v_s, v_d, label)$ and v_s, v_d are source and destination vertices. Two vertices v_x, v_y are equal (denoted $v_x = v_y$) if $v_x.type = v_y.type$ and $v_x.value = v_y.value$. Two edges e_x and e_y are equal (denoted $e_x = e_y$) if $e_x.v_s = e_y.v_s$, $e_x.v_d = e_y.v_d$, and $e_x.label = e_y.label$. We use the union operator $\cup$ over edge sets in the usual way of the union of sets.

Types may take on one of the following: Agent, Entity, and Activity. An agent is an actor that is represented by an identifier (e.g., sensor or device name). An entity is an event, which represents data that is produced as a result of an activity. An activity is an action performed by an agent or entity (e.g., read, update, create, delete). $label$ takes on one of the following values: *wasGeneratedBy*, *used*, *wasInformedBy*, *wasDerivedFrom*, *wasAttributedTo*, *wasAssociatedWith*, *actedOnBehalfOf*.

B. Graph Similarity

Similarity is a measure of how identical two objects are, for example, by measuring the angle between objects (using cosine similarity) or a linear distance (using euclidean distance) between the objects. In this work, we use cosine similarity as our similarity metric. This was inspired by the use of information retrieval techniques for query ranking. e.g., given a corpus $D = \{d_1, ..., d_n\}$, and query, q, how do we find document(s) $\{d_x, ...d_y\}$ which are similar to q and rank them by order of importance. Cosine similarity is a measure of orientation between two non-zero vectors. It measures the cosine of the angle between the vectors. Two vectors which are at an angle of 90° have a similarity of 0, two vectors which are identical (with an angle of 0°) have a cosine of 1, and two vectors which are completely opposite (with an angle of 180°) have a similarity of -1. Since we are concerned with the similarity between vectors, we are only concerned with the positive values bounded in [0,1]. The cosine similarity between two vectors, X and Y, is computed by:

$$\cos(\theta) = \frac{X \cdot Y}{\|\mathbf{X}\| \cdot \|\mathbf{Y}\|} = \frac{\sum_i^n X_i Y_i}{\sqrt{\sum_i^n X_i^2} \times \sqrt{\sum_i^n Y_i^2}}$$

In order to apply cosine similarity between provenance graphs, we compute a vector representation which reduces the graph into an n-dimensional vector space where n represents the total number of edges contained in the union of all edge sets. Figure 2 illustrates the vector space conversion of provenance graphs. $\boldsymbol{p_1}$, and $\boldsymbol{p_2}$ which consists of vertices $A, B, E, F, G, I, J, S, R$ and edge labels $aOBO, wAW, wGB, wDB$. The vector space representation of u_1 is the occurrence of edges contained in the edge set of graph p_1, which are also found in the collective union of edge sets. Algorithm 1 further outlines the concept of graph to vector conversion.

C. Anomaly Detection on Provenance Graphs

Anomaly detection involves the use of rule-based, statistical, clustering or classification techniques to determine normal or anomalous data instances. The process of determining all anomalous instances in a given dataset is a complex task. A major challenge in anomaly detection is providing the right feature set from the data to use for detection. Another challenge exists in defining what constitutes as normal system behavior. Most anomaly detection using point-based data often fail to include the dependencies that exist between data points.

Due to the ubiquitous nature of IoT devices, there are a wide array of vulnerabilities associated with them. In designing our anomaly detection framework, we expect an attacker's footprint is reflected through the data flow as depicted in the provenance graph. Our algorithm detects attacks such as false data injection, and state change as depicted in information flow of sensor events in provenance graphs.

Algorithm 1: Graph to vector conversion.

```
1:  procedure GRAPHTOVECTOR(E, E_G)
2:      n ← |E_G|
3:      Q[k], Q[i] ← 0, 0 ≤ i < n
4:      for e_j ∈ E do
5:          for e_g ∈ E_G | 0 ≤ g < n do
6:              if e_j = e_g then
7:                  Q[g] ← Q[g] + 1
8:              end if
9:          end for
10:     end for
11:     return Q
12: end procedure
```

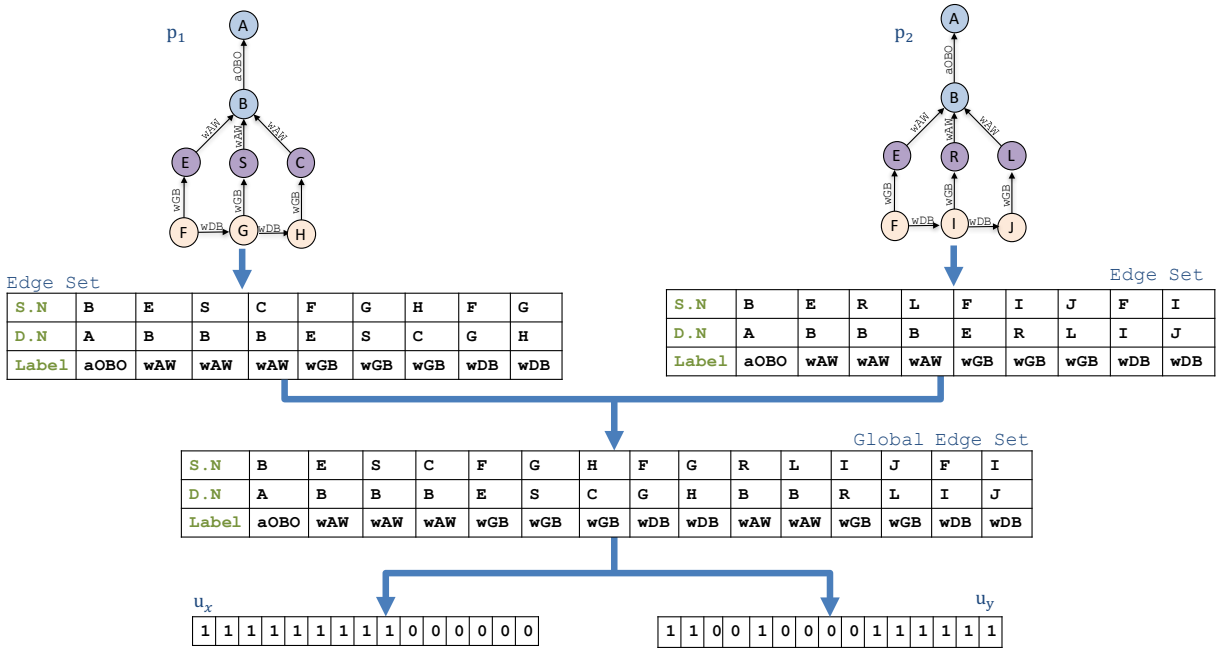


Fig. 2: Provenance graph conversion to vector space. u_x, u_y vectors generated from both provenance graphs.

Algorithm 2: Detection algorithm given an observation phase graph set, P, a detection phase graph, p, and a $threshold$.

procedure GRAPHANOMALY($P, p, threshold$)
  INPUT: $P = \{p_0, ..., p_n\} \mid p_i \leftarrow (V_i, E_i), 0 \leq i \leq n.$
  $E_G \leftarrow \cup_{i=0}^{n} E_i$
  $Q \leftarrow GraphtoVector(p, E_G)$
  $Z \leftarrow \{\}$
  for $p_i \in P$ **do**
    $N_i \leftarrow GraphtoVector(p_i, E_G)$
    $z \leftarrow Cosine\_Similarity(Q, N_i)$
    $Z \leftarrow Z \cup z_i$
  end for
  $s_{val} \leftarrow min(Z)$
  if $s_{val} \geq$ threshold **then**
    return normal
  end if
  return anomaly
end procedure

Many IoT devices implement a control systems in which sensor data is used as an input in a feedback loop to an actuator. The operations of most control systems are regular and predictable. For example, in a thermostat application, temperature readings generated might be converted from Celsius to Fahrenheit and utilized as feedback to an actuator. Each iteration of a control loop sequence generates a path in a provenance graph. This notion can be leveraged to define an expected provenance graph for each application.

The expected regularity of provenance graphs in IoT applications motivates a supervised learning approach to anomaly detection. This approach consists of two phases: observation phase, also known as the training phase, and the detection or test phase. In the observation phase, the system collects provenance data considered to be a representation of the normal system behavior. In the detection phase, the provenance graph set is compared with the provenance graph derived from subsequent observations to determine if an anomaly exists by measuring similarity between this graph and the provenance graph set. Note that provenance graphs from the observation and detection phase form a graph set. A global edge set, E_G represents the union of edge sets contained in a graph set. Algorithm 2 is the graph anomaly detection function given an observation phase graph set, P, and a detection phase graph, p. Z represents a list of the cosine scores from comparing each of the provenance graphs in the observation phase graph set with a detection phase graph. A function, $calculateAnomalyScore$ is used to determine the anomaly score, which is based on the minimum cosine similarity score of elements contained in Z.

III. EXPERIMENT

The experiment evaluation serves as a preliminary study to confirm the correctness of our theoretical approach in detecting anomalous instances between provenance graphs. We evaluate our intrusion detection algorithm by implementing an IoT application which simulates a climate control system. Climate control systems ensure a proper functional environment for people and machinery. Constant irregularities in temperature could have devastating effects on industrial machinery. This system consists primarily of a heating ventilation and air conditioning (HVAC) system which uses temperature and humidity data to regulate environmental conditions within a building. We utilize a publicly available dataset [9] which

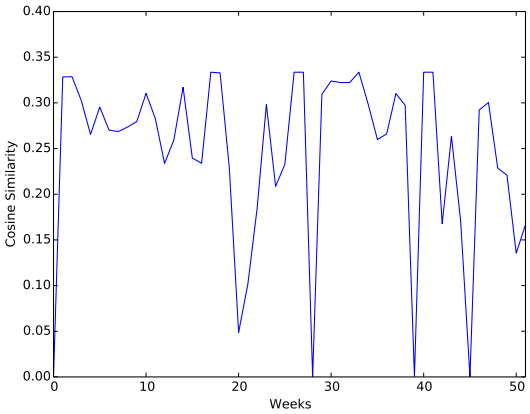


Fig. 3: Provenance graph comparison of climate control system by week.

consists of a year's worth of longitudinal data on the thermal conditions, related behaviors, and comfort of twenty-four occupants in a medium-sized building generated at a period of fifteen minutes. We utilize the temperature and humidity data as input to our simulation. We generate provenance graphs for each week of the year. We compare the provenance graph generated in consecutive weeks to see how they differ using our graph similarity approach (i.e., week 1 compared to week 2, week 2 compared to week 3 etc.).

Figure 3 depicts the cosine similarity between provenance graphs generated from the first occupant by preceding weeks. Ensuring a proper threshold score is used for detection is an important task that requires extensive knowledge of the application domain. The threshold is manually set to a value which is defined by domain experts. For automatic anomaly threshold detection, one can use prediction methods to define an anomaly score. As an example, a threshold might be set to 0.15 in which all of the scores below the threshold are considered anomalous. Since the dataset does not contain attacks, the declines shown in Figure 3 would likely cause false positives.

IV. Related Work

Liao et al [10] characterize a system's normal behavior by the frequency of unique system calls which are converted into a vector space representation. Stephanie et al. [11] define a human immune system inspired intrusion detection analyzing system call sequences. Yoon et al. [12] developed intrusion detection on embedded systems by analyzing system call frequencies clustered using k-means. Manzoor et al. [13] proposed a centroid based clustering anomaly detection for instances of streaming heterogeneous graphs in real time. Papadimitriou et al [14] proposed five similarity algorithms for comparing the similarity of web graphs. Xie et al. [15] proposed a provenance-aware intrusion detection and analysis (PIDAS) system using provenance graphs generated from system calls which reveals interactions between files and processes. Our approach differs from prior work because we focus on anomaly detection through information flow sequences of sensor data as represented by provenance graphs.

V. Summary and Conclusion

In this paper, we propose an anomaly detection algorithm for detecting anomalous instances of sensor based events in an IoT device using provenance graphs. We evaluate our approach with a preliminary study on an IoT application which simulates a climate control system. Current implementation of our anomaly detection algorithm works with offline data. Future work would include implementation for real-time detection. We also plan on conducting further experimentation to identify the false and true positive rates of our algorithm using select IoT application dataset.

References

[1] A. Greenberg, "The jeep hackers are back to prove car hacking can get much worse," 2015. [Online]. Available: https://www.wired.com/2016/08/jeep-hackers-return-high-speed-steering-acceleration-hacks/

[2] D. Raywood, "Defcon: Thermostat control hacked to host ransomware," 2016. [Online]. Available: https://www.infosecurity-magazine.com/news/defcon-thermostat-control-hacked/

[3] A. Lazarevic, V. Kumar, and J. Srivastava, *Intrusion Detection: A Survey*. Boston, MA: Springer US, 2005, pp. 19–78.

[4] E. Bertino, *Data Trustworthiness—Approaches and Research Challenges*. Cham: Springer International Publishing, 2015, pp. 17–25.

[5] T. Pasquier, X. Han, M. Goldstein, T. Moyer, D. Eyers, M. Seltzer, and J. Bacon, "Practical whole-system provenance capture," in *Symposium on Cloud Computing (SoCC'17)*, ACM. ACM, 2017.

[6] R. H. Zakon, Ed., *28th Annual Computer Security Applications Conference, ACSAC 2012, Orlando, FL, USA, 3-7 December 2012*. ACM, 2012.

[7] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. Seltzer, "Provenance-aware storage systems," in *Proceedings of the Annual Conference on USENIX '06 Annual Technical Conference*, ser. ATEC '06. Berkeley, CA, USA: USENIX Association, 2006, pp. 4–4.

[8] E. Nwafor, D. Hill, A. Campbell, and G. Bloom, "Towards a provenance aware framework for internet of things devices," in *Proceedings of the 14th International Conference on Ubiquitous Intelligence and Computing*, ser. UIC '17. San Fransisco, CA, USA: IEEE Computer Society, 2017.

[9] J. Langevin, P. L. Gurian, and J. Wen, "Tracking the human-building interaction: A longitudinal field study of occupant behavior in air-conditioned offices," *Journal of Environmental Psychology*, vol. 42, no. Supplement C, pp. 94 – 115, 2015.

[10] Y. Liao and V. R. Vemuri, "Using Text Categorization Techniques for Intrusion Detection," in *Proceedings of the 11th USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, 2002, pp. 51–59.

[11] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *J. Comput. Secur.*, vol. 6, no. 3, pp. 151–180, Aug. 1998.

[12] M.-K. Yoon, S. Mohan, J. Choi, M. Christodorescu, and L. Sha, "Learning execution contexts from system call distribution for anomaly detection in smart embedded system," in *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, ser. IoTDI '17. New York, NY, USA: ACM, 2017, pp. 191–196.

[13] E. Manzoor, S. M. Milajerdi, and L. Akoglu, "Fast Memory-efficient Anomaly Detection in Streaming Heterogeneous Graphs," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 1035–1044.

[14] P. Papadimitriou, A. Dasdan, and H. Garcia-Molina, "Web graph similarity for anomaly detection," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 19–30, May 2010.

[15] Y. Xie, D. Feng, Z. Tan, and J. Zhou, "Unifying intrusion detection and forensic analysis via provenance awareness," *Future Gener. Comput. Syst.*, vol. 61, no. C, pp. 26–36, Aug. 2016.

SIOTOME: An Edge-ISP Collaborative Architecture for IoT Security

Hamed Haddadi*, Vassilis Christophides†, Renata Teixeira†, Kenjiro Cho‡, Shigeya Suzuki§, Adrian Perrig¶
*Imperial College London
†Inria Paris
‡IIJ Research Lab
§Keio University
¶ETH Zurich

***Abstract*—Modern households are deploying Internet of Things (IoT) devices at a fast pace. The heterogeneity of these devices, which range from low-end sensors to smart TVs, make securing home IoT particularly challenging. To make matters worse, many consumer-IoT devices are hard or impossible to secure because device manufacturers fail to adopt security best practices (e.g., regular software patches). In this paper we propose a novel, cooperative system between the home gateway and the Internet Service Provider (ISP) to provide data driven security solutions for detecting and isolating IoT security attacks. Our approach is based on a combination of a large-scale view from the ISP (using powerful machine learning techniques on traffic traces), and the fine-grained view of the per-device activity from the home (using edge processing techniques) to provide efficient, yet privacy-aware IoT security services.**

I. Introduction

Today we are observing an increasing rate in the introduction of connected, Internet of Things (IoT) [9] devices in our everyday life.[1] Homes and public buildings and spaces (e.g., campuses, pedestrian zones, airports) are increasingly instrumented with a variety of IoT devices that can interact with each other and/or be remotely monitored and controlled. These devices range from voice-enabled personal assistants, entertainment systems, health and well-being monitoring devices (i.e., quantified self), home automation (i.e., smart plugs and pet doors) and connected appliances, as well as monitoring equipment such as light, temperature, and humidity sensors, cameras, and motion detectors. As IoT devices are typically embedded inside the networks (i.e., continuously interacting using primary local and third party cloud-based services), they are attractive attack targets for breaking into a secure network infrastructure [6], [20], or for leaking sensitive information about users and their behaviors [2], [11], [10], [19].

The rapid development of the consumer IoT sector and the focus on time-to-market has been generally at the sacrifice of privacy and security. Many of the current devices remain vulnerable to attacks, do not receive regular updates without user intervention, or use insecure communication methods such as telnet[2] or HTTP-based communication. Often, device vendors and manufacturers may be unable or unwilling to release software updates that address vulnerabilities.[3] A study identified more than 500,000 insecure, publicly accessible embedded networked devices [18]. Vulnerable IoT devices make home networks open to attacks or privacy leaks and make the Internet subject to large-scale Distributed Denial of Service (DDoS) attacks such as the Dyn Attack by the Mirai botnet.[4][5] Providing security for the consumer IoT market will be a big challenge in the next decade.[6]

Traditional network security solutions combining *static perimeter network defenses* (e.g., firewalls and intrusion detection/prevention systems), with ubiquitous use of end-host based defenses (e.g., antivirus), and software patches from vendors (e.g., Patch Tuesday) [20] are challenged by the dynamic landscape of the IoT threats and the *technical skills* required by the end-users for maintaining secure IoT devices. An operation deep inside the network renders traditional perimeter defenses ineffective while the longevity of IoT devices implies that despite IT security best practices, several vulnerabilities (e.g., default passwords, unpatched bugs) will remain deployed long after vendors cease to produce or support them. Moreover, devices can be moved between private, communal, or public spaces. Given overlapping wireless connectivity within or across spaces, it became easier for a device on one network to inadvertently or maliciously breach the security and mismanage another device on another overlapping network [1]. The existing rule-based security measures cannot cope with unpredictability in ever-changing traffic behaviors, as IoT interactions are evolving with increasing complexity. Last but not least, end-users in the consumer IoT space often lack access to a technically skilled network administrator [8]. As the number of devices increase (even within a household), the traditional firewall and port-based monitoring approaches will

[1] https://www.forbes.com/sites/louiscolumbus/2017/01/29/internet-of-things-market-to-reach-267b-by-2020/

[2] https://arstechnica.com/information-technology/2017/08/leak-of-1700-valid-passwords-could-make-the-iot-mess-much-worse/

[3] http://krebsonsecurity.com/2016/02/iot-reality-smart-devices-dumb-defaults/

[4] https://www.schneier.com/blog/archives/2016/11/lessons_from_th_5.html

[5] https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html

[6] http://www.gartner.com/smarterwithgartner/navigating-the-security-landscape-in-the-iot-era/

1st International Workshop on Security and Privacy for the Internet-of-Things (IoTSec)
17 April 2018
http://dx.doi.org/XYZ

not be effective at mitigating the threats, while enabling the range of services and applications in the IoT ecosystem to operate flawlessly.

The complex inter-dependancies of the IoT ecosystem force the network to (re)emerge as the key vantage point for enforcing security policies [20]. Network-based security solutions are better suited to the scale of deployed IoT devices, the nature of Machine-to-Machine (M2M) communication, the sheer diversity of the device hardware, as well as interoperability constraints (e.g., devices of the same type but from different vendors cannot always communicate) [19]. We thus propose to move the responsibility of securing consumer IoT devices from users to a *collaborative system between the network edge and the Internet Service Provider* (ISP).

In this context, we propose a security system that learns and adapts to the changing environment, and reacts to unexpected events in a quick and autonomous manner, by means of collecting data, performing analytics, creating network access rules, and controlling traffic accordingly for defense. As dynamic IoT threat detection requires a close view on traffic from the users' devices, we should employ security analytics methods that guarantee privacy of raw users' data. Furthermore, we need to develop mechanisms that protect against a wide range of network-based attacks such as vulnerability scanning, intrusion attacks, network eavesdropping, data alteration, as well as Denial-of-Service (DoS) attacks. Given that network conditions and device behaviors can change rapidly, we need to continuously reassess and update the IoT security posture. Our overarching goal here is to create a protection system that enables secure operation of an IoT deployment despite potentially vulnerable or even compromised IoT devices.

In this paper we present SIOTOME,[7] a cooperative architecture between the edge network and the ISP for early detection and mitigation of security vulnerabilities and threats due to IoT device misconfigurations and malicious attacks. In SIOTOME, instead of trying to secure an increasing number of heterogeneous devices, we focus on securing the network connecting them. With no communication, malicious devices cannot compromise other devices or launch attacks. We propose to design, develop, and evaluate a system that relies on the *cooperation among defense mechanisms deployed at multiple layers at the network edge*: the cloud, the ISP, and the home gateway. We advocate a *data-driven* approach to detect and isolate security threats based on a combination of large-scale view from the cloud (using machine learning techniques on traffic traces) and the fine-grained view of the per-device activity from within the home. To mitigate security breaches, SIOTOME relies on a set of defense mechanisms, for example, *network isolation* for limiting the attack surface, *key management* approaches to establish cryptographic keys between devices to provide communication secrecy and authenticity, and *allowed network input and output* to prevent vulnerability scanning and DDoS.

We assume that home users are independent and autonomous for protecting user's privacy. Thus, the ISP does not know the details of devices deployed in the home. An obvious case that requires cooperation between the edge and the ISP is when identifying an infected device in the home. When the ISP detects a suspicious communication originated from a certain home, the ISP can only tell the home gateway about the threat information and its signature for detection. Then, it is the home gateway that identifies the device using the provided signature, and notifies the user along with augmented device information such as model and installation date extracted from the home-internal device registry.

[7]The word "siotome" is taken from a Japanese medieval word, a water pool for protecting city canals from tidal influences.

II. System Framework

Building up on existing network-based intrusion detection and security systems, which focus on defending a single domain/service with rule-based approaches, SIOTOME leverages data from a large number of domains to learn from the environment to identify attack signals so that it can react more quickly and effectively to emerging threats. A domain in SIOTOME can be an individual home network, a cloud provider, or an individual ISP network (the traditional definition of Autonomous System in Internet routing). As shown in Fig. 1, SIOTOME has two high-level types of domains: SIOTOME/edge and SIOTOME/cloud.

Inside the user's home, we find the following components:

- The home gateway provides network connectivity to the access ISP and enforces local connectivity under the control of the home controller.
- The edge data collector is responsible to observe network traffic to monitor the behavior of IoT devices. It can also run active probing tests to profile devices. This home collector can be hosted in the home gateway or in a separate device that is directly connected to the gateway.
- The edge analyzer takes information from the home data collector to profile the behavior of local IoT devices and identify threats and attacks. Upon the detection of a threat, it will notify the home controller. It also shares relevant information with SIOTOME/cloud after applying privacy-preserving data modifications.
- The edge controller is responsible for configuring the home gateway to steer local network traffic: among devices in the home as well as with the outside world. The edge controller contains an SDN controller for fine-gained control of network traffic from/to connected devices in the home, as well as additional management functions. Examples of management functions include simple mechanisms to create small groups of devices (similar to Virtual LANs but more convenient for IoT devices and easier to handle for users), and also all classical control and management functions in the home, e.g., DHCP, firewall, user management. It is responsible for applying specific countermeasures to protect users' security and privacy. The home controller functionality can also be offloaded to SIOTOME/cloud when needed as equipment in user's homes may have limited resources.

SIOTOME/cloud hosts the following components of the system:

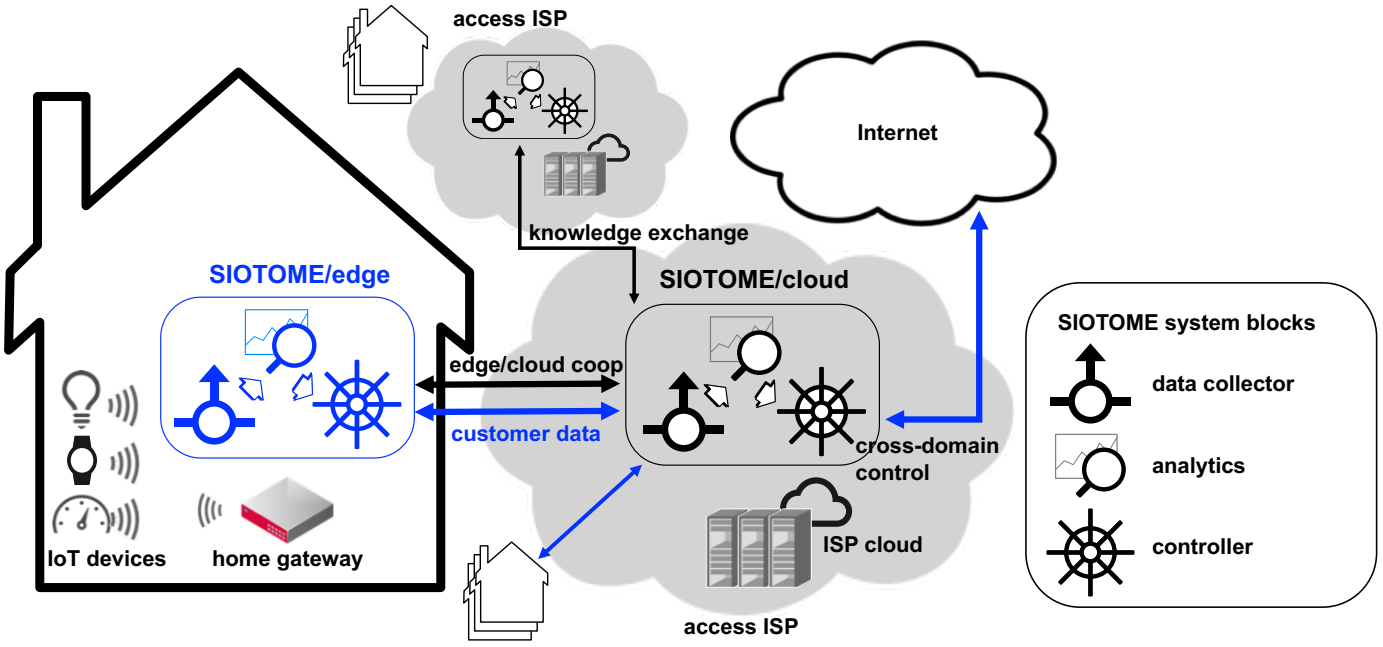


Fig. 1. SIOTOME Architecture and System Components

- The Cloud collector is the software system that collects reports from home collectors as well as performs additional monitoring at the ISP level (when SIOTOME/cloud is running in the ISP), so it can observe malicious patterns that span several customers.
- The Cloud analyzer is similar to the edge analyzer in that it analyses network traffic to identify threats. The methods running in the cloud analyzer benefit from the large volume of data coming from multiple homes and ISP traffic. It is responsible for collecting the device profiles learned across homes into a central database as well as for populating this database with signatures of attacks it discovers or learns from edge analyzers.
- The Cloud controller is an SDN controller that can steer local network traffic at the ISP level and trigger countermeasures to the threats identified by the cloud analyzer.
- The cross-domain controller steers traffic between domains. It can make a destination reachable from only a subset of sources or ensure that outgoing traffic stays within a selected network region.
- The secure communication component maintains secure communication between various SIOTOME components.

SIOTOME allows for *delegating parts of such security functionality from the cloud to the edge*, enabled by a common framework called SIOTOME/cloud and SIOTOME/edge. It aims to balance local learning/defense and global learning/defense, and to quickly propagate detected threat information among users. The SIOTOME/edge in a user's home adapts to individual user environments, and provides front-end defense mechanisms close to IoT devices. It also *preserves user privacy* by processing sensitive data locally without exposing them to a third-party [5]. We rely on the home gateway architectures such as the Databox system [14], where privacy-preserving IoT and sensor data analytics can be performed using containerized libraries and isolated data sources, while minimizing the risk of sensitive inferences from third parties and the ISP [13], [12]. Collaborative and hybrid machine learning frameworks have recently been developed, leveraging edge processing to aid in preserving privacy, and increasing the resource efficiency of IoT systems [7], [16].

The SIOTOME/cloud in the access ISP has a more global view by collecting and analyzing data from a large number of customers, as well as exchanging knowledge information with SIOTOME/clouds in other ISPs. It also provides back-end defense mechanisms for isolating individual customers and for cross-domain communications. The SIOTOME/cloud and SIOTOME/edge can run the same set of security primitives, although the edge has only limited resources. A specific security service is composed by chaining security primitives; each security primitive can be dynamically created, deleted, or migrated between the SIOTOME/cloud and the SIOTOME/edge.

Finally, SIOTOME makes extensive use of *network slicing* for isolating IoT device communications; devices are grouped by attributes and observed behaviors, and then, assigned to a network slice with a specific security policy. SIOTOME will rely on intra- and cross-domain network environments that only permit approved network communications, which we call *permissioned network input and output*. Intra-domain mechanisms will rely on a technique called SDN-based home network steering that whitelists communication between groups of devices and devices and external entities (i.e., websites) in network-isolated slices, leveraging the Majord'Home platform [3], [4]. For cross-domain mechanisms, we plan to leverage the SCION secure Internet architecture [17], an

inter-domain architecture that provides source-controlled path selection, multipath operation, and DoS defenses. For intra-domain, cross-domain and edge-to-cloud coordination, secure communucation mechanism is essential. SIOTOME plan to make use of blockchain as a secure broadcast channel based on [15].

III. Future Direction

In this paper we proposed SIOTOME, an architecture for a collaborative, privacy-preserving analytics architecture between the edge of the network and an ISP, to provide a first step security defense against distributed attacks by compromised IoT devices. As the first step towards realizing this vision, we are evaluating the interactive behavior of a number of IoT devices to advance our understanding of IoT security threats in the wild. Understanding these interactions and network utilization profiles allows us to train machine learning models and establish optimal operational configurations between the edge and the cloud.

SIOTOME is inspired by the vision to make IoT security analysis, threat detection, and defenses intuitive and effective for the non-expert users at the home environment. Our ambition is to build a service where data and inferences from the edge are combined with the insights gained from the cloud to provide a coherent system for early detection of security threats and take autonomous action, and consequently alert the user and the ISP. Most importantly, privacy-preserving inferences of the normal device behavior and network characteristics, and cooperative sharing of this knowledge in combination with the ISP traffic characterization, allows SIOTOME to monitor an IoT network, providing user security and privacy, despite potentially vulnerable or compromised devices.

Acknowledgment

The authors would like to thank Felipe Huici, Matthieu Boussard, Nicolas Le Sauze, Ludovic Noirie, Romain Fontugne, Kensuke Fukuda, Ping Du, Aki Nakao, and Nick Feamster for their extensive discussions on SIOTOME. We also appreciate constructive feedback from the anonymous reviewers at IoTSec 2018. Hamed Haddadi was supported by the EPSRC Databox grant (Ref: EP/N028260/1) and a Microsoft Azure for Research grant.

References

[1] W. Aman, "Assessing the feasibility of adaptive security models for the internet of things," in *Human Aspects of Information Security, Privacy, and Trust*, T. Tryfonas, Ed. Cham: Springer International Publishing, 2016, pp. 201–211.

[2] N. Apthorpe, D. Reisman, S. Sundaresan, A. Narayanan, and N. Feamster, "Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic," *CoRR*, vol. abs/1708.05044, 2017. [Online]. Available: http://arxiv.org/abs/1708.05044

[3] S. Bera, S. Misra, and A. V. Vasilakos, "Software-defined networking for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1994–2008, Dec 2017.

[4] D. T. Bui, R. Douville, and M. Boussard, "Supporting multicast and broadcast traffic for groups of connected devices," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, June 2016, pp. 48–52.

[5] A. Chaudhry, J. Crowcroft, H. Howard, A. Madhavapeddy, R. Mortier, H. Haddadi, and D. McAuley, "Personal data: Thinking inside the box," in *Proceedings of The Fifth Decennial Aarhus Conference on Critical Alternatives*, ser. AA '15. Aarhus University Press, 2015, pp. 29–32. [Online]. Available: http://dx.doi.org/10.7146/aahcc.v1i1.21312

[6] E. Fernandes, A. Rahmati, K. Eykholt, and A. Prakash, "Internet of things security research: A rehash of old ideas or new intellectual challenges?" *IEEE Security Privacy*, vol. 15, no. 4, pp. 79–84, 2017.

[7] L. Georgopoulos and M. Hasler, "Distributed machine learning in networks by consensus," *Neurocomputing*, vol. 124, pp. 2 – 12, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231213003639

[8] R. E. Grinter, W. K. Edwards, M. Chetty, E. S. Poole, J.-Y. Sung, J. Yang, A. Crabtree, P. Tolmie, T. Rodden, C. Greenhalgh, and S. Benford, "The ins and outs of home networking: The case for useful and usable domestic networking," *ACM Trans. Comput.-Hum. Interact.*, vol. 16, no. 2, pp. 8:1–8:28, Jun. 2009. [Online]. Available: http://doi.acm.org/10.1145/1534903.1534905

[9] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[10] M. Krämer, D. Aspinall, and M. Wolters, "Poster: Weighing in ehealth security," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 1832–1834. [Online]. Available: http://doi.acm.org/10.1145/2976749.2989044

[11] D. Leibenger, F. Möllers, A. Petrlic, R. Petrlic, and C. Sorge, "Privacy challenges in the quantified self movement - an EU perspective," *PoPETs*, vol. 2016, no. 4, pp. 315–334, 2016. [Online]. Available: https://doi.org/10.1515/popets-2016-0042

[12] M. Malekzadeh, R. G. Clegg, A. Cavallaro, and H. Haddadi, "Protecting sensory data against sensitive inferences," *the 1st EuroSys Workshop on Privacy by Design in Distributed Systems*, 2018.

[13] M. Malekzadeh, R. G. Clegg, and H. Haddadi, "Replacement autoencoder: A privacy-preserving algorithm for sensory data analysis," *The 3rd ACM/IEEE International Conference on Internet-of-Things Design and Implementation*, 2018.

[14] R. Mortier, J. Zhao, J. Crowcroft, L. Wang, Q. Li, H. Haddadi, Y. Amar, A. Crabtree, J. Colley, T. Lodge, T. Brown, D. McAuley, and C. Greenhalgh, "Personal data management with the databox: What's inside the box?" in *Proceedings of the 2016 ACM Workshop on Cloud-Assisted Networking*, ser. CAN '16. New York, NY, USA: ACM, 2016, pp. 49–54. [Online]. Available: http://doi.acm.org/10.1145/3010079.3010082

[15] J. Murai and S. Suzuki, "Blockchain as an audit-able communication channel," in *Computer Software and Applications Conference (COMPSAC), 2017 IEEE 41st Annual*, July 2017, pp. 516–522.

[16] S. A. Osia, A. S. Shamsabadi, A. Taheri, H. R. Rabiee, N. Lane, and H. Haddadi, "A hybrid deep learning architecture for privacy-preserving mobile analytics," *arXiv preprint arXiv:1703.02952*, 2017.

[17] A. Perrig, P. Szalachowski, R. M. Reischuk, and L. Chuat, *The SCION Architecture*. Cham: Springer International Publishing, 2017, pp. 17–42. [Online]. Available: https://doi.org/10.1007/978-3-319-67080-5_2

[18] A. Sivanathan, D. Sherratt, H. H. Gharakheili, V. Sivaraman, and A. Vishwanath, "Low-cost flow-based security solutions for smart-home iot devices," in *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Nov 2016, pp. 1–6.

[19] V. Sivaraman, H. H. Gharakheili, A. Vishwanath, R. Boreli, and O. Mehani, "Network-level security and privacy control for smart-home iot devices," in *WiMob*. IEEE Computer Society, 2015, pp. 163–167.

[20] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu, "Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things," in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XIV. New York, NY, USA: ACM, 2015, pp. 5:1–5:7. [Online]. Available: http://doi.acm.org/10.1145/2834050.2834095

Butterfly Effect: Causality from Chaos in the IoT

Ioannis Agadakos*, Gabriela Ciocarlie†, Bogdan Copos‡, Tancrède Lepoint†, Ulf Lindqvist‡, and Michael Locasto†
* Stevens Institute of Technology, Hoboken, New Jersey
iagadako@stevens.edu
† SRI International, New York, New York
{gabriela.ciocarlie,tancrede.lepoint,michael.locasto}@sri.com
‡ SRI International, Menlo Park, California
{bogdan.copos,ulf.lindqvist}@sri.com

***Abstract*—The interconnection of devices into networked services and applications is collectively called the Internet of Things (IoT). Significant efficiencies are gained from this ecosystem that automates many aspects of everyday life. Unfortunately, this automation often occludes the understanding of causes and effects of actions taken in an IoT network (including those of rogue devices). As we observe the effects of a particular device's actions on its environment (e.g., the light turns on), we want to capture the full causality path to the originating device. This paper describes a promising approach that enables one to infer a potential sequence of actions (or any deviation thereof) that led to a particular observation.**

I. Introduction

The automation underlying the Internet of Things aims to make many aspects of our lives easier or more efficient. Smarter sensing, control, and actuation help decreasing the need for physical presence or action of individuals even in their own home. From a computational perspective, devices become smarter by using distributed, yet external computing engines commonly referred to as "the Cloud". As a run-through example in this paper, we consider a simple smart-home setting containing personal assistants, connected lightbulbs, and smart hubs—see Fig. 1.

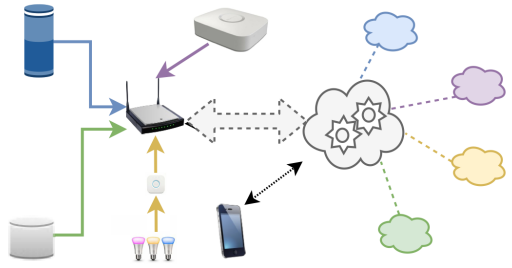
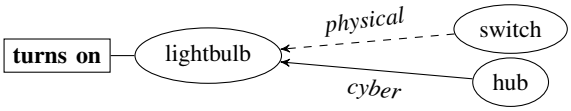


Fig. 1. A simple smart home that includes two personal assistants (an Amazon Echo and a Google Home), a connected lightbulb and its hub (Philips Hue), itself connected to a smart home hub (Samsung SmartThings), and a router. The Philips Hue application is also installed on a smartphone that uses a cellular network. Different colors encode that each smart device has a different Cloud end-point.

As lights turn on in this smart home, we want to understand what sequence of actions led to the activation of the connected lightbulb, and whether everything happened "as planned". The research question that we focus on in this paper therefore becomes: *Can we identify the cause or causes of an effect observed in an IoT device? Were they anomalous?* While the IoT is often described as a network of interconnected devices, the reality is quite different: most IoT devices are connected directly (or through a hub) to the router, and interactions between devices happen after network packets from Cloud end-points are received. For example, when one asks Echo to turn the lights on, the voice stream is encrypted and sent to an Amazon end-point, and a few moments later, a network packet emanates from a Philips end-point and goes to the hub; the hub then sends a network packet to the lightbulb, which eventually turns the lights on.

To answer the above question, this paper proposes a methodology of causal inference in an IoT setting. As an action is observed on an IoT device, we propose to automatically investigate the potential sequences of causes that led to this particular action. The first step of the methodology identifies every potential source of cyber and physical interactions with the device. For every such source, the second step is to inspect its recent behavior against pre-existing models of behaviors associated to the final action. These two steps are iterated to construct a tree, where leafs are potential causes. Using our running example, as the lights turns on, we inspect the two potential interaction paths with the lightbulb: the physical path via a physical switch, or the cyber path via the hub.



The recent behavior of each input source (i.e., the switch and the hub) is inspected and compared against models of predefined behaviors. For example, it will be checked whether recent network packets have been issued from the hub and directed to the lightbulb. When no predefined behavior is known or when the input source cannot be observed (e.g., the switch), a generic root cause "change" may be associated to the input sources. By bootstrapping the previous approach and considering multiple expected behaviors (from physical sources such as side-channels or cyber sources such as network traffic), it may be possible to identify the root cause (or causes) that led to an effect, such as "a voice command to Amazon

Echo caused the lightbulb to turn on".

There are several approaches to applying provenance analysis to the IoT domain. Previous efforts [1]–[3] instrument IoT devices to generate provenance records. The approach presented in this paper lies on the opposite end of the spectrum and is designed to work in uncooperative IoT environments. As such, our black-box approach relies on learning device behavior by analyzing readily available information streams such as network communications and side channel modalities. By analyzing omnipresent evidence, our analysis can easily support new behaviors while instrumentation needs to be reapplied with every firmware update. The lack of instrumentation also implies that there is no overhead imposed on the IoT devices and network. On the other hand, the completeness of the learned behavior models depends heavily on extensive exercising of the devices.

Significant challenges arise along the way, such as modeling an IoT environment and its cyber and physical interactions paths, or measuring and learning expected behaviors. To model the IoT environment and its cyber-physical interactions, we leverage an approach proposed by Agadakos et al. [4] that builds a "pessimistic assumption" model, where device interactions are defined by the (hardware) capabilities of the devices and are not limited by the software stack driving the devices and their interactions. We review this modeling technique in Section II-A. As for expected behaviors, we define an expected behavior as a vector of physical and cyber measurements computed when an event is observed. In particular, our expected behavior captures side-channels, such as power consumption and electromagnetic emanations, and network traffic fingerprints. We provide specific methods for measuring such expected behaviors in Section II-B.

In the rest of this paper, we elaborate on our methodology and report early experimentation results in the setting described in Fig. 1 to motivate its viability in Section III. We discuss limitations and future work in Section IV.

II. Causal Inference Methodology

Our causal inference is comprised of three stages, described below: (i) mapping of the IoT environment to identify devices and potential interactions, (ii) learning of expected behavior, and (iii) detection of causality.

A. Mapping the IoT Environment

The first step towards identifying causal inference is identifying and characterizing the devices available in the IoT environment. To achieve this, we rely on the approach described in [4]. The approach leverages sniffers that passively monitor network transmissions between devices across several network technologies (e.g., Wi-Fi, Bluetooth, ZigBee). Once the observed traffic is analyzed and devices are fingerprinted, a "pessimistic assumption" model of the cyber and physical channels between devices is generated. The model identifies potential interaction paths between devices based on their capabilities. The model is "pessimistic" in that the feasibility of an interaction path is basely solely on devices' capabilities and not other properties (e.g., physical obstacles, software constraints). This assumption provides us the flexibility required to reason about a variety of threat models including attackers with various capabilities and worst-case scenarios.

The interaction model supports our causality analysis by reducing the number of evidence sources to be analyzed. Given the observed behavior of a device, the interaction model indicates which devices should be considered for the causality analysis and which can be safely ignored (since it is improbable they impacted the current device). This is particularly important for interactions over physical channels which are difficult to trace to an origin.

B. Measuring Expected Behaviors

The model of cyber and physical interactions of the IoT devices depicts the world of possible interaction paths between devices according to their capabilities. However, the model is a hypothetical representation and fails to capture information about what interactions are actually occurring. To capture live information regarding interactions, our approach leverages network traffic communications and side-channel information.

We passively monitor network traffic and extract patterns of network communications between devices across multiple network technologies (e.g., Wi-Fi, Bluetooth, Zigbee). Information regarding protocols, size, and format of packets exchanged is recorded. However, network traffic only covers a subset of interactions (i.e., cyber interactions over supported technologies) and discloses limited information about the impact of communications on device behavior. To account for these limitations, our approach also relies on side channel analysis. Side channels such as electromagnetic emanations and power consumption provide additional insight into device behavior, especially regarding interactions over physical channels (that may be otherwise difficult to identify). Time-series side channel modalities for each device are processed using signal processing (both time and frequency domain features) along with machine learning techniques to identify and differentiate between patterns of behavior. This approach enables flexibility in the side channels considered for each device. While some side channels (e.g., power) require instrumentation, others (e.g., acoustics, EM) do not, increasing the applicability of the proposed approach.

Currently, our approach learns the expected behavior over repeated observations. An alternative solution is to leverage an active probing approach that stimulates the devices and triggers various behavior. While this approach is more involved, it achieves higher confidence and can better determine relationships between input and device behavior.

Behavioral patterns from network traffic, side channel modalities, and physical environment sensors are combined to create representations of expected behaviors for each device. Using these patterns, we construct a tree-based model of the expected behaviors. The nodes of the tree represent events, with leafs indicating potential causes. The edges of the model connect events and depict temporal order of the events. In the future, we wish to explore the use of Dynamic Bayesian

Network (DBN) [5] and Probabilistic Suffix Tree (PST) [6] as the underlying predictive model, as seen in [7].

C. Detecting Causality

Given the model of the potential IoT interactions and the learned expected behavior model, it is possible to trace the causality of an event to the original cause by traversing the tree model. The analysis begins at the root of the tree with the device that triggered the event and its observed behavior. Starting at the root, the approach performs two recursive steps: (i) compare observed behavior with expected behavior and (ii) identify potential sources of input and analyze. From there, the IoT network model is used to determine which devices may have impacted the triggered device. The observed behavior for those devices is compared against the expected behavior. This process is repeated until either there are no unvisited neighboring nodes or a leaf node in the expected behavior model was reached. The last node(s) traversed represents the cause(s) that triggered the event. When no predefined behavior is known, a generic root cause "change" is associated with the input. If the input source behavior does not match a predefined one, an alert is raised as the occurrence may be indicative of malicious activity. A similar approach can be used to perform forward-search and identify subsequent events associated with a particular event.

III. Experimentation and Early Results

A. Test Bed

We test our approach in an IoT network comprised of an Amazon Echo, a Google Home, a Samsung SmartThings Hub, a Philips Hue bridge and an Iris light, and an Android smartphone device. The network and its topology are depicted in Figure 1. All devices are installed according to their manual instructions and run native software. To capture network traffic, we install an open-source OpenWRT-based router firmware on a commodity wireless router. We collect two types of side channel modalities; power is collected using a low cost off-the-shelf OORT BLE smartplug, while EM emanations are measured using the HackRF software defined radio. This simple IoT network example is sufficient to show that the chain of events and the infrastructure at play are far from trivial.

B. Power Analyses

We perform several experiments while measuring power consumption of the Amazon Echo and Google Home devices, independently. As shown in Figure 2, devices exhibit different patterns in their power consumption in response to various voice commands.

C. Electromagnetic Emanations

In our experiments, we also explore the feasibility of relying on electromagnetic emanations. Using the HackRF software-defined radio, we measure the electromagnetic emanations generated by the Philips Hue bridge. The raw signal is split into windows. For each window, we compute the power spectral density (PSD) to determine the frequency components of the signal. These frequencies compose the features that help us differentiate between behavior phases.

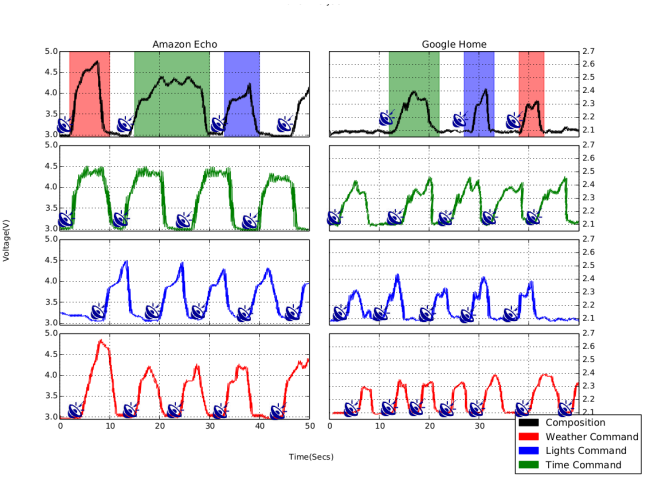


Fig. 2. Power Analysis for Amazon Echo (1st column) and Google Home (2nd column) over three verbal commands. First row contains an example run with all three tested commands in the same experiment, the commands are easily deciphered from each power consumption signature. All commands given over a text2speech timed script. Note that Google Home responds to commands faster than Amazon Echo, hence we observe more commands/responses in the same time interval.

D. Scenarios

We evaluate our approach using two scenarios.

a) Scenario 1: Determining causality source: The first scenario demonstrates our approach's ability to identify the causality source of the Philis Hue Iris light turning on. In this scenario, we send voice commands to the Google Home and Amazon Echo devices to turn on the lights. Our approach monitors the network communications and side channel information, and follows the chain of events starting from the effect event, the Hue Iris light activation event, as depicted in Figure 3a. Based on power consumption patterns of the Google Home and Amazon Echo, our approach is able to correctly identify which virtual personal assistant device received the voice command that triggers the light activation.

b) Scenario 2: Identifying anomalous activity: The second scenario, illustrated in Figure 3b, uses the same setup to show how our approach can be leveraged for monitoring and debugging purposes. This scenario demonstrates our approach's ability to not only identify where a chain of events breaks down, but also what parts of the expected interactions are missing. To simulate a broken link in a chain of interactions, we place the Philips Hue bridge in a faraday cage to isolate it from Zigbee radio communications. While the Hue bridge is still connected to the Internet via a physical wire, it is unable to communicate with the Hue Iris and turn the light on. We repeat the "turn on lights" voice command and use our approach to determine where the chain of events terminates. Unlike in the previous scenario, our approach performs forward-search to determine the sequence of effects created by the voice command. By comparing the gathered evidence against the learned model we successfully find that the Zigbee "lights on" message to the Iris light is the missing link.

(a) Scenario 1: Determining causality source

(b) Scenario 2: Identifying anomalous activity

Fig. 3. Our approach is capable of performing both forward and backward search to analyze a chain of events.

IV. Limitations and Future Work

Limitations. The reliance on learning device behavior limits our coverage to only previously-observed behaviors. Extensive exercising of devices is crucial to achieving completeness. The dynamic nature of devices and the high number of factors that can dictate their behavior makes this nontrivial. Another limitation of our approach is that it currently only supports cause-effect relationships that are temporally adjacent and does not handle events interrupted by delays.

Related Work. Several prior efforts [1]–[3] apply provenance analysis to the IoT domain. While similar in scope, we do not employ instrumentation-based provenance analysis. Previous work on modeling systems and their behavior has been done in many domains, including IoT. Researchers have proposed models for IoT and Cyber-physical environments [4] to identify potential interaction paths. Other previous efforts [7] propose threat detection frameworks that rely on models of communication patterns in industrial control systems. In contrast to these works, our work combines evidence from disjoint information streams to construct a probabilistic model of device behavior. Unlike previous efforts that apply side channel analysis for anomaly detection [8], [9], our work leverages side channels to characterize behavior.

Future Work. There are several ways to improve and extend our approach in the future. We plan on extending the model proposed by Agadakos et. al. [4] to capture more information regarding potential interactions between devices, particularly with respect to time. We wish to address temporal discontinuity of events by keeping track of the device behavior and associated states. Finally, we plan to explore the use of mathematical correlation on edges to fascilitate selection of most probable paths. One way of extending our approach is to add support for handling "if this, then that" (IFTT) rules (e.g., as in the Mozilla Web of Things Gateway) and leverage our approach to verify the expected behavior.

V. Conclusion

The autonomy of the Internet of Things provides many benefits. However, this automation also raises security concerns and introduces new challenges. As the IoT network scales, it becomes increasingly difficult to determine how and why devices interact. In this paper, we propose an approach that enables one to infer a potential sequence of actions that led to a particular observation. The proposed aproach relies on a model of IoT device interactions and collected data to learn expected behavior and trace an event to the original cause. We demonstrate our methodology in a small smart-home IoT network and show that it is possible to identify the root cause of various events such as a smart-light turning on. The many benefits and applications of our approach include that it can be applied as an anomaly detector for IoT device interactions, and that it can be used for planning (e.g., configuring devices to minimizes attack surface or providing certain isolation properties) and debugging (e.g., indicating where the chain of interactions failed).

Acknowledgment

This work was performed at SRI International's Internet of Things Security and Privacy Center.

References

[1] Q. Wang, W. U. Hassan, A. Bates, and C. Gunter, "Fear and logging in the Internet of things," in *Network and Distributed Systems Symposium*, 2018.

[2] M. N. Aman, K. C. Chua, and B. Sikdar, "Secure data provenance for the Internet of things," in *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security*. ACM, 2017, pp. 11–14.

[3] S. Bauer and D. Schreckling, "Data provenance in the Internet of things," in *EU Project COMPOSE, Conference 2013*, 2013.

[4] I. Agadakos, C.-Y. Chen, M. Campanelli, P. Anantharaman, M. Hasan, B. Copos, T. Lepoint, M. Locasto, G. F. Ciocarlie, and U. Lindqvist, "Jumping the air gap: Modeling cyber-physical attack paths in the Internet-of-Things," in *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy*, ser. CPS '17, 2017, pp. 37–48.

[5] Z. Ghahramani, "Learning dynamic Bayesian networks," in *Adaptive processing of sequences and data structures*. Springer, 1998, pp. 168–197.

[6] D. Ron, Y. Singer, and N. Tishby, "Learning probabilistic automata with variable memory length," in *Proceedings of the seventh annual conference on Computational learning theory*. ACM, 1994, pp. 35–46.

[7] M.-K. Yoon and G. Ciocarlie, "Communication pattern monitoring: Improving the utility of anomaly detection for industrial control systems," in *Proceedings of the 2014 NDSS Workshop on Security of Emerging Networking Technologies (SENT)*, 2014.

[8] C. Aguayo Gonzalez and A. Hinton, "Detecting Malicious Software Execution in Programmable Logic Controllers Using Power Fingerprinting," in *Critical Infrastructure Protection VIII*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 15–27.

[9] A. Nazari, N. Sehatbakhsh, M. Alam, A. Zajic, and M. Prvulovic, "EDDIE: EM-Based Detection of Deviations in Program Execution," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA '17. New York, NY, USA: ACM, 2017, pp. 333–346. [Online]. Available: http://doi.acm.org/10.1145/3079856.3080223

Cognitive Enhancement as an Attack Surface

Daniel J. Sanchez
Computer Science Laboratory
SRI International
Menlo Park, CA, USA
daniel.sanchez@sri.com

Bogdan Copos
Computer Science Laboratory
SRI International
Menlo Park, CA, USA
bogdan.copos@sri.com

***Abstract*—Imagine that an attacker with an off-the-shelf Bluetooth exploit could go beyond gaining access to your connected devices and directly manipulate the neurons in your brain. The emergence of wearable Internet-of-Things devices has introduced consumer-available neural stimulator devices that are capable of modulating brain activity and function. The level of stimulation varies from seemingly innocuous sound stimulation to electrical current delivered to the scalp. We outline how these new stimulation technologies are opening the door to a novel type of attack surface. One use case shows how it is possible to manipulate stimulation parameters with off-the-shelf Bluetooth man-in-the-middle and replay attacks. Another scenario outlines a risk analysis conducted during development of an auditory sleep enhancement system. Lastly, we discuss safeguards and recommendations to address the emerging threats related to the nascent market of consumer wearables with actuation capabilities.**

***Index Terms*—Internet-of-Things security, wearables, attack surface, cognition, human stimulation, Bluetooth**

Introduction

Security research of "neuro-devices" has focused on privacy concerns related to devices that measure neural activity [1]. However, off-the-shelf wearables now feature stimulation capabilities. Consumer-available, connected, wearable devices that are capable of neurostimulation naturally raise concerns about their security and physical-safety risks. Recent high-profile exploits of smart guns [2] and automated car wash stations [3] have demonstrated the feasibility of physical harm that can be accomplished when an attacker compromises a connected system. Here we outline capabilities for abuse in this emerging market of systems that are able to directly stimulate a user's brain, and provide guidance on practices for safety guided by both computer science and neuroscience.

Cyber-security risks related to consumer neurostimulators are fundamentally different from those of cyber-physical systems for a number of reasons. **1) Neurobiology:** The difficulty of accurately modeling the effects of neurostimulation makes it challenging to validate ongoing system operation with closed-loop feedback, **2) Scale:** Consumer-available stimulators are much cheaper and more widely accessible than their medical-grade counterparts, **3) Oversight:** Despite their actuation capabilities, the FDA brands these devices as "health and wellness" as opposed to a "medical device" [4], putting them in the same category as fitness trackers and aroma therapy. **4) Misuse:** There are no doctors monitoring dosage and patient outcomes, so users must monitor their own progress and performance. In this paper, we show how to manipulate two different neurostimulation technologies to produce adverse effects and provide recommendations for system creators.

Neurostimulation Overview

Systems that can modulate neural activity are no longer reserved for research labs with huge budgets. Advancements in technology and computing have made these tools cheap enough to be embedded into consumer devices that can be purchased online or at specialty stores. Brain sensing was the first neurotechnology to gain consumer traction, with companies such as NeuroSky, Emotiv, and Muse producing electroencephalography (EEG) headbands for users to enhance their lives through "biofeedback," which is the effortful modulation of one's own brain waves. However, humans are intrigued with the idea of enhancing cognition and human performance. A dedicated biohacker community has long pursued brain-enhancing nootropics and techniques for ultimate performance, and now digital technology has entered the space of enhancement tools. Professional sports teams are exploring these tools, and it has been reported that the Golden State Warriors, NBA Champions in 2015 and 2017, have used consumer neurostimulators to enhance their performance [5]. Additionally, sleep enhancement is a new segment of this emerging market, with startups and large companies taking part in developing devices that provide stimulation for modulating brain function. Consumer neurostimulators vary in their approach, from mild electrical stimulation [6] to change cortical excitability, to auditory cues that are precisely-timed to modulate sleep or memory function [7]. A key realization, from a security perspective, is that while this stimulation seems fairly innocuous, these devices work with a high degree of specificity such that changes in the parameter space can render the system useless, or worse, a threat to the user's wellbeing. To provide a thorough picture of the space, we present an exploitation scenario for a consumer electrical stimulation system and a risk analysis for an auditory stimulation system.

Transcranial Direct Current Stimulation

Transcranial Direct Current Stimulation (tDCS) has been, by far, the most popular form of consumer-level "brain-hacking," with devices that are currently available for purchase [8]. tDCS works by applying a mild electrical current to the scalp (usually between 1-2 mA), where anodal (positive) stimulation

excites neurons, making them more likely to fire, and cathodal stimulation (negative) depresses neuronal excitability [9], [10]. Although tDCS is generally considered safe when used according to evidence-based guidelines, the consequences of higher stimulation intensities and durations are not yet well understood [11].

Use Case: tDCS Bluetooth Exploit

The system we analyzed is marketed for enhancing skilled motor performance in athletes. We selected it for our study because the stimulation sites are easily accessible for monitoring our test results. The system consists of three components: the stimulation headset, a back-end server, and a smartphone application (app). The headset is a set of over-the-ear headphones with three tDCS stimulator pads built into the band (see Fig. 1). The pads in the band are situated over key motor cortex areas to stimulate neurons and enhance performance. The app communicates to the server for updates and information about the neurostimulation sessions. Session profiles are downloaded from the cloud when the application starts, outlining the stimulation protocol that dictates stimulation intensity, electrical polarity, and session duration. Relative stimulation intensity can then be controlled within the app during a session. The session profiles (downloaded from the server) and user commands are communicated to the headset over Bluetooth Low-Energy (BTLE) 4.1. The app performs read requests for headset status information to obtain battery level, and, critically, the impedance value that determines whether the headset has a connection with the user's scalp. If the impedance value or battery level are below a threshold, the application does not allow a user to start a stimulating session.

Threat Model and Attack Methodology

We focus on a particular threat around tDCS neurostimulation. Specifically, we consider an attacker who wants to harm the performance or well-being of a user by maliciously taking control of the system. Considering these devices are marketed for performance enhancement, we also discuss users as a potential threat in the *Discussion* section.

Our man-in-the-middle (MitM) attack exploits the lack of authentication in the Bluetooth communications between the smartphone and the headset and demonstrates that such an attack gives a malicious party full control. We use an open-source BTLE MitM framework that is comprised of two components: an interception core and a proxy. The proxy spoofs the slave (headset) device, while the interception core connects to the actual headset and enables an attacker to capture and modify traffic between the smartphone and headset. The traffic was analyzed during repeated observations to determine association between Bluetooth payloads and headset actions. Once the payload format was determined, our MitM exploit modified particular payload bytes while the headset was monitored by an oscilloscope to assess impact. Once a user pairs their mobile device to the inconspicuously spoofed headset, we demonstrate that an attacker can trigger

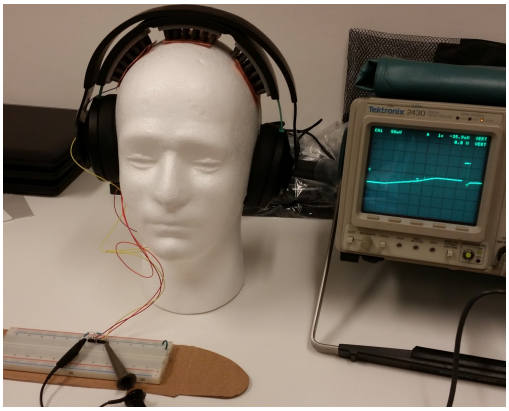


Fig. 1. Hardware testbench for assessing the operations of a tDCS headset using a custom sensor rig and oscilloscope.

neurostimulation sessions, spoof impedance values, and adjust stimulation intensity and session duration. Our ability to spoof impedance readings allows us to cause the device to stimulate even when it is not being worn. We were also successful in altering values relating to stimulation intensity to set the amplitude beyond the limit defined by the app. Using an oscilloscope, as shown in Figure 1, we determined it was possible to increase the intensity to 140% of the maxmimum level (i.e., increase to 3.2 mA vs. the app-defined level 10 at 2.0 mA).

Auditory Stimulation

Poor sleep leads to a host of negative consequences [12]. To this end, another form of neurostimulation that is becoming popular is sleep and memory enhancement with the use of precisely-timed sound stimulation. During sleep, the brain produces stereotyped activity that can be reliably measured with EEG. Slow-wave oscillations (SOs) are patterns of activity, around 1 Hz in frequency, that correspond to neurons cycling between highly active and inactive states. These SOs have a crucial role in the restorative qualities of sleep and memory consolidation [13]. There are two strategies for cognitive enhancement that are achieved by time-locking audio stimulation to key oscillation phases. *General sleep enhancement* is accomplished by presenting a short burst of pink noise (50ms) to increase the amplitude and power of the SOs [14], [15]. *Targeted memory enhancement* employs a similar strategy, but in this case auditory stimulation that was presented during learning is presented again during the SOs to bias the neural activity for that information and memory representation, preferentially strengthening that knowledge [16], [17].

Use Case: Auditory Cognitive and Sleep Enhancement

Rythm (Dreem) and Philips (SmartSleep) have consumer devices that attempt to enhance slow-wave sleep by using auditory stimulation driven by real-time monitoring of EEG signals. While these headsets target general sleep enhancement, the platform and framework are the same for a system aimed at targeted memory enhancement. Here we outline risk analyses

assessed during development of an at-home device capable of delivering auditory stimulation for cognitive enhancement during sleep. It is reasonable to argue that the attacks outlined in the tDCS section are also applicable to auditory stimulation devices, but our hands-on experience developing a targeted memory enhancement system provides novel insight into the security concerns and possible negative outcomes with these technologies. The following sections outline negative consequences we discovered during early stages of development of an auditory neurostimulation system through studies with in-lab interviews (N=3), overnight clinical sleep monitoring (N=5), and surveys (N=24).

Cognitive Impairment through Sleep Disruption

Auditory-based sleep enhancement is exceptionally sensitive to precisely-timed stimulation [18]. Stimulation is useless if timing is slightly off, or worse, actually harmful to sleep quality. The algorithms typically work in two phases: (1) deep sleep identification, and then (2) slow-wave monitoring and stimulation. The first phase is critical, as it is easy to falsely identify SOs outside of deep sleep due to oscillating signals during wake or other stages of sleep. Thus, an attacker (or system developer) needs only to cause the system to deliver the stimulation outside of the ideal sleep stage.

During system development, clinicians noticed that many participants experienced "sleep arousals" triggered by auditory stimulation outside of deep sleep by volumes low enough to not cause awakenings, but loud enough to disturb sleep. "Sleep arousals" are rapid transitions from deep to lighter stages of sleep that are detrimental to sleep quality, but do not necessarily cause individuals to wake up and become aware their sleep is disturbed. Additionally, in a follow-up study we purposefully stimulated during all stages of sleep. In this study, 83% of users reported hearing noises during sleep (i.e., awakenings). However, only 20% believed it negatively impacted their sleep and 33% believed it actually helped their sleep quality! This disconnection between sleep disruption and user perception highlights how people are not accurately aware of their sleep quality, indicating that sleep enhancement exploits would likely go unnoticed [12].

Targeted Nightmare Enhancement

While users may not be consciously aware of the detriments of poor sleep quality, some negative consequences clearly reach a user's consciousness. Targeted memory enhancement is achieved by presenting auditory cues during deep sleep that were present during an earlier learning event in order to bias consolidation for that information. One natural application is enhancing foreign language vocabulary learning [19], as it is naturally auditory in nature, and it is a highly desired learning activity. Despite academic successes with enhancing vocabulary learning in a laboratory environment, the movement to real-world applications introduces novel complications that normally would not be detected in a research environment. During initial testing with an at-home device that used foreign vocabulary words, in-person interviews revealed that some users woke up and/or had nightmares, fearing that "...someone was in [their] bedroom" because of the mysterious voices. It is interesting to note that these adverse events were experienced outside of the lab environment and were only reported during interviews and not expressed in self-report surveys. In a lab environment, it is not surprising to hear random voices of other participants or researchers, but in a home environment it is reasonably alarming to hear unfamiliar voices. This demonstrates the potential for enhancement systems to become detrimental to a user's health and warrants serious consideration of how security and safety can be embedded based on principles from neuroscience and computer science.

Discussion

Security researchers have been concerned with wearables because of privacy-related problems, but now wearables also feature actuation capabilities that can physically interact with users. To this end, our work shows that well-established and device-agnostic exploits can directly influence brain function, producing anything from a change in motor stimulation to inducing nightmares. The trivial effort required to find and utilize such exploits demonstrates the grave concern for IoT security. These vulnerabilities reveal a need for wearable device security practices that are well-informed by both computer science and neuroscience. It is important to follow security design principles [20] and best practices, and here we outline three principles that should be carefully considered.

Recommended Security Practices

Independent Protection Mechanisms: Many IoT and wearable systems are distributed amongst disjoint subsystems, increasing the attack surface. Following with the tDCS example, while the stimuli intensity levels are well-defined in the app, we show how their values can be successfully modified by a MitM and used by the headset without any additional validation. At the heart of this problem is the lack of independent protection mechanisms and over-privilege. We argue that it is crucial that safeguards are introduced to protect the data (1) at rest and in transit, (2) in both hardware and software, and (3) in all components of the system (e.g., app, hub, wearable). Given the actuating capabilities of such devices and their potential impact on users' well-being, we believe safeguards should be introduced at the hardware level as a fail-safe default that would protect users from potentially unsafe levels of stimuli.

Feedback-loop Validation: A security mechanism guided by neuroscience that could reduce exploitation likelihood is a robust closed-loop mechanism that has a predictive model of stimulation effects for continuous system validation. A simplified framework for this closed-loop feedback is shown in Figure 2. By having a predictive model of how the neural signal should be affected during stimulation, the system is able to incorporate this information into the stimulation protocol. For example, predicted slow-wave architecture changes due to auditory stimulation, such as wave amplitude and slope. In this case, it is much more difficult for the attacker to perturb

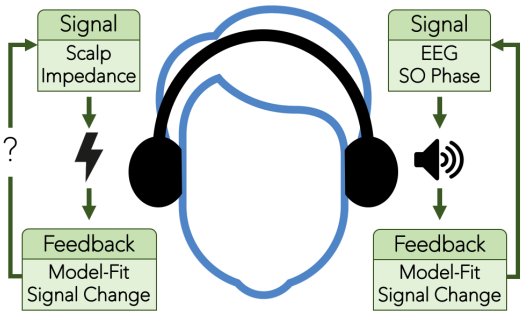


Fig. 2. Diagram with closed-loop feedback signal for neurostimulation. On the left is a highly simplified diagram of a tDCS stimulation loop with an unknown level of closed-loop feedback. On the right is a simplified auditory stimulation loop with closed-loop feedback based on how well the ongoing EEG signal matches the predicted change given the stimulation.

the system and affect the user because the predictive model would not match the feedback signal and stimulation would stop. The authors note this mechanism is likely achievable with tDCS systems, as noted on the left portion of Figure 2, but determining a feasible closed-loop marker was outside the scope of this research.

Anti-Abuse Safeguards: Lastly, wearable enhancement devices also present a unique challenge from a security standpoint in their potential for user abuse. Guided by the history of abuse potential with performance-enhancing drugs in sports [21], it is important to stress the need for security that protects a user from self-inflicted harm by misuse of such devices. While many of these devices implement safeguards to restrict use, these safeguards are often trivially bypassed. In the case of the tDCS system we assessed, creating new accounts or stopping a session prematurely enables a user to freely use the headset as often as desired. Thus, it is important to embed safeguards at multiple levels (hardware, server-side, algorithms) to ensure that misuse is mitigated.

Main Takehome

Given the potential for harm and lack of oversight with off-the-shelf cognitive enhancement systems, we believe it is critical to introduce redundant, independent security mechanisms informed by neuroscience, computer science, and human factors. While completely mediating these issues is challenging, we believe that all system developers, from neuroscientists to engineers and computer scientists, can work together to ensure that safeguards are introduced into all system components at multiple layers. Although we understand system designers are rightfully focused on the benefits and positive impacts of their systems, it is important to always be vigilant about the potential risks that could arise with these IoT devices.

Acknowledgment

System testing was performed at SRI International's Internet-of-Things Security and Privacy Center.

References

[1] Tamara Bonaci, Jeffrey Herron, Charlie Matlack, and Howard Jay Chizeck. Securing the exocortex: A twenty-first century cybernetics challenge. In *Norbert Wiener in the 21st Century (21CW), 2014 IEEE Conference*, pages 1–8. IEEE, 2014.

[2] Plore. Popping a smart gun. https://media.defcon.org/DEF CON 25/DEF CON 25 presentations/DEFCON-25-Plore-Popping-a-Smart-Gun-UPDATED.pdf, July 2017.

[3] Billy Rios. When IoT attacks: Understanding the safety risks associated with connected devices. https://www.blackhat.com/docs/us-17/wednesday/us-17-Rios-When-IoT-Attacks-Understanding-The-Safety-Risks-Associated-With-Connected-Devices.pdf, July 2017.

[4] Halo Neuroscience. Is Halo Sport FDA approved? https://support.haloneuro.com/hc/en-us/articles/115010397027-Is-Halo-Sport-FDA-approved/, 2017.

[5] Alex Hutchinson. For the golden state warriors, brain zapping could provide an edge. http://www.newyorker.com/tech/elements/for-the-golden-state-warriors-brain-zapping-could-provide-an-edge, 2016.

[6] Michael A Nitsche and Walter Paulus. Excitability changes induced in the human motor cortex by weak transcranial direct current stimulation. *The Journal of Physiology*, 527(3):633–639, 2000.

[7] Delphine Oudiette and Ken A Paller. Upgrading the sleeping brain with targeted memory reactivation. *Trends in Cognitive Sciences*, 17(3):142–149, 2013.

[8] Jennifer Alsever. Can electric 'brain training' devices make you smarter? http://fortune.com/2015/11/17/electric-brain-training-devices-cognitive-enhancement/, 2015.

[9] Giuliana Grimaldi, Georgios P Argyropoulos, Amy Bastian, Mar Cortes, Nicholas J Davis, Dylan J Edwards, Roberta Ferrucci, Felipe Fregni, Joseph M Galea, Masahi Hamada, et al. Cerebellar transcranial direct current stimulation (ctdcs) a novel approach to understanding cerebellar function in health and disease. *The Neuroscientist*, 22(1):83–97, 2016.

[10] Gowri Jayaram, Byron Tang, Rani Pallegadda, Erin VL Vasudevan, Pablo Celnik, and Amy Bastian. Modulating locomotor adaptation with cerebellar stimulation. *Journal of Neurophysiology*, 107(11):2950–2957, 2012.

[11] Csaba Poreisz, Klára Boros, Andrea Antal, and Walter Paulus. Safety aspects of transcranial direct current stimulation concerning healthy subjects and patients. *Brain Research Bulletin*, 72(4):208–214, 2007.

[12] Hans Van Dongen, Greg Maislin, Janet M Mullington, and David F Dinges. The cumulative cost of additional wakefulness: dose-response effects on neurobehavioral functions and sleep physiology from chronic sleep restriction and total sleep deprivation. *Sleep*, 26(2):117–126, 2003.

[13] Giulio Tononi and Chiara Cirelli. Sleep and the price of plasticity: From synaptic and cellular homeostasis to memory consolidation and integration. *Neuron*, 81(1):12–34, 01 2014.

[14] HV Ngo, Thomas Martinetz, Jan Born, and Matthias Mölle. Auditory closed-loop stimulation of the sleep slow oscillation enhances memory. *Neuron*, 78(3):545–553, 2013.

[15] Ju Lynn Ong, June C. Lo, Nicholas I. Y. N. Chee, Giovanni Santostasi, Ken A. Paller, Phyllis C. Zee, and Michael W. L. Chee. Effects of phase-locked acoustic stimulation during a nap on eeg spectra and declarative memory consolidation. *Sleep Medicine*, 20:88–97, 2016.

[16] Daniel Bendor and Matthew A Wilson. Biasing the content of hippocampal replay during sleep. *Nature Neuroscience*, 15(10):1439, 2012.

[17] John D Rudoy, Joel L Voss, Carmen E Westerberg, and Ken A Paller. Strengthening individual memories by reactivating them during sleep. *Science*, 326(5956):1079–1079, 2009.

[18] Arne Weigenand, Matthias Mölle, Friederike Werner, Thomas Martinetz, and Lisa Marshall. Timing matters: Open-loop stimulation does not improve overnight consolidation of word pairs in humans. *European Journal of Neuroscience*, 44(6):2357–2368, 2016.

[19] Laura J Batterink, Carmen E Westerberg, and Ken A Paller. Vocabulary learning benefits from REM after slow-wave sleep. *Neurobiology of Learning and Memory*, 144:102–113, 2017.

[20] Jerome H Saltzer and Michael D Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[21] Claudia L Reardon and Shane Creado. Drug abuse in athletes. *Substance Abuse and Rehabilitation*, 5:95–105, 2014.

Control-hijacking Vulnerabilities in IoT Firmware: A Brief Survey

Abhinav Mohanty*, Islam Obaidat*, Fadi Yilmaz* and Meera Sridhar*
Software and Informations Systems, University of North Carolina at Charlotte
Email: *amohant1@uncc.edu, *iobaidat@uncc.edu, *fyilmaz@uncc.edu, *msridhar@uncc.edu

***Abstract*—Security issues in the Internet of Things (IoT) are rapidly growing. While much recent research has focused on network security, authentication, authorization, protocol, web and mobile (interfaces for IoT devices) security, less attention has been devoted to software vulnerabilities, especially in IoT *firmware*. As a first step towards gathering research community attention and developing robust defenses, we present a classification of recently discovered control-hijacking vulnerabilities in select IoT firmware.**

I. Introduction

The number of devices in the *Internet of Things* (IoT) is expected to exceed 26 billion [27] within the next four years. Industry experts estimate that 70% of IoT devices presently on the market are vulnerable to cyber attacks [33], and Gartner predicts that by 2020 more than 25% of enterprise attacks will involve exploits of IoT devices [31]. Taken together, these statistics forecast an alarming perfect storm brewing in the world of cyber security. With such an enormous network of devices, many of which are likely to be deeply integrated into the daily lives of users, and considering the complexity of the multilayered IoT architecture, securing the IoT is one of the most essential and yet highly challenging tasks facing technologists today.

A large percentage of IoT security efforts focus primarily on network and systems security solutions ([47], [42]); but a report from NIST has shown that 77% of all reported vulnerabilities are in software applications, not in networks or operating systems [41]. Unfortunately, while industrial and governmental communities are now slowly becoming more aware that software security is a vital consideration for the design, development, and evaluation of IoT products (e.g., IoT security expenditures are expected to exceed $500 million this year [31]), it is not yet sufficiently widely recognized that the nature of many IoT security problems differs fundamentally from conventional cyber security problems, and that critical gaps still exist in the science of security that will impede conventional software defense approaches from scaling adequately to the expected growth of the IoT. For example:

- **Low expertise:** Many industrial producers of IoT software are not software companies, and therefore lack comprehensive training in cyber security or even foundational software engineering best-practices (cf., [37]). Additionally, typical IoT consumers also lack technical expertise needed for specifying/enforcing policies, updating device firmware frequently, and applying critical security patches.
- **Device heterogeneity:** Many conventional approaches to software security confine their attention to one hardware architecture at a time; but IoT attacks frequently cross platforms or abuse interactions between the many diverse hardware elements that comprise the IoT (cf., [32], [39], [34]).
- **Light-weight architectures:** Consumer-side software protection mechanisms often draw upon the relatively extensive computing resources of modern PCs—for example, to implement virtual machines, hypervisors, or intrusion detection systems. IoT software defenses require the innovation of much lighter-weight solutions, since they typically lack the resources necessary to sustain these conventional, resource-heavy approaches.

Much of IoT security research ([25], [35], [23], [47]) has focused on authentication and authorization issues, light-weight crypto systems, and security protocols, such as encrypted communication between IoT devices and their counterparts (hub, router, cloud, mobile app, etc.), hard-coded encryption keys/credentials in device firmware or mobile apps, and prevalence of default credentials and weak passwords. Other works have also focused on securing the cloud back-end, and web and mobile interfaces of the device.

One of the prerequisites for building robust software defense systems for the IoT is a comprehensive study of the IoT attack surface. Surveys on IoT security issues prevail in the literature ([40], [43], [36], [44]), but most of them are too broad and general.

In this paper, we focus on a very specific software-security issue, *control-flow hijacking*, and present a classification of control-hijacking vulnerabilities in IoT *firmware*. Control-flow hijacking is a highly significant class of attacks in software security, where attackers can gain control of the entire system [46].

General purpose computing devices can be protected against such attacks by traditional security mechanisms such as binary randomization, memory layout randomization, stack canaries, tainting of suspect data, forcing pages to either be writable or executable (DEP), and *Control Flow Integrity* (CFI) enforcement(cf., [30]). However, these countermeasures typically require high computation capabilities and memory usage, and often rely on hardware that is unavailable to simple microcontrollers such as a Memory Management Unit (MMU) or

execution rings that most IoT devices lack. Moreover, they mostly use software solutions, since hardware modifications (for example on the IA-32 architecture) are difficult and likely to cause problems with legacy applications [30].

IoT firmware attacks and vulnerabilities remain fairly less explored in the literature; however, vulnerabilities such as *buffer overflows* [4], *integer overflows* [19], *heap overflows* [14] and *format string* [10] vulnerabilities abound in device firmware, operating systems, and utility programs running on the device.

Although control-hijacking attacks are not widespread in IoT yet, it is just a matter of time, when trivial bugs such as weak or default passwords are fixed, and attackers will move towards more sophisticated attacks such as these [24].

We hope that this small study would be the first step towards focusing research attention and defense solutions on this significant class of vulnerabilities.

II. Control-Flow Vulnerabilities in IoT Firmware

Table I presents the most widespread and interesting control-flow hijacking vulnerabilities in the recent past discovered in IoT firmware. This section also presents more detailed descriptions of select vulnerabilities from the table.

We use the following metrics for the characterization of these vulnerabilities:

- Is the source code available for analysis?
- How many devices or users were affected?
- What is the architecture/OS on the device?
- Which version of the firmware or the device model was affected?
- Was a CVE number [38] assigned to the vulnerability?
- When was the vulnerability first reported?
- What risks does the vulnerability pose? (For example, Remote Code Execution, Denial of Service, etc.)

Foscam C1: The Foscam C1 [11] is a highly popular IP camera designed to be accessible remotely via a web interface/mobile application. In June 2017, researchers working at Cisco Talos [5] reported numerous vulnerabilities in the firmware version 2.52.2.37 of the product. Among these was a stack overflow which could be triggered by a specially crafted *HTTP* request (CVE-2017-2805). Once again in November 2017, the same security firm reported multiple buffer overflow vulnerabilities in the same product which could be used for DoS and RCE attacks. The researchers also demonstrated that they could also spoof the video stream as well. This could easily be used in conducting criminal activities and escaping the aftermath.

D-Link: In July 2016, researchers at security firm Senrio [9] revealed a stack buffer overflow vulnerability in the firmware (v1.12) [1], which is common to more than 40 distinct D-Link devices [6]. According to Shodan [21], there are 414,949 publicly accessible devices impacted by this vulnerability.

Devil's Ivy: On July 18 2017, security firm Senrio, revealed a stack buffer overflow vulnerability (CVE-2017-9765) in the implementation of the *gSOAP* [12] toolkit that could be exploited to take over thousands of IoT devices across the world; the vulnerability was termed as *'Devil's Ivy'* [13]. gSOAP is an open-source software development toolkit for XML web services and generic XML data bindings. The gSOAP tools are widely used in physical security products to allow them to perform XML serializations efficiently with zero-copy overhead. According to Axis [3], the market leader in IP cameras, Devil's Ivy is present in 249 distinct camera models manufactured by the company.

Dnsmasq: Dnsmasq [7] is an open-source, lightweight DNS forwarder and DHCP server for small-scale networks. It is widely used in home networks and cloud environments as a caching DNS stub resolver and to manage DHCP leases. In October 2017, researchers at Google, discovered a heap-based overflow vulnerability (CVE-2017-14491) in Dnsmasq versions 2.70–2.77 [8]. A carefully constructed DNS request can be used to overflow the heap and perform remote code execution or denial of service. According to Shodan, almost 1,098,179 devices were affected by this vulnerability.

MatrixSSL: MatrixSSL [16] is an open-source TLS/SSL implementation designed for embedded hardware environments. More than 900 products from about 50 vendors employ MatrixSSL [45]. The implementation of parsing X.509 certificates in MatrixSSL version 3.8.7b contains several vulnerabilities, which include heap-based buffer overflow and integer overflow [15]. The vulnerabilities can be exploited to achieve remote code execution and sensitive information leakage.

Amazon Echo and Samsung Gear S3: At *BlackHat Europe 2017*, security researchers Ben Seri and Gregory Vishnepolsky disclosed numerous critical vulnerabilities in the implementation of the *Bluetooth* protocol, which is widely is used in IoT device communication. They demonstrated a successful exploitation of buffer overflows in both the Echo and Gear S3, and pointed out that even the devices from top IT companies lack basic security mechanisms as *Address Space Layout Randomization* (ASLR) [2], *Stack Canary* [29] or *NX_bit* (DEP) [18].

Mirai: Mirai [17] made its first appearance in September 2016, with dramatic flair. After flooding a prominent security journalist's website with traffic from a massive botnet created using vulnerable Internet of Things devices, it managed to make much of the Internet unavailable for millions of people by overwhelming Dyn [20], a company that provides a significant portion of the Internet backbone for United States. Surprisingly, a month later in October 2016, researchers at security firm Invincea Labs, discovered multiple vulnerabilities in the code of Mirai itself, one of which was a stack buffer overflow. The vulnerability could be exploited to crash the process and stop an infected device from attacking.

III. Related Work

Related work on control-hijacking defenses for traditional systems is vast; we omit their discussion here due to space constraints.

Costin et al. [28] have conducted a large-scale analysis of embedded firmware. However, their analysis does not cover

Table I
CONTROL FLOW HIJACKING ATTACKS IN IOT SPACE

Device/API	Open Source	# of Users/Devices Affected	Architecture/OS	Affected version/model	#CVE	Date of Report	RISKS
D-Link	Yes	∼414949 devices	MIPS architecture	39 models	CVE-2016-6563	06/08/2016	RCE, DOS
Mirai	Yes	N/A	Embedded Linux	Mirai Source Code	N/A	10/28/2016	
MatrixSSL	Yes	>900 products >50 vendors	Embedded and IoT systems	<version 3.8.5	CVE-2016-6890	10/11/2016	RCE
Foscam C1	Yes	>500,000 users	Embedded Linux	System f/w: v1.9.3.17 App f/w: v2.52.2.37 Web: v2.0.1.1 Plug-In: v3.3.0.5	CVE-2017-2805 CVE-2017-2830–2831 CVE-2017-2851	6/19/2017	RCE, DoS Complete Takeover
Devil's Ivy	Yes	>million devices	249 unique Axis cameras	>34 companies	CVE-2017-9765	06/21/2017	RCE, DoS Complete Takeover
MatrixSSL	Yes	>900 products >50 vendors	Embedded and IoT systems	version 3.8.7b	CVE-2017-2780–2782	6/22/2017	RCE
Zmodo_Greet	No	>100,000	BusyBox, UART			08/11/2017	RCE, DoS
ConnMan	Yes	usage unknown	Embedded Linux	<v1.34 of ConnMan	CVE-2017-12865 CVE-2017-14491–14493	08/15/2017	RCE, DoS
Dnsmasq	Yes	>million devices	Linux, Android, IoT	<v2.68 of Dnsmasq	CVE-2017-14496 CVE-2017-13704	10/02/2017	RCE, DoS
Foscam C1	Yes	>500,000 users	Embedded Linux	System f/w: v1.9.3.18 App f/w: v2.52.2.43 Plug-In: v3.3.0.26	CVE-2017-2854–2857 CVE-2017-2876 CVE-2017-2878–2879	11/13/2017	RCE, DoS Complete Takeover
Amazon Echo	Yes	>20 million	Fire OS (via bluetooth)	Kernel 2.6.37 (!) Arm 32bit	N/A	12/06/2017	RCE, Complete Takeover
Samsung Gear S3	Yes	>100 million	Tizen OS (via bluetooth)	Kernel 3.18.14 Arm 64bit	N/A	12/06/2017	RCE, DoS
Arduino Yun	Yes		Linux based OS ATmega32u4 chip Atheros AR9331 chip		N/A	05/16/2016	RCE, DoS

control-hijacking attacks. Chen et al. [26] have presented Firmadyne, which is an automated dynamic analysis tool to discover vulnerabilities in firmware. However, their tool finds overflow vulnerabilities through the web interface of the firmware, which does not guarantee complete code coverage. Abera et al. [22] present C-FLAT, which provides control-flow attestation in embedded systems firmware. While C-FLAT is a promising solution, it has not been tested extensively yet (only two cyber physical systems).

Further investigation might be in order, to determine whether these related works on control-hijack defenses for embedded systems can be readily adpated to IoT firmware. The enormous heterogeneity in the IoT firmware space, combined with the proprietary nature of the firmware and reluctance on the part of IoT vendors to incorporate security, might be substantial issues to consider in this process.

IV. Conclusion

We present a classification of recently discovered control-hijacking vulnerabilities in select IoT firmware, using metrics such as source code availability, architecture, risks associated and more. We hope that this study would serve as a first step towards gathering research community attention and developing robust defenses for IoT software security.

Acknowledgements

This research was supported by the NSF CRII award #1566321.

References

[1] 400,000 publicly available IoT devices vulnerable to single flaw. https://tinyurl.com/ycb2p7q4.
[2] Address Space Layout Random Randomization. https://tinyurl.com/zb6p2vm.
[3] Axis Communication. https://www.axis.com/ae/en.
[4] Buffer Overflows. https://www.owasp.org/index.php/Buffer_Overflow.
[5] Cisco Talos - Meet Cisco Talos, the industry-leading threat intelligence group fighting the good fight. https://tinyurl.com/y8ha6mf9.
[6] D-link. http://www.dlink.co.in/products/category/?cid=3.
[7] Dnsmasq. http://www.thekelleys.org.uk/dnsmasq/doc.html.
[8] Dnsmasq: Multiple critical and important vulnerabilities. https://access.redhat.com/security/vulnerabilities/3199382.
[9] Enterprise Security for IoT. http://senr.io/.
[10] Format String Attack. https://tinyurl.com/6t5qwph.
[11] FOSCAM Home Security. https://foscam.com/C1.html.
[12] gSOAP. https://www.genivia.com/dev.html.
[13] Hack Brief: 'Devil's Ivy' vulnerability could afflict millions of IoT devices. https://www.wired.com/story/devils-ivy-iot-vulnerability/.
[14] Heap Overflow: Vulnerability and heap internals explained. https://tinyurl.com/ybeqsqwe. Accessed: 2018-01-24.
[15] Inside secure MatrixSSL x509 certificate SubjectDomainPolicy remote code execution vulnerability. https://tinyurl.com/yav6dda4.
[16] Matrixssl. www.matrixssl.org.
[17] Mirai: what you need to know about the botnet behind recent major DDoS attacks . https://tinyurl.com/yas5p757.
[18] NX bit. http://index-of.es/EBooks/NX-bit.pdf. Accessed: 2018-01-24.
[19] OWASP periodic table of vulnerabilities - Integer Overflow/Underflow. https://tinyurl.com/ycfjrg9l.
[20] Rethink DNS. https://dyn.com/.
[21] Shodan, The search engine for the Internet of Things. https://www.shodan.io/.
[22] T. Abera, N. Asokan, L. Davi, J. Ekberg, T. Nyman, A. Paverd, A. Sadeghi, and G. Tsudik. C-FLAT: Control-flow attestation for embedded systems software. In *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 743–754, 2016.
[23] 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*, 17(4):2347–2376, 2015.
[24] Altium Designer. Internet of Things security vulnerabilities: All about buffer overflow. https://tinyurl.com/ybfdaob3. Accessed: 2018-01-24.
[25] L. Atzori, A. Iera, and G. Morabito. The Internet of Things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010.
[26] D. D. Chen, M. Egele, M. Woo, and D. Brumley. Towards automated dynamic analysis for linux-based embedded firmware. In *Proceedings of the 23rd Annual Network and Distributed System Security Symposium (NDSS)*, 2016.
[27] Cisco. Cisco visual networking index predicts near-tripling of IP traffic by 2020. https://tinyurl.com/mp8r9kw.
[28] A. Costin, J. Zaddach, A. Francillon, and D. Balzarotti. A large-scale analysis of the security of embedded firmwares. In *Proceedings of the 23rd USENIX Security Symposium (USENIX)*, pages 95–110, 2014.
[29] C. Cowan, C. Pu, D. Maier, J. Walpole, P. Bakke, S. Beattie, A. Grier, P. Wagle, and Q. Zhang. Stackguard: Automatic adaptive detection and prevention of buffer-overflow attacks. In *Proceedings of the 7th USENIX Security Symposium*, 1998.
[30] A. Francillon, D. Perito, and C. Castelluccia. Defending embedded systems against control flow attacks. In *Proceedings of the First ACM Workshop on Secure Execution of Untrusted Code*, pages 19–26, 2009.
[31] Garther. Forecast: IoT security, worldwide, 2016. http://www.gartner.com/newsroom/id/3291817, 2016.
[32] Gartner. Gartner says potential size and diversity of the Internet of Things mask immediate opportunities. http://www.gartner.com/newsroom/id/2564916, 2013.
[33] Hewlett-Packard. Internet of Things research study - 2015 report. https://tinyurl.com/zv2bdkm, 2015.
[34] S. Koegler. IoT device diversity, data volumes drive management platform adoption. https://tinyurl.com/yangfkle, 2015.
[35] J. Y. Lee, W. C. Lin, and Y. H. Huang. A lightweight authentication protocol for Internet of Things. In *Proceedings of the 3rd International Symposium on Next-Generation Electronics (ISNE)*, pages 1–2, 2014.
[36] Lieberman Software Corporation. IoT security survey. http://go.liebsoft.com/IoT-Security-Survey, January 2017.
[37] McAfee. McAfee labs 2016 threats predictions. https://tinyurl.com/y9vqs44h, September 2016. Retrieved 10-1-2016.
[38] MITRE. Common vulnerabilities and exposures database. https://cve.mitre.org/. Accessed: 2018-01-24.
[39] B. O'Donnell. Commentary: Tech device diversity set to explode with IoT. https://tinyurl.com/y6vemmk4.
[40] A. Oracevic, S. Dilek, and S. Ozdemir. Security in Internet of Things: A survey. In *Proceedings of the 5th International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–6, May 2017.
[41] G. A. W. Paul E. Black, Larry Feldman. Dramatically reducing software vulnerabilities. https://tinyurl.com/ybqtc7fj, January 2017.
[42] R. Roman, P. Najera, and J. Lopez. Securing the Internet of Things. *Computer*, 44(9):51–58, Sept 2011.
[43] M. Sain, Y. J. Kang, and H. J. Lee. Survey on security in Internet of Things: State of the art and challenges. In *Proceedings of the 19th International Conference on Advanced Communication Technology (ICACT)*, pages 699–704, Feb 2017.
[44] SANS Institute. Securing the Internet of Things survey. https://tinyurl.com/pv4oaa2, 2017.
[45] SEC Consult. House of keys: Industry-wide HTTPS certificate and SSH key reuse endangers millions of devices worldwide. https://tinyurl.com/ycn3rwyl, 2015.
[46] H. Shacham. The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86). In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 552–561, New York, NY, USA, 2007. ACM.
[47] S. Sicari, A. Rizzardi, L. Grieco, and A. Coen-Porisini. Security, privacy and trust in Internet of Things: The road ahead. *Computer Networks*, 76:146–164, 2015.

SDN-based In-network Honeypot: Preemptively Disrupt and Mislead Attacks in IoT Networks

Hui Lin
Computer Science and Engineering Department
University of Nevada at Reno
Reno, NV, USA
hlin2@unr.edu

***Abstract*—Detecting cyber attacks in the network environments used by Internet-of-things (IoT) and preventing them from causing physical perturbations play an important role in delivering dependable services. To achieve this goal, we propose in-network Honeypot based on Software-Defined Networking (SDN) to disrupt and mislead adversaries into exposures while they are in an early stage of preparing an attack. Different from traditional Honeypot requiring dedicated hardware setup, the in-network Honeypot directly reroutes traffic from suspicious nodes and intelligently spoofs the network traffic to them by adding misleading information into normal traffic. Preliminary evaluations on real networks demonstrate that the in-network Honeypot can have little impact on the performance of IoT networks.**

***Index Terms*—IoT, Software-Defined Networking, Honeypot**

I. Introduction

Many critical infrastructures, e.g., smart meters and smart homes, have used Internet-of-things (IoT) to monitor and control physical processes. Detecting cyber attacks in the IoT network environment and preventing them from causing physical perturbations play an important role in delivering dependable services. Based on historical incidents in industrial control systems, "remote insider" attacks can become a big threat to IoT networks [1]. After penetrating into the IoT networks, adversaries stay in a preparation stage, during which they use the existing network traffic to study computing context and collect information related to the physical processes. With the help of collected information, adversaries can develop and execute attack concept of operations by crafting malicious commands in legitimate formats without raising anomaly alerts at network levels.

Current detection approaches for IoT networks are passive [2]. Those approaches rely on anomaly patterns of communication networks after adversaries perform malicious activities, e.g., propagating malware, issuing malicious commands [3]. The communication nodes in an IoT networks are used to operate physical processes, e.g., opening or closing a valve, adjusting room temperature. After adversaries inject malicious activities into IoT networks, the impact of attacks can quickly propagate through underneath physical processes over a wide geographic area [4]. Consequently, even though some passive detection approaches can identify malicious activities, it is very challenging to prevent physical damage.

We argue that preemptive approaches, such as Honeypot or Honeynet, which can expose adversaries at an early stage are effective ways to prevent damage caused by attacks. In recent years, multiple projects use traditional Honeypot for cyber-physical industrial control systems, to attract adversaries and trace their activities [5] [6]. However, it is challenging to apply traditional Honeypot into IoT networks, due to three reasons. *First*, IoT networks can contain a large number of communication nodes, which makes it challenging to mimic the network of the similar size. *Second*, IoT network has a very dynamic feature; the participating nodes and their connections can experience fast changes. Once a Honeypot is built, it is difficult to update its implementation according to run-time changes. *Third*, traditional Honeypots lack the support for constructing meaningful application-layer payloads, e.g., measurements exchanged between communication nodes in IoT networks. Randomly generated measurements communicated in Honeypot can reveal the presence of a bogus environment to adversaries.

Compared to traditional Honeypots, we propose in this paper an in-network Honeypot by using traffic-manipulation capability enabled by Software-Defined Networking (SDN). The in-network Honeypot does not require setting up a dedicated network environment; it directly reroutes the traffic from suspicious nodes identified at run time to an SDN controller, which effectively quarantines the suspicious nodes from other communication nodes. The SDN controller spoofs network communications, which are used to interact with suspicious nodes, on behalf of nonexistent nodes, which we refer as *phantom nodes*. There is no dedicated hardware or software resources allocated for the phantom nodes; their existence, e.g., the IP addresses, are only reflected on spoofed packets issued from the SDN controller to the suspicious nodes. Furthermore, we include in the spoofed traffic misleading information, such as vulnerability of certain physical processes to mislead adversaries into targeting on phantom nodes to perform malicious activities. Consequently, the in-network Honeypot can detect adversaries' malicious activities in a quarantined environment without causing real physical disruptions of the protected IoT networks.

II. System Assumption and Attack Model

In this paper, we consider IoT networks which rely on IP-based networks for communications. As shown in Figure 1, communication nodes exchange information about underneath physical processes, such as thermostats or smart meters. There are two common operations performed in IoT networks: control operations used to configure or operate physical processes

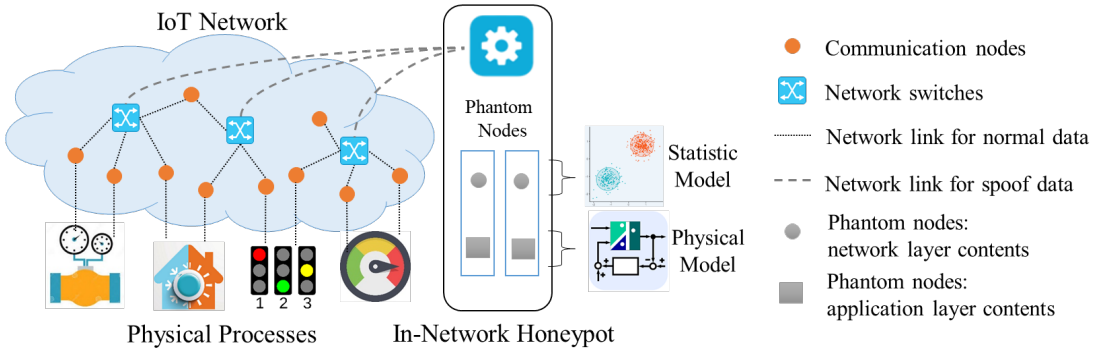


Fig. 1. The architecture of in-network Honeypot for IoT.

and periodic polling operations used to collect analog measurements indicating the state of the physical processes.

We consider the "remote insider" threat model in this paper. *First*, we assume that adversaries can penetrate communication nodes in IoT networks from public networks, which make adversaries insiders. To become insiders, adversaries can exploit vulnerabilities in employee's devices, e.g., laptops, smart phones, or USB drives, that are connected to the IoT network. *Second*, we assume that adversaries are "remote" (i.e., not an expert) to the configuration of IoT networks as well as the characteristics of underneath physical processes. Under these assumptions, adversaries can monitor information exchanged over IoT networks and thus obtain the knowledge on physical processes to prepare for malicious operations.

We assume that the IoT networks under protections can support SDN-enabled switches and the functionality of SDN controlling plane is trusted. SDN is a new network paradigm whose key feature is the separation of the control plane and the data plane [7]. In SDN, network switches are simple forwarding devices, whose forwarding rules can be dynamically configured by a central controller. In recent years, many telecommunication companies, e.g., Huawei, began integrating SDN into their core wireless networks [8]. In the survey shown in [9], Bera et al. has presented the opportunities for SDN to address critical challenges in IoT, e.g., dynamic network management, resource allocations, and resilience. Even though SDN can facilitate the implementation of the in-network Honeypot, the concept of spoofing network traffic to disrupt and mislead adversaries is not restricted by SDN but can be implemented by any traffic manipulation techniques.

III. Architecture of In-network Honeypot

We present the overall design of the in-network Honeypot in Figure 1, whose implementation relies on SDN. An SDN controller can observe all communication going through network switches under its control and use the global knowledge of a communication network to make a traffic-management decision that can achieve optimal network performance, resource utilization, or reliability. The proposed in-network Honeypot, however, is to use the knowledge obtained by the SDN controller and its programmable capability to achieve two objectives: (1) quarantine suspicious or potential malicious nodes and (2) mislead adversaries targeting on nonexistent communication nodes (i.e., phantom nodes). Achieving these two objectives, we can detect adversaries while they are preparing attacks and prevent them from causing damage to physical processes.

A. Quarantine suspicious nodes

We assume that existing IoT networks have intrusion detection systems (IDS) to raise alerts on suspicious activities from communication nodes. After IDS identifies suspicious nodes, the in-network Honeypot uses SDN controllers to stop forwarding the traffic from the suspicious nodes to other communication nodes. In other words, the network traffic from the suspicious nodes cannot reach any communication nodes, which make them quarantined from the IoT networks. Regarding all network traffic from the suspicious nodes, the SDN controller responds with spoofed information on behalf of the destination nodes. Consequently, the suspicious nodes communicate with phantom nodes spoofed by the SDN controller with no attachment to physical machines or software processes.

B. Spoof communication

It is critical for the in-network Honeypot to spoof communication that can mimic the real network traffic, which can maintain highly active interactions between phantom nodes and suspicious nodes. Based on these interactions, the in-network Honeypot can collect more information about those suspicious nodes to make a trustworthy decision.

We spoof communications by adding variations into normal communication patterns. In addition, we include some misleading information in the variations to mislead adversaries into targeting phantom nodes instead of real communication nodes. Note that an SDN controller can observe all traffic going through the switches under its control. We can integrate into the SDN controller anomaly-based intrusion detection techniques to build a normal pattern of each communication node.

As shown in Figure 1, the in-network Honeypot needs to construct contents at both network layer and application layer to spoof network packets on behalf of phantom nodes. We

propose using different approaches to construct contents, i.e., using statistic models for network layer contents and using physical models for application layer contents.

Construct contents at network layer

To construct contents at network layers, we first build statistic models that can classify communication nodes based on their network-layer characteristics. The example of these characteristics includes the range of IP-addresses, the length of network packets, the latency between different types of network packets. Because many physical processes associated with communication nodes run fixed operations and follow deterministic patterns, the in-network Honeypot can use network traffic observed at the SDN controller to fingerprint physical processes and classify network communications associated with them [10]. The advantage of using SDN controllers is that the in-network Honeypot does not need to interfere the normal physical processes but to rely on observed network traffic to build statistic models. Then at runtime, the in-network Honeypot creates contents by following these statistic models.

Construct contents at application layer

In order to prevent adversaries from disrupting underneath physical processes, we construct contents at the application layer to (1) mimic the state of physical processes and (2) mislead adversaries into disrupting the nonexistent physical processes controlled by phantom nodes.

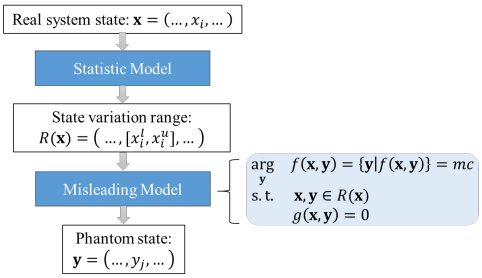


Fig. 2. Procedure to construct contents at application layer.

In Figure 2, we present the high-level procedure to construct contents at the application layer. We use a vector $\boldsymbol{x}$ to represent the state of physical processes managed by the IoT network. Through the traffic collected by the SDN controller, we first monitor those state variables and model their statistic characteristics (in "statistic model"), such as the variation range of each state determined by its lower bound and upper bound (i.e., x_i^l and x_i^u in the figure). In the next step, we include the state variation range in an optimization problem in the "misleading model." The solution to this problem is "phantom state", which represents the state of nonexistent physical processes that are controlled by phantom nodes. In other words, *phantom states are the contents at the application layer of the network packets issued from the phantom nodes*.

Assume that we use function $f(x, y)$ to represent decision procedure of adversaries to cause physical damage, i.e., mc, based on the observed system state and phantom state. In the "misleading model," we set mc as the operations that can cause significant disruption on the phantom state but little impact on real physical processes. Consequently, the solutions to the optimization problem determine the phantom states that can mislead adversaries into designing ineffective attack strategies, which introduce no physical damage. Note that the optimization makes both physical and phantom states subject to the constraints of state variation range observed at runtime. This constraint makes the resulting phantom state follow normal variations and avoid adversaries' suspicions. Also, because physical processes need to follow physical laws, we add the constraint to make physical and phantom state consistent with the physical laws in a mathematical expression, i.e., $g(x, y) = 0$ in the figure. Because this optimization problem mimics adversaries' decision procedure, solving the problem requires similar computation complexity as required to prepare attacks.

C. Handle false detection

If IDS makes a false positive detection and a suspicious node is mistakenly quarantined, the in-network Honeypot can use the SDN controller to restore its communication path. Additionally, the SDN controller can profile the physical changes initiated by the suspicious nodes. After restoring communications for suspicious nodes, the in-network Honeypot can use the profile as reference points to update the physical process, to avoid repeated operations from the suspicious nodes. To profile the physical changes initiated by the suspicious nodes at runtime and without causing real physical changes, we can use simulations, which represent the mathematical models of the physical process, to estimate the consequences of the commands and record them.

In addition to handling the false positive detection after their occurrence, the in-network Honeypot can reduce the number of false detection in advance by increasing the accuracy of anomaly-based IDS. Anomaly-based IDSes raise an alert when they observe any network traffic that deviates from normal patterns. They suffer from two drawbacks: (1) raising false positive alerts on anomaly not due to attacks and (2) introducing false negative detection if the IDSes build the normal patterns based on the network traffic that has already been contaminated by malicious activities [11]. With the help of the in-network Honeypot, we can remedy the negative impacts caused by these two drawbacks. When the in-network Honeypot identifies a suspicious node as malicious, it can use the interactions with them to build the model of adversaries in parallel with the model of normal traffic. The adversaries' model can help to (1) reduce false positive detection by distinguishing attacks from anomalies and (2) reduce false negative detection by removing malicious traffic of suspicious nodes from the model to build the normal traffic patterns.

IV. Evaluation

A. Environment

We used the GENI testbed, a nationwide network experiment platform, to construct IoT networks with a dumbbell topology shown in Figure 3 [12]. In the GENI testbed, we used real SDN-enabled hardware switches to connect virtual machines that simulate communication nodes. Because in GENI testbed we can use virtual machines physically located

in three different areas, the evaluations can reflect the performance of wide area communications in IoT networks. We also constructed networks of three different sizes by changing the number of communication nodes connected to switches. When indicating a network, we add the number of nodes with the name of the network topology in parentheses.

We used DNP3 as the network protocol to exchange information [13]. Although the DNP3 protocol is mainly used in electric and water companies, its complex structure and rich data formats can cover wide varieties of measurements and operations used in different IoT networks.

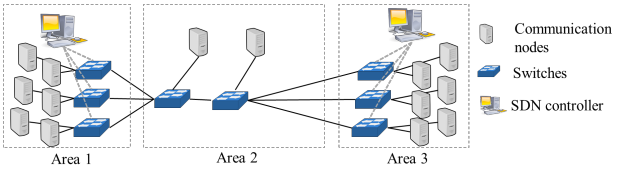


Fig. 3. Dumbbell topology to simulate IoT networks.

We implemented the in-network Honeypot in ONOS, an open-source network operating systems commonly used as SDN controllers in commercial networks [14]. Because we used DNP3 as the protocol for communications, we included in ONOS an encoder to spoof DNP3 packets.

B. Evaluation results

In this section, we evaluate the impact of spoofing network traffic on the performance of IoT networks. Specifically, we spoofed the traffic of 80% computing nodes to simulate a worst case scenario. As shown in Figure 4, we use two performance metrics for evaluations, i.e., the round trip times (RTT) and the throughput of the SDN controller.

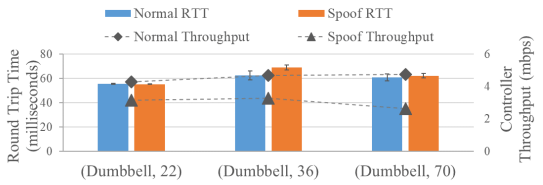


Fig. 4. Performance evaluations on spoofing network traffic (with 99% confidence interval).

On the primary y-axis, we compare the RTT when networks uses the in-network Honeypot to spoof traffic ("spoof RTT") to "normal RTT." From the figure, we can observe that the change of RTT is within 10% (less than 10 milliseconds). Consequently, it can be challenging for adversaries to distinguish the spoofed traffic from the real one based on the observed variations in RTTs.

On the secondary y-axis, we compare the throughput of the SDN controller spoofing network traffic ("spoof throughput") with the throughput when it directly forwards network traffic ("normal throughput"). Compared to the "normal throughput," we can see that there was an approximately 30% decrease on average; throughput of spoofing network packets varied between 2.8 Mbps and 3.5 Mbps. For a DNP3 packet of 1 kilobyte (KB), which can contain more than 200 32-bit measurements, the SDN controller can spoof more than 300 packets per second.

V. Conclusions

In this paper, we propose an in-network Honeypot, which reroutes network traffic from suspicious nodes to an SDN controller to quarantine their communications in an IoT network. After quarantining the suspicious nodes, the SDN controller spoofs network communication with suspicious nodes on behalf of nonexistent phantom nodes. We use both statistic model and physical model to construct contents of the spoofed packets. The spoofed packets can mislead adversaries into targeting phantom nodes and thus to prevent potential physical damage from happening on real communication nodes and the underneath physical processes. Preliminary evaluations on real SDN networks demonstrate that deploying in-network Honeypot introduces small overhead on normal network communications.

References

[1] R. M. Lee, M. J. Assante, and T. Conway, "Analysis of the cyber attack on the Ukrainian power grid," SANS E-ISAC, Tech. Rep., March 2016.

[2] A. A. Gendreau and M. Moorman, "Survey of intrusion detection systems towards an end to end secure internet of things," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, Aug 2016, pp. 84–90.

[3] E. Hodo, X. Bellekens, A. Hamilton, P. L. Dubouilh, E. Iorkyase, C. Tachtatzis, and R. Atkinson, "Threat analysis of iot networks using artificial neural network intrusion detection system," in *2016 International Symposium on Networks, Computers and Communications (ISNCC)*, May 2016, pp. 1–6.

[4] E. Ronen, A. Shamir, A. O. Weingarten, and C. OFlynn, "Iot goes nuclear: Creating a zigbee chain reaction," in *2017 IEEE Symposium on Security and Privacy (SP)*, May 2017, pp. 195–212.

[5] K. Wilhoit and S. Hilt, "The GasPot experiment: Unexamined perils in using gas tank monitoring systems," A TrendLabs Research Paper, Trend Micro Inc.

[6] D. I. Buza, F. Juhász, G. Miru, M. Félegyházi, and T. Holczer, "Cryplh: Protecting smart energy systems from targeted attacks with a plc honeypot," in *International Workshop on Smart Grid Security*. Springer, 2014, pp. 181–192.

[7] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4D approach to network control and management," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, 2005.

[8] R. Jack, "Achieving next generation oss with the tmf zoom, onos and huawei," https://onosproject.org/tag/huawei/.

[9] S. Bera, S. Misra, and A. V. Vasilakos, "Software-defined networking for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1994–2008, Dec 2017.

[10] D. Formby, P. Srinivasan, A. Leonard, J. Rogers, and R. Beyah, "Who's in control of your control system? Device fingerprinting for cyber-physical systems," in *Network and Distributed System Security Symposium (NDSS)*, 2016.

[11] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE Symposium on Security and Privacy*, May 2010, pp. 305–316.

[12] "Geni (global environment for network innovations) exploring networks of the future," Raytheon BBN Technologies, www.geni.net.

[13] "IEEE standard for electric power systems communications-distributed network protocol (dnp3)," *IEEE Std 1815-2012 (Revision of IEEE Std 1815-2010)*, pp. 1–821, Oct 2012.

[14] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "Onos: towards an open, distributed sdn os," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 1–6.